YAYASAN PRIMA AGUS TEKNIK

# Komputasi Awan
## (Cloud Computing)

Dr. Joseph Teguh Santoso, M.Kom

**Komputasi Awan (Cloud Computing)**

**Penulis :**
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

**ISBN : 9 786238 120079**

**Editor :**
Muhammad Sholikan, M.Kom

**Penyunting :**
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

**Desain Sampul dan Tata Letak :**
Irdha Yunianto, S.Ds., M.Kom.

**Penebit** :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

**Redaksi :**
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

**Distributor Tunggal :**
**Universitas STEKOM**
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

# KATA PENGANTAR

Puji Syukur penulis panjatkan atas selesainya buku yang berjudul ***"Komputasi Awan (Cloud Computing)"***. Komputasi awan tersebar luas di lanskap TI. Didorong oleh beberapa faktor yang menyatu dan saling melengkapi, komputasi awan maju sebagai model pengiriman layanan TI yang layak dengan kecepatan yang luar biasa. Hal ini telah menyebabkan perubahan paradigma dalam cara memberikan, menggunakan, dan memanfaatkan berbagai layanan TI yang ditawarkannya. Ini juga menawarkan beberapa keuntungan dibandingkan dengan model komputasi on-premis tradisional, termasuk pengurangan biaya dan peningkatan kegunaan dan fleksibilitas. Potensi transformasinya sangat besar dan melaju dengan cepat, dan akibatnya komputasi awan mulai dilirik oleh pengguna individu, bisnis, lembaga pendidikan, pemerintah, dan organisasi masyarakat. Komputasi awan membantu menutup kesenjangan digital (informasi).

Oleh karena itu, perusahaan dengan penuh semangat berinvestasi dalam teknologi dan layanan komputasi awan yang menjanjikan tidak hanya di negara maju tetapi juga semakin meningkat di negara. Komputasi awan menjadi minat yang cukup besar di antara beberapa pemangku kepentingan bisnis, industri TI, pengembang aplikasi, administrator dan manajer TI, peneliti, dan mahasiswa yang bercita-cita menjadi profesional TI yang sukses. Namun, untuk berhasil merangkul paradigma komputasi baru ini, mereka perlu memperoleh pengetahuan dan keterampilan komputasi awan baru. Sebagai jawaban atas hal ini, universitas mulai menawarkan kursus baru tentang komputasi awan.

Buku ini, dimulai dengan tinjauan singkat tentang berbagai paradigma komputasi dan potensi dari paradigma tersebut, menguraikan dasar-dasar komputasi awan. Kemudian, berkaitan dengan jenis layanan cloud, model penyebaran cloud, teknologi yang mendukung dan menggerakkan cloud, model proses perangkat lunak dan model pemrograman untuk cloud, dan pengembangan aplikasi perangkat lunak yang menjalankan cloud. Buku ini juga membahas masalah keamanan dan masalah dalam komputasi cloud.

Buku ini terbagi menjadi 14 Bab dengan setiap bab dilengkapi dengan pertanyaan ulasan yang membantu pembaca untuk memeriksa pemahaman mereka tentang topik dan masalah yang dieksplorasi dalam bab tersebut. Bab pertama buku ini membahas tentang paradigma atau cara pandang tentang komputasi awan. Bab selanjutnya membahas tentang dasar-dasar komputasi awan, bab ini merujuk pada manfaat dan kerugian komputasi. Bab 3 mejabar kan tentang arsitektur dan manajemen komputasi awan, mencakup pengelolaan nya. Bab 4 membahas tentang model penerapan pada komputasi awan. Melanjutkan bab selanjutnya, bab 5 membahas tentang model layanan komputasi awan. Bab 6 akan membahas tentang teknologi cakupan komputasi awan.

Bab ke 7 tentang virtualisasi yaitu teknologi yang memanfaatkan layanan komputasi awan. Selanjutnya bab 8 membahas tentang model pemrograman komputasi awan. Bab 9 akan membahas lanjutan dari bab 8 yaitu pengembangan perangkat lunak untuk komputas awan. Bab 10 membahas tentang konsep jaringan komputasi awan. Bab 11 membahas

tentang penyedia layanan cloud dari yang gratis hingga berbayar. Bab 12 melanjutkan pemahaman dari bab sebelumnya, yaitu membahas tentang *open source* komputasi awan. Bab 13 akan mengacu dalam pembahasan tentang keamanan yang terkait *platform Cloud computing* hingga kebijakan yang diberikan oleh komputasi awan. Bab terakhir dalam buku ini, akan menjelaskan tentang konsep lanjutan komputasi awan, dari manajerial komputasi awan hingga *cloud analytics.* Saya yakin buku ini informatif, ringkas, komprehensif, dan bermanfaat bagi pembaca untuk mendapatkan pengetahuan cloud.

Semarang, Januari 2023
Penulis

Dr. Joseph Teguh Santoso, M.Kom

# DAFTAR ISI


Halaman Judul ..... i
Kata Pengantar ..... iii
Daftar isi ..... v
**BAB 1 PARADIGMA KOMPUTASI** ..... 1
1.1 Komputasi Berkinerja Tinggi ..... 1
1.2 Komputasi Paralel ..... 1
1.3 Komputasi Terdistribusi ..... 2
1.4 Komputasi Klaster ..... 2
1.5 Komputasi Grid ..... 3
1.6 Komputasi Awan ..... 4
1.7 Biokomputer ..... 4
1.8 Komputasi Seluler ..... 4
1.9 Komputasi Kuantum ..... 5
1.10 Komputasi Optik ..... 5
1.11 Komputasi Nano ..... 5
1.12 Komputasi Jaringan ..... 6
1.13 Ringkasan ..... 6
**BAB 2 DASAR-DASAR KOMPUTASI AWAN ..... 7**
2.1 Motivasi *Cloud Computing* ..... 7
2.2 Definisi *Cloud Computing* ..... 9
2.3 5-4-3 Prinsip *Cloud Computing* ..... 10
2.4 Ekosistem Awan ..... 14
2.5 Persyaratan untuk Layanan *Cloud* ..... 15
2.6 Aplikasi *Cloud* ..... 17
2.7 Manfaat dan Kerugian ..... 18
2.8 Ringkasan ..... 19
**BAB 3 ARSITEKTUR DAN MANAJEMEN *CLOUD COMPUTING* ..... 22**
3.1 Pendahuluan ..... 22
3.2 Arsitektur Awan ..... 23
3.3 Anatomi Awan ..... 24
3.4 Konektivitas Jaringan di *Cloud Computing* ..... 25
3.5 Aplikasi di *Cloud* ..... 27
3.6 Mengelola *Cloud* ..... 30
3.7 Memigrasikan Aplikasi ke *Cloud* ..... 32
3.8 Ringkasan ..... 34
**BAB 4 MODEL PENERAPAN *CLOUD* ..... 36**
4.1 Pendahuluan ..... 36
4.2 *Cloud* Pribadi ..... 37
4.3 *Cloud* Publik ..... 42

4.4 *Cloud* Komunitas ..... 45
4.5 *Cloud* Hibrida ..... 49
4.6 Ringkasan ..... 51
**BAB 5 MODEL LAYANAN *CLOUD* ..... 53**
5.1 Pendahuluan ..... 53
5.2 Infrastruktur sebagai Layanan ..... 56
5.3 Platform sebagai Layanan ..... 61
5.4 Perangkat Lunak sebagai Layanan ..... 68
5.5 Model Layanan *Cloud* Lainnya ..... 73
5.6 Ringkasan ..... 75
**BAB 6 PENGGERAK TEKNOLOGI UNTUK *CLOUD COMPUTING*** ..... 76
6.1 Pendahuluan ..... 76
6.2 SOA dan Cloud ..... 78
6.3 Virtualisasi ..... 84
6.4 Teknologi Multicore ..... 87
6.5 Teknologi Memori dan Penyimpanan ..... 88
6.6 Teknologi Jaringan ..... 91
6.7 Web 2.0 ..... 94
6.8 Web 3.0 ..... 98
6.9 Model Proses Perangkat Lunak untuk Cloud ..... 103
6.10 Model Pemrograman ..... 109
6.11 Komputasi Pervasif ..... 114
6.12 Sistem Operasi ..... 117
6.13 Lingkungan Aplikasi ..... 122
6.14 Ringkasan ..... 127
**BAB 7 VIRTUALISASI ..... 130**
7.1 Pendahuluan ..... 130
7.2 Peluang Virtualisasi ..... 132
7.3 Pendekatan Virtualisasi ..... 136
7.4 Hypervisor ..... 141
7.5 Dari Virtualisasi ke *Cloud Computing* ..... 145
7.6 Ringkasan ..... 149
**BAB 8 MODEL PEMROGRAMAN UNTUK *CLOUD COMPUTING* ..... 152**
8.1 Pendahuluan ..... 153
8.2 Model Pemrograman yang Diperluas untuk *Cloud* ..... 153
8.3 Model Pemrograman Baru Diusulkan untuk *Cloud* ..... 164
8.4 Ringkasan ..... 169
**BAB 9 PENGEMBANGAN PERANGKAT LUNAK DI *CLOUD* ..... 172**
9.1 Pendahuluan ..... 172
9.2 Perspektif Berbeda tentang Pengembangan SaaS ..... 175
9.3 Tantangan Baru ..... 179
9.4 Pengembangan Perangkat Lunak Cloud-Aware Pada Teknologi PaaS ..... 181

9.5 Ringkasan ..... 189
**BAB 10 JARINGAN UNTUK *CLOUD COMPUTING* ..... 192**
10.1 Pendahuluan ..... 192
10.2 Tinjauan Lingkungan Pusat Data ..... 194
10.3 Masalah Jaringan di Pusat Data ..... 199
10.4 Masalah Transport Layer di DCN ..... 199
10.5 Peningkatan TCP untuk DCN ..... 204
10.6 Ringkasan ..... 216
**BAB 11 PENYEDIA LAYANAN CLOUD ..... 217**
11.1 Pendahuluan ..... 217
11.2 EMC ..... 218
11.3 Google ..... 220
11.4 Layanan Web Amazon ..... 223
11.5 Microsoft ..... 227
11.6 IBM ..... 228
11.7 SAP Laps ..... 230
11.8 Salesforce.com ..... 231
11.9 Rackspace ..... 232
11.10 VMware ..... 233
11.11 Majrasoft ..... 235
11.12 Ringkasan ..... 236
**BAB 12 DUKUNGAN OPEN SOURCE UNTUK CLOUD COMPUTING ..... 239**
12.1 Pendahuluan ..... 239
12.2 Layanan *Open Source* untuk IaaS ..... 241
12.3 Layanan *Open Source* untuk PaaS ..... 247
12.4 Layanan *Open Source* untuk SaaS ..... 249
12.5 Layanan *Open Source* untuk Riset ..... 252
12.6 Alat Komputasi Terdistribusi untuk Pengelolaan Sistem Terdistribusi ..... 255
12.7 Ringkasan ..... 259
**BAB 13 KEAMANAN DI *CLOUD COMPUTING* ..... 260**
13.1 Pendahuluan ..... 260
13.2 Aspek Keamanan ..... 262
13.3 Keamanan Terkait Platform ..... 271
13.4 Audit dan Kepatuhan ..... 275
13.5 Ringkasan ..... 277
**BAB 14 KONSEP LANJUTAN DALAM CLOUD COMPUTING ..... 279**
14.1 Intercloud ..... 279
14.2 Manajemen Awan ..... 282
14.3 Awan Seluler ..... 283
14.4 Media Cloud ..... 285
14.5 Interoperabilitas dan Standar ..... 287
14.6 Tata Kelola *Cloud* ..... 288

14.7 Kecerdasan Komputasi di Cloud ..... 290
14.8 Awan Hijau ..... 291
14.9 *Cloud Analytics* ..... 293
14.10 Ringkasan ..... 295
**Daftar Pustaka ..... 297**

# BAB 1
# PARADIGMA KOMPUTASI

**Tujuan pembelajaran**

Tujuan dari bab ini adalah untuk

- Berikan penjelasan singkat tentang jurusan komputasi
- Periksa potensi paradigma ini

**Pengantar**

Istilah paradigma menyampaikan bahwa ada seperangkat praktik yang harus diikuti untuk menyelesaikan suatu tugas. Dalam domain komputasi, ada banyak praktik standar berbeda yang diikuti berdasarkan penemuan dan kemajuan teknologi. Dalam bab ini, kita melihat ke dalam berbagai paradigma komputasi: yaitu komputasi kinerja tinggi, komputasi cluster, komputasi grid, komputasi awan, komputasi bio, komputasi seluler, komputasi kuantum, komputasi optik, komputasi nano, dan komputasi jaringan. Ketika sistem komputasi menjadi lebih cepat dan lebih mampu, fitur-fitur komputasi modern perlu diperhatikan untuk menghubungkan diri kita dengan judul buku ini tentang komputasi awan, dan oleh karena itu menjadi penting untuk mengetahui sedikit tentang berbagai paradigma komputasi.

## 1.1 KOMPUTASI BERKINERJA TINGGI

Dalam sistem komputasi berkinerja tinggi, kumpulan prosesor (mesin prosesor atau unit pemrosesan pusat [CPU]) terhubung (dalam jaringan) dengan sumber daya lain seperti memori, penyimpanan, dan perangkat input dan output, dan perangkat lunak yang diterapkan diaktifkan untuk berjalan di seluruh sistem komponen yang terhubung.

Mesin pemroses dapat berupa tipe homogen atau heterogen. Makna warisan komputasi kinerja tinggi (HPC) adalah superkomputer; Namun, itu tidak benar dalam skenario komputasi saat ini. Oleh karena itu, HPC juga dapat dikaitkan dengan paradigma komputasi lain yang dibahas di bagian yang akan datang, karena ini adalah nama umum untuk semua sistem komputasi ini.

Jadi, contoh HPC termasuk sekelompok kecil komputer desktop atau komputer pribadi (PC) hingga superkomputer tercepat. Sistem HPC biasanya ditemukan dalam aplikasi yang memerlukan penggunaan atau pemecahan masalah ilmiah. Sebagian besar waktu, tantangan dalam menangani masalah semacam ini adalah melakukan studi simulasi yang sesuai, dan ini dapat diselesaikan oleh HPC tanpa kesulitan. Contoh ilmiah seperti pelipatan protein dalam biologi molekuler dan studi tentang pengembangan model dan aplikasi berdasarkan fusi nuklir patut dicatat sebagai aplikasi potensial untuk HPC.

## 1.2 KOMPUTASI PARALEL

Komputasi paralel juga merupakan salah satu aspek HPC. Di sini, satu set prosesor bekerja secara kooperatif untuk memecahkan masalah komputasi. Mesin prosesor atau CPU ini sebagian besar bertipe homogen. Oleh karena itu, definisi ini sama dengan HPC dan cukup luas untuk mencakup superkomputer yang memiliki ratusan atau ribuan prosesor yang saling

berhubungan dengan sumber daya lainnya. Seseorang dapat membedakan antara komputer konvensional (juga dikenal sebagai serial atau berurutan atau Von Neumann) dan komputer paralel dalam cara aplikasi dijalankan.

Di komputer serial atau berurutan, hal berikut berlaku:

- Ini berjalan pada satu komputer/mesin prosesor yang memiliki CPU tunggal.
- Masalah dipecah menjadi serangkaian instruksi diskrit.
- Instruksi dieksekusi satu demi satu.

Dalam komputasi paralel, karena ada penggunaan beberapa mesin prosesor secara bersamaan, hal berikut berlaku:

- Dijalankan menggunakan banyak prosesor (multiple CPUs).
- Suatu masalah dipecah menjadi bagian-bagian diskrit yang dapat diselesaikan secara bersamaan.
- Setiap bagian selanjutnya dipecah menjadi serangkaian instruksi.
- Instruksi dari setiap bagian dijalankan secara bersamaan pada prosesor yang berbeda.
- Mekanisme kontrol/koordinasi keseluruhan digunakan.

## 1.3 KOMPUTASI TERDISTRIBUSI

Komputasi terdistribusi juga merupakan sistem komputasi yang terdiri dari beberapa komputer atau mesin prosesor yang terhubung melalui jaringan, yang bisa homogen atau heterogen, tetapi dijalankan sebagai satu sistem. Konektivitas dapat sedemikian rupa sehingga CPU dalam sistem terdistribusi dapat secara fisik berdekatan dan terhubung oleh jaringan lokal, atau mereka dapat jauh secara geografis dan dihubungkan oleh jaringan area luas. Heterogenitas dalam sistem terdistribusi mendukung sejumlah kemungkinan konfigurasi dalam mesin prosesor, seperti mainframe, PC, workstation, dan komputer mini. Tujuan komputasi terdistribusi adalah membuat jaringan seperti itu berfungsi sebagai satu komputer.

Sistem komputasi terdistribusi lebih menguntungkan daripada sistem terpusat, karena ada dukungan untuk fitur karakteristik berikut:

1. Skalabilitas: Merupakan kemampuan sistem untuk dapat diperluas dengan mudah dengan menambahkan lebih banyak mesin sesuai kebutuhan, dan sebaliknya, tanpa mempengaruhi pengaturan yang ada.
2. Redundansi atau replikasi: Di sini, beberapa mesin dapat menyediakan layanan yang sama, sehingga meskipun tidak tersedia (atau gagal), pekerjaan tidak berhenti karena dukungan komputasi lain yang serupa akan tersedia.

## 1.4 KOMPUTASI KLASTER

Sistem komputasi cluster terdiri dari satu set mesin prosesor yang sama atau serupa yang terhubung menggunakan infrastruktur jaringan khusus. Semua mesin prosesor berbagi sumber daya seperti direktori home umum dan memiliki perangkat lunak seperti implementasi *message passing interface* (MPI) yang diinstal untuk memungkinkan program dijalankan di semua node secara bersamaan. Ini juga semacam kategori HPC. Komputer individu dalam sebuah cluster dapat disebut sebagai node. Alasan untuk mewujudkan cluster sebagai HPC adalah karena fakta bahwa masing-masing node dapat bekerja sama untuk

memecahkan masalah yang lebih besar daripada yang dapat diselesaikan dengan mudah oleh komputer mana pun. Dan, node perlu berkomunikasi satu sama lain untuk bekerja secara kooperatif dan bermakna bersama untuk memecahkan masalah di tangan. Jika kita memiliki mesin prosesor dengan tipe heterogen dalam sebuah cluster, cluster semacam ini menjadi subtipe dan sebagian besar masih dalam tahap percobaan atau penelitian.

## 1.5 KOMPUTASI JARINGAN

Sumber daya komputasi di sebagian besar organisasi kurang dimanfaatkan tetapi diperlukan untuk operasi tertentu. Gagasan komputasi grid adalah memanfaatkan daya komputasi yang tidak terpakai oleh organisasi yang membutuhkan, dan dengan demikian laba atas investasi (ROI) pada investasi komputasi dapat ditingkatkan.

Dengan demikian, komputasi grid adalah jaringan mesin komputasi atau prosesor yang dikelola dengan semacam perangkat lunak seperti middleware, untuk mengakses dan menggunakan sumber daya dari jarak jauh. Aktivitas pengelolaan sumber daya jaringan melalui middleware disebut layanan jaringan. Layanan grid menyediakan kontrol akses, keamanan, akses ke data termasuk perpustakaan digital dan database, dan akses ke fasilitas penyimpanan interaktif dan jangka panjang berskala besar.

**Tabel 1.1** Jaringan Tenaga Listrik dan Komputasi Jaringan

| Jaringan Tenaga Listrik | Komputasi Jaringan |
|---|---|
| Jangan pernah khawatir dari mana listrik yang kita gunakan berasal; yaitu, apakah itu dari batu bara di Australia, dari tenaga angin di Amerika Serikat, atau dari pembangkit nuklir di Prancis, seseorang cukup mencolokkan alat listrik ke stopkontak yang terpasang di dinding dan alat itu akan mendapatkan daya listrik yang kita butuhkan. untuk mengoperasikan alat. | Jangan pernah khawatir tentang dari mana asal daya komputer yang kita gunakan; yaitu, apakah itu dari superkomputer di Jerman, peternakan komputer di India, atau laptop di Selandia Baru, seseorang cukup menyambungkan komputer dan Internet dan eksekusi aplikasi akan selesai. |
| Infrastruktur yang memungkinkan ini disebut jaringan listrik. Ini menghubungkan berbagai jenis pembangkit listrik dengan rumah kita, melalui stasiun transmisi, pembangkit listrik, trafo, saluran listrik, dan sebagainya. | Infrastruktur yang memungkinkan ini disebut jaringan komputasi. Ini menghubungkan bersama sumber daya komputasi, seperti PC, workstation, server, dan elemen penyimpanan, dan menyediakan mekanisme yang diperlukan untuk mengaksesnya melalui Internet. |
| Jaringan listrik tersebar luas: listrik pada dasarnya tersedia di mana-mana, dan seseorang dapat dengan mudah | Grid juga meresap dalam arti bahwa sumber daya komputasi jarak jauh dapat diakses dari berbagai platform, termasuk laptop dan ponsel, dan seseorang dapat |

| | |
|---|---|
| mengaksesnya melalui soket standar yang terpasang di dinding. | dengan mudah mengakses daya komputasi grid melalui browser web. |
| Jaringan listrik adalah utilitas: kami meminta listrik dan kami mendapatkannya. Kami juga membayar untuk apa yang kami dapatkan. | Komputasi grid juga merupakan utilitas: kami meminta daya komputasi atau kapasitas penyimpanan dan kami mendapatkannya. Kami juga membayar untuk apa yang kami dapatkan. |

## 1.6 KOMPUTASI AWAN

Kecenderungan komputasi bergerak menuju awan dari konsep komputasi grid, terutama ketika sumber daya komputasi yang besar diperlukan untuk memecahkan satu masalah, dengan menggunakan ide daya komputasi sebagai utilitas dan konsep sekutu lainnya. Namun, potensi perbedaan antara grid dan *cloud* adalah bahwa komputasi grid mendukung pemanfaatan beberapa komputer secara paralel untuk menyelesaikan aplikasi tertentu, sedangkan komputasi awan mendukung pemanfaatan banyak sumber daya, termasuk sumber daya komputasi, untuk memberikan layanan terpadu kepada pengguna akhir.

Dalam komputasi awan, sumber daya TI dan bisnis, seperti server, penyimpanan, jaringan, aplikasi, dan proses, dapat disediakan secara dinamis untuk memenuhi kebutuhan dan beban kerja pengguna. Selain itu, sementara *cloud* dapat menyediakan dan mendukung grid, *cloud* juga dapat mendukung lingkungan nongrid, seperti arsitektur web tiga tingkat yang berjalan pada aplikasi tradisional atau Web 2.0. Kami akan melihat detail komputasi awan di berbagai bab buku ini.

## 1.7 BIOKOMPUTER

Sistem biokomputasi menggunakan konsep molekul (atau model) yang diturunkan atau disimulasikan secara biologis yang melakukan proses komputasi untuk memecahkan masalah. Model yang diturunkan secara biologis membantu menyusun program komputer yang menjadi bagian dari aplikasi.

Biocomputing memberikan latar belakang teoretis dan alat praktis bagi para ilmuwan untuk mengeksplorasi protein dan DNA. DNA dan protein adalah bahan penyusun alam, tetapi bahan penyusun ini tidak persis digunakan sebagai batu bata; fungsi molekul akhir sangat bergantung pada urutan blok-blok ini. Dengan demikian, ilmuwan biokomputasi bekerja untuk menemukan urutan yang cocok untuk berbagai aplikasi yang meniru biologi. Oleh karena itu, biocomputing akan mengarah pada pemahaman yang lebih baik tentang kehidupan dan penyebab molekuler dari penyakit tertentu.

## 1.8 KOMPUTASI SELULER

Dalam komputasi seluler, elemen pemrosesan (atau komputasi) berukuran kecil (yaitu, perangkat genggam) dan komunikasi antara berbagai sumber daya berlangsung menggunakan

media nirkabel. Komunikasi seluler untuk aplikasi suara (mis., telepon seluler) secara luas didirikan di seluruh dunia dan menyaksikan pertumbuhan yang sangat pesat di semua dimensi termasuk peningkatan jumlah pelanggan dari berbagai jaringan seluler. Perpanjangan dari teknologi ini adalah kemampuan untuk mengirim dan menerima data di berbagai jaringan seluler dengan menggunakan perangkat kecil seperti telepon pintar. Ada banyak aplikasi berdasarkan teknologi ini; misalnya, panggilan video atau konferensi adalah salah satu aplikasi penting yang orang lebih suka gunakan sebagai pengganti komunikasi suara (hanya) yang ada di ponsel. Aplikasi berbasis komputasi seluler menjadi sangat penting dan berkembang pesat dengan berbagai kemajuan teknologi karena memungkinkan pengguna untuk mengirimkan data dari lokasi terpencil ke lokasi jauh atau tetap lainnya.

## 1.9 KOMPUTASI KUANTUM

Produsen sistem komputasi mengatakan bahwa ada batas untuk menjejalkan lebih banyak transistor ke dalam ruang sirkuit terpadu (IC) yang lebih kecil dan lebih kecil dan dengan demikian menggandakan kekuatan pemrosesan setiap 18 bulan. Masalah ini harus diatasi dengan solusi baru berbasis komputasi kuantum, di mana ketergantungannya ada pada informasi kuantum, aturan yang mengatur dunia subatomik. Komputer kuantum jutaan kali lebih cepat daripada superkomputer terkuat kita saat ini. Karena komputasi kuantum bekerja secara berbeda pada tingkat yang paling mendasar dari teknologi saat ini, dan meskipun ada prototipe yang berfungsi, sistem ini sejauh ini belum terbukti menjadi alternatif untuk mesin berbasis silikon saat ini.

## 1.10 KOMPUTASI OPTIK

Sistem komputasi optik menggunakan foton dalam cahaya tampak atau sinar inframerah, bukan arus listrik, untuk melakukan perhitungan digital. Arus listrik mengalir hanya sekitar 10% dari kecepatan cahaya. Ini membatasi kecepatan pertukaran data dalam jarak jauh dan merupakan salah satu faktor yang menyebabkan evolusi serat optik. Dengan menerapkan beberapa keunggulan jaringan tampak dan/atau IR pada skala perangkat dan komponen, komputer dapat dikembangkan yang dapat melakukan operasi 10 kali atau lebih cepat daripada komputer elektronik konvensional.

## 1.11 KOMPUTASI NANO

Nanocomputing (Komputasi Nano) mengacu pada sistem komputasi yang dibangun dari komponen berskala nano. Transistor silikon di komputer tradisional dapat digantikan oleh transistor berbasis karbon nanotube.

Keberhasilan realisasi nanokomputer berkaitan dengan skala dan integrasi nanotube atau komponen ini. Masalah skala berhubungan dengan dimensi komponen; mereka, paling banyak, beberapa nanometer setidaknya dalam dua dimensi. Masalah integrasi komponen ada dua: pertama, pembuatan pola arbitrer kompleks mungkin tidak layak secara ekonomi, dan kedua, komputer nano dapat mencakup perangkat dalam jumlah besar. Para peneliti sedang mengerjakan semua masalah ini untuk mewujudkan komputasi nano.

## 1.12 KOMPUTASI JARINGAN

Komputasi jaringan adalah cara merancang sistem untuk memanfaatkan teknologi terbaru dan memaksimalkan dampak positifnya pada solusi bisnis dan kemampuan mereka untuk melayani pelanggan mereka menggunakan jaringan dasar yang kuat dari sumber daya komputasi. Dalam solusi komputasi jaringan apa pun, komponen klien dari arsitektur atau aplikasi jaringan akan bersama pelanggan atau klien atau pengguna akhir, dan di zaman modern, mereka menyediakan seperangkat fungsi penting yang diperlukan untuk mendukung fungsi klien yang sesuai di biaya minimum dan kesederhanaan maksimum. Tidak seperti PC konvensional, mereka tidak perlu dikonfigurasi dan dipelihara secara individual sesuai dengan tujuan penggunaannya. Ujung lain dari komponen klien dalam arsitektur jaringan akan menjadi lingkungan server tipikal untuk mendorong layanan aplikasi ke ujung klien.

Hampir semua paradigma komputasi yang telah dibahas sebelumnya bersifat seperti ini. Bahkan di masa depan, jika ada orang yang menciptakan paradigma komputasi yang benar-benar baru, itu akan didasarkan pada arsitektur jaringan, yang tanpanya tidak mungkin mewujudkan manfaat bagi pengguna akhir mana pun.

## 1.13 RINGKASAN

Kita memasuki era pasca-PC, di mana lebih banyak dan beragam komputer dan paradigma komputasi dengan ukuran dan fungsi yang berbeda dapat digunakan di mana saja dan dengan setiap manusia; jadi, tujuan dari bab ini adalah untuk mengilustrasikan secara singkat ide-ide dari semua domain komputasi ini, karena sebagian besar ada di mana-mana dan meresap dalam akses dan lingkungan kerjanya.

**Poin Kunci**

- Komputasi bergerak: Komputasi bergerak terdiri dari elemen pemrosesan kecil (yaitu, perangkat genggam) dan komunikasi antara berbagai sumber daya dilakukan dengan menggunakan media nirkabel.
- Nanocomputing: Memanfaatkan komponen berskala nano.

**Latihan Soal**

1. Mengapa perlu memahami berbagai paradigma komputasi?
2. Bandingkan komputasi jaringan dengan jaringan tenaga listrik
3. Akankah komputasi seluler memainkan peran dominan di masa depan? Membahas
4. Bagaimana komputasi terdistribusi dan komputasi jaringan berbeda atau serupa?
5. Bagaimana komputasi nano dapat membentuk perangkat masa depan?

# BAB 2
# DASAR-DASAR *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan dari bab ini adalah untuk

- Memahami ide dasar dan motivasi untuk komputasi awan
- Untuk mendefinisikan komputasi awan
- Memahami prinsip 5-4-3 komputasi awan dan ekosistem awan
- Memahami cara kerja aplikasi *cloud*
- Memiliki pemahaman singkat mengenai keuntungan dan kerugian dalam *cloud computing*

**Pengantar**

Komputasi modern dengan laptop atau desktop kita atau bahkan dengan tablet/smartphone menggunakan Internet untuk mengakses data dan detail yang kita inginkan, yang terletak/disimpan di tempat/komputer yang jauh, melalui wajah aplikasi seperti Facebook, e-mail, dan YouTube, menghadirkan kekuatan informasi aktual yang kita butuhkan secara instan dalam waktu singkat. Bahkan jika jutaan pengguna terhubung dengan cara ini, dari mana saja di dunia, aplikasi ini melayani apa yang diinginkan pengguna-pelanggan ini. Fenomena penyediaan informasi atau data lain dan perincian kepada semua pelanggan yang membutuhkan, sebagaimana dan ketika diminta, adalah pemahaman konseptual dan cara kerja dari apa yang dikenal sebagai *cloud computing*. Bab ini dikhususkan untuk memberikan pemahaman dasar tentang *cloud computing.*

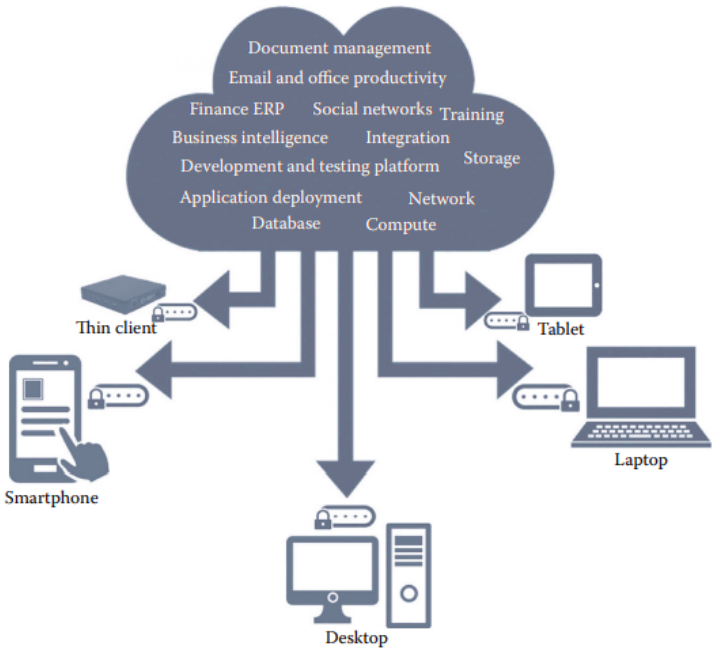
## 2.1 MOTIVASI UNTUK *CLOUD COMPUTING*

Mari kita tinjau skenario komputasi sebelum pengumuman dan ketersediaan komputasi awan: Pengguna yang membutuhkan komputasi diharapkan menginvestasikan uang untuk sumber daya komputasi seperti perangkat keras, perangkat lunak, jaringan, dan penyimpanan; investasi ini secara alami menghabiskan banyak uang bagi pengguna karena mereka harus membeli sumber daya komputasi ini, menyimpannya di tempat mereka, dan memelihara serta membuatnya beroperasi—semua tugas ini akan menambah biaya. Dan, ini adalah pengeluaran yang benar dan sangat besar untuk perusahaan yang membutuhkan daya dan sumber daya komputasi yang sangat besar, dibandingkan dengan akademisi dan individu klasik.

Di sisi lain, mudah dan praktis untuk mendapatkan daya dan sumber daya komputasi yang diperlukan dari beberapa penyedia (atau pemasok) jika diperlukan dan hanya membayar untuk penggunaan tersebut. Ini hanya akan menelan biaya investasi atau pengeluaran yang masuk akal, dibandingkan dengan investasi besar saat membeli seluruh infrastruktur komputasi. Fenomena ini dapat dilihat sebagai belanja modal versus belanja operasional. Karena seseorang dapat dengan mudah menilai lump sum yang sangat besar yang diperlukan untuk pengeluaran modal (seluruh investasi dan pemeliharaan untuk infrastruktur komputasi) dan membandingkannya dengan lump sum sedang atau lebih kecil yang diperlukan untuk

mempekerjakan atau mendapatkan infrastruktur komputasi hanya sesuai dengan waktu yang dibutuhkan, dan sisa waktu bebas dari itu. Oleh karena itu, komputasi awan adalah mekanisme membawa-mempekerjakan atau mendapatkan layanan daya atau infrastruktur komputasi ke tingkat organisasi atau individu sejauh yang diperlukan dan membayar hanya untuk layanan yang dikonsumsi.

Seseorang dapat membandingkan situasi ini dengan penggunaan listrik (layanannya) dari produsen sekaligus distributornya (di India, dewan listrik milik negara/pemerintah yang memberikan pasokan listrik ke semua tempat tinggal dan organisasi) ke rumah-rumah. atau organisasi; di sini, kami tidak menghasilkan listrik (sebanding dengan tugas yang berhubungan dengan produksi listrik); sebaliknya, kami menggunakannya hanya untuk menyetel kebutuhan kami di tempat kami, seperti untuk penerangan dan penggunaan peralatan listrik lainnya, dan membayar sesuai dengan nilai pembacaan meteran listrik. Oleh karena itu, *cloud computing* sangat dibutuhkan dalam mendapatkan layanan sumber daya komputasi. Dengan demikian, dapat dikatakan sebagai jawaban satu baris untuk kebutuhan komputasi awan yang menghilangkan investasi komputasi yang besar tanpa mengurangi penggunaan komputasi pada tingkat pengguna dengan biaya operasional. *Cloud computing* sangat ekonomis dan menghemat banyak uang. Manfaat buta dari komputasi ini adalah bahwa bahkan jika kita kehilangan laptop kita atau karena beberapa krisis komputer pribadi kita — dan sistem desktop — rusak, tetap saja data dan file kita akan tetap aman dan terlindungi karena ini tidak ada di mesin lokal kita (tetapi jarak jauh di tempat penyedia — mesin).




**Gambar 2.1** Komputasi Awan.

Selain itu, seseorang dapat berpikir untuk menambahkan keamanan saat mengakses sumber daya komputasi jarak jauh ini seperti yang digambarkan pada Gambar 2.1.

Gambar 2.1 menunjukkan beberapa aplikasi *cloud computing. Cloud* mewakili sumber daya komputasi berbasis Internet, dan aksesibilitasnya melalui beberapa dukungan konektivitas yang aman. Ini adalah solusi komputasi yang semakin populer, terutama di kalangan individu dan perusahaan kecil dan menengah (UKM). Dalam model komputasi awan, kekuatan komputer inti organisasi berada di luar lokasi dan pada dasarnya berlangganan daripada dimiliki.

Dengan demikian, komputasi awan menjadi fokus dan sangat dibutuhkan hanya ketika kita berpikir tentang sumber daya komputasi dan solusi teknologi informasi (TI) apa yang dibutuhkan. Kebutuhan ini memenuhi cara untuk meningkatkan kapasitas atau menambah kemampuan dengan cepat tanpa berinvestasi dalam infrastruktur baru, melatih personel baru, atau melisensikan perangkat lunak baru. Komputasi awan mencakup model layanan berbasis langganan atau bayar per penggunaan yang menawarkan komputasi kepada pengguna akhir atau pelanggan melalui Internet dan dengan demikian memperluas kemampuan TI yang ada.

## 2.2 DEFINISI *CLOUD COMPUTING*

Dalam istilah yang paling sederhana, komputasi awan berarti menyimpan dan mengakses data dan program melalui Internet dari lokasi atau komputer yang jauh, bukan dari hard drive komputer kita. Lokasi jauh yang disebut ini memiliki beberapa sifat seperti skalabilitas, elastisitas, dll., Yang sangat berbeda dari mesin jarak jauh yang sederhana. Awan hanyalah metafora untuk Internet. Saat kami menyimpan data atau menjalankan program dari hard drive komputer lokal, itu disebut penyimpanan dan komputasi lokal. Agar dapat dianggap komputasi awan, kita perlu mengakses data atau program kita melalui Internet. Hasil akhirnya adalah sama; Namun, dengan koneksi online, komputasi awan dapat dilakukan di mana saja, kapan saja, dan dengan perangkat apa saja.

**NIST Pengertian *Cloud computing***

Definisi formal komputasi awan berasal dari National Institute of Standards and Technology (NIST): “Komputasi awan adalah model untuk memungkinkan akses jaringan di mana-mana, nyaman, sesuai permintaan ke kumpulan bersama sumber daya komputasi yang dapat dikonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat disediakan dan dirilis dengan cepat dengan upaya manajemen minimal atau interaksi penyedia layanan. Model *cloud* ini terdiri dari lima karakteristik penting, tiga model layanan, dan empat model penyebaran. Ini berarti bahwa sumber daya atau infrastruktur komputasi — baik itu perangkat keras server, penyimpanan, jaringan, atau perangkat lunak aplikasi — semuanya tersedia dari vendor *cloud* atau situs/premis penyedia, dapat diakses melalui Internet dari lokasi jarak jauh mana pun dan oleh perangkat komputasi lokal mana pun. Selain itu, penggunaan atau aksesibilitas adalah biaya hanya untuk tingkat penggunaan kepada pelanggan berdasarkan kebutuhan dan permintaan mereka, juga dikenal sebagai model *pay-as-you-go* atau *pay-as-per-use.* Jika kebutuhan lebih banyak, lebih banyak sumber daya komputasi kuantum disediakan (penyediaan dengan elastisitas) oleh penyedia. Upaya manajemen minimal menyiratkan bahwa di sisi pelanggan, pemeliharaan sistem komputasi sangat minim karena mereka harus melihat tugas-tugas ini hanya untuk perangkat komputasi

lokal mereka yang digunakan untuk mengakses sumber daya berbasis *cloud*, bukan untuk sumber daya komputasi yang dikelola di penyedia.

***Cloud computing* Adalah Layanan**

Hal paling sederhana yang dilakukan komputer mana pun adalah memungkinkan kita menyimpan dan mengambil informasi. Kita dapat menyimpan foto keluarga kita, lagu favorit kita, atau bahkan menyimpan film di dalamnya, yang juga merupakan layanan paling dasar yang ditawarkan oleh *cloud computing*. Mari kita lihat contoh aplikasi populer bernama Flickr untuk mengilustrasikan arti dari bagian ini.

Sementara Flickr dimulai dengan penekanan pada berbagi foto dan gambar, Flickr telah muncul sebagai tempat yang bagus untuk menyimpan gambar tersebut. Dalam banyak hal, lebih baik menyimpan gambar di komputer Anda:

1. Pertama, Flickr memungkinkan kita mengakses gambar dengan mudah di mana pun kita berada atau jenis perangkat apa pun yang kita gunakan. Meskipun kita mungkin mengunggah foto-foto liburan kita dari komputer di rumah, nanti kita dapat dengan mudah mengaksesnya dari laptop kita di kantor.
2. Kedua, Flickr memungkinkan kita berbagi gambar. Tidak perlu membakarnya ke CD atau menyimpannya di flash drive. Kami dapat mengirimkan alamat Flickr kami kepada seseorang untuk membagikan foto atau gambar ini.
3. Ketiga, Flickr menyediakan keamanan data. Dengan mengunggah gambar ke Flickr, kami memberi diri kami keamanan data dengan membuat cadangan di web. Dan, meskipun yang terbaik adalah menyimpan salinan lokal—baik di komputer, CD, atau flash drive—kenyataannya adalah kita jauh lebih mungkin kehilangan gambar yang kita simpan secara lokal daripada Flickr yang kehilangan gambar kita.

***Cloud computing* Adalah Sebuah Platform**

World Wide Web (WWW) dapat dianggap sebagai sistem operasi untuk semua aplikasi berbasis Internet kami. Namun, perlu dipahami bahwa kita akan selalu membutuhkan sistem operasi lokal di komputer kita untuk mengakses aplikasi berbasis web.

Arti dasar dari istilah platform adalah dukungan di mana aplikasi berjalan atau memberikan hasil kepada pengguna. Misalnya, Microsoft Windows adalah sebuah platform. Tapi, platform tidak harus berupa sistem operasi. Java adalah platform meskipun itu bukan sistem operasi.

Melalui komputasi awan, web menjadi platform. Dengan tren (aplikasi) seperti Office 2.0, semakin banyak aplikasi yang semula tersedia di komputer desktop kini diubah menjadi aplikasi web-*cloud*. Pengolah kata seperti Buzzword dan office suites seperti Google Docs sekarang tersedia di *cloud* sebagai rekanan desktop mereka. Semua jenis tren dalam menyediakan aplikasi melalui *cloud* mengubah komputasi awan menjadi platform atau bertindak sebagai platform.

## 2.3 5-4-3 PRINSIP *CLOUD COMPUTING*

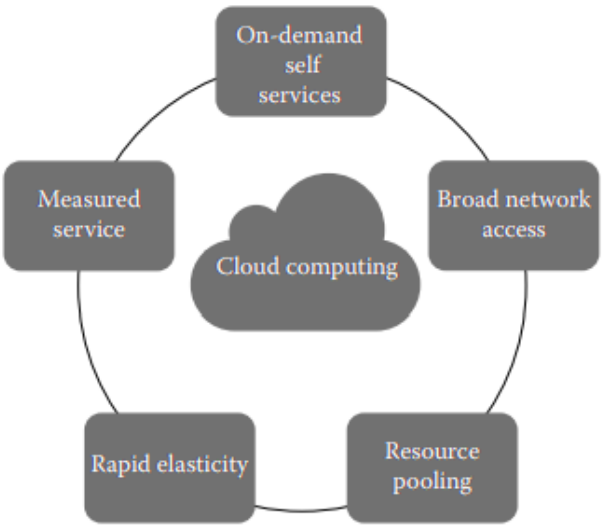
Prinsip 5-4-3 yang dikemukakan oleh NIST menggambarkan (a) lima fitur karakteristik penting yang mempromosikan komputasi awan, (b) empat model penerapan yang digunakan

untuk menceritakan peluang komputasi awan bagi pelanggan sambil melihat model arsitektur, dan (c) tiga model penawaran layanan penting dan dasar dari komputasi awan.

**Lima Karakteristik Esensial**

Komputasi awan memiliki lima karakteristik penting, yang ditunjukkan pada Gambar 2.2. Pembaca dapat mencatat kata esensial, yang artinya jika salah satu dari karakteristik ini hilang, maka itu bukan komputasi awan:

1. Layanan mandiri sesuai permintaan: Konsumen dapat secara sepihak menyediakan kemampuan komputasi, seperti waktu server dan penyimpanan jaringan, sesuai kebutuhan secara otomatis tanpa memerlukan interaksi manusia dengan masing-masing penyedia layanan.
2. Akses jaringan yang luas: Kemampuan tersedia melalui jaringan dan diakses melalui mekanisme standar yang mendorong penggunaan platform klien tipis atau tebal yang heterogen (misalnya ponsel, laptop, dan asisten digital pribadi [PDA]).




**Gambar 2.2** Karakteristik penting dari komputasi awan.

3. Penyatuan sumber daya elastis: Sumber daya komputasi penyedia dikumpulkan untuk melayani banyak konsumen menggunakan model multitenant, dengan sumber daya fisik dan virtual yang berbeda secara dinamis ditetapkan dan ditetapkan kembali sesuai dengan permintaan konsumen. Ada rasa independensi lokasi di mana pelanggan umumnya tidak memiliki kendali atau pengetahuan atas lokasi yang tepat dari sumber daya yang disediakan tetapi mungkin dapat menentukan lokasi pada tingkat abstraksi yang lebih tinggi (misalnya, negara, negara bagian, atau data). tengah). Contoh sumber daya termasuk penyimpanan, pemrosesan, memori, dan bandwidth jaringan.
4. Elastisitas yang cepat: Kemampuan dapat disediakan secara cepat dan elastis, dalam beberapa kasus secara otomatis, untuk diskalakan dengan cepat dan dilepaskan dengan cepat untuk diskalakan dengan cepat. Bagi konsumen, kemampuan yang tersedia untuk penyediaan sering tampak tidak terbatas dan dapat dibeli dalam jumlah berapa pun kapan saja.

5. Layanan terukur: Sistem *cloud* secara otomatis mengontrol dan mengoptimalkan penggunaan sumber daya dengan memanfaatkan kemampuan pengukuran pada beberapa tingkat abstraksi yang sesuai dengan jenis layanan (misalnya, penyimpanan, pemrosesan, bandwidth, dan akun pengguna aktif). Penggunaan sumber daya dapat dipantau, dikendalikan, dan dilaporkan memberikan transparansi bagi penyedia dan konsumen layanan yang digunakan.

**Empat Model Penerapan *Cloud***

Model penerapan menjelaskan cara layanan *cloud* dapat diterapkan atau disediakan untuk pelanggannya, bergantung pada struktur organisasi dan lokasi penyediaan. Seseorang juga dapat memahaminya dengan cara ini: sumber daya komputasi berbasis *cloud* (Internet)—yaitu, lokasi di mana data dan layanan diperoleh dan disediakan untuk pelanggannya—dapat mengambil berbagai bentuk. Empat model penerapan biasanya dibedakan, yaitu penggunaan layanan *cloud* publik, pribadi, komunitas, dan hybrid:

1. *Cloud* pribadi: Infrastruktur *cloud* disediakan untuk penggunaan eksklusif oleh satu organisasi yang terdiri dari banyak konsumen (mis., unit bisnis). Ini mungkin dimiliki, dikelola, dan dioperasikan oleh organisasi, pihak ketiga, atau beberapa kombinasi dari mereka, dan mungkin ada di dalam atau di luar lokasi.
2. Awan publik: Infrastruktur awan disediakan untuk penggunaan terbuka oleh masyarakat umum. Itu mungkin dimiliki, dikelola, dan dioperasikan oleh organisasi bisnis, akademik, atau pemerintah, atau beberapa kombinasi dari mereka. Itu ada di tempat penyedia *cloud*.
3. Awan komunitas: Infrastruktur awan digunakan bersama oleh beberapa organisasi dan mendukung komunitas tertentu yang memiliki perhatian bersama (mis., misi, persyaratan keamanan, kebijakan, dan pertimbangan kepatuhan). Itu mungkin dikelola oleh organisasi atau pihak ketiga dan mungkin ada di lokasi atau di luar lokasi.
4. *Cloud* hibrid: Infrastruktur *cloud* adalah komposisi dari dua atau lebih infrastruktur *cloud* yang berbeda (swasta, komunitas, atau publik) yang tetap menjadi entitas unik tetapi terikat bersama oleh teknologi standar atau hak milik yang memungkinkan portabilitas data dan aplikasi (misalnya, *cloud* bursting untuk load balancing antar *cloud*).

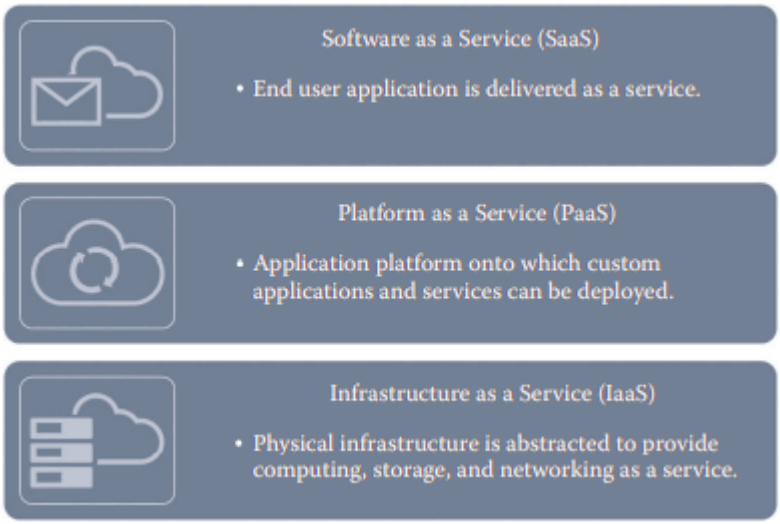
**Tiga Model Penawaran Layanan**

Tiga jenis layanan yang menyediakan sumber daya komputasi berbasis *cloud* untuk pelanggan akhir adalah sebagai berikut: Perangkat Lunak sebagai Layanan (SaaS), Platform sebagai Layanan (PaaS), dan Infrastruktur sebagai Layanan (IaaS). Ini juga dikenal sebagai model *cloud* layanan-platform-infrastruktur (SPI) dan ditunjukkan pada Gambar 2.3. SaaS adalah model distribusi perangkat lunak di mana aplikasi (perangkat lunak, yang merupakan salah satu sumber daya komputasi terpenting) dihosting oleh vendor atau penyedia layanan dan tersedia untuk pelanggan melalui jaringan, biasanya Internet. PaaS adalah paradigma untuk memberikan sistem operasi dan layanan terkait (misalnya, alat rekayasa perangkat lunak berbantuan komputer [CASE], lingkungan pengembangan terintegrasi [IDE] untuk mengembangkan solusi perangkat lunak) melalui Internet tanpa mengunduh atau menginstal.

IaaS melibatkan outsourcing peralatan yang digunakan untuk mendukung operasi, termasuk penyimpanan, perangkat keras, server, dan komponen jaringan.

1. SaaS Awan: Kemampuan yang diberikan kepada konsumen adalah menggunakan aplikasi penyedia yang berjalan di infrastruktur awan, termasuk jaringan, server, sistem operasi, penyimpanan, dan bahkan kemampuan aplikasi individual, dengan kemungkinan pengecualian pengguna terbatas -Pengaturan konfigurasi aplikasi khusus. Aplikasi dapat diakses dari berbagai perangkat klien baik melalui antarmuka klien tipis, seperti browser web (misalnya, email berbasis web), atau antarmuka program. Konsumen tidak mengelola atau mengontrol infrastruktur *cloud* yang mendasarinya. Aplikasi umum yang ditawarkan sebagai layanan meliputi manajemen hubungan pelanggan (CRM), analitik intelijen bisnis, dan perangkat lunak akuntansi online.
2. *Cloud* PaaS: Kemampuan yang diberikan kepada konsumen adalah untuk menyebarkan ke infrastruktur *cloud* aplikasi yang dibuat atau diperoleh konsumen yang dibuat menggunakan bahasa pemrograman, perpustakaan, layanan, dan alat yang didukung oleh penyedia. Konsumen tidak mengelola atau mengontrol infrastruktur *cloud* yang mendasarinya tetapi memiliki kontrol atas aplikasi yang diterapkan dan kemungkinan pengaturan konfigurasi untuk lingkungan hosting aplikasi. Dengan kata lain, ini adalah kerangka kerja pengembangan atau operasi yang dikemas dan siap dijalankan. Vendor PaaS menyediakan jaringan, server, dan penyimpanan serta mengelola tingkat skalabilitas dan pemeliharaan. Klien biasanya membayar untuk layanan yang digunakan. Contoh penyedia PaaS termasuk Google App Engine dan Microsoft Azure Services.
3. *Cloud* IaaS: Kemampuan yang diberikan kepada konsumen adalah untuk menyediakan pemrosesan, penyimpanan, jaringan, dan sumber daya komputasi fundamental lainnya berdasarkan pembayaran per penggunaan di mana dia dapat menyebarkan dan menjalankan perangkat lunak arbitrer, yang dapat mencakup pengoperasian sistem dan aplikasi. Konsumen tidak mengelola atau mengontrol infrastruktur *cloud* yang mendasarinya tetapi memiliki kontrol atas sistem operasi, penyimpanan, dan aplikasi yang diterapkan dan kemungkinan kontrol terbatas atas komponen jaringan tertentu (mis., firewall host). Penyedia layanan memiliki peralatan dan bertanggung jawab atas perumahan, operasi pendinginan, dan pemeliharaan. Amazon Web Services (AWS) adalah contoh populer dari penyedia IaaS besar.

Perbedaan utama antara PaaS dan IaaS adalah jumlah kendali yang dimiliki pengguna. Intinya, PaaS memungkinkan vendor untuk mengatur semuanya, sedangkan IaaS membutuhkan lebih banyak pengelolaan dari sisi pelanggan. Secara umum, organisasi yang sudah memiliki paket perangkat lunak atau aplikasi untuk tujuan tertentu dan ingin menginstal dan menjalankannya di *cloud* sebaiknya memilih untuk menggunakan IaaS daripada PaaS.



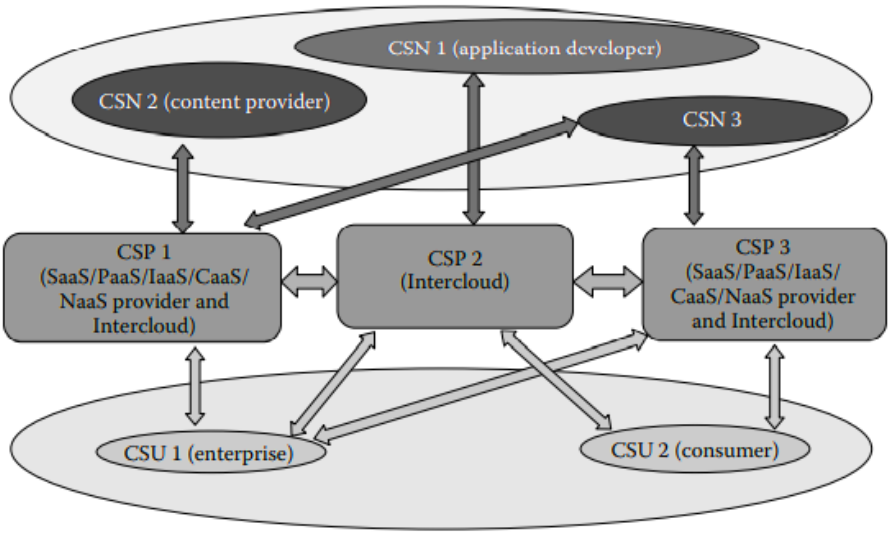

**Gambar 2.3** SPI—model penawaran layanan *cloud*.

## 2.4 EKOSISTEM AWAN

Ekosistem *cloud* adalah istilah yang digunakan untuk menggambarkan lingkungan atau sistem lengkap dari komponen atau entitas yang saling bergantung yang bekerja sama untuk mengaktifkan dan mendukung layanan *cloud*. Lebih tepatnya, ekosistem komputasi awan adalah lingkungan kompleks yang mencakup deskripsi setiap item atau entitas beserta interaksinya; entitas yang kompleks mencakup elemen tradisional komputasi awan seperti perangkat lunak (SaaS), perangkat keras (PaaS dan/atau IaaS), infrastruktur lain (misalnya, jaringan, penyimpanan), dan juga pemangku kepentingan seperti konsultan, integrator, mitra, pihak ketiga, dan apa pun di lingkungan mereka yang berkaitan dengan komponen *cloud* lainnya.

Ekosistem *cloud* dari komponen dan organisasi yang berinteraksi dengan individu, bersama-sama dikenal sebagai aktor yang dapat bertanggung jawab untuk menyediakan atau menggunakan layanan *cloud*, dapat dikategorikan sebagai berikut:

1. Pengguna layanan *cloud* (CSU): Konsumen (individu/orang), perusahaan (termasuk administrator perusahaan), dan/atau lembaga atau organisasi pemerintah/publik yang menggunakan layanan *cloud* yang disampaikan; CSU dapat menyertakan pengguna perantara yang akan mengirimkan layanan *cloud* yang disediakan oleh penyedia layanan *cloud* (CSP) kepada pengguna sebenarnya dari layanan *cloud*, yaitu pengguna akhir. Pengguna akhir dapat berupa orang, mesin, atau aplikasi.
2. CSP: Organisasi yang menyediakan atau memberikan dan memelihara atau mengelola layanan *cloud*, yaitu, penyedia SaaS, PaaS, IaaS, atau infrastruktur komputasi terkait lainnya.
3. Mitra layanan *cloud* (CSN): Seseorang atau organisasi (misalnya, pengembang aplikasi; penyedia konten, perangkat lunak, perangkat keras, dan/atau peralatan; integrator sistem; dan/atau auditor) yang menyediakan dukungan untuk pembangunan layanan yang ditawarkan oleh CSP (contohnya integrasi layanan).




**Gambar 2.4** Aktor dengan beberapa kemungkinan perannya dalam ekosistem *cloud*.

Gambar 2.4 mengilustrasikan ide ekosistem awan. Dalam istilah awam, ekosistem *cloud* menggambarkan penggunaan dan nilai setiap entitas dalam ekosistem, dan ketika semua entitas dalam ekosistem disatukan, pengguna kini dapat memiliki rangkaian terintegrasi yang terdiri dari solusi terbaik. Contoh dari ekosistem ini dapat berupa solusi *cloud accounting* seperti Tally; sementara vendor SaaS ini berfokus pada dukungan mereka untuk akuntansi dan solusi penggajian terintegrasi, mereka dapat terlibat (berkolaborasi) dengan CSP pihak ketiga lainnya yang dapat mendukung fitur tambahan dalam perangkat lunak akuntansi seperti alat pelaporan, dasbor, kertas kerja, alur kerja, manajemen proyek , dan CRM, mencakup sebagian besar kebutuhan perangkat lunak klien atau perusahaan pelanggan. Dan, persyaratan tambahan lainnya yang mungkin penting kemungkinan akan ditambahkan oleh mitra yang bergabung dengan ekosistem dalam waktu dekat.

## 2.5 PERSYARATAN UNTUK LAYANAN *CLOUD*

Dari konsep yang diilustrasikan di bagian sebelumnya, dapat dipahami bahwa layanan *cloud* atau model penawaran layanan memerlukan fitur tertentu untuk ditampilkan agar dianggap sebagai layanan. Berikut adalah persyaratan dasar untuk apa saja yang dapat dianggap sebagai layanan oleh para pelaku ekosistem *cloud computing*, yang dapat ditawarkan atau disediakan melalui *cloud:*

1. *Multitenancy: Multitenancy* adalah karakteristik penting dari sistem *cloud* yang bertujuan untuk menyediakan isolasi bagi pengguna yang berbeda dari sistem *cloud* (penyewa) sekaligus memaksimalkan pembagian sumber daya. Diharapkan *multitenancy* didukung di berbagai tingkat infrastruktur *cloud*. Sebagai contoh, pada level aplikasi, *multitenancy* adalah fitur yang memungkinkan satu instance aplikasi (katakanlah, sistem database) dan memanfaatkan skala ekonomis untuk memuaskan beberapa pengguna pada saat yang bersamaan.
2. Manajemen siklus hidup layanan: Layanan *cloud* dibayar sesuai penggunaan dan dapat dimulai dan diakhiri kapan saja. Oleh karena itu, diperlukan layanan *cloud* yang

mendukung penyediaan layanan otomatis. Selain itu, *metering* dan *charging* atau *billing settlement* perlu disediakan untuk layanan yang secara dinamis dibuat, dimodifikasi, dan kemudian dirilis di lingkungan virtual.

3. Keamanan: Keamanan setiap layanan individu perlu dilindungi di lingkungan *cloud* multitenant; pengguna (penyewa) juga mendukung layanan aman yang diperlukan, yang berarti bahwa *cloud* memberikan kontrol ketat untuk akses layanan penyewa ke sumber daya yang berbeda untuk menghindari penyalahgunaan sumber daya *cloud* dan untuk memfasilitasi pengelolaan CSU oleh CSP.
4. Daya tanggap: Ekosistem *cloud* diharapkan memungkinkan deteksi dini, diagnosis, dan perbaikan masalah terkait layanan untuk membantu pelanggan menggunakan layanan dengan setia.
5. Penerapan layanan cerdas: *Cloud* diharapkan memungkinkan penggunaan sumber daya yang efisien dalam penerapan layanan, yaitu, memaksimalkan jumlah layanan yang diterapkan sambil meminimalkan penggunaan sumber daya dan tetap menghormati SLA. Misalnya, karakteristik aplikasi spesifik (misalnya, unit pemrosesan pusat [CPU]-intensif, input/output [IO]-intensif) yang dapat disediakan oleh pengembang atau melalui pemantauan aplikasi dapat membantu CSP dalam memanfaatkan sumber daya secara efisien.
6. Portabilitas: Layanan *cloud* diharapkan mendukung portabilitas fitur-fiturnya melalui berbagai sumber daya yang mendasarinya dan bahwa CSP harus dapat mengakomodasi portabilitas beban kerja *cloud* (misalnya, portabilitas VM) dengan gangguan layanan terbatas.
7. Interoperabilitas: Diharapkan memiliki spesifikasi yang terdokumentasi dengan baik dan teruji dengan baik yang memungkinkan sistem heterogen di lingkungan *cloud* untuk bekerja sama.
8. Aspek peraturan: Semua peraturan yang berlaku harus dihormati, termasuk perlindungan privasi.
9. Kelestarian lingkungan: Karakteristik utama komputasi awan adalah kemampuan untuk mengakses, melalui jaringan luas dan klien kecil, kumpulan sumber daya yang dapat dikonfigurasi sesuai permintaan yang dapat disediakan dan dirilis dengan cepat. Komputasi awan kemudian dapat dianggap pada intinya sebagai model konsolidasi konsumsi energi TIK, mendukung teknologi arus utama yang bertujuan untuk mengoptimalkan konsumsi energi (misalnya, di pusat data) dan kinerja aplikasi. Contoh teknologi tersebut termasuk virtualisasi dan multitenancy.
10. Keandalan layanan, ketersediaan layanan, dan jaminan kualitas: CSU menuntut jaminan kualitas layanan (QoS) *end-to-end* layanan mereka, tingkat keandalan yang tinggi, dan ketersediaan berkelanjutan untuk CSP mereka.
11. Akses layanan: Infrastruktur *cloud* diharapkan memberi CSU akses ke layanan *cloud* dari perangkat pengguna apa pun. CSU diharapkan memiliki pengalaman yang konsisten saat mengakses layanan *cloud.*
12. Fleksibilitas: Diharapkan layanan *cloud* mampu mendukung beberapa model penyebaran *cloud* dan kategori layanan *cloud.*

13. Akuntansi dan pengisian daya: Layanan *cloud* diharapkan mampu mendukung berbagai model dan kebijakan akuntansi dan pengisian daya.
14. Pemrosesan data besar-besaran: *Cloud* diharapkan mendukung mekanisme untuk pemrosesan data besar-besaran (misal, Ekstraksi, transformasi, dan pemuatan data). Perlu dicatat dalam konteks ini bahwa sistem pemrosesan terdistribusi dan/atau paralel akan digunakan dalam penerapan infrastruktur *cloud* untuk menyediakan penyimpanan data terintegrasi berskala besar dan kemampuan pemrosesan yang diskalakan dengan toleransi kesalahan berbasis perangkat lunak.

Persyaratan yang diharapkan untuk layanan dalam kategori IaaS meliputi:

- Persyaratan perangkat keras komputasi (termasuk pemrosesan, memori, disk, antarmuka jaringan, dan mesin virtual)
- Persyaratan perangkat lunak komputasi (termasuk OS dan perangkat lunak prainstal lainnya)
- Persyaratan penyimpanan (termasuk kapasitas penyimpanan)
- Persyaratan jaringan (termasuk spesifikasi QoS, seperti bandwidth dan volume lalu lintas)
- Persyaratan ketersediaan (termasuk rencana perlindungan/pencadangan untuk komputasi, penyimpanan, dan sumber daya jaringan)

Persyaratan layanan yang diharapkan untuk layanan dalam kategori PaaS meliputi:

- Persyaratan serupa dengan kategori IaaS
- Opsi penerapan aplikasi buatan pengguna (misal, opsi penskalaan)

Persyaratan layanan yang diharapkan untuk layanan dalam kategori SaaS meliputi:

- Persyaratan khusus aplikasi (termasuk opsi lisensi)
- Persyaratan jaringan (termasuk spesifikasi QoS seperti lebar pita dan volume lalu lintas)

## 2.6 APLIKASI *CLOUD*

Aplikasi *cloud* adalah program aplikasi yang berfungsi atau dijalankan di *cloud*; aplikasi dapat menunjukkan beberapa karakteristik aplikasi desktop murni dan beberapa karakteristik aplikasi berbasis web murni. Aplikasi desktop berada sepenuhnya pada satu perangkat di lokasi pengguna (tidak harus berupa komputer desktop), dan di sisi lain, aplikasi web disimpan seluruhnya di server jarak jauh dan dikirimkan melalui Internet melalui antarmuka peramban.

Seperti aplikasi desktop, aplikasi *cloud* dapat memberikan respon yang cepat dan dapat bekerja secara offline. Seperti aplikasi web, aplikasi *cloud* tidak perlu secara permanen berada di perangkat lokal, tetapi dapat dengan mudah diperbarui secara online. Oleh karena itu, aplikasi *cloud* berada di bawah kendali konstan pengguna, namun tidak selalu membutuhkan ruang penyimpanan di komputer atau perangkat komunikasi pengguna. Dengan asumsi bahwa pengguna memiliki koneksi Internet yang cukup cepat, aplikasi *cloud* yang ditulis dengan baik menawarkan semua interaktivitas aplikasi desktop bersama dengan portabilitas aplikasi web.

Aplikasi *cloud* dapat digunakan dengan browser web yang terhubung ke Internet. Sekarang, bagian antarmuka pengguna dari aplikasi mungkin ada di perangkat lokal dan bagi

pengguna untuk menyimpan data secara lokal, memungkinkan mode luring penuh bila diinginkan. Selain itu, aplikasi *cloud*, tidak seperti aplikasi web, dapat digunakan dalam situasi sensitif apa pun di mana perangkat nirkabel—konektivitas—tidak diizinkan (yaitu, meskipun tidak ada koneksi Internet tersedia selama beberapa waktu).

Contoh aplikasi *cloud* adalah email berbasis web (misal, Gmail, Yahoo mail); dalam aplikasi ini, pengguna email menggunakan *cloud* — semua email di kotak masuk mereka disimpan di server di lokasi terpencil di penyedia layanan email. Namun, ada banyak layanan lain yang menggunakan *cloud* dengan cara yang berbeda. Ini contoh lainnya: Dropbox adalah layanan penyimpanan *cloud* yang memungkinkan kita menyimpan dan berbagi file dengan mudah dengan orang lain dan juga mengakses file dari perangkat seluler.

## 2.7 MANFAAT DAN KERUGIAN

Salah satu daya tarik *cloud computing* adalah aksesibilitas. Jika aplikasi dan dokumen kita ada di *cloud* dan tidak disimpan di server kantor, maka kita dapat mengakses dan menggunakannya kapan saja, di mana saja untuk pekerjaan kita, baik di kantor, di rumah, atau bahkan di rumah teman. Komputasi awan juga memungkinkan jumlah daya dan sumber daya komputasi yang tepat untuk digunakan untuk aplikasi. Vendor komputasi awan menyediakan layanan terkait komputasi sebagai kumpulan daya komputasi dan membaginya sesuai permintaan. Pelanggan dapat menggambar dan menggunakan daya komputasi sebanyak atau sesedikit yang mereka butuhkan, hanya dikenakan biaya untuk waktu penggunaan/daya komputasi; karenanya, skema ini dapat menghemat uang. Ini juga menyiratkan bahwa skalabilitas adalah salah satu manfaat besar komputasi awan. Saat kita membutuhkan lebih banyak daya komputasi, *cloud computing* dapat memberikan akses instan ke apa yang kita butuhkan.

Dalam model *cloud*, kekuatan komputer inti organisasi berada di luar lokasi dan pada dasarnya berlangganan daripada dimiliki. Tidak ada belanja modal, hanya belanja operasional. Ini juga membebaskan kami dari tanggung jawab dan biaya pemeliharaan seluruh infrastruktur komputasi dan mendorong semua ini ke vendor atau penyedia *cloud*. *Cloud* juga menawarkan tingkat keandalan yang baru. Teknologi virtualisasi memungkinkan perangkat lunak *cloud* vendor untuk secara otomatis memindahkan data dari perangkat keras yang rusak atau ditarik offline ke bagian sistem atau perangkat keras yang berfungsi atau beroperasi. Oleh karena itu, klien mendapatkan akses mulus ke data. Sistem pencadangan terpisah, dengan strategi pemulihan bencana *cloud*, menyediakan lapisan ketergantungan dan keandalan lainnya. Akhirnya, komputasi awan juga mempromosikan alternatif ramah lingkungan untuk fungsi kantor yang padat kertas. Itu karena membutuhkan lebih sedikit perangkat keras komputasi on-premise, dan semua tugas yang berhubungan dengan komputasi berlangsung dari jarak jauh dengan kebutuhan perangkat keras komputasi minimal dengan bantuan inovasi teknologi seperti virtualisasi dan *multitenancy.*

Sudut pandang lain pada aspek hijau adalah bahwa komputasi awan dapat mengurangi dampak lingkungan dari pembangunan, pengiriman, perumahan, dan pada akhirnya menghancurkan (atau mendaur ulang) peralatan komputer karena tidak seorang pun akan memiliki banyak sistem seperti itu di tempat mereka dan mengelola kantor. dengan lebih

sedikit komputer yang mengkonsumsi lebih sedikit energi secara komparatif. Serangkaian poin terkonsolidasi yang menjelaskan manfaat komputasi awan adalah sebagai berikut:

1. Mencapai skala ekonomi: Kita dapat meningkatkan volume keluaran atau produktivitas dengan sistem yang lebih sedikit dan dengan demikian mengurangi biaya per unit proyek atau produk.
2. Mengurangi pengeluaran untuk infrastruktur teknologi: Akses data dan informasi mudah dengan pengeluaran di muka yang minimal dengan pendekatan pay-as-you-go, dalam arti penggunaan dan pembayarannya mirip dengan pembacaan meteran listrik di rumah, yang didasarkan pada permintaan.
3. Mengglobalkan tenaga kerja: Orang-orang di seluruh dunia dapat mengakses *cloud* dengan koneksi internet.
4. Merampingkan proses bisnis: Anda dapat menyelesaikan lebih banyak pekerjaan dalam waktu yang lebih singkat dengan sumber daya yang lebih sedikit.
5. Mengurangi biaya modal: Tidak perlu menghabiskan banyak uang untuk perangkat keras, perangkat lunak, atau biaya lisensi.
6. Aksesibilitas luas: Data dan aplikasi dapat diakses kapan saja, di mana saja, menggunakan perangkat komputasi pintar apa pun, membuat hidup kita jauh lebih mudah.
7. Memantau proyek secara lebih efektif: Dimungkinkan untuk membatasi alokasi anggaran dan dapat mendahului waktu siklus penyelesaian.
8. Lebih sedikit pelatihan personel yang dibutuhkan: Dibutuhkan lebih sedikit orang untuk melakukan lebih banyak pekerjaan di *cloud*, dengan kurva pembelajaran minimal pada masalah perangkat keras dan perangkat lunak.
9. Minimalkan pemeliharaan dan lisensi perangkat lunak: Karena sumber daya komputasi lokal tidak terlalu banyak, pemeliharaan menjadi sederhana dan pembaruan serta pembaruan sistem perangkat lunak bergantung pada vendor atau penyedia *cloud*.
10. Fleksibilitas yang lebih baik: Perubahan cepat di lingkungan kerja kita dapat dilakukan tanpa masalah serius yang dipertaruhkan.

Kelemahan komputasi awan sudah jelas. Poin utama dalam konteks ini adalah jika kita kehilangan koneksi Internet, kita kehilangan tautan ke *cloud* dan dengan demikian ke data dan aplikasi. Ada juga kekhawatiran tentang keamanan karena seluruh pekerjaan kami dengan data dan aplikasi bergantung pada daya komputasi (vendor atau penyedia *cloud*) pihak lain. Juga, sementara komputasi awan mendukung skalabilitas (yaitu, dengan cepat meningkatkan dan menurunkan sumber daya komputasi tergantung pada kebutuhan), itu tidak mengizinkan kontrol pada sumber daya ini karena ini tidak dimiliki oleh pengguna atau pelanggan. Bergantung pada vendor atau penyedia *cloud*, pelanggan mungkin menghadapi batasan pada ketersediaan aplikasi, sistem operasi, dan opsi infrastruktur. Dan, terkadang, semua platform pengembangan mungkin tidak tersedia di *cloud* karena fakta bahwa vendor *cloud* mungkin tidak mengetahui solusi semacam itu. Hambatan utama komputasi awan adalah interoperabilitas aplikasi, yang merupakan kemampuan dua aplikasi atau lebih yang diperlukan untuk mendukung kebutuhan bisnis untuk bekerja sama dengan berbagi data dan sumber daya terkait bisnis lainnya. Biasanya, ini tidak terjadi di *cloud* karena aplikasi ini

mungkin tidak tersedia dengan satu vendor *cloud* dan dua vendor berbeda yang memiliki aplikasi ini tidak bekerja sama satu sama lain.

## 2.8 RINGKASAN

Untuk pemahaman yang jelas tentang komputasi awan, ada beberapa konsep dasar yang harus diketahui, seperti yang dibahas dalam bab ini. Bab ini dimulai dengan motivasi untuk komputasi awan dan membahas secara singkat alasan diperkenalkannya awan, kebutuhan akan komputasi awan, dan definisi dasar awan. NIST memberikan definisi standar untuk komputasi awan. *Cloud* didasarkan pada prinsip 5-4-3. *Cloud* memiliki lingkungan yang berbeda. Jadi, ekosistem *cloud* dibahas, yang secara singkat menunjukkan berbagai peran yang terlibat dalam komputasi *cloud*. Selanjutnya beberapa fitur penting dari *cloud computing* diuraikan. Aplikasi di *cloud* juga dibahas secara singkat. Bab ini diakhiri dengan catatan mendetail tentang keuntungan dan kerugian *cloud*.

**Tinjau Poin**

- Komputasi awan: Komputasi awan adalah model untuk memungkinkan akses jaringan di mana-mana, nyaman, sesuai permintaan ke kumpulan bersama dari sumber daya komputasi yang dapat dikonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat disediakan dengan cepat sioned dan dirilis dengan upaya manajemen minimal atau interaksi penyedia layanan. Model *cloud* ini terdiri dari lima karakteristik penting, tiga model layanan, dan empat model penerapan (lihat Bagian 2.2.1).
- Ekosistem *cloud*: Seseorang atau organisasi (mis., pengembang aplikasi; penyedia konten, perangkat lunak, perangkat keras, dan/atau peralatan; integrator sistem; dan/atau auditor) yang memberikan dukungan untuk pembangunan layanan yang ditawarkan oleh CSP (mis., integrasi layanan) (lihat Bagian 2.4).
- Penyedia layanan *cloud*: Suatu organisasi yang menyediakan atau memberikan dan memelihara atau mengelola layanan *cloud*, yaitu, penyedia SaaS, PaaS, IaaS, atau infrastruktur komputasi terkait lainnya (lihat Bagian 2.4).
- Multitenancy: Multitenancy merupakan karakteristik penting dari sistem *cloud* yang bertujuan untuk menyediakan isolasi dari pengguna yang berbeda dari sistem *cloud* (penyewa) sekaligus memaksimalkan pembagian sumber daya (lihat Bagian 2.5).

**Latihan Soal**

1. Apa itu komputasi awan? Mengapa itu dibutuhkan?
2. Jelaskan contoh kehidupan nyata untuk mengilustrasikan konsep di balik komputasi awan.
3. Bedakan antara definisi *cloud computing* dengan layanan dan komputasi awan adalah sebuah platform.
4. Apakah benar bahwa semua fitur karakteristik penting dari awan diperlukan untuk menggambarkannya secara lengkap?
5. Apa saja model penawaran layanan *cloud*?
6. Apa saja model penerapan *cloud*?

7. Apa saja aktor dan peran mereka dalam ekosistem *cloud* pada umumnya?
8. Sebutkan dan jelaskan persyaratan yang perlu diperhatikan untuk layanan *cloud*.
9. Jelaskan bagaimana aplikasi *cloud* diakses.
10. Berikan catatan singkat tentang kelebihan dan kekurangan komputasi awan.

# BAB 3
# ARSITEKTUR DAN MANAJEMEN *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan dari bab ini adalah untuk

- Memberikan gambaran tentang arsitektur *cloud*
- Berikan wawasan tentang anatomi awan
- Jelaskan peran konektivitas jaringan di *cloud*
- Berikan deskripsi tentang aplikasi di *cloud*
- Berikan penjelasan rinci tentang mengelola *cloud*
- Memberikan ikhtisar tentang migrasi aplikasi ke *cloud*

**Pengantar**

Komputasi awan adalah teknologi baru yang telah menjadi salah satu teknologi komputasi paling populer. Setiap teknologi memiliki konsep tertentu yang menjadi dasar kerjanya. Demikian pula, ada beberapa aspek teknologi yang perlu dicermati sebelum menggali lebih dalam. Dengan demikian, ada beberapa masalah mendasar dalam komputasi awan yang perlu dibahas sebelum masuk ke pembahasan mendetail tentang *cloud*. Bab ini pertama-tama menjelaskan arsitektur *cloud*. Arsitektur *cloud* terdiri dari kumpulan komponen hierarkis yang secara kolektif menjelaskan cara kerja *cloud*. Bagian selanjutnya menjelaskan tentang anatomi *cloud*, diikuti dengan konektivitas jaringan di *cloud*, lalu detail halus tentang pengelolaan aplikasi *cloud*. Terakhir, ikhtisar tentang migrasi aplikasi ke *cloud* dibahas. Beberapa topik yang dibahas dalam bab ini diuraikan dalam bab-bab selanjutnya.

## 3.1 PENDAHULUAN

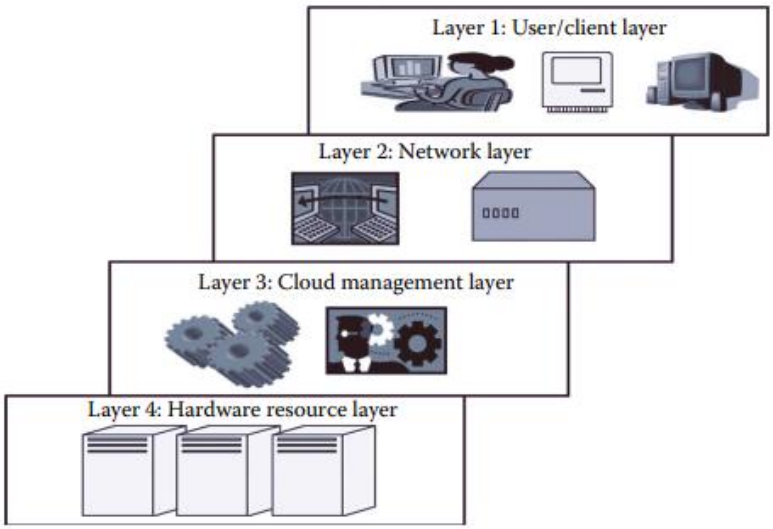
Komputasi awan mirip dengan teknologi lain dengan cara yang juga memiliki beberapa konsep dasar yang harus dipelajari seseorang sebelum mengetahui konsep intinya. Ada beberapa proses dan komponen komputasi awan yang perlu dibahas. Salah satu topik yang sangat penting adalah arsitektur. Arsitektur adalah pandangan hirarkis menggambarkan teknologi. Ini biasanya mencakup komponen di mana teknologi yang ada dibangun dan komponen yang bergantung pada teknologi tersebut. Topik lain yang berhubungan dengan arsitektur adalah anatomi. Anatomi menggambarkan struktur inti awan. Setelah struktur *cloud* jelas, koneksi jaringan di *cloud* dan detail tentang aplikasi *cloud* perlu diketahui. Ini penting karena *cloud* adalah teknologi yang sepenuhnya bergantung pada Internet. Demikian pula, manajemen *cloud* membahas masalah manajemen penting dan cara pengelolaan skenario *cloud* saat ini. Ini menjelaskan cara aplikasi dan infrastruktur di *cloud* dikelola. Manajemen penting karena faktor kualitas layanan (QoS) yang terlibat dalam *cloud*. Faktor QoS ini membentuk dasar untuk komputasi awan. Semua layanan diberikan berdasarkan faktor QoS ini. Demikian pula, migrasi aplikasi ke *cloud* juga memainkan peran yang sangat penting. Tidak semua aplikasi dapat langsung disebarkan ke *cloud*. Sebuah aplikasi perlu dimigrasikan dengan benar ke *cloud* untuk dianggap sebagai aplikasi *cloud* yang tepat yang akan memiliki semua properti *cloud*.

## 3.2 ARSITEKTUR AWAN

Setiap model teknologi terdiri dari arsitektur yang didasarkan pada fungsi model, yang merupakan pandangan hierarkis untuk menggambarkan teknologi. *Cloud* juga memiliki arsitektur yang menggambarkan mekanisme kerjanya. Ini termasuk dependensi tempat kerjanya dan komponen yang berfungsi di atasnya. *Cloud* adalah teknologi terkini yang sepenuhnya bergantung pada Internet untuk fungsinya. Gambar 3.1 menggambarkan arsitektur. Arsitektur *cloud* dapat dibagi menjadi empat lapisan berdasarkan akses *cloud* oleh pengguna. Mereka adalah sebagai berikut.

**Lapisan 1 (Lapisan Pengguna/Klien)**

Lapisan ini merupakan lapisan paling bawah dalam arsitektur *cloud*. Semua pengguna atau klien termasuk dalam lapisan ini. Ini adalah tempat klien/pengguna memulai koneksi ke *cloud*. Klien dapat berupa perangkat apa pun seperti klien tipis, klien tebal, atau seluler atau perangkat genggam apa pun yang akan mendukung fungsi dasar untuk mengakses aplikasi web. Thin client di sini mengacu pada perangkat yang sepenuhnya bergantung pada beberapa sistem lain untuk fungsionalitasnya yang lengkap. Secara sederhana, mereka memiliki kemampuan pemrosesan yang sangat rendah. Demikian pula, klien tebal adalah komputer umum yang memiliki kemampuan pemrosesan yang memadai. Mereka memiliki kemampuan yang memadai untuk pekerjaan mandiri. Biasanya, aplikasi *cloud* dapat diakses dengan cara yang sama seperti aplikasi web. Namun secara internal, properti aplikasi *cloud* sangat berbeda. Dengan demikian, lapisan ini terdiri dari perangkat klien.




**Gambar 3.1** Arsitektur Awan.

**Lapisan 2 (Lapisan Jaringan)**

Lapisan ini memungkinkan pengguna untuk terhubung ke *cloud*. Seluruh infrastruktur *cloud* bergantung pada koneksi ini di mana layanan ditawarkan kepada pelanggan. Ini terutama Internet dalam kasus *cloud* publik. *Cloud* publik biasanya ada di lokasi tertentu dan pengguna tidak akan mengetahui lokasi tersebut karena bersifat abstrak. Dan, *cloud* publik dapat diakses di seluruh dunia. Dalam kasus *cloud* pribadi, konektivitas dapat disediakan oleh jaringan area lokal (LAN). Bahkan dalam hal ini, *cloud* sepenuhnya bergantung pada jaringan

yang digunakan. Biasanya, saat mengakses *cloud* publik atau privat, pengguna memerlukan bandwidth minimum, yang terkadang ditentukan oleh penyedia *cloud*. Lapisan ini tidak berada di bawah lingkup perjanjian tingkat layanan (SLA), yaitu, SLA tidak memperhitungkan koneksi Internet antara pengguna dan *cloud* untuk kualitas layanan (QoS).

**Lapisan 3 (Lapisan Manajemen *Cloud*)**

Lapisan ini terdiri dari perangkat lunak yang digunakan dalam mengelola *cloud*. Perangkat lunak dapat berupa sistem operasi *cloud* (OS), perangkat lunak yang bertindak sebagai antarmuka antara pusat data (sumber daya aktual) dan pengguna, atau perangkat lunak manajemen yang memungkinkan pengelolaan sumber daya. Perangkat lunak ini biasanya memungkinkan manajemen sumber daya (penjadwalan, penyediaan, dll.), optimalisasi (konsolidasi server, konsolidasi beban kerja penyimpanan), dan tata kelola *cloud* internal. Lapisan ini berada di bawah lingkup SLA, yaitu, operasi yang terjadi di lapisan ini akan memengaruhi SLA yang diputuskan antara pengguna dan penyedia layanan. Keterlambatan apa pun dalam pemrosesan atau ketidaksesuaian apa pun dalam penyediaan layanan dapat menyebabkan pelanggaran SLA. Sesuai aturan, setiap pelanggaran SLA akan mengakibatkan penalti yang akan diberikan oleh penyedia layanan. SLA ini untuk *cloud* pribadi dan publik Penyedia layanan populer adalah Amazon Web Services (AWS) dan Microsoft Azure untuk *cloud* publik. Demikian pula, OpenStack dan Eucalyptus memungkinkan pembuatan, penerapan, dan manajemen *cloud* pribadi.

**Lapisan 4 (Lapisan Sumber Daya Perangkat Keras)**

Lapisan 4 terdiri dari ketentuan untuk sumber daya perangkat keras yang sebenarnya. Biasanya, dalam kasus *cloud* publik, pusat data digunakan di bagian belakang. Demikian pula, di *cloud* pribadi, ini bisa menjadi pusat data, yang merupakan kumpulan besar sumber daya perangkat keras yang saling berhubungan satu sama lain yang ada di lokasi tertentu atau sistem konfigurasi tinggi. Lapisan ini berada di bawah lingkup SLA. Ini adalah lapisan terpenting yang mengatur SLA. Lapisan ini paling memengaruhi SLA dalam hal pusat data. Setiap kali pengguna mengakses *cloud*, itu harus tersedia untuk pengguna secepat mungkin dan harus dalam waktu yang ditentukan oleh SLA. Seperti disebutkan, jika ada ketidaksesuaian dalam penyediaan sumber daya atau aplikasi, penyedia layanan harus membayar denda. Oleh karena itu, pusat data terdiri dari koneksi jaringan berkecepatan tinggi dan algoritme yang sangat efisien untuk mentransfer data dari pusat data ke pengelola. Mungkin ada sejumlah pusat data untuk *cloud*, dan demikian pula, sejumlah *cloud* dapat berbagi pusat data.

Jadi, inilah arsitektur *cloud*. Pelapisannya ketat, dan untuk aplikasi *cloud* apa pun, ini diikuti. Mungkin ada sedikit isolasi longgar antara lapisan 3 dan lapisan 4 tergantung pada cara *cloud* diterapkan.

## 3.3 ANATOMI AWAN

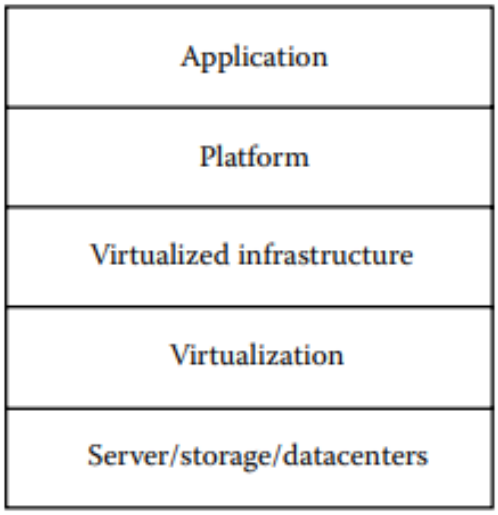
Anatomi awan secara sederhana dapat didefinisikan sebagai struktur awan. Anatomi *cloud* tidak dapat dianggap sama dengan arsitektur *cloud*. Ini mungkin tidak termasuk ketergantungan apa pun di mana atau di mana teknologi itu bekerja, sedangkan arsitektur sepenuhnya mendefinisikan dan menggambarkan teknologi tempat ia bekerja. Arsitektur adalah pandangan struktural hierarkis yang mendefinisikan teknologi serta teknologi yang

bergantung padanya atau / dan teknologi yang bergantung padanya. Dengan demikian, anatomi dapat dianggap sebagai bagian dari arsitektur. Struktur dasar awan dijelaskan pada Gambar 3.2, yang dapat diuraikan, dan detail struktural dapat diberikan. Gambar 3.2 menggambarkan anatomi paling standar yang menjadi dasar awan. Itu tergantung pada orang yang memilih kedalaman deskripsi *cloud*. Pandangan anatomi yang berbeda diberikan oleh Referensi.

Pada dasarnya ada lima komponen *cloud*:

1. Aplikasi: Lapisan atas adalah lapisan aplikasi. Di lapisan ini, semua aplikasi dijalankan.
2. Platform: Komponen ini terdiri dari platform yang bertanggung jawab atas eksekusi aplikasi. Platform ini berada di antara infrastruktur dan aplikasi.
3. Infrastruktur: Infrastruktur terdiri dari sumber daya di mana komponen lainnya bekerja. Ini memberikan kemampuan komputasi kepada pengguna.
4. Virtualisasi: Virtualisasi adalah proses pembuatan komponen logis dari sumber daya di atas sumber daya fisik yang ada. Komponen logis terisolasi dan independen, yang membentuk infrastruktur.
5. Perangkat keras fisik: Perangkat keras fisik disediakan oleh server dan unit penyimpanan.

Komponen-komponen ini adalah dasar dan dijelaskan secara rinci dalam bab-bab selanjutnya.




**Gambar 3.2** Struktur Awan.

## 3.4 KONEKTIVITAS JARINGAN DI *CLOUD COMPUTING*

Komputasi awan adalah teknik berbagi sumber daya di mana server, penyimpanan, dan infrastruktur komputasi lainnya di beberapa lokasi dihubungkan oleh jaringan. Di awan, saat aplikasi diajukan untuk pelaksanaannya, sumber daya yang diperlukan dan sesuai dialokasikan dari kumpulan sumber daya ini; karena sumber daya ini terhubung melalui Internet, pengguna mendapatkan hasil yang diperlukan. Untuk banyak aplikasi komputasi awan, kinerja jaringan akan menjadi isu utama kinerja komputasi awan. Karena komputasi awan memiliki berbagai pilihan penyebaran, kami sekarang mempertimbangkan aspek

penting yang terkait dengan model penyebaran awan dan aksesibilitasnya dari sudut pandang konektivitas jaringan.

**Jaringan Akses *Cloud* Publik**

Dalam opsi ini, konektivitas seringkali dilakukan melalui Internet, meskipun beberapa penyedia awan mungkin dapat mendukung jaringan pribadi virtual (VPN) untuk pelanggan. Mengakses layanan *cloud* publik akan selalu menimbulkan masalah terkait keamanan, yang pada akhirnya terkait dengan kinerja. Salah satu pendekatan yang mungkin untuk mendukung keamanan adalah untuk mempromosikan konektivitas melalui terowongan terenkripsi, sehingga informasi dapat dikirim melalui pipa aman di Internet. Prosedur ini akan menjadi overhead dalam konektivitas, dan menggunakannya pasti akan meningkatkan penundaan dan dapat mempengaruhi kinerja.

Jika kita ingin mengurangi penundaan tanpa mengorbankan keamanan, maka kita harus memilih metode perutean yang sesuai seperti yang mengurangi penundaan dengan meminimalkan lompatan transit dalam konektivitas end-to-end antara penyedia *cloud* dan konsumen *cloud*. Karena dukungan konektivitas end-to-end dilakukan melalui Internet, yang merupakan federasi kompleks dari penyedia yang saling terhubung (dikenal sebagai penyedia layanan Internet [ISP]), kita harus melihat opsi untuk memilih jalur.

**Jaringan Akses *Cloud* Pribadi**

Dalam model penyebaran *cloud* pribadi, karena *cloud* adalah bagian dari jaringan organisasi, teknologi dan pendekatannya bersifat lokal untuk struktur jaringan in-house. Ini mungkin termasuk VPN Internet atau layanan VPN dari operator jaringan. Jika akses aplikasi dilakukan dengan benar dengan jaringan organisasi—konektivitas dalam konfigurasi pre*cloud*—transisi ke komputasi *cloud* pribadi tidak akan mempengaruhi kinerja akses.

**Jaringan Intracloud untuk Layanan *Cloud* Publik**

Pertimbangan konektivitas jaringan lainnya dalam komputasi awan adalah jaringan intra-awan untuk layanan *cloud* publik. Di sini, sumber daya penyedia *cloud* dan dengan demikian layanan *cloud* kepada pelanggan didasarkan pada sumber daya yang secara geografis terpisah satu sama lain tetapi tetap terhubung melalui Internet. Jaringan komputasi awan publik bersifat internal ke penyedia layanan dan karenanya tidak terlihat oleh pengguna/pelanggan; namun, aspek keamanan konektivitas dan mekanisme akses sumber daya adalah penting. Masalah lain yang harus dicari adalah QoS di sumber daya yang terhubung di seluruh dunia. Sebagian besar masalah kinerja dan pelanggaran dari hal ini dibahas dalam SLA secara komersial.

**Jaringan Intracloud Pribadi**

Masalah yang paling rumit untuk jaringan dan konektivitas dalam komputasi awan adalah jaringan intracloud pribadi. Apa yang membuat masalah khusus ini begitu kompleks adalah tergantung pada seberapa banyak konektivitas intracloud yang terkait dengan aplikasi yang dijalankan di lingkungan ini. Jaringan intra-*cloud* pribadi biasanya didukung melalui konektivitas antara situs pusat data utama yang dimiliki oleh perusahaan. Minimal, semua implementasi *cloud computing* akan bergantung pada jaringan intracloud untuk menghubungkan pengguna dengan sumber daya tempat aplikasi mereka ditugaskan. Setelah hubungan sumber daya dibuat, sejauh mana jaringan intracloud digunakan tergantung pada

apakah aplikasi tersebut dikomponen berdasarkan arsitektur berorientasi layanan (SOA) atau tidak, di antara beberapa sistem. Jika prinsip SOA diikuti, maka lalu lintas dapat berpindah antar komponen aplikasi, serta antara aplikasi dan pengguna. Kinerja koneksi tersebut kemudian akan berdampak pada kinerja komputasi awan secara keseluruhan. Di sini juga, dampak kinerja *cloud computing* adalah perbedaan yang ada antara aplikasi saat ini dan hubungan jaringan dengan aplikasi tersebut.

Ada alasan untuk mempertimbangkan jaringan dan konektivitas dalam komputasi awan dengan pendekatan yang lebih baru karena globalisasi dan persyaratan jaringan yang berubah, terutama yang terkait dengan peningkatan penggunaan Internet, menuntut lebih banyak fleksibilitas dalam arsitektur jaringan perusahaan saat ini. Bagaimana ini terkait dengan kita? Jawabannya dibahas nanti.

**Aspek Baru di Jaringan Pribadi**

Jaringan pribadi konvensional telah dirancang untuk aplikasi lokal dan keamanan Internet maksimum. Biasanya, aplikasi seperti email, berbagi file, dan sistem perencanaan sumber daya perusahaan (ERP) dikirimkan ke server berbasis lokasi di setiap pusat data perusahaan. Semakin hari ini, vendor perangkat lunak menawarkan Perangkat Lunak sebagai Layanan (SaaS) sebagai alternatif untuk dukungan perangkat lunak mereka ke kantor perusahaan, yang membawa lebih banyak tantangan dalam mekanisme akses dan penggunaan perangkat lunak dari server pusat data dan dalam konektivitas jaringan. ilmu bangunan. Arsitektur jaringan tradisional untuk perusahaan global ini tidak dirancang untuk mengoptimalkan kinerja aplikasi *cloud*, sekarang banyak aplikasi termasuk aplikasi mission-critical sedang bertransisi (bergerak) dari berbasis on-premise ke berbasis *cloud*, di mana ketersediaan jaringan menjadi sebagai misi. kritis seperti listrik: bisnis tidak dapat berfungsi jika tidak dapat mengakses aplikasi seperti ERP dan email.

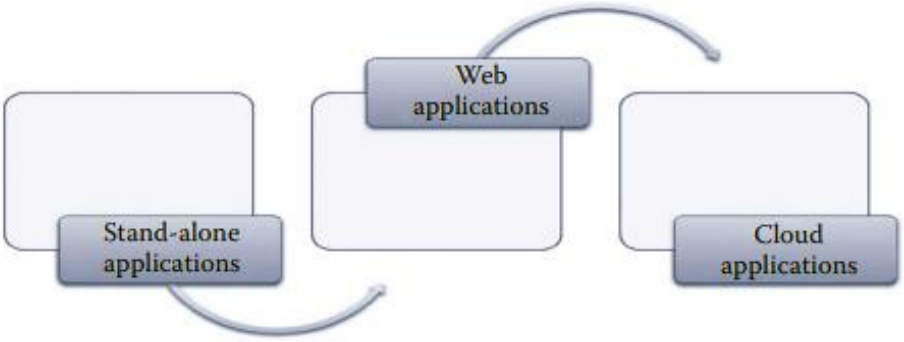
**Jalur Lalu Lintas Internet**

Lalu lintas Internet tradisional melalui serangkaian gateway Internet terbatas menimbulkan masalah kinerja dan ketersediaan bagi pengguna akhir yang menggunakan aplikasi berbasis *cloud*. Ini dapat ditingkatkan jika infrastruktur dan konektivitas gateway Internet yang didistribusikan lebih luas didukung untuk mengakses aplikasi, karena mereka akan memberikan akses latensi rendah ke aplikasi *cloud* mereka. Seiring meningkatnya volume lalu lintas ke aplikasi *cloud*, persentase kapasitas jaringan lama dalam hal lalu lintas ke gateway regional meningkat. Aplikasi seperti konferensi video akan memakan lebih banyak bandwidth sementara aplikasi mission-critical seperti ERP akan mengkonsumsi lebih sedikit bandwidth, dan karenanya, seseorang harus merencanakan konektivitas dan jalur yang benar antara penyedia dan konsumen.

## 3.5 APLIKASI DI *CLOUD*

Kekuatan komputer diwujudkan melalui aplikasi. Ada beberapa jenis aplikasi. Jenis aplikasi pertama yang dikembangkan dan digunakan adalah aplikasi yang berdiri sendiri. Sebuah aplikasi yang berdiri sendiri dikembangkan untuk dijalankan pada satu sistem yang tidak menggunakan jaringan untuk fungsinya. Sistem yang berdiri sendiri ini hanya menggunakan mesin tempat mereka dipasang. Berfungsinya sistem semacam ini sepenuhnya

bergantung pada sumber daya atau fitur yang tersedia di dalam sistem. Sistem ini tidak memerlukan data atau kekuatan pemrosesan dari sistem lain; mereka mandiri. Namun seiring berjalannya waktu, persyaratan pengguna berubah dan diperlukan aplikasi tertentu, yang dapat diakses oleh pengguna lain di luar sistem. Hal ini menyebabkan dimulainya aplikasi web.




**Gambar 3.3** Evolusi Aplikasi Komputer.

Aplikasi web berbeda dari aplikasi yang berdiri sendiri dalam banyak aspek. Perbedaan utama adalah arsitektur client server yang diikuti oleh aplikasi web. Tidak seperti aplikasi yang berdiri sendiri, sistem ini sepenuhnya tergantung pada jaringan untuk bekerja. Di sini, pada dasarnya ada dua komponen, yang disebut klien dan server. Server adalah mesin *high-end* yang terdiri dari aplikasi web yang diinstal. Aplikasi web ini diakses dari sistem klien lain. Klien dapat berada di mana saja di jaringan. Itu dapat mengakses aplikasi web melalui Internet. Jenis aplikasi ini sangat berguna, dan ini digunakan secara luas sejak awal dan sekarang telah menjadi bagian penting dari kehidupan sehari-hari. Meskipun aplikasi ini banyak digunakan, ada kekurangan seperti yang dibahas berikut ini:

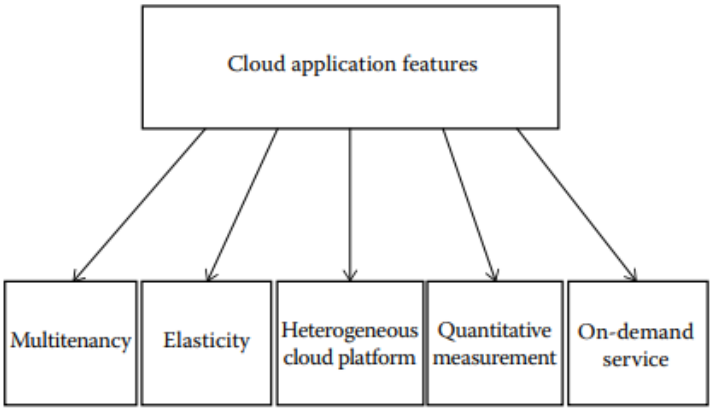
- Aplikasi web tidak elastis dan tidak dapat menangani beban yang sangat berat, yaitu tidak dapat melayani beban yang sangat bervariasi.
- Aplikasi web bukan multitenant.
- Aplikasi web tidak memberikan pengukuran kuantitatif atas layanan yang diberikan kepada pengguna, meskipun mereka dapat memantau pengguna.
- Aplikasi web biasanya dalam satu platform tertentu.
- Aplikasi web tidak disediakan berdasarkan pembayaran sesuai pemakaian; dengan demikian, layanan tertentu diberikan kepada pengguna untuk penggunaan permanen atau uji coba dan biasanya pengaturan waktu akses pengguna tidak dapat dipantau.
- Karena sifatnya yang tidak elastis, transaksi beban puncak tidak dapat ditangani.

Terutama untuk mengatasi masalah yang disebutkan sebelumnya, aplikasi *cloud* dikembangkan. Gambar 3.3 menggambarkan perbaikan dalam aplikasi. *Cloud* sebagaimana disebutkan dapat diklasifikasikan ke dalam tiga model akses atau layanan yang luas, Perangkat Lunak sebagai Layanan (SaaS), Platform sebagai Layanan (PaaS), dan Infrastruktur sebagai Layanan (IaaS). Aplikasi *cloud* secara umum mengacu pada aplikasi SaaS.

Aplikasi *cloud* berbeda dari aplikasi lain; mereka memiliki fitur unik. Aplikasi *cloud* biasanya dapat diakses sebagai aplikasi web tetapi propertinya berbeda. Menurut NIST [3],

fitur yang membuat aplikasi *cloud* unik dijelaskan sebagai berikut (Gambar 3.4 menggambarkan fitur aplikasi *cloud*):

1. Multitenancy: Multitenancy adalah salah satu properti penting *cloud* yang membuatnya berbeda dari jenis aplikasi lain di mana perangkat lunak dapat digunakan bersama oleh pengguna yang berbeda dengan kemandirian penuh. Di sini, independensi mengacu pada independensi logis.

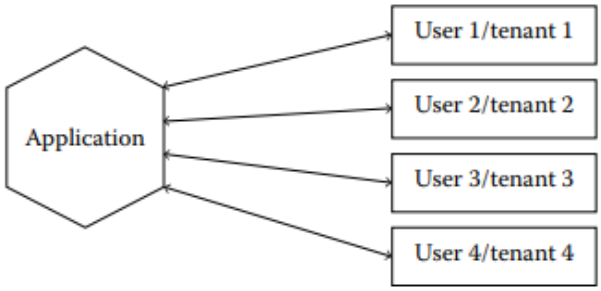



**Gambar 3.4** Fitur Awan.

Setiap pengguna akan memiliki instance aplikasi terpisah dan perubahan dalam satu aplikasi tidak akan memengaruhi aplikasi lainnya. Secara fisik, perangkat lunak tersebut digunakan bersama dan tidak berdiri sendiri. Tingkat isolasi fisik sangat kurang. Independensi logis adalah apa yang dijamin. Tidak ada batasan jumlah aplikasi yang dibagikan. Kesulitan dalam menyediakan isolasi logis tergantung pada isolasi fisik sampai batas tertentu. Jika aplikasi secara fisik terlalu dekat, maka sulit untuk menyediakan multitenancy. Aplikasi web dan aplikasi *cloud* serupa karena pengguna menggunakan cara yang sama untuk mengakses keduanya. Gambar 3.5 menggambarkan sebuah aplikasi multitenant dimana beberapa pengguna berbagi aplikasi yang sama.

2. Elastisitas: Elastisitas juga merupakan properti unik yang memungkinkan *cloud* berfungsi lebih baik. Elastisitas dapat didefinisikan sebagai sejauh mana suatu sistem dapat beradaptasi dengan perubahan beban kerja dengan penyediaan dan deprovisioning sumber daya secara otonom sehingga pada setiap titik waktu, sumber daya yang tersedia sesuai dengan permintaan saat ini. mungkin. Elastisitas memungkinkan penyedia *cloud* menangani jumlah pengguna secara efisien, dari satu hingga beberapa ratus pengguna sekaligus. Selain itu, ini mendukung fluktuasi beban yang cepat, yaitu peningkatan atau penurunan jumlah pengguna dan penggunaannya dapat berubah dengan cepat.
3. Platform *cloud* heterogen: Platform *cloud* mendukung heterogenitas, di mana semua jenis aplikasi dapat digunakan di *cloud*. Karena properti ini, *cloud* menjadi fleksibel bagi para pengembang, yang memfasilitasi penyebaran. Aplikasi yang biasanya dikerahkan dapat diakses oleh pengguna menggunakan web browser.

4. Pengukuran kuantitatif: Layanan yang diberikan dapat diukur secara kuantitatif. Pengguna biasanya ditawari layanan berdasarkan biaya tertentu. Di sini, aplikasi atau sumber daya diberikan sebagai utilitas berdasarkan pembayaran per penggunaan. Dengan demikian, penggunaannya dapat dipantau dan diukur. Tidak hanya layanan yang dapat diukur, tetapi juga penggunaan tautan dan beberapa parameter lain yang mendukung aplikasi *cloud* dapat diukur. Properti pengukuran penggunaan ini biasanya tidak tersedia di aplikasi web dan merupakan fitur unik untuk aplikasi berbasis *cloud*.
5. Layanan sesuai permintaan: Aplikasi *cloud* menawarkan layanan kepada pengguna, sesuai permintaan, yaitu kapan pun pengguna membutuhkannya. Layanan *cloud* akan memungkinkan pengguna untuk mengakses aplikasi web biasanya tanpa batasan waktu, durasi, dan jenis perangkat yang digunakan.




**Gambar 3.5** Multitenansi.

Properti yang disebutkan sebelumnya adalah beberapa fitur yang menjadikan *cloud* sebagai platform aplikasi yang unik. Properti yang disebutkan ini khusus untuk *cloud* sehingga menjadikannya sebagai salah satu dari sedikit teknologi yang memungkinkan pengembang aplikasi untuk memenuhi kebutuhan pengguna secara mulus tanpa gangguan apa pun.

## 3.6 MENGELOLA *CLOUD*

Mangelola *cloud* ditujukan untuk mengelola *cloud* secara efisien untuk menjaga QoS. Ini adalah salah satu pekerjaan utama yang harus dipertimbangkan. Seluruh *cloud* bergantung pada cara pengelolaannya. Manajemen *cloud* dapat dibagi menjadi dua bagian:

1. Mengelola infrastruktur *cloud*
2. Mengelola aplikasi *cloud*

**Mengelola Infrastruktur *Cloud***

Infrastruktur *cloud* dianggap sebagai tulang punggung *cloud*. Komponen ini terutama bertanggung jawab atas faktor QoS. Jika infrastruktur tidak dikelola dengan baik, maka seluruh *cloud* bisa gagal dan QoS akan terpengaruh secara negatif. Inti dari manajemen *cloud* adalah manajemen sumber daya. Manajemen sumber daya melibatkan beberapa tugas internal seperti penjadwalan sumber daya, penyediaan, dan penyeimbangan muatan. Tugas-tugas ini terutama dikelola oleh kemampuan perangkat lunak inti penyedia layanan *cloud* seperti OS *cloud* yang bertanggung jawab untuk menyediakan layanan ke *cloud* dan yang mengontrol *cloud* secara internal. Infrastruktur *cloud* adalah sistem yang sangat kompleks yang terdiri dari banyak sumber daya. Sumber daya ini biasanya digunakan bersama oleh beberapa pengguna.

Manajemen sumber daya yang buruk dapat menyebabkan beberapa inefisiensi dalam hal kinerja, fungsionalitas, dan biaya. Jika sumber daya tidak dikelola secara efisien, kinerja seluruh sistem akan terpengaruh. Performa adalah aspek terpenting dari *cloud*, karena semua yang ada di *cloud* bergantung pada SLA dan SLA hanya dapat dipenuhi jika performanya bagus. Demikian pula, fungsionalitas dasar *cloud* harus selalu disediakan dan dipertimbangkan dengan biaya berapa pun. Bahkan jika ada perbedaan kecil dalam menyediakan fungsionalitas, seluruh tujuan pemeliharaan *cloud* menjadi sia-sia. Awan yang berfungsi sebagian tidak akan memenuhi SLA.

Terakhir, alasan pengembangan *cloud* adalah biaya. Biaya adalah kriteria yang sangat penting sejauh menyangkut prospek bisnis *cloud*. Di pihak penyedia layanan, jika mereka mengeluarkan lebih sedikit biaya untuk mengelola *cloud*, maka mereka akan berusaha mengurangi biaya untuk mendapatkan basis pengguna yang kuat. Oleh karena itu, banyak pengguna akan menggunakan layanan tersebut, meningkatkan margin keuntungan mereka. Demikian pula, jika biaya pengelolaan sumber daya tinggi, maka pasti biaya untuk mengakses sumber daya akan tinggi dan tidak pernah ada bisnis yang merugi dari organisasi mana pun sehingga penyedia layanan tidak akan menanggung biayanya dan karenanya pengguna harus membayar lebih. Demikian pula, ini akan terbukti mahal bagi penyedia layanan karena mereka memiliki peluang tinggi kehilangan basis pengguna yang luas, yang hanya menyebabkan pertumbuhan marjinal dalam industri. Dan, bersaing dengan saingan industrinya akan menjadi masalah besar. Oleh karena itu, manajemen yang efisien dengan biaya yang lebih rendah diperlukan.

Pada tingkat yang lebih tinggi, selain ketiga isu tersebut, masih ada beberapa isu lagi yang bergantung pada pengelolaan sumber daya. Ini adalah konsumsi daya dan optimalisasi berbagai tujuan untuk mengurangi biaya lebih lanjut. Untuk menyelesaikan tugas ini, ada beberapa pendekatan yang diikuti, yaitu konsolidasi beban kerja server dan penyimpanan. Konsolidasi akan mengurangi konsumsi energi dan dalam beberapa kasus akan meningkatkan kinerja *cloud*. Konsolidasi server menurut definisi adalah pendekatan penggunaan sumber daya server komputer yang efisien untuk mengurangi jumlah total server atau lokasi server yang dibutuhkan organisasi.

Prospek yang dibahas sebelumnya sebagian besar cocok untuk IaaS. Demikian pula, ada berbagai metode manajemen yang diikuti untuk berbagai jenis model penyampaian layanan. Masing-masing jenis memiliki cara pengelolaannya sendiri. Semua metodologi manajemen didasarkan pada fluktuasi beban. Fluktuasi beban adalah titik di mana beban kerja sistem berubah secara terus menerus. Ini adalah salah satu kriteria dan masalah penting yang harus dipertimbangkan untuk aplikasi *cloud*. Fluktuasi beban dapat dibagi menjadi dua jenis: dapat diprediksi dan tidak dapat diprediksi. Fluktuasi beban yang dapat diprediksi mudah ditangani. *Cloud* dapat dikonfigurasikan sebelumnya untuk menangani fluktuasi semacam itu. Sedangkan fluktuasi beban yang tidak dapat diprediksi sulit untuk ditangani, ironisnya ini adalah salah satu alasan mengapa *cloud* lebih disukai oleh beberapa pengguna.

Ini sejauh menyangkut manajemen *cloud*. Tata kelola *cloud* adalah topik lain yang terkait erat dengan manajemen *cloud*. Tata kelola *cloud* berbeda dengan manajemen *cloud*. Tata kelola secara umum adalah istilah dalam dunia korporasi yang umumnya melibatkan

proses penciptaan nilai bagi suatu organisasi dengan menciptakan tujuan strategis yang akan mengarah pada pertumbuhan perusahaan dan akan mempertahankan tingkat kontrol tertentu atas perusahaan. Mirip dengan itu, di sini organisasi *cloud* terlibat.

Ada beberapa aspek tata kelola *cloud* di mana SLA adalah salah satu aspek penting. SLA adalah seperangkat aturan yang ditentukan antara pengguna dan penyedia layanan *cloud* yang memutuskan faktor QoS. Jika SLA tidak diikuti, maka mangkir harus membayar denda. Seluruh *cloud* diatur dengan mengingat SLA ini. Tata kelola *cloud* dibahas secara rinci di bab selanjutnya.

**Mengelola Aplikasi *Cloud***

Perusahaan bisnis semakin mencari untuk memindahkan atau membangun aplikasi korporat mereka pada platform *cloud* untuk meningkatkan kelincahan atau untuk memenuhi persyaratan dinamis yang ada dalam globalisasi bisnis dan daya tanggap terhadap permintaan pasar. Namun, pergeseran atau pemindahan aplikasi ke lingkungan *cloud* ini membawa kompleksitas baru. Aplikasi menjadi lebih komposit dan kompleks, yang tidak hanya membutuhkan peningkatan kemampuan seperti penyimpanan dan basis data yang ditawarkan oleh penyedia *cloud*, tetapi juga kemampuan SaaS pihak ketiga seperti email dan perpesanan. Jadi, memahami ketersediaan aplikasi memerlukan pemeriksaan infrastruktur, layanan yang digunakannya, dan pemeliharaan aplikasi. Sifat gabungan dari aplikasi *cloud* memerlukan visibilitas ke semua layanan untuk menentukan ketersediaan dan waktu aktif secara keseluruhan.

Manajemen aplikasi *cloud* adalah untuk mengatasi masalah ini dan mengusulkan solusi untuk memungkinkan untuk memiliki wawasan tentang aplikasi yang berjalan di *cloud*, serta menerapkan atau menegakkan kebijakan perusahaan seperti tata kelola dan audit dan manajemen lingkungan saat aplikasi disebarkan di *cloud*. Layanan pemantauan dan pengelolaan berbasis *cloud* ini dapat mengumpulkan banyak peristiwa, menganalisisnya, dan mengidentifikasi informasi penting yang memerlukan tindakan perbaikan tambahan seperti menyesuaikan kapasitas atau menyediakan layanan baru. Selain itu, manajemen aplikasi harus didukung dengan alat dan proses yang diperlukan untuk mengelola lingkungan lain yang mungkin hidup berdampingan, memungkinkan operasi yang efisien.

## 3.7 MEMIGRASIKAN APLIKASI KE *CLOUD*

Migrasi *cloud* mencakup pemindahan satu atau lebih aplikasi perusahaan dan lingkungan TI mereka dari jenis hosting tradisional ke lingkungan *cloud*, baik publik, pribadi, atau hibrid. Migrasi *cloud* menghadirkan peluang untuk secara signifikan mengurangi biaya yang dikeluarkan pada aplikasi. Kegiatan ini terdiri dari berbagai fase seperti evaluasi, strategi migrasi, pembuatan prototipe, penyediaan, dan pengujian.

**Tahapan Migrasi *Cloud***

1. Evaluasi: Evaluasi dilakukan untuk semua komponen seperti infrastruktur saat ini dan arsitektur aplikasi, lingkungan dalam hal komputasi, penyimpanan, pemantauan, dan manajemen, SLA, proses operasional, pertimbangan keuangan, risiko, keamanan, kepatuhan, dan perizinan kebutuhan diidentifikasi untuk membangun kasus bisnis untuk pindah ke *cloud*.

2. Strategi migrasi: Berdasarkan evaluasi, strategi migrasi ditarik—strategi hotplug digunakan di mana aplikasi dan datanya serta dependensi antarmuka diisolasi dan aplikasi ini dapat dioperasionalkan sekaligus. Sebuah strategi fusi digunakan dimana aplikasi dapat dimigrasikan sebagian; tetapi untuk sebagiannya, terdapat ketergantungan berdasarkan lisensi yang ada, persyaratan server khusus seperti mainframe, atau interkoneksi ekstensif dengan aplikasi lain.
3. Pembuatan Prototipe: Aktivitas migrasi didahului oleh aktivitas pembuatan prototipe untuk memvalidasi dan memastikan bahwa sebagian kecil aplikasi diuji di lingkungan *cloud* dengan penyiapan data pengujian.
4. Penyediaan: Pengoptimalan pramigrasi yang teridentifikasi diterapkan. Server *cloud* disediakan untuk semua lingkungan yang teridentifikasi, perangkat lunak platform yang diperlukan dan aplikasi disebarkan, konfigurasi disetel agar sesuai dengan ukuran lingkungan baru, dan basis data serta file direplikasi. Semua titik integrasi internal dan eksternal dikonfigurasi dengan benar. Layanan web, pekerjaan batch, dan perangkat lunak operasi dan manajemen diatur di lingkungan baru.
5. Pengujian: Pengujian pascamigrasi dilakukan untuk memastikan bahwa migrasi berhasil. Pengujian kinerja dan beban, pengujian kegagalan dan pemulihan, dan pengujian skala dilakukan terhadap beban lalu lintas yang diharapkan dan tingkat pemanfaatan sumber daya.

**Pendekatan untuk Migrasi *Cloud***

Berikut adalah empat pendekatan luas untuk migrasi *cloud* yang telah diadopsi secara efektif oleh vendor:

1. Migrasi aplikasi yang ada: Bangun kembali atau rancang bangun beberapa atau semua aplikasi, manfaatkan beberapa teknologi virtualisasi yang ada untuk mempercepat pekerjaan. Tapi, itu membutuhkan insinyur top untuk mengembangkan fungsionalitas baru. Ini dapat dicapai selama beberapa rilis dengan waktu yang ditentukan oleh permintaan pelanggan.
2. Mulai dari awal: Daripada mengkanibalisasi penjualan, membingungkan pelanggan dengan pilihan, dan mengikat teknisi yang mencoba membangun kembali aplikasi yang ada, mungkin lebih mudah untuk memulai lagi. Banyak keputusan R&D akan berbeda sekarang, dan dengan beberapa lingkungan pengembangan yang lebih canggih, seseorang dapat mencapai lebih banyak bahkan dengan tim kerja kecil yang terfokus.
3. Perusahaan terpisah: Seseorang mungkin ingin membuat perusahaan baru dengan merek, manajemen, R&D, dan penjualan yang terpisah. Investasi dan protokol internet (IP) mungkin berasal dari perusahaan yang sudah ada, tetapi banyak konflik hilang begitu perusahaan *cloud* yang baru lahir didirikan. Perusahaan yang terpisah bahkan dapat menjadi anak perusahaan dari perusahaan yang ada. Yang penting adalah perusahaan baru dapat bertindak, beroperasi, dan berperilaku seperti start-up berbasis *cloud*.
4. Beli vendor *cloud* yang sudah ada: Untuk vendor besar yang sudah mapan, membeli pesaing berbasis *cloud* mencapai dua hal. Pertama, ini menghilangkan pesaing, dan kedua, memungkinkan vendor untuk mulai beroperasi di ruang *cloud*. Risikonya tentu

saja inovasi, dorongan, dan pendekatan operasional perusahaan berbasis *cloud* hancur saat digabungkan ke dalam pengakuisisi yang lebih besar.

## 3.8 RINGKASAN

Komputasi awan memiliki beberapa konsep yang harus dipahami sebelum memulai dengan detail tentang awan, yang termasuk salah satu konsep penting arsitektur awan. Ini terdiri dari struktur hierarki dasar dengan dependensi komponen yang ditentukan. Demikian pula, anatomi juga penting karena menggambarkan struktur dasar tentang *cloud*, meskipun tidak mempertimbangkan ketergantungan seperti dalam arsitektur. Selanjutnya, konektivitas jaringan awan yang membentuk inti dari model awan adalah penting. Jaringan adalah basis yang digunakan *cloud* untuk bekerja. Demikian pula, manajemen *cloud* adalah salah satu konsep penting yang menggambarkan cara *cloud* dikelola, dan memiliki dua komponen: manajemen infrastruktur dan manajemen aplikasi. Keduanya penting karena keduanya mempengaruhi QoS. Terakhir, aplikasi harus berhasil dimigrasikan ke *cloud*. Aplikasi akan memancarkan properti lengkapnya sebagai *cloud* hanya jika telah dimigrasikan dengan sempurna.

**Tinjau Poin**

- Arsitektur awan: Arsitektur awan terdiri dari serangkaian komponen hierarkis yang secara kolektif menjelaskan cara kerja awan. Ini adalah pandangan dari sistem.
- Anatomi awan: Anatomi awan adalah struktur dasar awan
- SLA: SLA adalah sekumpulan perjanjian yang ditandatangani antara pengguna dan penyedia layanan.
- Elastisitas: Elastisitas dapat didefinisikan sebagai sejauh mana sistem mampu beradaptasi dengan perubahan beban kerja dengan penyediaan dan deprovisioning sumber daya secara otonom, sehingga pada setiap titik waktu, sumber daya yang tersedia cocok dengan permintaan saat ini sebagai sedekat mungkin.
- Multitenancy: Multitenancy adalah properti *cloud* yang dengannya perangkat lunak dapat digunakan bersama oleh pengguna yang berbeda dengan kemandirian penuh.
- Aplikasi stand-alone: Aplikasi stand-alone dikembangkan untuk dijalankan pada sistem tunggal yang tidak menggunakan jaringan untuk fungsinya.
- Konsolidasi server: Konsolidasi server menurut definisi adalah pendekatan untuk penggunaan sumber daya server komputer yang efisien untuk mengurangi jumlah total server atau lokasi server yang dibutuhkan organisasi.

**Latihan Soal**

1. Apa itu konsolidasi server?
2. Bagaimana anatomi *cloud* berbeda dari arsitektur *cloud*?
3. Apa saja properti unik dari aplikasi *cloud*?
4. Apa dua klasifikasi manajemen yang berbeda?
5. Mengapa SLA penting?
6. Jelaskan beberapa pendekatan migrasi awan.
7. Apa itu jaringan akses *cloud* publik?

8. Buat daftar fase migrasi *cloud*.
9. Apa kekurangan dari aplikasi web?
10. Apa itu elastisitas?
11. Jelaskan paradigma pay-as-you-go.

# BAB 4
# MODEL PENERAPAN *CLOUD*

**Tujuan pembelajaran**

Tujuan bab buku ini adalah untuk

- Perkenalkan pembaca ke model penerapan *cloud*
- Jelaskan penerapan *cloud* secara detail
- Menganalisis keuntungan dan kerugian dari masing-masing model yang diterapkan
- Mendiskusikan masalah yang terkait dengan masing-masing model penerapan
- Menguraikan model penerapan berdasarkan properti seperti SLA dan keamanan

**Pengantar**

Bab ini secara luas membahas model penerapan yang tersedia di *cloud* yang merupakan salah satu konsep terpenting terkait komputasi *cloud*. Model penyebaran adalah berbagai cara di mana lingkungan komputasi awan dapat diatur, yaitu beberapa cara di mana awan dapat digunakan. Penting untuk memiliki gagasan tentang model penerapan karena menyiapkan awan adalah persyaratan paling dasar sebelum memulai studi lebih lanjut tentang komputasi awan. Komputasi awan berorientasi pada bisnis, dan popularitas awan dikreditkan dengan sifatnya yang berorientasi pasar. Dalam perspektif bisnis, membuat keputusan yang tepat terkait model penerapan sangatlah penting. Model harus dipilih berdasarkan kebutuhan, persyaratan, anggaran, dan keamanan. Keputusan yang salah dalam model penerapan dapat sangat memengaruhi organisasi. Oleh karena itu, sangat penting untuk mengetahui tentang model penyebaran. Ada banyak pengguna *cloud*, dan setiap pengguna memiliki kebutuhan yang berbeda. Satu model penerapan tidak akan cocok untuk semua pengguna *cloud*. Berdasarkan pengaturan *cloud*, properti *cloud* berubah. Ada empat jenis model penerapan yang tersedia di *cloud*, yaitu privat, publik, komunitas, dan hybrid. Masing-masing dan setiap jenis memiliki kelebihan dan kekurangannya sendiri seperti yang dibahas di bagian selanjutnya.

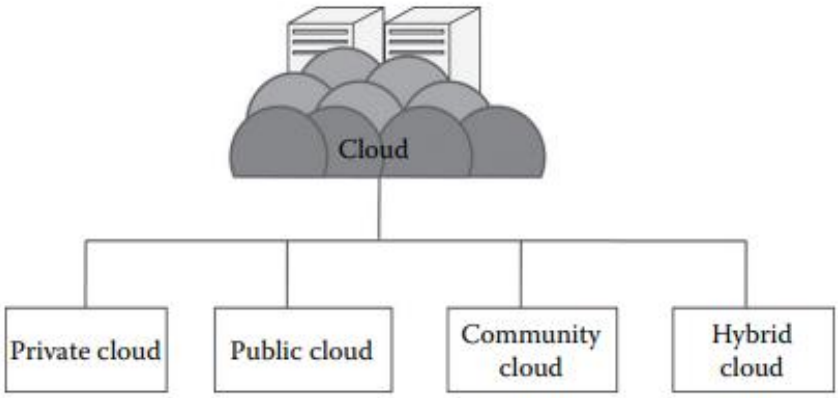
## 4.1 PENDAHULUAN

Model penyebaran dapat didefinisikan sebagai berbagai cara di mana *cloud* dapat digunakan. Model-model ini sepenuhnya berpusat pada pengguna, yaitu bergantung pada kebutuhan dan kenyamanan pengguna. Seorang pengguna memilih model berdasarkan kebutuhan dan kebutuhannya. Pada dasarnya, ada empat jenis model penerapan di *cloud*:

1. *Cloud* pribadi
2. Awan publik
3. Awan komunitas
4. Awan hibrida

Klasifikasi *cloud* didasarkan pada beberapa parameter seperti ukuran *cloud* (jumlah sumber daya), jenis penyedia layanan, lokasi, jenis pengguna, keamanan, dan masalah lainnya. Ukuran terkecil adalah private *cloud* (Gambar 4.1).

*Cloud* pribadi adalah model penyebaran paling dasar yang dapat digunakan oleh satu organisasi untuk penggunaan pribadinya. Itu tidak dibagikan oleh organisasi lain, dan tidak diizinkan untuk penggunaan publik. *Cloud* pribadi adalah untuk melayani orang-orang dari suatu organisasi. Biasanya di lokasi tetapi dapat dialihdayakan juga. Yang berikutnya adalah *cloud* komunitas, yang merupakan perpanjangan dari *cloud* pribadi. Di sini, *cloud* sama dengan private *cloud* tetapi digunakan bersama oleh beberapa organisasi. *Cloud* komunitas didirikan untuk tujuan bersama.




**Gambar 4.1** Model penerapan *cloud*.

Penyebabnya bisa apa saja, tetapi biasanya mengarah pada keuntungan bersama di antara organisasi yang berpartisipasi. Berikutnya adalah *cloud* publik, yang merupakan kebalikan dari *cloud* pribadi. *Cloud* ini memungkinkan akses dari mana saja di dunia dan terbuka untuk umum. *Cloud* ini memiliki ukuran terbesar di antara semua model penyebaran lainnya. Model *cloud* publik adalah salah satu model penerapan yang paling populer. Penyedia layanan *cloud* publik mengenakan biaya kepada pengguna setiap jam dan melayani pengguna sesuai dengan perjanjian tingkat layanan (SLA), yang dibahas di bagian selanjutnya. Yang berikutnya adalah *cloud* hybrid, yang merupakan kombinasi dari penerapan lainnya. Biasanya, ini terdiri dari gabungan *cloud* privat dan publik. Beberapa properti *cloud* pribadi digunakan dengan properti *cloud* publik. *Cloud* ini adalah salah satu model *cloud* mendatang yang tumbuh di industri. Keempat jenis penyebaran *cloud* dibahas secara rinci di bagian selanjutnya.

## 4.2 *CLOUD* PRIBADI

Di bagian ini, model penerapan *cloud* pribadi dibahas. Menurut National Institute of Standards and Technology (NIST), private *cloud* dapat didefinisikan sebagai infrastruktur *cloud* yang disediakan untuk penggunaan eksklusif oleh satu organisasi yang terdiri dari beberapa konsumen (misalnya, unit bisnis). Itu mungkin dimiliki, dikelola, dan dioperasikan oleh organisasi, pihak ketiga, atau beberapa kombinasi dari mereka, dan mungkin ada di dalam atau di luar lokasi.

*Cloud* pribadi dalam istilah sederhana adalah lingkungan *cloud* yang dibuat untuk satu organisasi. Biasanya pribadi untuk organisasi tetapi dapat dikelola oleh organisasi atau pihak

ketiga lainnya. *Cloud* pribadi dapat digunakan menggunakan alat Opensource seperti Openstack, Eucalyptus.

*Cloud* pribadi berukuran kecil dibandingkan dengan model *cloud* lainnya. Di sini, *cloud* digunakan dan dikelola oleh organisasi itu sendiri.

**Karakteristik**

Karakteristik tertentu dari *cloud* pribadi adalah sebagai berikut:

1. Aman: *Cloud* pribadi aman. Hal ini karena biasanya private *cloud* digunakan dan dikelola oleh organisasi itu sendiri, dan karenanya kecil kemungkinan data bocor keluar dari *cloud*. Dalam kasus awan outsourcing, penyedia layanan dapat melihat awan (meskipun diatur oleh SLA), tetapi tidak ada risiko lain dari orang lain karena semua pengguna milik organisasi yang sama.
2. Kontrol pusat: Sebagian besar organisasi memiliki kendali penuh atas *cloud* karena biasanya *cloud* pribadi dikelola oleh organisasi itu sendiri. Dengan demikian, ketika dikelola oleh organisasi itu sendiri, organisasi tidak perlu bergantung pada siapa pun.
3. SLA yang lemah: SLA formal mungkin ada atau tidak ada di *cloud* pribadi. Tetapi jika ada, mereka lemah karena berada di antara organisasi dan pengguna organisasi yang sama. Dengan demikian, ketersediaan tinggi dan layanan yang baik mungkin tersedia atau tidak. Hal ini bergantung pada organisasi yang mengendalikan *cloud*.

**Kesesuaian**

Kesesuaian mengacu pada contoh di mana model *cloud* ini dapat digunakan. Ini juga menandakan kondisi dan lingkungan yang paling sesuai di mana model *cloud* ini dapat digunakan, seperti berikut ini:

- Organisasi atau perusahaan yang memerlukan *cloud* terpisah untuk penggunaan pribadi atau resmi mereka.
- Organisasi atau perusahaan yang memiliki dana yang cukup karena mengelola dan memelihara *cloud* adalah urusan yang mahal.
- Organisasi atau perusahaan yang menganggap penting keamanan data.
- Organisasi yang menginginkan otonomi dan kendali penuh atas *cloud*.
- Organisasi yang memiliki jumlah pengguna lebih sedikit.
- Organisasi yang memiliki infrastruktur bawaan untuk menerapkan *cloud* dan siap melakukan pemeliharaan *cloud* secara tepat waktu untuk fungsi yang efisien.
- Perhatian khusus harus diberikan dan sumber daya harus tersedia untuk pemecahan masalah.

Platform *cloud* pribadi tidak cocok untuk yang berikut ini:

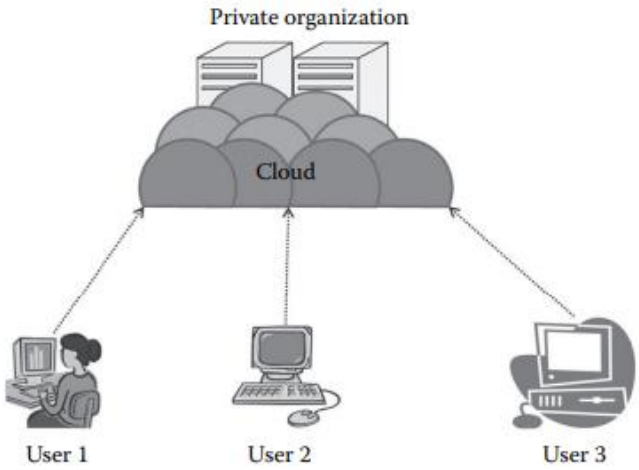
- Organisasi yang memiliki basis pengguna yang tinggi
- Organisasi yang memiliki kendala keuangan
- Organisasi yang tidak memiliki infrastruktur prebuilt
- Organisasi yang tidak memiliki cukup tenaga untuk memelihara dan mengelola *cloud*

Menurut NIST, private *cloud* dapat diklasifikasikan menjadi beberapa jenis berdasarkan lokasi dan pengelolaannya:

- *Cloud* pribadi di tempat
- *Cloud* pribadi yang dialihdayakan

### *Cloud* Pribadi Di Tempat

*Cloud* pribadi di tempat adalah *cloud* pribadi tipikal yang dikelola oleh satu organisasi. Di sini, *cloud* disebarkan di tempat organisasi dan terhubung ke jaringan organisasi. Gambar 4.2 menggambarkan private *cloud* (on premis).




**Gambar 4.2** *Cloud* pribadi di tempat.

#### Masalah

Ada beberapa masalah yang terkait dengan *cloud* pribadi seperti yang dibahas berikut ini:

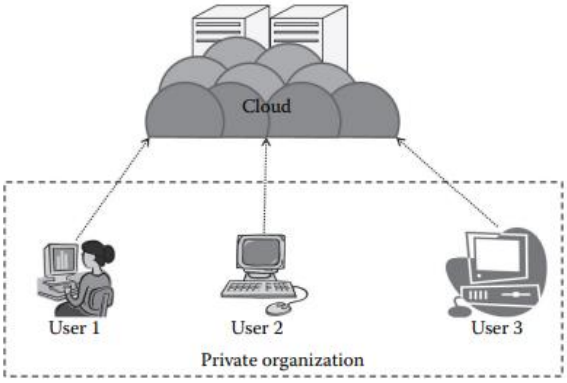
1. SLA: SLA memainkan peran yang sangat penting dalam model penerapan layanan *cloud* apa pun. Agar *cloud* apa pun dapat beroperasi, harus ada perjanjian tertentu antara pengguna dan penyedia layanan. Penyedia layanan akan menyepakati syarat dan ketentuan tertentu terkait pemberian layanan. Syarat dan ketentuan ini harus diikuti dengan ketat; jika tidak, akan ada penalti dari pihak yang wanprestasi. Jika penyedia layanan gagal memberikan layanan sesuai SLA, maka ia harus membayar denda kepada pengguna; hukuman ini bisa dalam bentuk apapun, yang disebut menurut SLA. SLA ini memiliki efek berbeda pada model pengiriman *cloud* yang berbeda. Di sini, di *cloud* pribadi, SLA ditentukan antara organisasi dan penggunanya, yaitu sebagian besar karyawan. Biasanya, para pengguna ini memiliki hak akses yang lebih luas daripada pengguna *cloud* publik pada umumnya. Demikian pula di sisi penyedia layanan, penyedia layanan dapat menyediakan layanan secara efisien karena basis pengguna yang kecil dan sebagian besar jaringan yang efisien.
2. Jaringan: *Cloud* sepenuhnya bergantung pada jaringan yang ditata. Jaringan biasanya terdiri dari bandwidth yang tinggi dan memiliki latensi yang rendah.
   Ini karena koneksi hanya ada di dalam organisasi. Manajemen jaringan lebih mudah dalam hal ini, dan menyelesaikan masalah jaringan lebih mudah.
3. Performa: Performa model pengiriman *cloud* terutama bergantung pada jaringan dan sumber daya. Karena di sini jaringan dikelola secara internal, kinerja dapat

dikontrol oleh tim manajemen jaringan, dan sebagian besar akan memiliki kinerja yang baik karena jumlah sumber dayanya rendah.

4. Keamanan dan privasi data: Keamanan dan privasi data, meskipun merupakan masalah dengan setiap jenis model layanan, paling tidak memengaruhi *cloud* pribadi. Karena data pengguna hanya dikelola oleh perusahaan dan sebagian besar data akan terkait dengan organisasi atau perusahaan, di sini kecil kemungkinan data akan bocor ke orang luar karena tidak ada pengguna di luar. organisasi. Oleh karena itu, secara komparatif, private *cloud* lebih tahan terhadap serangan daripada jenis *cloud* lainnya semata-mata karena jenis pengguna dan jaringan area lokal. Namun, pelanggaran keamanan mungkin terjadi jika pengguna internal menyalahgunakan hak istimewa tersebut.
5. Lokasi: *Cloud* pribadi tidak memiliki masalah terkait lokasi data yang disimpan. Di *cloud* pribadi, data bersifat internal dan biasanya disimpan di lokasi geografis yang sama di mana pengguna *cloud*, yaitu organisasi, hadir (*cloud* di lokasi). Jika sebuah perusahaan memiliki beberapa lokasi fisik, maka *cloud* didistribusikan ke beberapa tempat. Dalam hal ini, ada kemungkinan sumber daya *cloud* harus diakses menggunakan Internet (dengan membangun jaringan pribadi virtual [VPN] atau tanpa VPN).
6. Manajemen *cloud*: Manajemen *cloud* adalah area luas di mana seluruh tugas terkait *cloud* dikelola untuk memberikan layanan yang mulus kepada pelanggan. Ini melibatkan beberapa tugas seperti penjadwalan sumber daya, penyediaan sumber daya, dan manajemen sumber daya. Jumlah pengguna, ukuran jaringan, dan jumlah sumber daya adalah beberapa parameter penting yang memengaruhi pengelolaan *cloud*. Di sini, jaringannya kecil, dan jumlah pengguna serta sumber dayanya lebih sedikit.
7. Multitenancy: *Cloud* pada dasarnya memiliki arsitektur multitenant. Karena arsitektur multitenant mendukung banyak penyewa dengan sumber daya fisik atau perangkat lunak yang sama, ada kemungkinan akses data yang tidak diinginkan, dan efeknya akan lebih kecil di *cloud* pribadi karena semua masalah akan bersifat intraorganisasi.
8. Pemeliharaan: *Cloud* dikelola oleh organisasi tempat *cloud* diterapkan. Sumber daya yang rusak (drive dan prosesor) diganti dengan sumber daya yang baik. Jumlah sumber daya di *cloud* pribadi lebih sedikit, sehingga pemeliharaan relatif lebih mudah.

### *Cloud* Pribadi yang Dialihdayakan

*Cloud* pribadi yang dialihdayakan memiliki *cloud* yang dialihdayakan ke pihak ketiga. Pihak ketiga mengelola seluruh *cloud*. Semuanya sama seperti *cloud* pribadi biasa kecuali di sini *cloud* dialihdayakan. Ada beberapa keuntungan dan kerugian dari outsourcing *cloud*. Berikut ini adalah properti yang memiliki perubahan signifikan karena sifat outsourcing *cloud*. Semua aspek lainnya sama dengan *cloud* pribadi di tempat. Gambar 4.3 menggambarkan private *cloud* outsourcing.




**Gambar 4.3** *Cloud* pribadi outsourcing.

**Masalah**

Masalah yang khusus untuk outsourcing private *cloud* dibahas sebagai berikut:

1. SLA: SLA antara pihak ketiga dan organisasi outsourcing. Di sini, seluruh *cloud* dikelola oleh pihak ketiga yang biasanya tidak tersedia di lokasi. SLA biasanya diikuti secara ketat karena merupakan organisasi pihak ketiga.
2. Jaringan: *Cloud* sepenuhnya diterapkan di situs pihak ketiga. Jaringan internal *cloud* dikelola oleh pihak ketiga, dan organisasi terhubung ke pihak ketiga melalui koneksi khusus atau melalui Internet. Jaringan internal organisasi dikelola oleh organisasi, dan tidak berada di bawah lingkup SLA.
3. Keamanan dan privasi: Keamanan dan privasi perlu dipertimbangkan saat *cloud* dialihdayakan. Di sini, *cloud* kurang aman dibandingkan *cloud* pribadi di tempat. Privasi dan keamanan data terutama bergantung pada pihak ketiga hosting karena mereka memiliki kendali *cloud*. Namun, pada dasarnya ancaman keamanan berasal dari pihak ketiga dan karyawan internal.
4. Hukum dan konflik: Jika *cloud* ini digunakan di luar negeri, maka undang-undang keamanan yang berkaitan dengan itu akan berlaku pada data dan data tersebut masih belum sepenuhnya aman. Biasanya private *cloud* tidak digunakan di luar, tetapi jika lokasi off-site berada di luar batas negara, maka beberapa masalah mungkin muncul.
5. Lokasi: *Cloud* pribadi biasanya terletak di luar lokasi di sini. Ketika ada perubahan lokasi, data perlu ditransmisikan melalui jarak jauh. Dalam beberapa kasus, mungkin di luar negeri, yang akan menyebabkan masalah tertentu terkait data dan transfernya.
6. Performa: Performa *cloud* bergantung pada pihak ketiga yang melakukan outsourcing *cloud*.
7. Pemeliharaan: *Cloud* dikelola oleh organisasi pihak ketiga tempat *cloud* diterapkan. Seperti disebutkan, sumber daya yang rusak (drive dan prosesor) diganti dengan sumber daya yang baik. Di sini, sekali lagi prosesnya tidak sekompleks *cloud* publik. Biaya perawatan adalah masalah besar. Jika sebuah organisasi memiliki *cloud*, maka biaya yang terkait dengan *cloud* harus ditanggung oleh organisasi tersebut dan ini biasanya tinggi.

Penyebaran *cloud* pribadi ke mesin (konfigurasi) berukuran sedang kini menjadi tugas yang lebih mudah. Untuk mengalami real *cloud*, private *cloud* dapat digunakan. Konfigurasi minimum bervariasi untuk setiap jenis platform, tetapi secara umum, mesin dengan RAM 8 GB, hard disk 250 GB, dan setidaknya prosesor i7 akan memungkinkan pengguna untuk menginstal *cloud* pribadi di dalamnya. Selanjutnya, *cloud* pribadi (Infrastruktur sebagai Layanan [IaaS]) ini dapat digunakan untuk membuat mesin virtual, dan kemudian pengguna dapat menguji mesin virtual ini. Berdasarkan konfigurasi, efisiensi *cloud* bervariasi. Penyebaran ini mungkin tidak menawarkan *cloud* pribadi yang lengkap untuk beberapa pengguna tetapi bisa sangat berguna untuk memahami cara kerja *cloud* pribadi.
Ada beberapa keuntungan dan kerugian dari *cloud* pribadi.

**Keuntungan**

- *Cloud* berukuran kecil dan mudah dipelihara.
- Memberikan tingkat keamanan dan privasi yang tinggi kepada pengguna.
- Itu dikendalikan oleh organisasi.

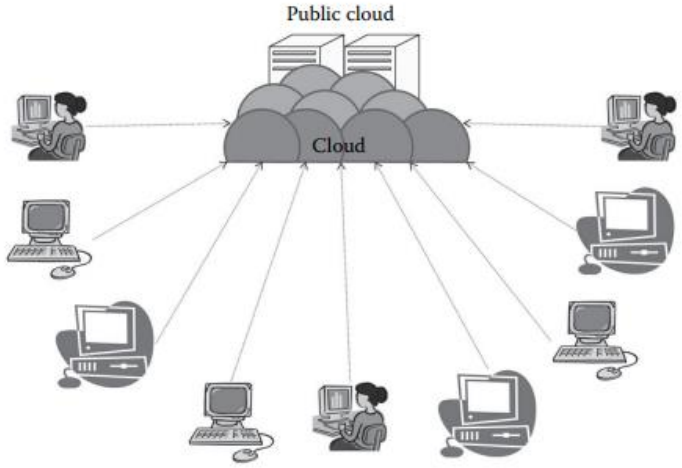
**Kerugian**

- Untuk *cloud* pribadi, anggaran menjadi kendala.
- *Cloud* pribadi memiliki SLA yang longgar.

## 4.3 *CLOUD* PUBLIK

Menurut NIST, *cloud* publik adalah infrastruktur *cloud* yang disediakan untuk penggunaan terbuka oleh masyarakat umum. Ini mungkin dimiliki, dikelola, dan dioperasikan oleh organisasi bisnis, akademik, atau pemerintah, atau beberapa kombinasi dari mereka.

Tipikal awan publik digambarkan pada Gambar 4.4. *Cloud* publik terdiri dari pengguna dari seluruh dunia. Seorang pengguna cukup membeli sumber daya setiap jam dan bekerja dengan sumber daya tersebut. Tidak diperlukan infrastruktur prebuilt untuk menggunakan *cloud* publik. Sumber daya ini tersedia di lokasi penyedia *cloud*. Biasanya, penyedia *cloud* menerima semua permintaan, dan karenanya, sumber daya di ujung penyedia layanan dianggap tidak terbatas dalam satu aspek. Beberapa contoh *cloud* publik yang terkenal adalah Amazon AWS, Microsoft Azure, dll.




**Gambar 4.4** Awan publik.

**Karakteristik**

1. Sangat dapat diskalakan: *Cloud* publik sangat dapat diskalakan. Sumber daya di *cloud* publik jumlahnya besar dan penyedia layanan memastikan bahwa semua permintaan dikabulkan. Oleh karena itu, *cloud* publik dianggap dapat diskalakan.
2. Terjangkau: *Cloud* publik ditawarkan kepada publik dengan sistem bayar sesuai pemakaian; karenanya, pengguna harus membayar hanya untuk apa yang dia gunakan (biasanya per jam). Dan, ini tidak melibatkan biaya apa pun yang terkait dengan penerapan.
3. Kurang aman: *Cloud* publik kurang aman dari keempat model penerapan. Ini karena *cloud* publik ditawarkan oleh pihak ketiga dan mereka memiliki kendali penuh atas *cloud* tersebut. Meskipun SLA memastikan privasi, masih ada risiko tinggi kebocoran data.
4. Sangat tersedia: *Cloud* publik sangat tersedia karena siapa pun dari belahan dunia mana pun dapat mengakses *cloud* publik dengan izin yang sesuai, dan ini tidak mungkin dilakukan di model lain karena pembatasan akses geografis atau lainnya mungkin ada.
5. SLA yang ketat: SLA sangat ketat dalam hal *cloud* publik. Karena reputasi bisnis dan kekuatan pelanggan penyedia layanan sepenuhnya bergantung pada layanan *cloud*, mereka mengikuti SLA dengan ketat dan pelanggaran dapat dihindari. SLA ini sangat kompetitif.

**Kesesuaian**

Ada beberapa kesempatan dan lingkungan di mana *cloud* publik cocok. Dengan demikian, kesesuaian *cloud* publik dijelaskan. *Cloud* publik dapat digunakan kapan pun hal berikut berlaku:

- Kebutuhan sumber daya besar, yaitu basis pengguna yang besar.
- Persyaratan sumber daya bervariasi.
- Tidak ada infrastruktur fisik yang tersedia.
- Sebuah organisasi memiliki kendala keuangan.

*Cloud* publik tidak cocok, jika yang berikut ini berlaku:

- Keamanan sangat penting.
- Organisasi mengharapkan otonomi.
- Keandalan pihak ketiga tidak disukai.

**Masalah**

Beberapa masalah yang berkaitan dengan *cloud* publik adalah sebagai berikut:

1. SLA: Berbeda dengan *cloud* pribadi, di sini jumlah pengguna lebih banyak dan begitu pula jumlah perjanjian layanan. Penyedia layanan bertanggung jawab kepada semua pengguna. Pengguna di sini beragam. SLA akan mencakup semua pengguna dari seluruh penjuru dunia. Penyedia layanan harus menjamin semua pengguna mendapat bagian yang adil tanpa prioritas apa pun. Memiliki SLA yang sama untuk semua pengguna biasanya diharapkan, tetapi bergantung pada penyedia layanan untuk memiliki SLA yang sama untuk semua pengguna di mana pun mereka berada.
2. Jaringan: Jaringan memainkan peran utama dalam *cloud* publik. Setiap pengguna yang mendapatkan layanan *cloud* mendapatkannya melalui Internet. Layanan diakses

melalui Internet oleh semua pengguna, dan karenanya, pengiriman layanan sepenuhnya bergantung pada jaringan. Berbeda dengan *cloud* pribadi di mana organisasi bertanggung jawab atas jaringan, di sini penyedia layanan tidak bertanggung jawab atas jaringan. Penyedia layanan bertanggung jawab untuk menyediakan layanan yang tepat kepada pelanggan, dan setelah layanan diberikan dari penyedia layanan, layanan tersebut diteruskan ke pengguna. Pengguna akan dikenakan biaya bahkan jika dia memiliki masalah karena jaringan. Jaringan biasanya terdiri dari bandwidth yang tinggi dan memiliki latensi yang rendah. Ini karena koneksi hanya ada di dalam organisasi. Manajemen jaringan lebih mudah dalam hal ini.

3. Performa: Seperti yang telah disebutkan, performa model pengiriman *cloud* terutama bergantung pada jaringan dan sumber daya. Penyedia layanan harus mengelola sumber daya dan jaringan secara memadai. Seiring bertambahnya jumlah pengguna, merupakan tugas yang menantang bagi penyedia layanan untuk memberikan kinerja yang baik.
4. Multitenancy: Sumber daya dibagi, yaitu, banyak pengguna berbagi sumber daya, oleh karena itu istilah multitenant. Karena properti ini, ada risiko tinggi kebocoran data atau kemungkinan akses tanpa hak.
5. Lokasi: Lokasi *cloud* publik menjadi masalah. Karena *cloud* publik terfragmentasi dan terletak di berbagai wilayah, akses ke *cloud* ini melibatkan banyak transfer data melalui Internet. Ada beberapa hal yang berkaitan dengan lokasi. Misalnya, pengguna dari India mungkin menggunakan *cloud* publik dan dia mungkin harus mengakses sumber daya pribadinya dari negara lain. Ini tidak baik karena data disimpan di beberapa negara lain.
6. Keamanan dan privasi data: Keamanan dan privasi data adalah tantangan terbesar di *cloud* publik. Karena data disimpan di berbagai tempat di seluruh dunia, keamanan data adalah masalah yang sangat besar. Seorang pengguna yang menyimpan data di luar negaranya memiliki risiko data dilihat oleh orang lain karena hal itu tidak berada di bawah yurisdiksi negara pengguna. Meskipun ini mungkin tidak selalu benar, tetapi itu mungkin terjadi.
7. Hukum dan konflik: Data disimpan di berbagai tempat di dunia di berbagai negara. Oleh karena itu, pusat data terikat pada undang-undang negara tempat mereka berada. Ini menciptakan banyak konflik dan masalah bagi penyedia layanan dan pengguna.
8. Manajemen *cloud*: Di sini, jumlah pengguna lebih banyak, sehingga pengelolaannya sulit. Pekerjaan di sini kritis terhadap waktu, dan seiring bertambahnya jumlah pengguna, ini menjadi lebih sulit. Manajemen sumber daya yang tidak efisien akan menyebabkan kekurangan sumber daya, dan layanan pengguna mungkin terpengaruh. Ini berdampak langsung pada SLA dan dapat menyebabkan pelanggaran SLA.
9. Pemeliharaan: Memelihara seluruh *cloud* adalah tugas lain. Ini melibatkan pemeriksaan terus menerus terhadap sumber daya, jaringan, dan parameter lain semacam itu untuk penyampaian layanan yang efisien dan tahan lama. Penyedia sumber daya harus terus mengubah komponen sumber daya dari waktu ke waktu.

Tugas pemeliharaan sangat krusial di *cloud* publik. Semakin baik *cloud* dipertahankan, semakin baik kualitas layanannya. Di sini, pusat data *cloud* adalah tempat pemeliharaan dilakukan; terus menerus, disk diganti dari waktu ke waktu.

Masalah yang dibahas sebelumnya akan membantu untuk memahami *cloud* publik. Sebelum menggunakan *cloud* publik, seseorang harus memilih penyedia layanan *cloud*. Seseorang dapat memilih *cloud* publik berdasarkan parameter tertentu seperti pelanggaran SLA, keamanan, dan biaya sumber daya. Jadi, kualitas *cloud* ditentukan oleh pelanggaran SLA yang dilakukannya. Semakin sedikit pelanggaran SLA yang dilakukannya, semakin baik *cloud* tersebut. Ini adalah salah satu cara memilih *cloud* publik; cara lain adalah dengan biaya. Jika pekerjaan yang menggunakan sumber daya tidak peka terhadap waktu, maka penyedia layanan yang menawarkan biaya paling murah akan dipilih.

Berikut ini adalah beberapa kelebihan dan kekurangan *cloud* publik.

**Keuntungan**

- Tidak perlu membangun infrastruktur untuk menyiapkan *cloud*.
- Tidak perlu memelihara *cloud*.
- Harganya relatif lebih murah dibandingkan model *cloud* lainnya.
- SLA yang ketat dipatuhi.
- Tidak ada batasan jumlah pengguna.
- *Cloud* publik sangat terukur.

**Kerugian**

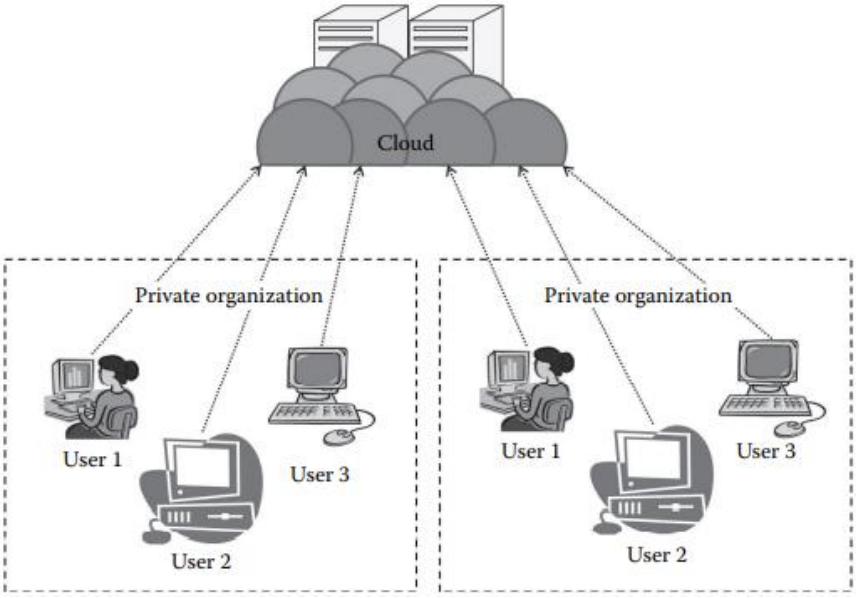
- Keamanan adalah masalah.
- Privasi dan otonomi organisasi tidak dimungkinkan.

## 4.4 *CLOUD* KOMUNITAS

Menurut NIST, *cloud* komunitas adalah infrastruktur *cloud* yang disediakan untuk penggunaan eksklusif oleh komunitas konsumen tertentu dari organisasi yang memiliki kepedulian yang sama (mis., misi, persyaratan keamanan, kebijakan, dan pertimbangan kepatuhan). Ini mungkin dimiliki, dikelola, dan dioperasikan oleh satu atau lebih organisasi di masyarakat, pihak ketiga, atau beberapa kombinasi dari mereka, dan mungkin ada di dalam atau di luar tempat. Ini adalah perpanjangan lebih lanjut dari *cloud* pribadi. Di sini, *cloud* pribadi dibagikan di antara beberapa organisasi. Baik organisasi atau satu organisasi dapat secara kolektif memelihara *cloud*.

Keuntungan utama *cloud* publik adalah bahwa organisasi dapat berbagi sumber daya di antara mereka sendiri berdasarkan masalah tertentu. Dengan demikian, di sini organisasi dapat mengekstraksi kekuatan *cloud*, yang jauh lebih besar daripada *cloud* pribadi, dan pada saat yang sama, mereka dapat menggunakannya dengan biaya yang biasanya lebih murah. Komunitas dibentuk berdasarkan penyebab umum apa pun, tetapi pada akhirnya, semua anggota komunitas diuntungkan.

Model ini sangat cocok untuk organisasi yang tidak mampu membeli *cloud* pribadi dan juga tidak dapat mengandalkan *cloud* publik. Gambar 4.5 menggambarkan *cloud* komunitas.




**Gambar 4.5** Awan komunitas.

**Karakteristik**

1. Pemeliharaan kolaboratif dan distributif: *Cloud* komunitas sepenuhnya kolaboratif, dan biasanya tidak ada satu pihak pun yang memiliki kendali penuh atas seluruh *cloud* (dalam beberapa kasus, mungkin dikendalikan oleh satu pihak). Ini biasanya distributif, dan karenanya, kerja sama yang lebih baik memberikan hasil yang lebih baik. Meskipun dapat dialihdayakan, kolaborasi berdasarkan tujuan selalu terbukti bermanfaat.
2. Sebagian aman: Sebagian aman mengacu pada properti *cloud* komunitas di mana beberapa organisasi berbagi *cloud*, sehingga ada kemungkinan data dapat bocor dari satu organisasi ke organisasi lain, meskipun aman dari dunia luar.
3. Hemat biaya: *Cloud* komunitas hemat biaya karena seluruh *cloud* digunakan bersama oleh beberapa organisasi atau komunitas. Biasanya, tidak hanya biaya tetapi setiap tanggung jawab lain yang dapat dibagi juga dibagi atau dibagi di antara kelompok.

**Kesesuaian**

Jenis *cloud* ini cocok untuk organisasi yang

- Ingin membuat *cloud* pribadi tetapi memiliki kendala keuangan
- Tidak ingin menyelesaikan tanggung jawab pemeliharaan *cloud*
- Ingin mendirikan *cloud* untuk berkolaborasi dengan *cloud* lain
- Ingin memiliki *cloud* kolaboratif dengan lebih banyak fitur keamanan daripada *cloud* publik

*Cloud* ini tidak cocok untuk organisasi yang:

- Lebih memilih otonomi dan kendali atas *cloud*
- Tidak ingin berkolaborasi dengan organisasi lain Ada dua jenis penerapan *cloud* komunitas:
    1. *Cloud* komunitas lokal
    2. *Cloud* komunitas outsourcing

### *Cloud* Komunitas Lokal

*Cloud* komunitas di lokasi terdiri dari *cloud* yang diterapkan di dalam lokasi dan dikelola oleh organisasi itu sendiri.

**Masalah**

Masalah yang terkait dengan *cloud* komunitas di tempat adalah sebagai berikut:

1. SLA: Di sini, SLA sedikit lebih ketat daripada *cloud* pribadi tetapi tidak seketat *cloud* publik. Karena lebih dari satu organisasi yang terlibat, SLA harus ada agar ada permainan yang adil di antara pengguna *cloud* dan di antara organisasi itu sendiri.
2. Jaringan: *Cloud* pribadi dapat ada di lokasi mana pun karena *cloud* ini digunakan bersama oleh lebih dari satu organisasi. Di sini, setiap organisasi akan memiliki jaringan terpisah, dan mereka akan terhubung ke *cloud*. Setiap organisasi bertanggung jawab untuk menjaga jaringan mereka sendiri. Penyedia layanan tidak bertanggung jawab atas masalah jaringan dalam organisasi. Jaringannya tidak besar dan kompleks seperti di *cloud* publik.
3. Kinerja: Dalam jenis penerapan ini, lebih dari satu organisasi berkoordinasi bersama dan menyediakan layanan *cloud*. Jadi, pada tim pemeliharaan dan manajemen kinerja tergantung.
4. Multitenancy: Ada risiko sedang karena multitenancy. Karena awan ini dimaksudkan untuk beberapa organisasi, akses yang tidak diistimewakan ke dalam data antarorganisasi dapat menyebabkan beberapa masalah.
5. Lokasi: Lokasi *cloud* sangat penting dalam hal ini. Biasanya, *cloud* digunakan di salah satu organisasi atau dikelola di luar lokasi oleh pihak ketiga mana pun. Dalam kedua kasus tersebut, organisasi harus mengakses *cloud* dari lokasi lain.
6. Keamanan dan privasi: Keamanan dan privasi adalah masalah di *cloud* komunitas karena beberapa organisasi terlibat di dalamnya. Privasi antar organisasi perlu dijaga. Karena data disimpan secara kolektif, situasinya lebih seperti *cloud* publik dengan lebih sedikit pengguna. Organisasi harus memiliki kepercayaan penuh pada penyedia layanan, dan seperti semua model *cloud* lainnya, ini menjadi penghambat.
7. Hukum dan konflik: Hal ini berlaku jika organisasi berada di negara yang berbeda. Jika organisasi tersebut berada di negara yang sama, maka tidak ada masalah, tetapi jika organisasi ini berada di tempat lain, yaitu di negara yang berbeda, maka mereka harus mematuhi aturan negara tempat infrastruktur *cloud* berada. hadir, sehingga membuat proses sedikit lebih kompleks.
8. Manajemen *cloud*: Manajemen *cloud* dilakukan oleh penyedia layanan, dalam hal ini oleh organisasi secara kolektif. Organisasi akan memiliki tim manajemen khusus untuk *cloud* ini dan yang bertanggung jawab atas semua operasi terkait manajemen *cloud*.
9. Pemeliharaan *cloud*: Pemeliharaan *cloud* dilakukan oleh organisasi secara kolektif. Tim pemeliharaan secara kolektif memelihara semua sumber daya. Ini bertanggung jawab untuk penggantian sumber daya yang berkelanjutan. Di *cloud* komunitas, jumlah sumber daya lebih sedikit daripada *cloud* publik, tetapi biasanya lebih banyak daripada *cloud* pribadi.

***Cloud* Komunitas Outsourcing**

Di *cloud* komunitas yang dialihdayakan, *cloud* dialihdayakan ke pihak ketiga. Pihak ketiga bertanggung jawab atas pemeliharaan dan pengelolaan *cloud*.

**Masalah**

Berikut adalah beberapa aspek di *cloud* komunitas yang berubah karena sifat outsourcing *cloud* komunitas:

1. SLA: SLA berada di antara grup organisasi dan penyedia layanan. SLA di sini ketat karena melibatkan pihak ketiga. SLA di sini ditujukan untuk pembagian sumber daya yang adil di antara organisasi. Penyedia layanan tidak bertanggung jawab atas masalah teknis dalam organisasi.
2. Jaringan: Masalah yang terkait dengan jaringan sama dengan *cloud* komunitas di tempat, tetapi di sini penyedia layanan dialihdayakan dan karenanya organisasi bertanggung jawab atas jaringan mereka sendiri dan penyedia layanan bertanggung jawab atas jaringan *cloud*.
3. Kinerja: Kinerja sepenuhnya tergantung pada penyedia layanan outsourcing. Penyedia layanan bertanggung jawab atas layanan yang efisien, kecuali untuk masalah jaringan di sisi klien.
4. Keamanan dan privasi: Seperti yang telah dibahas sebelumnya, ada masalah keamanan dan privasi karena beberapa organisasi terlibat di dalamnya, tetapi selain itu, keterlibatan pihak ketiga sebagai penyedia layanan akan menciptakan lebih banyak masalah karena organisasi tersebut harus sepenuhnya bergantung pada pihak ketiga.
5. Hukum dan konflik: Selain masalah hukum karena lokasi organisasi, ada masalah besar yang terkait dengan lokasi penyedia layanan *cloud*. Jika penyedia layanan berada di luar negara, maka terdapat konflik terkait undang-undang data di negara tersebut.
6. Manajemen dan pemeliharaan *cloud*: Manajemen dan pemeliharaan *cloud* dilakukan oleh penyedia layanan. Kompleksitas pengelolaan dan pemeliharaan meningkat seiring dengan banyaknya organisasi di masyarakat. Namun, ini tidak sekompleks *cloud* publik.
7. *Cloud* komunitas seperti yang dikatakan merupakan perpanjangan dari *cloud* pribadi. Isu yang dibahas sebelumnya kurang lebih sama dengan isu yang berkaitan dengan private *cloud* dengan perbedaan yang sangat sedikit. *Cloud* komunitas akan terbukti berhasil jika sekelompok organisasi bekerja secara kooperatif.

Berikut ini diuraikan beberapa kelebihan dan kekurangan dari *community cloud*.

**Keuntungan**

- Memungkinkan pembentukan *cloud* pribadi berbiaya rendah.
- Memungkinkan kerja kolaboratif di *cloud*.
- Memungkinkan pembagian tanggung jawab di antara organisasi.
- Memiliki keamanan yang lebih baik daripada *cloud* publik.

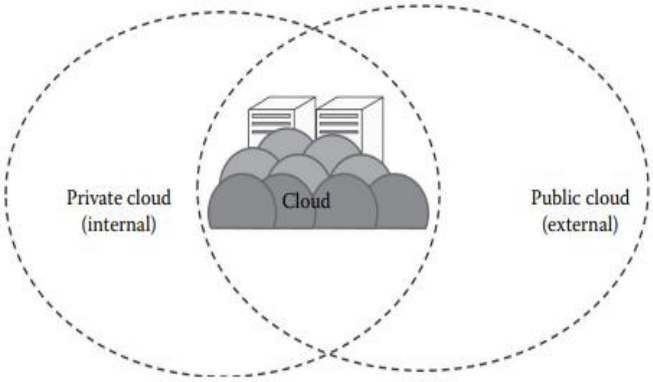
**Kerugian**

- Otonomi suatu organisasi hilang.
- Fitur keamanan tidak sebaik private *cloud*.
- Tidak cocok jika tidak ada kerjasama.

## 4.5 *CLOUD* HIBRIDA

Menurut NIST, *cloud* hybrid dapat didefinisikan sebagai infrastruktur *cloud* yang merupakan komposisi dari dua atau lebih infrastruktur *cloud* yang berbeda (swasta, komunitas, atau publik) yang tetap menjadi entitas unik namun terikat bersama oleh teknologi standar atau hak milik yang memungkinkan portabilitas data dan aplikasi.

*Cloud* hybrid biasanya merupakan kombinasi dari *cloud* publik dan privat. Ini bertujuan untuk menggabungkan keunggulan *cloud* pribadi dan publik. Metode biasa menggunakan *cloud* hybrid adalah memiliki *cloud* pribadi pada awalnya, dan kemudian untuk sumber daya tambahan, *cloud* publik digunakan. Ada beberapa keuntungan dari hybrid *cloud*. *Cloud* hybrid dapat dianggap sebagai *cloud* pribadi yang diperluas ke *cloud* publik. Ini bertujuan untuk memanfaatkan kekuatan *cloud* publik dengan mempertahankan properti *cloud* pribadi. Salah satu contoh *cloud* hybrid yang populer adalah Eucalyptus. Eucalyptus awalnya dirancang untuk *cloud* pribadi dan pada dasarnya adalah *cloud* pribadi, tetapi sekarang juga mendukung *cloud* hybrid. Gambar 4.6 menunjukkan *cloud* hybrid. Awan hibrid dapat diperluas lebih jauh ke wilayah awan federasi yang luas yang akan dibahas di bab-bab berikutnya.




**Gambar 4.6** Awan hibrida.

**Karakteristik**

1. Dapat diskalakan: *Cloud* hibrid adalah kombinasi dari satu atau beberapa model penerapan. Biasanya private dengan public *cloud* memberikan hybrid *cloud*. Alasan utama memiliki *cloud* hybrid adalah menggunakan properti *cloud* publik dengan lingkungan *cloud* pribadi. *Cloud* publik digunakan kapan pun dibutuhkan; karenanya, karena *cloud* publik dapat diskalakan, *cloud* hybrid dengan bantuan mitra publiknya juga dapat diskalakan.
2. Sebagian aman: *Cloud* hybrid biasanya merupakan kombinasi antara publik dan pribadi. *Cloud* pribadi dianggap aman, tetapi karena *cloud* hybrid juga menggunakan *cloud* publik, ada risiko pelanggaran keamanan yang tinggi. Dengan demikian, tidak dapat sepenuhnya disebut aman tetapi sebagian aman.
3. SLA yang ketat: Karena *cloud* hybrid melibatkan intervensi *cloud* publik, SLA ketat dan sesuai dengan penyedia layanan *cloud* publik. Namun secara keseluruhan, SLA lebih ketat daripada *cloud* pribadi.

4. Manajemen *cloud* yang kompleks: Manajemen *cloud* rumit dan merupakan tugas yang sulit di *cloud* hybrid karena melibatkan lebih dari satu jenis model penerapan dan juga jumlah pengguna yang tinggi.

**Kesesuaian**

Lingkungan *cloud* hybrid cocok untuk

- Organisasi yang menginginkan lingkungan *cloud* pribadi dengan skalabilitas *cloud* publik
- Organisasi yang membutuhkan keamanan lebih dari *cloud* publik *Cloud* hybrid tidak cocok untuk itu
- Organisasi yang menganggap keamanan sebagai tujuan utama
- Organisasi yang tidak akan mampu menangani manajemen *cloud* hybrid

**Masalah**

Awan dapat dianalisis dalam aspek-aspek berikut:

1. SLA: SLA adalah salah satu aspek penting dari *cloud* hybrid karena melibatkan swasta dan publik. Ada kombinasi SLA yang tepat di antara *cloud*. *Cloud* pribadi tidak memiliki perjanjian yang ketat, sedangkan *cloud* publik memiliki aturan ketat tertentu yang harus dicakup. SLA yang akan dicakup dalam setiap bidang didefinisikan dengan jelas, dan sepenuhnya bergantung pada penyedia layanan (*cloud* pribadi) untuk memberikan layanan yang efisien kepada pelanggan.
2. Jaringan: Jaringan biasanya merupakan jaringan pribadi, dan kapan pun diperlukan, *cloud* publik digunakan melalui Internet. Berbeda dengan *cloud* publik, di sini ada jaringan pribadi juga. Oleh karena itu, diperlukan upaya yang cukup besar untuk memelihara jaringan. Organisasi mengambil tanggung jawab dari jaringan.
3. Performa: *Cloud* hybrid adalah jenis *cloud* khusus di mana lingkungan pribadi dipertahankan dengan akses ke *cloud* publik kapan pun diperlukan. Jadi, di sini sekali lagi nuansa sumber daya tak terbatas dipulihkan. Penyedia *cloud* (awan pribadi) bertanggung jawab untuk menyediakan *cloud*.
4. Multitenancy: Multitenancy adalah masalah di *cloud* hybrid karena melibatkan *cloud* publik selain *cloud* pribadi. Dengan demikian, properti ini dapat disalahgunakan dan pelanggaran akan berdampak buruk karena beberapa bagian *cloud* menjadi publik.
5. Lokasi: Seperti *cloud* pribadi, lokasi *cloud* ini dapat berada di lokasi atau di luar lokasi dan dapat dialihdayakan. Mereka akan memiliki semua masalah yang terkait dengan *cloud* pribadi; selain itu, isu-isu terkait *cloud* publik juga akan muncul setiap kali ada akses terputus-putus ke *cloud* publik.
6. Keamanan dan privasi: Setiap kali pengguna diberikan layanan menggunakan *cloud* publik, keamanan dan privasi menjadi lebih ketat. Karena ini adalah *cloud* publik, ancaman kehilangan data sangat tinggi.
7. Hukum dan konflik: Beberapa undang-undang negara lain berada di bawah lingkup karena *cloud* publik terlibat, dan biasanya *cloud* publik ini terletak di luar batas negara.
8. Manajemen *cloud*: Di sini, semuanya dikelola oleh penyedia layanan *cloud* pribadi.

9. Pemeliharaan *cloud*: Pemeliharaan *cloud* memiliki kompleksitas yang sama dengan *cloud* pribadi; di sini, hanya sumber daya di bawah lingkup *cloud* pribadi yang perlu dipertahankan. Ini melibatkan biaya perawatan yang tinggi.

Hybrid *cloud* merupakan salah satu model deployment yang paling cepat berkembang, yang kini banyak diperbincangkan karena karakteristiknya seperti yang telah dibahas sebelumnya. Isu-isu yang dibahas memberikan gambaran tentang perbedaan antara model *cloud* lainnya dan model hybrid *cloud*. Ada bagian lain dari *cloud* yang disebut federated *cloud* yang dijelaskan di bab berikutnya.

Ada beberapa keuntungan dan kerugian dari *cloud* hybrid.

**Keuntungan**

- Ini memberikan kekuatan baik awan swasta dan publik.
- Hal ini sangat scalable.
- Memberikan keamanan yang lebih baik daripada *cloud* publik.

**Kerugian**

- Fitur keamanannya tidak sebaik *cloud* publik.
- Mengelola *cloud* hibrid itu rumit.
- Memiliki SLA yang ketat.

## 4.6 RINGKASAN

Komputasi awan membentuk dasar untuk banyak hal di dunia saat ini. Untuk memulainya, model penerapan membentuk dasar dan perlu diketahui sebelum memulai dengan aspek *cloud* lainnya. Model penerapan ini didasarkan pada beberapa properti seperti ukuran, lokasi, dan kompleksitas. Ada empat jenis model penerapan yang dibahas dalam bab ini. Uraian masing-masing model penyebaran dengan karakteristiknya dan kesesuaiannya dengan kebutuhan yang berbeda disediakan. Setiap jenis model penyebaran memiliki makna tersendiri. Setiap model penyebaran digunakan dalam satu atau aspek lainnya. Model penerapan ini sangat penting dan biasanya berdampak besar pada bisnis yang bergantung pada *cloud*. Pilihan model penerapan yang cerdas selalu terbukti bermanfaat, menghindari kerugian besar. Oleh karena itu, sangat penting diberikan untuk model penyebaran.

**Tinjau Poin**

- Model penerapan: Model penerapan dapat didefinisikan sebagai berbagai cara di mana *cloud* dapat diterapkan.
- Awan pribadi: Awan pribadi adalah lingkungan awan yang dibuat untuk satu organisasi.
- Awan publik: Awan publik adalah infrastruktur awan yang disediakan untuk penggunaan terbuka oleh masyarakat umum.
- Awan hibrid: Awan hibrid dapat didefinisikan sebagai infrastruktur awan yang merupakan komposisi dari dua atau lebih infrastruktur awan yang berbeda.
- Awan komunitas: Awan komunitas adalah infrastruktur awan yang disediakan untuk penggunaan eksklusif oleh komunitas konsumen tertentu dari organisasi yang memiliki kepedulian yang sama.
- SLA: SLA adalah syarat dan ketentuan yang dinegosiasikan antara penyedia layanan dan pengguna.

- Multitenancy: Multitenancy adalah properti *cloud* di mana banyak pengguna berbagi sumber daya perangkat lunak yang sama dengan penyewa.

**Latihan Soal**

1. Bandingkan dan bandingkan awan publik dan pribadi.
2. Apa itu SLA? Apakah SLA berbeda untuk setiap jenis penerapan *cloud*?
3. Menganalisis model penerapan *cloud* berdasarkan keamanan.
4. Bagaimana hukum di berbagai negara memengaruhi model *cloud* publik?
5. Membedakan community *cloud* dan hybrid *cloud* berdasarkan propertinya.
6. *Cloud* publik kurang aman. Membenarkan.
7. Apa itu *cloud* komunitas outsourcing?
8. Apa karakteristik *cloud* hybrid?
9. Apa keuntungan menggunakan *cloud* komunitas?

# BAB 5
# MODEL LAYANAN *CLOUD*

**Tujuan pembelajaran**

Tujuan utama dari bab ini adalah untuk memperkenalkan berbagai model penyampaian layanan komputasi awan. Setelah membaca bab ini, Anda akan

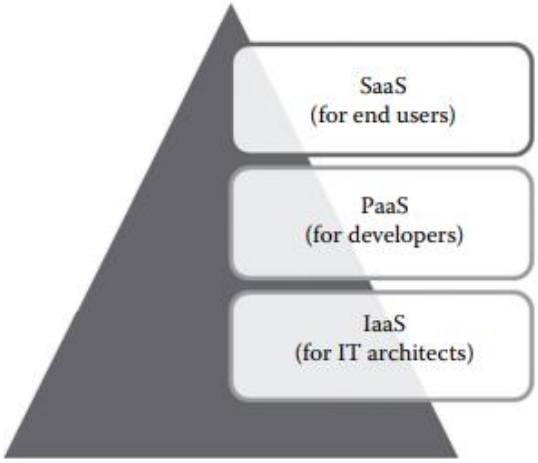
- Memahami dasar-dasar tumpukan komputasi awan dan model layanan awan
- Memahami bagaimana Infrastruktur sebagai Layanan (IaaS) mengubah komputasi
- Memahami bagaimana Platform as a Service (PaaS) mengubah pengembang aplikasi
- Memahami bagaimana Perangkat Lunak sebagai Layanan (SaaS) mengubah pengiriman aplikasi
- Memahami karakteristik, kesesuaian, serta kelebihan dan kekurangan IaaS, PaaS, dan SaaS
- Memahami model layanan *cloud* lainnya seperti Network as a Service (NaaS) dan Storage as a Service (STaaS)

**Pengantar**

Komputasi awan menyediakan sumber daya komputasi, platform pengembangan, dan aplikasi sebagai layanan kepada pengguna akhir. Industri teknologi informasi (TI) mulai berlangganan layanan *cloud* alih-alih membeli produk. Bab ini memberikan wawasan tentang tiga model layanan dasar komputasi awan, yaitu IaaS, PaaS, dan SaaS. Sesuai dengan layanan yang disediakan dan dilanggan, tanggung jawab pengguna akhir dan layanan dapat bervariasi. Bab ini juga membahas tanggung jawab pengguna akhir dan penyedia layanan IaaS, PaaS, dan SaaS. Karakteristik, kesesuaian, dan pro dan kontra dari berbagai model layanan *cloud* juga dibahas dalam bab ini bersama dengan ringkasan penyedia layanan populer. Di bagian akhir, bab ini memberikan gambaran singkat tentang model layanan *cloud* lainnya seperti NaaS, StaaS, Database as a Service (DBaaS), Security as a Service (SECaaS), dan Identity as a Service (IDaaS).

## 5.1 PENDAHULUAN

Komputasi awan adalah model yang memungkinkan pengguna akhir untuk mengakses kumpulan sumber daya bersama seperti komputasi, jaringan, penyimpanan, basis data, dan aplikasi sebagai layanan sesuai permintaan tanpa perlu membeli atau memilikinya. Layanan disediakan dan dikelola oleh penyedia layanan, mengurangi upaya pengelolaan dari sisi pengguna akhir. Karakteristik penting dari *cloud* termasuk layanan mandiri sesuai permintaan, akses jaringan luas, pengumpulan sumber daya, elastisitas cepat, dan layanan terukur. National Institute of Standards and Technology (NIST) mendefinisikan tiga model layanan dasar, yaitu IaaS, PaaS, dan SaaS, seperti yang ditunjukkan pada Gambar 5.1.



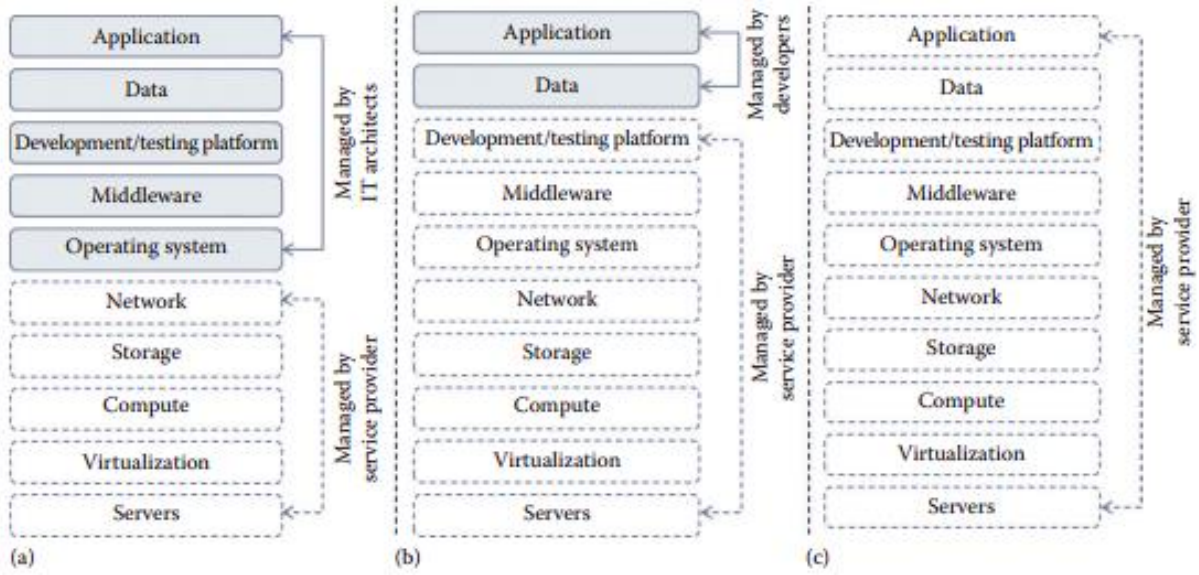

**Gambar 5.1** Model layanan *cloud* dasar.

Definisi NIST dari tiga model layanan dasar diberikan sebagai berikut:

1. IaaS: Kemampuan yang diberikan kepada arsitek infrastruktur untuk menyebarkan atau menjalankan perangkat lunak apa pun pada sumber daya komputasi yang disediakan oleh penyedia layanan. Di sini, infrastruktur dasar seperti komputasi, jaringan, dan penyimpanan dikelola oleh penyedia layanan. Dengan demikian, arsitek infrastruktur dibebaskan dari pemeliharaan pusat data atau infrastruktur yang mendasarinya. Pengguna akhir bertanggung jawab untuk mengelola aplikasi yang berjalan di atas infrastruktur *cloud* penyedia layanan. Umumnya, layanan IaaS disediakan dari pusat data *cloud* penyedia layanan. Pengguna akhir dapat mengakses layanan dari perangkat mereka melalui antarmuka baris perintah web (CLI) atau antarmuka pemrograman aplikasi (API) yang disediakan oleh penyedia layanan. Beberapa penyedia IaaS yang populer termasuk Amazon Web Services (AWS), Google Compute Engine, OpenStack, dan Eucalyptus.
2. PaaS: Kemampuan yang diberikan kepada pengembang untuk mengembangkan dan menyebarkan aplikasi pada platform pengembangan yang disediakan oleh penyedia layanan. Dengan demikian, pengembang dibebaskan dari pengelolaan platform pengembangan dan infrastruktur yang mendasarinya. Di sini, pengembang bertanggung jawab untuk mengelola aplikasi yang diterapkan dan mengonfigurasi lingkungan pengembangan. Secara umum, layanan PaaS disediakan oleh penyedia layanan di infrastruktur *cloud on-premise* atau *dedicated* atau host. Pengembang dapat mengakses platform pengembangan melalui Internet melalui web CLI, antarmuka pengguna web (UI), dan lingkungan pengembangan terintegrasi (IDE). Beberapa penyedia PaaS yang populer termasuk Google App Engine, Force.com, Red Hat OpenShift, Heroku, dan Engine Yard.
3. SaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses aplikasi melalui Internet yang dihosting dan dikelola oleh penyedia layanan. Dengan demikian, pengguna akhir dibebaskan dari pengelolaan atau pengendalian aplikasi, platform pengembangan, dan infrastruktur yang mendasarinya. Secara umum, layanan SaaS dihosting di infrastruktur *cloud* yang dikelola oleh penyedia layanan atau yang

dihosting oleh penyedia layanan. Pengguna akhir dapat mengakses layanan dari klien tipis atau browser web apa pun. Beberapa penyedia SaaS yang populer termasuk Saleforce.com, Google Apps, dan Microsoft office 365.

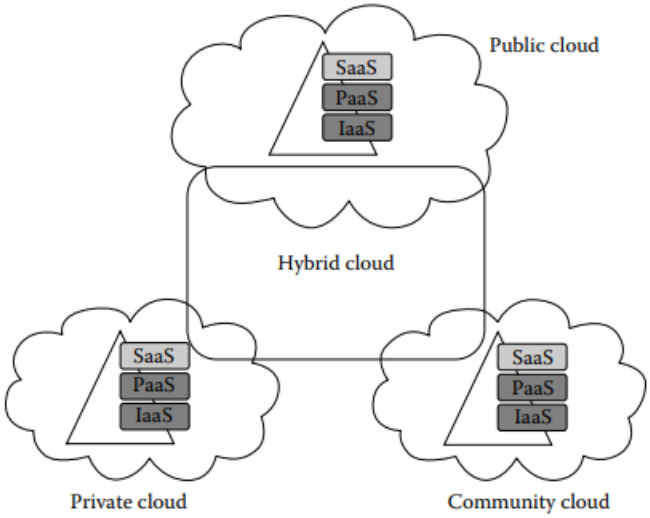
Model layanan *cloud* yang berbeda menargetkan audiens yang berbeda. Misalnya, model IaaS menargetkan arsitek teknologi informasi (TI), PaaS menargetkan pengembang, dan SaaS menargetkan pengguna akhir. Berdasarkan layanan yang dilanggan, tanggung jawab khalayak sasaran dapat bervariasi seperti yang ditunjukkan pada Gambar 5.2.




**Gambar 5.2** Tanggung jawab pengguna dan penyedia layanan model layanan *cloud*: (a) IaaS, (b) PaaS, dan (c) SaaS.

Di IaaS, pengguna akhir bertanggung jawab untuk memelihara platform pengembangan dan aplikasi yang berjalan di atas infrastruktur yang mendasarinya. Penyedia IaaS bertanggung jawab untuk memelihara perangkat keras yang mendasari seperti yang ditunjukkan pada Gambar 5.2a. Dalam PaaS, pengguna akhir bertanggung jawab untuk mengelola aplikasi yang mereka kembangkan. Infrastruktur dasar akan dipelihara oleh penyedia infrastruktur seperti yang ditunjukkan pada Gambar 5.2b. Di SaaS, pengguna akhir bebas dari pemeliharaan infrastruktur, platform pengembangan, dan aplikasi yang mereka gunakan. Semua pemeliharaan akan dilakukan oleh penyedia SaaS seperti yang ditunjukkan Gambar 5.2c.

Model layanan *cloud computing* yang berbeda dapat digunakan dan dikirim melalui salah satu model penyebaran *cloud*. NIST mendefinisikan empat jenis model penyebaran *cloud*, yaitu *cloud* publik, *cloud* pribadi, *cloud* komunitas, dan *cloud* hybrid. *Cloud* publik disediakan untuk masyarakat umum. *Cloud* pribadi digunakan oleh organisasi untuk beberapa unit bisnisnya. *Cloud* komunitas adalah untuk beberapa grup organisasi dengan tujuan yang sama. *Cloud* hybrid adalah kombinasi dari *cloud* publik, privat, dan komunitas. Penyampaian layanan *cloud* melalui model penyebaran yang berbeda ditunjukkan pada Gambar 5.3.

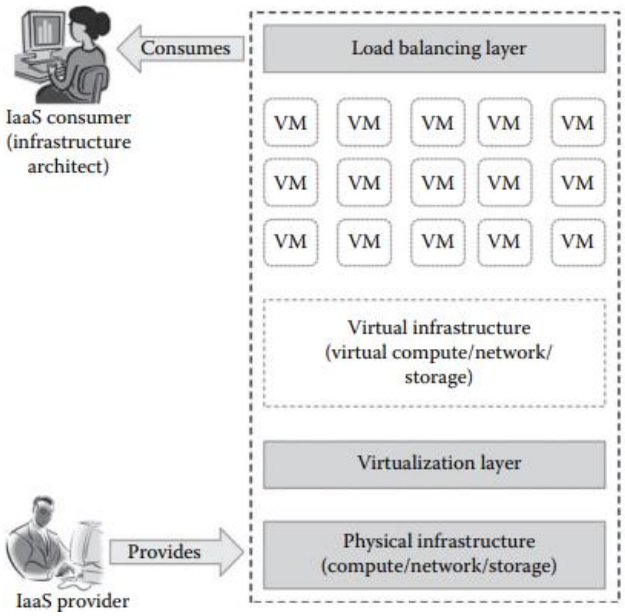



**Gambar 5.3** Penyebaran dan pengiriman berbagai model pengiriman layanan *cloud*.

Bab ini membahas tentang karakteristik, kesesuaian, dan pro dan kontra dari berbagai model layanan *cloud*. Selain itu, bab ini memberikan ringkasan penyedia IaaS, PaaS, dan SaaS yang populer.

## 5.2 INFRASTRUKTUR SEBAGAI LAYANAN

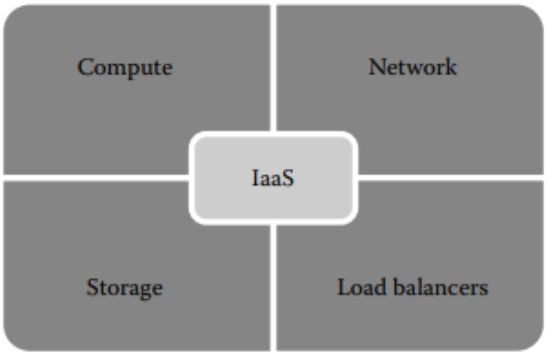
IaaS mengubah cara penggunaan sumber daya komputasi, penyimpanan, dan jaringan. Di pusat data tradisional, daya komputasi dikonsumsi dengan memiliki akses fisik ke infrastruktur. IaaS mengubah komputasi dari infrastruktur fisik menjadi infrastruktur virtual. IaaS menyediakan komputasi virtual, penyimpanan, dan sumber daya jaringan dengan mengabstraksikan sumber daya fisik. Virtualisasi teknologi digunakan untuk menyediakan sumber daya virtual. Semua sumber daya virtual diberikan ke mesin virtual (VM) yang dikonfigurasi oleh penyedia layanan. Pengguna akhir atau arsitek TI akan menggunakan sumber daya infrastruktur dalam bentuk VM seperti yang ditunjukkan pada Gambar 5.4.




**Gambar 5.4** Sekilas tentang IaaS.

Audiens yang ditargetkan dari IaaS adalah arsitek TI. Arsitek TI dapat merancang infrastruktur virtual, jaringan, penyeimbang muatan, dll., berdasarkan kebutuhan mereka. Arsitek TI tidak perlu memelihara server fisik seperti yang dipelihara oleh penyedia layanan. Infrastruktur fisik dapat dipelihara oleh penyedia layanan itu sendiri. Dengan demikian, ini menghilangkan atau menyembunyikan kerumitan pemeliharaan infrastruktur fisik dari arsitek TI. Penyedia IaaS tipikal dapat menyediakan layanan mengalir seperti yang ditunjukkan pada Gambar 5.5:

1. Compute: Computing as a Service mencakup virtual central processing unit (CPU) dan memori utama virtual untuk VM yang disediakan untuk pengguna akhir.
2. Penyimpanan: STaaS menyediakan penyimpanan back-end untuk image VM. Beberapa penyedia IaaS juga menyediakan back end untuk menyimpan file.
3. Jaringan: Jaringan sebagai Layanan (NaaS) menyediakan komponen jaringan virtual seperti router virtual, sakelar, dan jembatan untuk VM.
4. Load balancers: Load Balancing as a Service dapat memberikan kemampuan load balancing pada lapisan infrastruktur.




**Gambar 5.5** Layanan yang disediakan oleh penyedia IaaS.

**Karakteristik IaaS**

Penyedia IaaS menawarkan sumber daya komputasi virtual kepada konsumen dengan basis bayar sesuai penggunaan. IaaS mengandung karakteristik *cloud computing* seperti *on-demand self-service,* akses jaringan luas, *resource pooling*, elastisitas cepat, dan layanan terukur. Terlepas dari semua ini, IaaS memiliki karakteristik uniknya sendiri sebagai berikut:

1. Akses web ke sumber daya: Model IaaS memungkinkan pengguna TI untuk mengakses sumber daya infrastruktur melalui Internet. Saat mengakses daya komputasi yang sangat besar, pengguna TI tidak perlu mendapatkan akses fisik ke server. Melalui browser web atau konsol manajemen apa pun, pengguna dapat mengakses infrastruktur yang diperlukan.
2. Manajemen terpusat: Meskipun sumber daya fisik didistribusikan, manajemen akan berasal dari satu tempat. Sumber daya yang didistribusikan ke berbagai bagian dapat dikontrol dari konsol manajemen mana pun. Ini memastikan pengelolaan sumber daya yang efektif dan pemanfaatan sumber daya yang efektif.
3. Elastisitas dan penskalaan dinamis: IaaS menyediakan layanan elastis di mana penggunaan sumber daya dapat ditingkatkan atau dikurangi sesuai dengan kebutuhan.

Kebutuhan infrastruktur tergantung pada beban pada aplikasi. Menurut beban, layanan IaaS dapat menyediakan sumber daya. Beban pada aplikasi apa pun bersifat dinamis dan layanan IaaS mampu membuktikan layanan yang dibutuhkan secara dinamis.

4. Infrastruktur bersama: IaaS mengikuti model pengiriman satu-ke-banyak dan memungkinkan banyak pengguna TI untuk berbagi infrastruktur fisik yang sama. Pengguna TI yang berbeda akan diberikan VM yang berbeda. IaaS memastikan pemanfaatan sumber daya yang tinggi.
5. VM yang telah dikonfigurasi sebelumnya: Penyedia IaaS menawarkan VM yang telah dikonfigurasi sebelumnya dengan sistem operasi (OS), konfigurasi jaringan, dll. Pengguna TI dapat memilih segala jenis VM pilihan mereka. Pengguna TI bebas mengonfigurasi VM dari awal. Pengguna dapat langsung mulai menggunakan VM segera setelah mereka berlangganan layanan tersebut.
6. Layanan terukur: IaaS memungkinkan pengguna TI untuk menyewa sumber daya komputasi alih-alih membelinya. Layanan yang dikonsumsi oleh pengguna TI akan diukur, dan pengguna akan dikenakan biaya oleh penyedia IaaS berdasarkan jumlah penggunaan.

**Kesesuaian IaaS**

IaaS mengurangi total biaya kepemilikan (TCO) dan meningkatkan laba atas investasi (ROI) untuk perusahaan pemula yang tidak dapat berinvestasi lebih banyak dalam membeli infrastruktur.

IaaS dapat digunakan dalam situasi berikut:

1. Lonjakan penggunaan yang tidak dapat diprediksi: Ketika ada lonjakan yang signifikan dalam penggunaan sumber daya komputasi, IaaS adalah pilihan terbaik untuk industri TI. Ketika permintaan sangat fluktuatif, kami tidak dapat memprediksi lonjakan dan penurunan dalam hal permintaan infrastruktur. Dalam situasi ini, kami tidak dapat menambah atau menghapus infrastruktur dengan segera sesuai permintaan infrastruktur tradisional. Jika ada permintaan infrastruktur yang tidak dapat diprediksi, maka disarankan untuk menggunakan layanan IaaS.
2. Investasi modal terbatas: Perusahaan pemula baru tidak dapat berinvestasi lebih banyak untuk membeli infrastruktur untuk kebutuhan bisnis mereka. Jadi dengan menggunakan IaaS, perusahaan start-up dapat mengurangi investasi modal pada perangkat keras. IaaS adalah opsi yang cocok untuk perusahaan pemula dengan investasi modal lebih sedikit pada perangkat keras.
3. Infrastruktur sesuai permintaan: Beberapa organisasi mungkin membutuhkan infrastruktur yang besar untuk waktu yang singkat. Untuk tujuan ini, organisasi tidak mampu membeli lebih banyak sumber daya di tempat. Sebaliknya, mereka dapat menyewa infrastruktur yang dibutuhkan untuk jangka waktu tertentu. IaaS paling cocok untuk organisasi yang mencari infrastruktur sesuai permintaan atau untuk jangka waktu singkat.

IaaS membantu perusahaan pemula membatasi pengeluaran modalnya. Meskipun banyak digunakan oleh perusahaan pemula, ada beberapa situasi di mana IaaS mungkin bukan pilihan terbaik. Dalam situasi berikut, pengguna TI harus menghindari penggunaan IaaS:

1. Ketika kepatuhan peraturan tidak mengizinkan hosting di luar lokasi: Untuk beberapa perusahaan, peraturannya mungkin tidak mengizinkan aplikasi dan data dihosting di infrastruktur luar lokasi pihak ketiga.
2. Saat penggunaan minimal: Saat penggunaan minimal dan infrastruktur lokal yang tersedia mampu memenuhi kebutuhan mereka.
3. Ketika diperlukan kinerja yang lebih baik: Karena layanan IaaS diakses melalui Internet, terkadang kinerja mungkin tidak seperti yang diharapkan karena latensi jaringan.
4. Ketika ada kebutuhan untuk lebih mengontrol infrastruktur fisik: Beberapa organisasi mungkin memerlukan kontrol fisik atas infrastruktur yang mendasarinya. Karena layanan IaaS diabstraksi sebagai sumber daya virtual, tidak mungkin untuk memiliki kontrol lebih besar pada infrastruktur fisik yang mendasarinya.

**Kelebihan dan Kekurangan IaaS**

Menjadi salah satu model layanan penting komputasi awan, IaaS memberikan banyak manfaat bagi pengguna TI. Berikut adalah manfaat yang diberikan oleh IaaS:

1. Model bayar sesuai penggunaan: Layanan IaaS diberikan kepada pelanggan berdasarkan pembayaran per penggunaan. Ini memastikan bahwa pelanggan diharuskan membayar untuk apa yang telah mereka gunakan. Model ini menghilangkan pengeluaran yang tidak perlu untuk membeli perangkat keras.
2. Pengurangan TCO: Karena penyedia IaaS mengizinkan pengguna TI untuk menyewa sumber daya komputasi, mereka tidak perlu membeli perangkat keras fisik untuk menjalankan bisnis mereka. Para pengguna TI dapat menyewa infrastruktur TI daripada membelinya dengan mengeluarkan biaya besar. IaaS mengurangi kebutuhan untuk membeli sumber daya perangkat keras dan dengan demikian mengurangi TCO.
3. Sumber daya elastis: IaaS menyediakan sumber daya berdasarkan kebutuhan saat ini. Pengguna TI dapat memperbesar atau memperkecil sumber daya kapan pun mereka mau. Penskalaan dinamis ini dilakukan secara otomatis menggunakan beberapa penyeimbang beban. Penyeimbang beban ini mentransfer permintaan sumber daya tambahan ke server baru dan meningkatkan efisiensi aplikasi.
4. Pemanfaatan sumber daya yang lebih baik: Pemanfaatan sumber daya adalah kriteria terpenting untuk berhasil dalam bisnis TI. Infrastruktur yang dibeli harus dimanfaatkan dengan baik untuk meningkatkan ROI. IaaS memastikan pemanfaatan sumber daya yang lebih baik dan memberikan ROI yang tinggi untuk penyedia IaaS.
5. Mendukung TI Ramah Lingkungan: Dalam infrastruktur TI tradisional, server khusus digunakan untuk kebutuhan bisnis yang berbeda. Karena banyak server yang digunakan, konsumsi daya akan tinggi. Ini tidak menghasilkan Green IT. Di IaaS, kebutuhan untuk membeli server khusus dihilangkan karena infrastruktur tunggal dibagi di antara banyak pelanggan, sehingga mengurangi jumlah server yang harus dibeli dan konsumsi daya yang menghasilkan Green IT.

Meskipun IaaS memberikan manfaat terkait biaya untuk industri skala kecil, ia kurang memberikan keamanan pada data. Berikut ini adalah kekurangan dari IaaS:

1. Masalah keamanan: Karena IaaS menggunakan virtualisasi sebagai teknologi yang memungkinkan, hypervisor memainkan peran penting. Ada banyak serangan yang menargetkan hypervisor untuk mengkompromikannya. Jika hypervisor disusupi, maka setiap VM dapat diserang dengan mudah. Sebagian besar penyedia IaaS tidak dapat memberikan keamanan 100% ke VM dan data yang disimpan di VM.
2. Masalah interoperabilitas: Tidak ada standar umum yang diikuti di antara penyedia IaaS yang berbeda. Sangat sulit untuk memigrasi VM apa pun dari satu penyedia IaaS ke penyedia lainnya. Terkadang, pelanggan mungkin menghadapi masalah vendor lock-in.
3. Masalah kinerja: IaaS hanyalah konsolidasi sumber daya yang tersedia dari server *cloud* terdistribusi. Di sini, semua server terdistribusi terhubung melalui jaringan. Latensi jaringan memainkan peran penting dalam menentukan kinerja. Karena masalah latensi, terkadang VM mengalami masalah dengan kinerjanya.

**Ringkasan Penyedia IaaS**

Ada banyak penyedia IaaS publik dan swasta di pasar yang menyediakan layanan infrastruktur kepada pengguna akhir. Tabel 5.1 menyajikan ringkasan penyedia infrastruktur populer.

**Tabel 5.1** Ringkasan Penyedia IaaS Populer

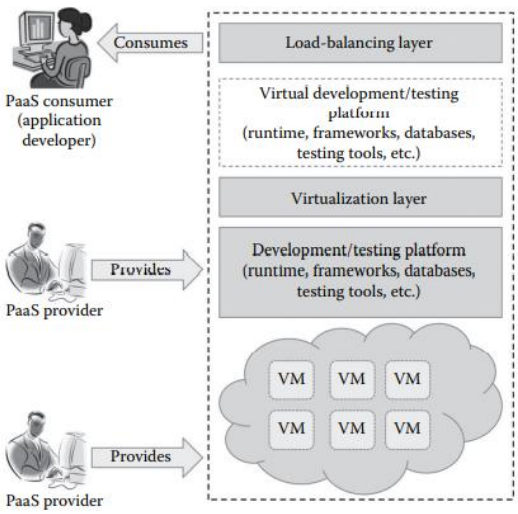
| Provider | License | Deployment Model | Host OS | Guest OS | Supported Hypervisor(s) |
|---|---|---|---|---|---|
| Amazon Web Services | Proprietary | Public | Not available | Red Hat Linux, Windows Server, SuSE Linux, Ubuntu, Fedora, Debian, CentOS, Gentoo Linux, Oracle Linux, and FreeBSD | Xen |
| Google Compute Engine | Proprietary | Public | Not available | Debian 7 Wheezy, CentOS 6, Red Hat Enterprise Linux, SUSE, Windows Server, CoreOS, FreeBSD, and SELinux | KVM |
| Microsoft Windows Azure | Proprietary | Public | Not available | Windows Server, CentOS, FreeBSD, openSUSE Linux, and Oracle Enterprise Linux | Windows Azure hypervisor |
| Eucalyptus | GPLv3 | Private and hybrid | Linux | Linux and Windows | Xen, KVM, VMware |
| Apache *Cloud*Stack | Apache 2 | Private | Linux | Windows, Linux, and various versions of BSD | KVM, vSphere, XenServer/ |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | XCP |
| OpenNebula | Apache 2 | Private, public, and hybrid | CentOS, Debian, and openSUSE | Microsoft Windows and Linux | Xen, KVM, VMware |
| OpenStack | Apache 2 | Private and public | CentOS, Debian, Fedora, RHEL, openSUSE, and Ubuntu | CentOS, Ubuntu, Microsoft Windows, and FreeBSD | libvirt, Hyper-V, VMware, XenServer 6.2, baremetal, docker, Xen, LXC via libvirt |

Dalam tabel, penyedia IaaS yang populer diklasifikasikan berdasarkan lisensi, model penerapan, dan OS host yang didukung, OS tamu, dan hypervisor. Pengguna akhir dapat memilih penyedia IaaS yang sesuai dengan kebutuhan mereka. Secara umum, konsumen IaaS publik tidak perlu mempertimbangkan OS host karena dikelola oleh penyedia layanan. Dalam mengelola *cloud* pribadi, pengguna harus melihat OS host yang didukung. Namun, sebagian besar IaaS pribadi mendukung OS tamu populer, sepenuhnya bergantung pada hypervisor yang didukung oleh penyedia IaaS.

## 5.3 PLATFORM SEBAGAI LAYANAN

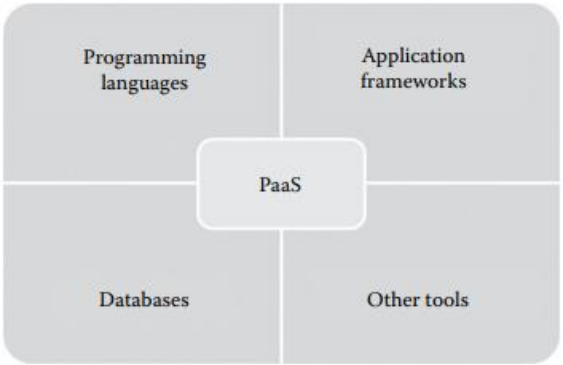
PaaS mengubah cara perangkat lunak dikembangkan dan digunakan. Dalam pengembangan aplikasi tradisional, aplikasi akan dikembangkan secara lokal dan akan dihosting di lokasi pusat. Dalam pengembangan aplikasi yang berdiri sendiri, aplikasi akan dikembangkan dan disampaikan sebagai executable. Sebagian besar aplikasi yang dikembangkan oleh platform pengembangan tradisional menghasilkan perangkat lunak berbasis lisensi, sedangkan PaaS mengubah pengembangan aplikasi dari mesin lokal menjadi online. Penyedia PaaS menyediakan pengembangan PaaS dari pusat data. Pengembang dapat menggunakan layanan melalui Internet seperti yang ditunjukkan pada Gambar 5.6.




**Gambar 5.6** Sekilas tentang PaaS.

PaaS memungkinkan para pengembang untuk mengembangkan aplikasi mereka secara online dan juga memungkinkan mereka untuk menerapkan langsung pada platform yang sama. Konsumen atau pengembang PaaS dapat menggunakan runtime bahasa, kerangka kerja aplikasi, basis data, antrian pesan, alat pengujian, dan alat penyebaran sebagai layanan melalui Internet. Dengan demikian, ini mengurangi kerumitan membeli dan memelihara alat yang berbeda untuk mengembangkan aplikasi. Penyedia PaaS tipikal dapat menyediakan bahasa pemrograman, kerangka kerja aplikasi, basis data, dan alat pengujian seperti yang ditunjukkan pada Gambar 5.7. Beberapa penyedia PaaS juga menyediakan alat bangun, alat penerapan, dan penyeimbang beban perangkat lunak sebagai layanan:

1. Bahasa pemrograman: Penyedia PaaS menyediakan berbagai macam bahasa pemrograman bagi pengembang untuk mengembangkan aplikasi. Beberapa bahasa pemrograman populer yang disediakan oleh vendor PaaS adalah Java, Perl, PHP, Python, Ruby, Scala, Clojure, dan Go.
2. Kerangka kerja aplikasi: Vendor PaaS menyediakan kerangka kerja aplikasi yang menyederhanakan pengembangan aplikasi. Beberapa framework pengembangan aplikasi populer yang disediakan oleh penyedia PaaS termasuk Node.js, Rails, Drupal, Joomla, WordPress, Django, EE6, Spring, Play, Sinatra, Rack, dan Zend.
3. Basis data: Karena setiap aplikasi perlu berkomunikasi dengan basis data, itu menjadi alat yang harus dimiliki untuk setiap aplikasi. Penyedia PaaS juga menyediakan database dengan platform PaaS mereka. Basis data populer yang disediakan oleh vendor PaaS yang populer adalah ClearDB, PostgreSQL, Cloudant, Membase, MongoDB, dan Redis.
4. Alat lain: Penyedia PaaS menyediakan semua alat yang diperlukan untuk mengembangkan, menguji, dan menggunakan aplikasi.




**Gambar 5.7** Layanan yang disediakan oleh penyedia PaaS.

**Karakteristik PaaS**

Platform pengembangan PaaS berbeda dari platform pengembangan aplikasi tradisional. Berikut ini adalah karakteristik penting yang membuat PaaS unik dari platform pengembangan tradisional:

1. Semua dalam satu: Sebagian besar penyedia PaaS menawarkan layanan untuk mengembangkan, menguji, menerapkan, menghosting, dan memelihara aplikasi dalam IDE yang sama. Selain itu, banyak penyedia layanan menyediakan semua bahasa

pemrograman, kerangka kerja, basis data, dan layanan terkait pengembangan lainnya yang membuat pengembang memilih dari beragam platform pengembangan.

2. Akses web ke platform pengembangan: Platform pengembangan tipikal menggunakan IDE apa pun untuk mengembangkan aplikasi. Biasanya, IDE akan dipasang di mesin pengembang. Namun, PaaS menyediakan akses web ke platform pengembangan. Menggunakan UI web, setiap pengembang dapat memperoleh akses ke platform pengembangan. UI berbasis web membantu pengembang membuat, memodifikasi, menguji, dan menerapkan aplikasi yang berbeda pada platform yang sama.
3. Akses offline: Pengembang mungkin tidak dapat terhubung ke Internet sepanjang hari untuk mengakses layanan PaaS. Ketika tidak ada konektivitas internet, para pengembang harus diizinkan untuk bekerja secara offline. Untuk mengaktifkan pengembangan offline, beberapa penyedia PaaS mengizinkan pengembang menyinkronkan IDE lokal mereka dengan layanan PaaS. Pengembang dapat mengembangkan aplikasi secara lokal dan menerapkannya secara online kapan pun mereka terhubung ke Internet.
4. Skalabilitas bawaan: Skalabilitas merupakan persyaratan penting untuk aplikasi web atau SaaS generasi baru. Sangat sulit untuk mengaktifkan skalabilitas dinamis untuk aplikasi apa pun yang dikembangkan menggunakan platform pengembangan tradisional. Namun, layanan PaaS menyediakan skalabilitas bawaan untuk aplikasi yang dikembangkan menggunakan PaaS tertentu. Ini memastikan bahwa aplikasi mampu menangani beban yang bervariasi secara efisien.
5. Platform kolaboratif: Saat ini, tim pengembangan terdiri dari pengembang yang bekerja dari berbagai tempat. Ada kebutuhan untuk platform bersama di mana para pengembang dapat bekerja sama secara kolaboratif dalam proyek yang sama. Sebagian besar layanan PaaS memberikan dukungan untuk pengembangan kolaboratif. Untuk mengaktifkan kolaborasi antar pengembang, sebagian besar penyedia PaaS menyediakan alat untuk perencanaan proyek dan komunikasi.
6. Alat klien yang beragam: Untuk mempermudah pengembangan, penyedia PaaS menyediakan berbagai macam alat klien untuk membantu pengembang. Alat klien termasuk CLI, CLI web, UI web, REST API, dan IDE. Pengembang dapat memilih alat apa pun pilihan mereka. Alat klien ini juga mampu menangani penagihan dan pengelolaan langganan.

**Kesesuaian PaaS**

Sebagian besar perusahaan pengembangan SaaS pemula dan vendor perangkat lunak independen (ISV) banyak menggunakan PaaS dalam mengembangkan aplikasi. Teknologi PaaS juga mendapat perhatian dari perusahaan pengembangan perangkat lunak tradisional lainnya. PaaS adalah opsi yang cocok untuk situasi berikut:

1. Pengembangan kolaboratif: Untuk meningkatkan waktu ke pasar dan efisiensi pengembangan, ada kebutuhan akan tempat bersama di mana tim pengembangan dan pemangku kepentingan aplikasi lainnya dapat berkolaborasi satu sama lain. Karena layanan PaaS menyediakan lingkungan pengembangan kolaboratif, ini

merupakan opsi yang cocok untuk aplikasi yang membutuhkan kolaborasi antara pengembang dan pihak ketiga lainnya untuk melakukan proses pengembangan.

2. Pengujian dan penerapan otomatis: Pengujian dan pembuatan aplikasi otomatis sangat berguna saat mengembangkan aplikasi dalam kerangka waktu yang sangat singkat. Alat pengujian otomatis mengurangi waktu yang dihabiskan dalam alat pengujian manual. Sebagian besar layanan PaaS menawarkan kemampuan pengujian dan penyebaran otomatis. Tim pengembangan perlu lebih berkonsentrasi pada pengembangan daripada pengujian dan penyebaran. Dengan demikian, layanan PaaS adalah pilihan terbaik di mana ada kebutuhan untuk pengujian otomatis dan penyebaran aplikasi.
3. Waktu ke pasar: Layanan PaaS mengikuti metodologi pengembangan berulang dan inkremental yang memastikan bahwa aplikasi ada di pasar sesuai kerangka waktu yang diberikan. Misalnya, layanan PaaS adalah pilihan terbaik untuk pengembangan aplikasi yang menggunakan metodologi pengembangan tangkas. Jika vendor perangkat lunak menginginkan aplikasinya segera tersedia di pasar, maka layanan PaaS adalah pilihan terbaik untuk pengembangan.

PaaS digunakan secara luas untuk mempercepat proses pengembangan aplikasi untuk memastikan waktu ke pasar. Sebagian besar perusahaan pemula dan ISV mulai bermigrasi ke layanan PaaS. Meskipun digunakan secara luas, ada beberapa situasi di mana PaaS mungkin bukan pilihan terbaik:

1. Migrasi aplikasi yang sering: Masalah utama dengan layanan PaaS adalah vendor lock-in. Karena tidak ada standar umum yang diikuti di antara penyedia PaaS, sangat sulit untuk memindahkan aplikasi dari satu penyedia PaaS ke penyedia lainnya.
2. Kustomisasi di tingkat infrastruktur: PaaS adalah layanan abstrak, dan pengguna PaaS tidak memiliki kendali penuh atas infrastruktur yang mendasarinya. Ada beberapa platform pengembangan aplikasi yang memerlukan beberapa konfigurasi atau penyesuaian infrastruktur yang mendasarinya. Dalam situasi ini, tidak mungkin menyesuaikan infrastruktur dasar dengan PaaS. Jika platform pengembangan aplikasi memerlukan konfigurasi apa pun pada tingkat perangkat keras, tidak disarankan untuk menggunakan PaaS.
3. Fleksibilitas di tingkat platform: PaaS menyediakan aplikasi berbasis template di mana semua bahasa pemrograman, database, dan antrean pesan yang berbeda telah ditentukan sebelumnya. Merupakan keuntungan jika aplikasi tersebut merupakan aplikasi generik.
4. Integrasi dengan aplikasi lokal: Perusahaan mungkin telah menggunakan layanan PaaS untuk beberapa rangkaian aplikasi. Untuk beberapa set aplikasi, mereka mungkin menggunakan platform on-premise. Karena banyak layanan PaaS menggunakan teknologi milik mereka sendiri untuk menentukan tumpukan aplikasi, ini mungkin tidak cocok dengan tumpukan aplikasi di tempat. Hal ini membuat integrasi aplikasi yang dihosting di platform on-premise dan platform PaaS menjadi pekerjaan yang sulit.

**Pro dan Kontra PaaS**

Keuntungan utama menggunakan PaaS adalah menyembunyikan kerumitan pemeliharaan platform dan infrastruktur yang mendasarinya. Hal ini memungkinkan pengembang untuk bekerja lebih banyak dalam mengimplementasikan fungsionalitas penting dari aplikasi. Selain itu, PaaS memiliki manfaat sebagai berikut:

1. Pengembangan dan penerapan cepat: PaaS menyediakan semua alat pengembangan dan pengujian yang diperlukan untuk mengembangkan, menguji, dan menerapkan perangkat lunak di satu tempat. Sebagian besar layanan PaaS mengotomatiskan proses pengujian dan penerapan segera setelah pengembang menyelesaikan pengembangan. Ini mempercepat pengembangan dan penyebaran aplikasi daripada platform pengembangan tradisional.
2. Mengurangi TCO: Pengembang tidak perlu membeli alat pengembangan dan pengujian berlisensi jika layanan PaaS dipilih. Sebagian besar platform pengembangan tradisional membutuhkan infrastruktur kelas atas agar dapat berfungsi, yang meningkatkan TCO perusahaan pengembangan aplikasi. Namun, PaaS mengizinkan pengembang untuk menyewa perangkat lunak, platform pengembangan, dan alat pengujian untuk mengembangkan, membangun, dan menyebarkan aplikasi. PaaS juga tidak memerlukan infrastruktur kelas atas untuk mengembangkan aplikasi, sehingga mengurangi TCO perusahaan pengembang.
3. Mendukung pengembangan perangkat lunak: Saat ini, sebagian besar aplikasi generasi baru dikembangkan menggunakan metodologi tangkas. Banyak perusahaan pengembangan ISV dan SaaS mulai mengadopsi metodologi tangkas untuk pengembangan aplikasi. Layanan PaaS mendukung metodologi tangkas yang dicari oleh ISV dan perusahaan pengembang lainnya.
4. Tim yang berbeda dapat bekerja sama: Platform pengembangan tradisional tidak memiliki dukungan ekstensif untuk pengembangan kolaboratif. Layanan PaaS mendukung pengembang dari berbagai tempat untuk bekerja sama dalam proyek yang sama. Ini dimungkinkan karena platform pengembangan umum online yang disediakan oleh penyedia PaaS.
5. Kemudahan penggunaan: Platform pengembangan tradisional menggunakan salah satu antarmuka berbasis CLI atau IDE untuk pengembangan. Beberapa pengembang mungkin tidak terbiasa dengan antarmuka yang disediakan oleh platform pengembangan aplikasi. Ini membuat pekerjaan pengembangan sedikit sulit. Namun, PaaS menyediakan berbagai macam alat klien seperti CLI, CLI web, UI web, API, dan IDE. Pengembang bebas memilih alat klien pilihan mereka. Terutama, layanan PaaS berbasis UI web meningkatkan kegunaan platform pengembangan untuk semua jenis pengembang.
6. Lebih sedikit biaya pemeliharaan: Dalam aplikasi di lokasi, perusahaan pengembang atau vendor perangkat lunak bertanggung jawab untuk memelihara perangkat keras yang mendasarinya. Mereka perlu merekrut administrator yang terampil untuk memelihara server. Overhead ini dihilangkan oleh layanan PaaS karena infrastruktur

yang mendasarinya dikelola oleh penyedia infrastruktur. Hal ini memberikan kebebasan kepada developer untuk mengerjakan pengembangan aplikasi.

7. Menghasilkan aplikasi yang dapat diskalakan: Sebagian besar aplikasi yang dikembangkan menggunakan layanan PaaS adalah aplikasi web atau aplikasi SaaS. Aplikasi ini membutuhkan skalabilitas yang lebih baik pada beban ekstra. Untuk menangani beban tambahan, vendor perangkat lunak perlu memelihara server tambahan. Sangat sulit bagi perusahaan start-up baru untuk menyediakan server tambahan berdasarkan beban tambahan. Namun, layanan PaaS menyediakan skalabilitas bawaan untuk aplikasi yang dikembangkan menggunakan platform PaaS.

PaaS memberikan banyak manfaat bagi pengembang jika dibandingkan dengan lingkungan pengembangan tradisional. Di sisi lain, ini mengandung kekurangan, yang dijelaskan sebagai berikut:

1. Vendor lock-in: Kelemahan utama penyedia PaaS adalah vendor lock-in. Alasan utama penguncian vendor adalah kurangnya standar. Tidak ada standar umum yang diikuti di antara penyedia PaaS yang berbeda. Alasan lain untuk penguncian vendor adalah teknologi hak milik yang digunakan oleh penyedia PaaS. Sebagian besar vendor PaaS menggunakan teknologi eksklusif yang tidak kompatibel dengan penyedia PaaS lainnya. Masalah vendor lock-in layanan PaaS tidak memungkinkan aplikasi dimigrasikan dari satu penyedia PaaS ke penyedia lainnya.
2. Masalah keamanan: Seperti di layanan *cloud* lainnya, keamanan adalah salah satu masalah utama dalam layanan PaaS. Karena data disimpan di server pihak ketiga di luar lokasi, banyak pengembang takut menggunakan layanan PaaS. Tentu saja, banyak penyedia PaaS menyediakan mekanisme untuk melindungi data pengguna, dan tidak cukup untuk merasakan keamanan penerapan di tempat. Saat memilih penyedia PaaS, pengembang harus meninjau peraturan, kepatuhan, dan kebijakan keamanan penyedia PaaS dengan persyaratan keamanan mereka sendiri. Jika tidak ditinjau dengan benar, pengembang atau pengguna berisiko mengalami pelanggaran keamanan data.
3. Kurang fleksibel: Penyedia PaaS tidak memberikan banyak kebebasan bagi pengembang untuk menentukan tumpukan aplikasi mereka sendiri. Sebagian besar penyedia PaaS menyediakan banyak bahasa pemrograman, basis data, dan alat pengembangan lainnya. Tapi, itu tidak luas dan tidak memenuhi semua kebutuhan pengembang. Hanya beberapa penyedia PaaS yang mengizinkan pengembang untuk memperluas alat PaaS dengan bahasa pemrograman kustom atau baru. Masih sebagian besar penyedia PaaS tidak memberikan fleksibilitas kepada para pengembang.
4. Bergantung pada koneksi Internet: Karena layanan PaaS dikirimkan melalui Internet, pengembang harus bergantung pada konektivitas Internet untuk mengembangkan aplikasi. Meskipun beberapa penyedia mengizinkan akses offline, sebagian besar penyedia PaaS tidak mengizinkan akses offline. Dengan koneksi Internet yang lambat, kegunaan dan efisiensi platform PaaS tidak memenuhi persyaratan pengembang.

**Ringkasan Penyedia PaaS**

Penyedia PaaS lebih banyak berada di pasar TI untuk *cloud* publik dan privat. Tabel 5.2 memberikan ringkasan penyedia PaaS swasta dan publik yang populer.

**Tabel 5.2** Ringkasan Penyedia PaaS Populer

| Provider | License | Deployment Model | Supported Languages | Supported Frameworks | Supported Databases | Client Tools |
|---|---|---|---|---|---|---|
| *Cloud* Foundry | Open source and proprietary | Public | Python, PHP, Java, Groovy, Scala, and Ruby | Spring, Grails, Play, Node.js, Lift, Rails, Sinatra, and Rack | MySQL, PostgreSQL, MongoDB, and Redis | cf. CLI, IDEs, and build tools |
| Google App Engine | Proprietary | Public | Python, Java, Groovy, JRuby, Scala, Clojure, Go, and PHP | Django, CherryPy, Pyramid, Flask, web2py, and webapp2. | Google *Cloud* SQL, Datastore, BigTable, and Blobstore | APIs |
| Heroku | Proprietary | Public | Ruby, Java, Scala, Clojure and Python, PHP, and Perl | Rails, Play, Django, and Node.js. | ClearDB, PostgreSQL, *Cloud*ant, Membase, MongoDB, and Redis | CLI and RESTful API |
| Microsoft Windows Azure | Proprietary | Public | .Net, PHP, Python, Ruby, and Java | Django, Rails, Drupal, Joomla, WordPress, DotNetNuke, and Node.js. | SQL Azure, MySQL, MongoDB, and CouchDB | RESTful API and IDEs |
| Red Hat OpenShift Online | Proprietary | Public | Java, Ruby, Python, PHP, and Perl | Node.js, Rails, Drupal, Joomla, WordPress, Django, EE6, Spring, Play, Sinatra, Rack, and Zend. | MySQL, PostgreSQL, and MongoDB | Web UI, APIs, CLI, and IDEs |
| ActiveState Stackato | Proprietary | Private | Java, Perl, PHP, Python, Ruby, Scala, Clojure, and Go | Spring, Node.js, Drupal, Joomla, WordPress, Django, Rails, and Sinatra. | MySQL, PostgreSQL, MongoDB, and Redis | CLI and IDE |

| | | | | | | |
|---|---|---|---|---|---|---|
| Apprenda | Proprietary | Private | .Net and Java | Most of the frameworks form .Net. | SQL Server | REST APIs |
| *Cloud*Bees | Proprietary | Private | Java, Groovy, and Scala | Spring, JRails, JRuby, and Grails. | MySQL, PostgreSQL, MongoDB, and CouchDB | API, SDK, and IDEs |
| Cumulogic | Proprietary | Private | Java, PHP, and Python | Spring and Grails. | MySQL, MongoDB, and Couchbase | RESTful API |
| Gigaspaces *Cloud*ify | Open source | Private | Any programming language specified by recipe | Rails, Play, and others. | MySQL, MongoDB, Couchbase, Cassandra, and others | CLI, web UI, and REST API |

## 5.4 PERANGKAT LUNAK SEBAGAI LAYANAN

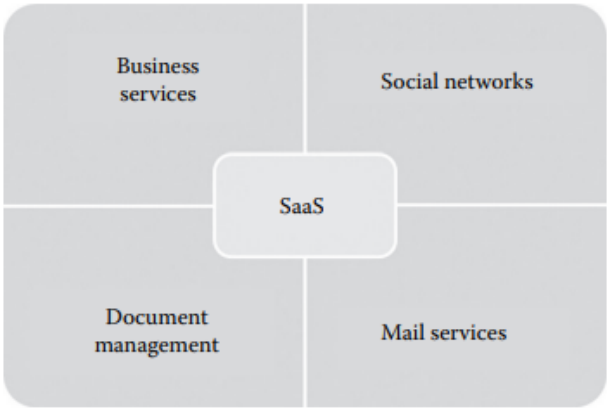
SaaS mengubah cara perangkat lunak dikirimkan ke pelanggan. Dalam model perangkat lunak tradisional, perangkat lunak dikirimkan sebagai produk berbasis lisensi yang perlu dipasang di perangkat pengguna akhir. Karena SaaS dikirimkan sebagai layanan sesuai permintaan melalui Internet, tidak perlu menginstal perangkat lunak ke perangkat pengguna akhir. Layanan SaaS dapat diakses atau diputuskan kapan saja berdasarkan kebutuhan pengguna akhir. Layanan SaaS dapat diakses dari browser web ringan apa pun di perangkat apa pun seperti laptop, tablet, dan ponsel cerdas. Beberapa layanan SaaS dapat diakses dari klien tipis yang tidak memiliki banyak ruang penyimpanan dan tidak dapat menjalankan banyak perangkat lunak seperti PC desktop tradisional. Manfaat penting menggunakan thin client untuk mengakses aplikasi SaaS adalah sebagai berikut: kurang rentan terhadap serangan, memiliki siklus hidup lebih lama, mengkonsumsi lebih sedikit daya, dan lebih murah.

Penyedia SaaS tipikal dapat menyediakan layanan bisnis, jejaring sosial, manajemen dokumen, dan layanan surat seperti yang ditunjukkan pada Gambar 5.8:

1. Layanan bisnis: Sebagian besar penyedia SaaS mulai menyediakan berbagai layanan bisnis yang menarik perusahaan pemula. Layanan SaaS bisnis meliputi ERP, CRM, penagihan, penjualan, dan sumber daya manusia.
2. Jejaring sosial: Karena situs jejaring sosial banyak digunakan oleh masyarakat umum, banyak penyedia layanan jejaring sosial mengadopsi SaaS untuk keberlanjutannya. Karena jumlah pengguna situs jejaring sosial meningkat secara eksponensial, komputasi awan adalah pasangan yang sempurna untuk menangani beban variabel.
3. Manajemen dokumen: Karena sebagian besar perusahaan secara ekstensif menggunakan dokumen elektronik, sebagian besar penyedia SaaS mulai menyediakan

layanan yang digunakan untuk membuat, mengelola, dan melacak dokumen elektronik.

4. Layanan surat: Layanan surat elektronik saat ini digunakan oleh banyak orang. Pertumbuhan penggunaan email di masa depan tidak dapat diprediksi. Untuk menangani jumlah pengguna yang tidak dapat diprediksi dan beban pada layanan email, sebagian besar penyedia email mulai menawarkan layanan mereka sebagai layanan SaaS.




**Gambar 5.8** Layanan yang disediakan oleh Penyedia SaaS.

**Karakteristik SaaS**

Layanan SaaS berbeda dan memberi lebih banyak manfaat bagi pengguna akhir daripada perangkat lunak tradisional. Berikut ini adalah karakteristik penting dari layanan SaaS yang membuatnya unik dari perangkat lunak tradisional:

1. Satu ke banyak: Layanan SaaS dikirimkan sebagai model satu-ke-banyak di mana satu contoh aplikasi dapat digunakan bersama oleh banyak penyewa atau pelanggan.
2. Akses web: Layanan SaaS menyediakan akses web ke perangkat lunak. Ini memungkinkan pengguna akhir untuk mengakses aplikasi dari lokasi mana pun jika perangkat terhubung ke Internet.
3. Manajemen terpusat: Karena layanan SaaS dihosting dan dikelola dari lokasi pusat, pengelolaan aplikasi SaaS menjadi lebih mudah. Biasanya, penyedia SaaS akan melakukan pembaruan otomatis yang memastikan bahwa setiap penyewa mengakses versi aplikasi terbaru tanpa pembaruan sisi pengguna.
4. Dukungan multiperangkat: Layanan SaaS dapat diakses dari semua perangkat pengguna akhir seperti desktop, laptop, tablet, smartphone, dan thin client.
5. Skalabilitas yang lebih baik: Karena sebagian besar layanan SaaS memanfaatkan PaaS dan IaaS untuk pengembangan dan penerapannya, ini memastikan skalabilitas yang lebih baik daripada perangkat lunak tradisional. Penskalaan dinamis sumber daya *cloud* yang mendasari membuat aplikasi SaaS bekerja secara efisien bahkan dengan beban yang bervariasi.
6. Ketersediaan tinggi: Layanan SaaS memastikan 99,99% ketersediaan data pengguna karena mekanisme pencadangan dan pemulihan yang tepat diterapkan di bagian belakang.

7. Integrasi API: Layanan SaaS memiliki kemampuan berintegrasi dengan perangkat lunak atau layanan lain melalui API standar.

**Kesesuaian SaaS**

SaaS populer di kalangan individu dan perusahaan pemula karena manfaat yang diberikannya. Sebagian besar pengguna perangkat lunak tradisional mencari versi perangkat lunak SaaS karena SaaS memiliki beberapa keunggulan dibandingkan aplikasi tradisional. Aplikasi SaaS adalah pilihan terbaik untuk yang berikut ini:

1. Perangkat lunak berdasarkan permintaan: Model perangkat lunak berbasis lisensi memerlukan pembelian perangkat lunak paket lengkap dan meningkatkan pengeluaran untuk membeli perangkat lunak. Beberapa perangkat lunak yang kadang-kadang digunakan tidak memberikan ROI apa pun. Karena itu, banyak pengguna akhir mencari perangkat lunak yang dapat mereka gunakan saat dan saat dibutuhkan. Jika pengguna akhir mencari perangkat lunak berdasarkan permintaan daripada perangkat lunak jangka penuh berbasis lisensi, maka model SaaS adalah pilihan terbaik.
2. Perangkat lunak untuk perusahaan pemula: Saat menggunakan perangkat lunak tradisional apa pun, pengguna akhir harus membeli perangkat dengan persyaratan minimum yang ditentukan oleh vendor perangkat lunak. Hal ini meningkatkan investasi untuk membeli perangkat keras bagi perusahaan start-up. Karena layanan SaaS tidak memerlukan infrastruktur kelas atas untuk mengaksesnya, ini adalah opsi yang cocok untuk perusahaan pemula yang dapat mengurangi pengeluaran awal untuk membeli perangkat keras kelas atas.
3. Perangkat lunak yang kompatibel dengan banyak perangkat: Beberapa aplikasi seperti pengolah kata atau layanan email memerlukan aksesibilitas yang lebih baik dari perangkat yang berbeda. Aplikasi SaaS dapat disesuaikan dengan hampir semua perangkat.
4. Software dengan beban yang bervariasi: Kami tidak dapat memprediksi beban pada aplikasi populer seperti situs jejaring sosial. Pengguna dapat menghubungkan atau memutuskan sambungan dari aplikasi kapan saja. Sangat sulit untuk menangani berbagai beban dengan infrastruktur tradisional. Dengan kemampuan penskalaan dinamis, aplikasi SaaS dapat menangani berbagai beban secara efisien tanpa mengganggu perilaku normal aplikasi.

Sebagian besar vendor perangkat lunak tradisional pindah ke bisnis SaaS karena merupakan model pengiriman perangkat lunak yang muncul yang menarik pengguna akhir. Namun masih banyak aplikasi tradisional yang tidak memiliki versi SaaS-nya. Ini menyiratkan bahwa aplikasi SaaS mungkin bukan pilihan terbaik untuk semua jenis perangkat lunak. Model pengiriman SaaS bukanlah pilihan terbaik untuk aplikasi yang disebutkan berikut ini:

1. Aplikasi waktu nyata: Karena aplikasi SaaS bergantung pada konektivitas Internet, ini mungkin tidak bekerja lebih baik dengan kecepatan Internet yang rendah. Jika data disimpan jauh dari pengguna akhir, masalah latensi dapat menunda waktu pengambilan data. Aplikasi waktu nyata memerlukan pemrosesan data yang cepat yang mungkin tidak dapat dilakukan dengan aplikasi SaaS karena ketergantungan pada konektivitas Internet berkecepatan tinggi dan masalah latensi.

2. Aplikasi dengan data rahasia: Keamanan data, tata kelola data, dan kepatuhan data selalu menjadi masalah dengan aplikasi SaaS. Karena data disimpan dengan penyedia layanan pihak ketiga, tidak ada jaminan bahwa data kami akan aman. Jika data rahasia yang disimpan hilang, itu akan membuat kerugian serius bagi organisasi. Tidak disarankan menggunakan SaaS untuk aplikasi yang menangani data rahasia.
3. Aplikasi lokal yang lebih baik: Beberapa aplikasi lokal mungkin memenuhi semua persyaratan organisasi. Dalam situasi seperti itu, bermigrasi ke model SaaS mungkin bukan pilihan terbaik.

**Pro dan Kontra SaaS**

Aplikasi SaaS digunakan oleh berbagai individu dan industri pemula untuk keuntungan terkait biaya. Terlepas dari manfaat terkait biaya, layanan SaaS memberikan manfaat berikut:

1. Tidak ada instalasi sisi klien: Layanan SaaS tidak memerlukan instalasi perangkat lunak sisi klien. Pengguna akhir dapat mengakses layanan langsung dari pusat data penyedia layanan tanpa instalasi apa pun. Tidak perlu perangkat keras kelas atas untuk menggunakan layanan SaaS. Itu dapat diakses dari klien tipis atau perangkat genggam apa pun, sehingga mengurangi pengeluaran awal untuk membeli perangkat keras kelas atas.
2. Penghematan biaya: Karena layanan SaaS mengikuti penagihan berbasis utilitas atau penagihan bayar sesuai penggunaan, ini menuntut pengguna akhir untuk membayar apa yang telah mereka gunakan. Sebagian besar penyedia SaaS menawarkan rencana berlangganan yang berbeda untuk menguntungkan pelanggan yang berbeda. Terkadang, layanan SaaS generik seperti pengolah kata diberikan secara gratis kepada pengguna akhir.
3. Lebih sedikit pemeliharaan: Layanan SaaS menghilangkan biaya tambahan pemeliharaan perangkat lunak dari sisi klien. Misalnya, dalam perangkat lunak tradisional, pengguna akhir bertanggung jawab untuk melakukan pembaruan massal. Namun di SaaS, penyedia layanan itu sendiri yang memelihara pembaruan otomatis, pemantauan, dan aktivitas pemeliharaan aplikasi lainnya.
4. Kemudahan akses: Layanan SaaS dapat diakses dari perangkat apapun jika terhubung dengan internet. Aksesibilitas layanan SaaS tidak terbatas pada perangkat tertentu. Ini dapat disesuaikan dengan semua perangkat karena menggunakan UI web responsif.
5. Penskalaan dinamis: Layanan SaaS terkenal dengan penskalaan dinamis elastis. Sangat sulit bagi perangkat lunak lokal untuk menyediakan kemampuan penskalaan dinamis karena memerlukan perangkat keras tambahan. Karena layanan SaaS memanfaatkan sumber daya elastis yang disediakan oleh komputasi awan, ia dapat menangani semua jenis beban yang bervariasi tanpa mengganggu perilaku normal aplikasi.
6. Pemulihan bencana: Dengan mekanisme pencadangan dan pemulihan yang tepat, replika dipertahankan untuk setiap layanan SaaS. Replika didistribusikan di banyak server. Jika ada server yang gagal, pengguna akhir dapat mengakses SaaS dari server lain. Ini menghilangkan masalah satu titik kegagalan. Ini juga memastikan ketersediaan aplikasi yang tinggi.

7. Multitenancy: Multitenancy adalah kemampuan yang diberikan kepada pengguna akhir untuk berbagi satu contoh aplikasi. Multitenancy meningkatkan pemanfaatan sumber daya dari sisi penyedia layanan.

Meskipun layanan SaaS digunakan oleh banyak individu dan industri start-up, adopsi dari industri besar sangat rendah. Masalah utama dengan layanan SaaS adalah keamanan data. Semua perusahaan mengkhawatirkan keamanan data mereka yang dihosting di pusat data penyedia layanan. Berikut ini adalah masalah utama dengan layanan SaaS:

1. Keamanan: Keamanan adalah perhatian utama dalam bermigrasi ke aplikasi SaaS. Karena aplikasi SaaS dibagikan di antara banyak pengguna akhir, ada kemungkinan kebocoran data. Di sini, data disimpan di pusat data penyedia layanan. Kami tidak dapat begitu saja mempercayai beberapa penyedia layanan pihak ketiga untuk menyimpan data sensitif dan rahasia perusahaan kami. Pengguna akhir harus berhati-hati saat memilih penyedia SaaS untuk menghindari kehilangan data yang tidak perlu.
2. Persyaratan konektivitas: Aplikasi SaaS memerlukan konektivitas Internet untuk mengaksesnya. Terkadang, koneksi Internet pengguna akhir mungkin sangat lambat. Dalam situasi seperti itu, pengguna tidak dapat mengakses layanan dengan mudah. Ketergantungan pada koneksi Internet berkecepatan tinggi merupakan masalah utama dalam aplikasi SaaS.
3. Kehilangan kendali: Karena data disimpan di lokasi pihak ketiga dan di luar lokasi, pengguna akhir tidak memiliki kendali apa pun atas data tersebut. Tingkat kontrol atas aplikasi dan data SaaS lebih rendah daripada aplikasi lokal.

**Ringkasan Penyedia SaaS**

Ada banyak penyedia SaaS yang menyediakan layanan SaaS seperti ERP, CRM, penagihan, manajemen dokumen, dan layanan surat. Tabel 5.3 memberikan ringkasan vendor SaaS yang populer di pasar.

**Tabel 5.3** Ringkasan Penyedia SaaS Populer

| **Pemberi** | **Pelayanan yang disediakan** |
|---|---|
| Salseforce.com | Solusi CRM sesuai permintaan |
| aplikasi Google | Gmail, Google Kalender, Talk, Docs, dan Sites |
| Microsoft Office 356 | Office suite online, perangkat lunak, plus layanan |
| NetSuite | ERP, akuntansi, manajemen pesanan, inventaris, CRM, otomatisasi layanan profesional (PSA), dan aplikasi e-niaga |
| Setuju | Solusi manajemen perjalanan dan pengeluaran terintegrasi |
| Pergi rapat | Rapat online, berbagi desktop, dan perangkat lunak konferensi video |
| Kontak konstan | Pemasaran email, pemasaran media sosial, survei online, pemasaran acara, etalase digital, dan alat penawaran lokal |
| Hari Kerja, Inc. | Manajemen modal manusia, penggajian, dan manajemen keuangan |
| Oracle CRM | aplikasi CRM |
| Utuh | Solusi perangkat lunak manajemen keuangan dan akuntansi |

## 5.5 MODEL LAYANAN *CLOUD* LAINNYA

Layanan *cloud* dasar seperti IaaS, PaaS, dan SaaS banyak digunakan oleh banyak perusahaan perorangan dan start-up. Kini, *cloud computing* menjadi teknologi dominan yang menggerakkan dunia IT. Karena ekstensif menggunakan layanan *cloud* dasar, pengguna akhir menyadari pentingnya dan manfaat dari layanan tertentu seperti jaringan, penyimpanan, dan database. Model layanan *cloud* dasar adalah model terpadu yang berisi banyak layanan di dalamnya. Sekarang, harapan pengguna akhir berubah, dan mereka mengharapkan layanan individu ditawarkan oleh penyedia layanan. Ini membuat sebagian besar penyedia layanan memikirkan layanan terpisah yang memenuhi persyaratan pengguna akhir. Banyak penyedia layanan sudah mulai menawarkan layanan terpisah seperti jaringan, desktop, database, dan penyimpanan sesuai permintaan seperti yang diberikan berikut ini:

1. NaaS adalah kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan jaringan virtual yang disediakan oleh penyedia layanan. Seperti model layanan *cloud* lainnya, NaaS juga merupakan model bisnis untuk memberikan layanan jaringan virtual melalui Internet dengan basis bayar per penggunaan. Di pusat data on-premise, industri TI menghabiskan banyak uang untuk membeli perangkat keras jaringan untuk mengelola jaringan internal. Namun, komputasi awan mengubah layanan jaringan menjadi layanan berbasis utilitas. NaaS memungkinkan arsitek jaringan untuk membuat jaringan virtual, kartu antarmuka jaringan virtual (NIC), router virtual, sakelar virtual, dan komponen jaringan lainnya. Selain itu, ini memungkinkan arsitek jaringan untuk menggunakan protokol perutean khusus dan memungkinkan desain layanan dalam jaringan yang efisien, seperti agregasi data, pemrosesan aliran, dan caching. Beberapa layanan populer yang disediakan oleh NaaS termasuk virtual private network (VPN), bandwidth on demand (BoD), dan virtualisasi jaringan seluler.
2. Desktop as a Service (DEaaS) adalah kemampuan yang diberikan kepada pengguna akhir untuk menggunakan virtualisasi desktop tanpa membeli dan mengelola infrastrukturnya sendiri. DEaaS adalah model pengiriman layanan *cloud* bayar per penggunaan di mana penyedia layanan mengelola tanggung jawab back-end penyimpanan data, pencadangan, keamanan, dan pemutakhiran. Pengguna akhir bertanggung jawab untuk mengelola gambar desktop, aplikasi, dan keamanan mereka sendiri. Mengakses desktop virtual yang disediakan oleh penyedia DEaaS tidak bergantung pada perangkat, lokasi, dan jaringan. Layanan DEaaS mudah digunakan, sangat aman, dan menghasilkan pengalaman yang lebih baik di hampir semua perangkat.
3. STaaS adalah kemampuan yang diberikan kepada pengguna akhir untuk menyimpan data pada layanan penyimpanan yang disediakan oleh penyedia layanan. STaaS memungkinkan pengguna akhir untuk mengakses file kapan saja dari mana saja. Penyedia STaaS menyediakan penyimpanan virtual yang dipisahkan dari penyimpanan fisik pusat data *cloud* mana pun. STaaS juga merupakan model bisnis *cloud* yang disampaikan sebagai utilitas. Di sini, pelanggan dapat menyewa penyimpanan dari penyedia STaaS. STaaS umumnya digunakan sebagai penyimpanan cadangan untuk pemulihan bencana yang efisien.

4. DBaaS adalah kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan basis data tanpa perlu menginstal dan memeliharanya. Penyedia layanan bertanggung jawab untuk menginstal dan memelihara database. Pengguna akhir dapat langsung mengakses layanan dan dapat membayar sesuai dengan penggunaannya. DBaaS mengotomatiskan proses administrasi database. Pengguna akhir dapat mengakses layanan database melalui API atau UI web apa pun yang disediakan oleh penyedia layanan. DBaaS memudahkan proses administrasi database. Contoh DBaaS yang populer termasuk SimpleDB, DynamoDB, MongoDB sebagai Layanan, penyimpanan data GAE, dan ScaleDB.
5. Data as a Service (DaaS) adalah kemampuan yang diberikan kepada pengguna akhir untuk mengakses data yang disediakan oleh penyedia layanan melalui Internet. DaaS menyediakan data sesuai permintaan. Data tersebut dapat berupa teks, gambar, suara, dan video. DaaS terkait erat dengan model layanan *cloud* lainnya seperti SaaS dan STaaS. DaaS dapat dengan mudah diintegrasikan dengan SaaS atau STaaS untuk menyediakan layanan komposit. DaaS sangat digunakan dalam layanan data geografi dan layanan data keuangan. Keunggulan DaaS meliputi kelincahan, efektivitas biaya, dan kualitas data.
6. SECaaS adalah kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan keamanan yang disediakan oleh penyedia layanan berdasarkan pembayaran per penggunaan. Di SECaaS, penyedia layanan mengintegrasikan layanan keamanan mereka untuk memberi manfaat bagi pengguna akhir. Secara umum, SECaaS mencakup autentikasi, antivirus, antimalware/spyware, deteksi intrusi, dan manajemen kejadian keamanan. Layanan keamanan yang disediakan oleh penyedia SECaaS biasanya digunakan untuk mengamankan infrastruktur dan aplikasi on-premise atau in-house. Beberapa penyedia SECaaS termasuk Cisco, McAfee, Panda Software, Symantec, Trend Micro, dan VeriSign.
7. IDaaS adalah kemampuan yang diberikan kepada pengguna akhir untuk mengakses infrastruktur otentikasi yang dikelola dan disediakan oleh penyedia layanan pihak ketiga. Pengguna akhir IDaaS biasanya adalah organisasi atau perusahaan. Dengan menggunakan layanan IDaaS, organisasi mana pun dapat dengan mudah mengelola identitas karyawannya tanpa biaya tambahan apa pun. Secara umum, IDaaS mencakup layanan direktori, layanan federasi, pendaftaran, layanan autentikasi, pemantauan risiko dan kejadian, layanan sistem masuk tunggal, serta manajemen identitas dan profil.

Berbagai model layanan baru yang dibahas di bagian ini muncul setelah diperkenalkannya komputasi awan. Bidang ini terus berkembang dan memperkenalkan model layanan baru berdasarkan kebutuhan pengguna akhir. Banyak peneliti dari industri dan akademisi sudah mulai memperkenalkan ide inovatif mereka untuk membawa komputasi awan ke tingkat berikutnya. Terlepas dari model layanan yang dibahas dalam bab ini, para peneliti komputasi awan berpikir untuk menambahkan lebih banyak model layanan. Sekarang, komputasi awan bergerak ke skenario di mana semuanya dapat diberikan sebagai layanan. Ini dapat disebut

sebagai Segalanya sebagai Layanan (XaaS). Di masa mendatang, kami mengharapkan banyak model layanan baru untuk mencapai tujuan XaaS. XaaS dapat mencakup Pencadangan sebagai Layanan (BaaS), Komunikasi sebagai Layanan (CaaS), Hadoop sebagai Layanan (HaaS), Pemulihan Bencana sebagai Layanan (DRaaS), Pengujian sebagai Layanan (TaaS), Firewall sebagai Layanan (FWaaS ), Virtual Private Network as a Service (VPNaaS), Load Balancers as a Service (LBaaS), Message Queue as a Service (MQaaS), dan Monitoring as a Service (MaaS).

## 5.6 RINGKASAN

Komputasi awan terdiri dari tiga model layanan dasar dan empat model penerapan. Model layanannya meliputi IaaS, PaaS, dan SaaS, dan model penerapannya mencakup *cloud* pribadi, publik, komunitas, dan hybrid. Model layanan menentukan jenis layanan yang disediakan oleh masing-masing model layanan. Model penyebaran memutuskan cara memberikan layanan. Komputasi *cloud* menghilangkan banyak overhead manajemen di setiap level. IaaS menyembunyikan kerumitan pemeliharaan perangkat keras yang mendasarinya. PaaS menyembunyikan kerumitan pemeliharaan platform pengembangan dan perangkat keras. Dengan cara yang sama, SaaS menyembunyikan kerumitan pemeliharaan aplikasi, platform pengembangan, dan perangkat keras. Semua model layanan awan dasar memiliki karakteristik penting dari komputasi awan: swalayan sesuai permintaan, akses jaringan luas, pengumpulan sumber daya, elastisitas cepat, dan layanan terukur. Terlepas dari karakteristik tersebut, setiap model layanan memiliki karakteristik uniknya sendiri.

Karakteristik IaaS termasuk akses web ke sumber daya, manajemen terpusat, elastisitas dan penskalaan dinamis, infrastruktur bersama, VM yang telah dikonfigurasi sebelumnya, dan layanan terukur. Karakteristik PaaS mencakup semua dalam satu, akses web ke platform pengembangan, akses offline, skalabilitas bawaan, platform kolaboratif, dan alat klien yang beragam. Karakteristik SaaS meliputi satu ke banyak, akses web, manajemen terpusat, dukungan multidevice, skalabilitas yang lebih baik, ketersediaan tinggi, dan integrasi API. Manfaat berbasis biaya dari layanan *cloud* membuat individu dan industri start-up menggunakan *cloud computing* untuk kebutuhan komputasi mereka. Meskipun layanan *cloud* digunakan oleh banyak individu dan industri pemula, kemampuan beradaptasi dari perusahaan besar sangat rendah. Kami tidak dapat menggunakan layanan *cloud* di semua tempat. Umumnya, layanan *cloud* dapat digunakan di perusahaan start-up yang modal awal investasinya sangat rendah. Layanan *cloud* tidak dapat digunakan saat aplikasi menggunakan data yang lebih sensitif dan rahasia. Manfaat umum dari layanan *cloud* adalah penghematan biaya, penskalaan elastis dan dinamis, dan manajemen terpusat. Kelemahan umum termasuk masalah keamanan, masalah interoperabilitas, dan masalah kinerja. Selain model layanan dasar, ada layanan *cloud* khusus lainnya yang juga disediakan oleh beberapa vendor, termasuk NaaS, STaaS, DBaaS, SECaaS, dan IDaaS.

**Tinjau Poin**

- SaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan perangkat lunak berdasarkan permintaan yang disediakan oleh penyedia SaaS melalui Internet.

- PaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses platform pengembangan dan penyebaran sesuai permintaan yang disediakan oleh penyedia PaaS melalui Internet.
- IaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses sumber daya komputasi on-demand yang disediakan oleh penyedia IaaS melalui Internet.
- NaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses komponen jaringan virtual berdasarkan permintaan yang disediakan oleh penyedia NaaS melalui Internet.
- DEaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan desktop virtual sesuai permintaan yang disediakan oleh penyedia DEaaS melalui Internet.
- STaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses penyimpanan *cloud* sesuai permintaan yang disediakan oleh penyedia STaaS melalui Internet.
- DBaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan basis data berdasarkan permintaan yang disediakan oleh penyedia DBaaS melalui Internet.
- DaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses data sesuai permintaan yang disediakan oleh penyedia DaaS melalui Internet.
- SECaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan keamanan yang disediakan oleh penyedia SECaaS melalui Internet.
- IDaaS: Kemampuan yang diberikan kepada pengguna akhir untuk mengakses layanan manajemen identitas yang disediakan oleh penyedia IDaaS melalui Internet.

**Latihan Soal**

1. Definisikan Infrastruktur sebagai Layanan (IaaS).
2. Tentukan Platform sebagai Layanan (PaaS).
3. Tentukan Perangkat Lunak sebagai Layanan (SaaS).
4. Tulis catatan singkat tentang tanggung jawab pengguna akhir dan penyedia layanan model layanan *cloud* dengan diagram yang sesuai.
5. Tulis catatan singkat tentang penerapan dan pengiriman model layanan *cloud* dengan diagram yang rapi.
6. Jelaskan secara detail tentang gambaran IaaS, PaaS, dan SaaS dengan diagram yang sesuai.
7. Tulis catatan singkat tentang karakteristik IaaS, PaaS, dan SaaS.
8. Jelaskan kesesuaian berbagai model layanan *cloud*.
9. Tulis catatan singkat tentang kelebihan dan kekurangan IaaS, PaaS, dan SaaS.
10. Tulis catatan singkat tentang model layanan *cloud* yang muncul setelah pengenalan *cloud computing*.

# BAB 6
# PENGGERAK TEKNOLOGI UNTUK *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan pembelajaran utama dari bab ini adalah sebagai berikut:

- Untuk mengidentifikasi berbagai driver teknologi paradigma komputasi awan
- Untuk menganalisis setiap teknologi yang mendasari secara rinci untuk memahami fitur karakteristik dan keunggulannya
- Untuk membiasakan diri dengan perkembangan terbaru di masing-masing teknologi yang memungkinkan ini
- Untuk memahami bagaimana masing-masing komponen teknologi ini berkontribusi pada keberhasilan komputasi awan
- Untuk memperkenalkan pembaca pada berbagai studi kasus di bidang ini
- Untuk memahami bagaimana penyedia layanan *cloud* dan konsumen layanan *cloud* diuntungkan dari kemajuan teknologi dalam skenario *cloud* ini

**Pengantar**

Komputasi awan adalah paradigma komputasi yang muncul di mana berbagai pengguna mengakses sumber daya dan layanan yang ditawarkan oleh penyedia layanan. Dari perspektif teknologi, komputasi awan adalah puncak dari banyak komponen yang memungkinkan paradigma komputasi awan saat ini untuk menawarkan layanan secara efisien kepada konsumen. Kemajuan di masing-masing bidang teknologi ini telah memberikan kontribusi yang signifikan terhadap adopsi komputasi awan secara luas. Bab ini berfokus pada pengidentifikasian penggerak teknologi komputasi awan dan juga membahas setiap komponen teknologi secara detail. Kemajuan terbaru dalam masing-masing teknologi ini disorot dengan kelebihan dan fitur karakteristiknya. Pembaca diharapkan mendapatkan pemahaman masing-masing komponen teknologi tersebut, kelebihan dan kekurangannya, serta manfaat yang diberikan oleh penggerak teknologi tersebut kepada berbagai pemangku kepentingan seperti penyedia layanan dan konsumen layanan.

## 6.1 PENDAHULUAN

Komputasi awan memungkinkan penyedia layanan untuk menawarkan berbagai sumber daya seperti infrastruktur, platform, dan perangkat lunak sebagai layanan kepada pengguna yang meminta pada model bayar sesuai pemakaian. Konsumen layanan *cloud* (CSC) diuntungkan dari pengurangan biaya dalam pengadaan sumber daya dan kualitas layanan (QoS) yang ditawarkan oleh paradigma komputasi yang menjanjikan ini. Saat ini, semakin banyak perusahaan dan perusahaan memasuki skenario *cloud* baik sebagai penyedia layanan atau sebagai konsumen layanan. Telah terjadi peningkatan yang cukup besar dalam tingkat adopsi komputasi awan di antara penyedia layanan dan pengguna di seluruh dunia.

Keberhasilan komputasi awan dapat dikaitkan erat dengan peningkatan teknologi di berbagai bidang seperti arsitektur berorientasi layanan (SOA), virtualisasi, teknologi multicore, teknologi memori dan penyimpanan, teknologi jaringan, Web 2.0, dan Web 3.0. Selain itu,

kemajuan dalam model pemrograman, model pengembangan perangkat lunak, komputasi pervasif, sistem operasi (OS), dan lingkungan aplikasi telah berkontribusi pada keberhasilan penerapan berbagai *cloud*.

Bab ini berfokus pada berbagai pendorong teknologi untuk komputasi awan dan juga menunjukkan bagaimana kemajuan terbaru di masing-masing teknologi yang memungkinkan ini berdampak pada adopsi luas komputasi awan. Bab ini juga membahas masing-masing komponen teknologi secara detail. Kemajuan terbaru dalam masing-masing teknologi ini disorot dengan kelebihan dan fitur karakteristiknya. Berbagai manfaat yang diberikan oleh penggerak teknologi ini kepada pemangku kepentingan yang berbeda seperti penyedia layanan dan konsumen layanan ditekankan.

## 6.2 SOA DAN CLOUD

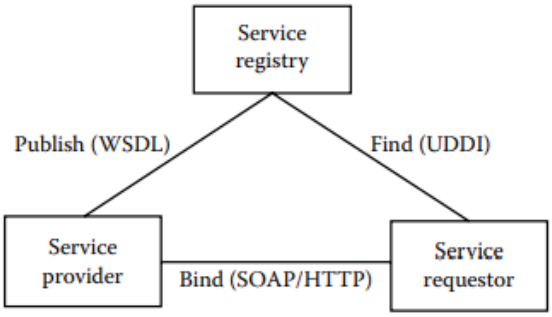
Banyak orang yang bingung mengira bahwa SOA dan *cloud computing* itu sama, atau *cloud computing* adalah nama lain dari SOA. Ini tidak benar. SOA adalah seperangkat prinsip dan standar desain yang fleksibel yang digunakan untuk pengembangan dan integrasi sistem. Sistem berbasis SOA yang diimplementasikan dengan benar menyediakan serangkaian layanan yang digabungkan secara longgar yang dapat digunakan oleh konsumen layanan untuk memenuhi persyaratan layanan mereka dalam berbagai domain bisnis. Komputasi awan adalah model pengiriman layanan di mana layanan dan sumber daya bersama dikonsumsi oleh pengguna di Internet seperti utilitas publik berdasarkan permintaan. Umumnya, SOA digunakan oleh aplikasi perusahaan, dan komputasi awan digunakan untuk memanfaatkan berbagai layanan berbasis Internet.

Perusahaan atau penyedia layanan yang berbeda dapat menawarkan berbagai layanan seperti layanan keuangan, layanan perawatan kesehatan, layanan manufaktur, dan layanan SDM. Berbagai pengguna dapat memperoleh dan memanfaatkan layanan yang ditawarkan melalui Internet. Dalam lingkungan seperti itu, *cloud* adalah tentang layanan dan komposisi layanan. *Cloud* menawarkan berbagai komponen infrastruktur seperti central processing unit (CPU), memori, dan penyimpanan. Ini juga menyediakan berbagai platform pengembangan untuk mengembangkan perangkat lunak yang menawarkan fitur terprogram mereka ke berbagai konsumen *cloud* melalui antarmuka pemrograman aplikasi berorientasi layanan (API). Program yang berjalan di *cloud* dapat diimplementasikan menggunakan teknologi terkait SOA. Pengguna *cloud* dapat menggabungkan layanan yang ditawarkan oleh penyedia layanan *cloud* (CSP) dengan layanan *cloud* internal dan publik lainnya untuk membuat aplikasi komposit berbasis SOA.

**SOA dan SOC**

Paradigma *service-oriented computing* (SOC) memanfaatkan layanan untuk pengembangan cepat dan murah dari aplikasi terdistribusi interoperable di lingkungan yang heterogen. Dalam paradigma ini, layanan adalah entitas otonom dan platform-independen yang dapat dideskripsikan, dipublikasikan, ditemukan, dan digabungkan secara longgar menggunakan berbagai protokol dan spesifikasi. SOC memungkinkan untuk membuat layanan kooperatif yang digabungkan secara longgar, dan layanan ini dapat digunakan untuk membuat proses bisnis yang dinamis dan aplikasi yang gesit dalam platform komputasi yang heterogen.

SOC menggunakan model arsitektur layanan SOA seperti yang ditunjukkan pada Gambar 6.1. Model ini terdiri dari entitas seperti penyedia layanan dan pemohon layanan. Penyedia layanan mempublikasikan detail layanan mereka di registri layanan menggunakan Extensible Markup Language (XML) yang disebut Bahasa Deskripsi Layanan Web (WSDL). Peminta layanan menemukan layanan yang sesuai dari registri layanan menggunakan spesifikasi seperti Deskripsi Universal, Penemuan, dan Integrasi (UDDI). Penyedia layanan dan pemohon layanan berkomunikasi satu sama lain menggunakan protokol seperti *Simple Object Access Protocol* (SOAP). SOAP memungkinkan sebuah program atau layanan yang berjalan pada satu platform untuk berkomunikasi dengan program atau layanan lain yang berjalan pada platform yang berbeda, menggunakan *Hypertext Transfer Protocol* (HTTP) dan XML-nya sebagai mekanisme pertukaran informasi.




**Gambar 6.1** Layanan model arsitektur SOA.

**Manfaat SOA**

SOA memungkinkan pertukaran data timbal balik antara program dari vendor yang berbeda tanpa perlu pemrograman tambahan atau perubahan pada layanan. Layanan harus independen, dan harus memiliki antarmuka standar yang dapat dipanggil untuk melakukan tugasnya dengan cara standar. Selain itu, layanan tidak perlu memiliki pengetahuan sebelumnya tentang aplikasi pemanggilan, dan aplikasi tidak perlu memiliki pengetahuan tentang bagaimana tugas dilakukan oleh layanan. Oleh karena itu, berbagai manfaat SOA adalah sebagai berikut:

1. Penggunaan kembali layanan: Berbagai layanan dapat digunakan kembali oleh aplikasi yang berbeda, yang menghasilkan biaya pengembangan dan pemeliharaan yang lebih rendah. Memiliki layanan yang dapat digunakan kembali tersedia juga menghasilkan waktu yang lebih cepat ke pasar.
2. Kelincahan: SOA dapat menghadirkan ketangkasan arsitektur dalam suatu perusahaan melalui penggunaan standar yang luas seperti layanan web. Ini adalah kemampuan untuk mengubah proses bisnis dengan cepat bila diperlukan untuk mendukung perubahan dalam aktivitas bisnis. Aspek ketangkasan ini membantu menangani perubahan sistem menggunakan lapisan konfigurasi alih-alih mengembangkan kembali sistem secara terus-menerus.
3. Pemantauan: Membantu memantau kinerja berbagai layanan untuk membuat perubahan yang diperlukan.

4. Jangkauan yang diperluas: Dalam kolaborasi antar perusahaan atau dalam kasus proses bersama, ini adalah kemampuan untuk mendapatkan layanan dari berbagai proses lain untuk menyelesaikan tugas tertentu.

Oleh karena itu, SOA dapat digunakan sebagai teknologi yang memungkinkan untuk memanfaatkan komputasi awan. Komputasi awan menawarkan sumber daya teknologi informasi (TI) sesuai permintaan yang dapat dimanfaatkan dengan memperluas SOA di luar firewall perusahaan ke domain CSP. Ini adalah proses SOA menggunakan komputasi awan.

**Teknologi yang Digunakan oleh SOA**

Ada banyak teknologi dan standar yang digunakan oleh SOA, yang juga dapat dimanfaatkan dalam domain komputasi awan untuk memberikan layanan secara efisien kepada pelanggan awan. Beberapa standar atau protokol diberikan sebagai berikut:

1. Layanan web: Layanan web dapat mengimplementasikan SOA. Layanan web membuat komponen atau layanan fungsional tersedia untuk akses melalui Internet, terlepas dari platform dan bahasa pemrograman yang digunakan. Spesifikasi UDDI mendefinisikan cara untuk mempublikasikan dan menemukan informasi tentang layanan web.
2. SOAP: Protokol SOAP digunakan untuk menggambarkan protokol komunikasi.
3. RPC: Remote procedure call (RPC) adalah protokol yang membantu suatu program untuk meminta layanan dari program lain yang berada di komputer lain dalam jaringan, tanpa perlu memahami detail jaringan.
4. RMI-IIOP: Ini menunjukkan antarmuka pemanggilan metode jarak jauh Java (RMI) melalui Internet Inter-ORB Protocol (IIOP). Protokol ini digunakan untuk mengirimkan kapabilitas komputasi terdistribusi Common Object Request Broker Architecture (CORBA) ke platform Java. Ini mendukung banyak platform dan bahasa pemrograman dan dapat digunakan untuk mengeksekusi RPC di komputer lain seperti yang didefinisikan oleh RMI.
5. REST: REpresentational State Transfer (REST) adalah arsitektur tanpa kewarganegaraan yang berjalan melalui HTTP. Ini digunakan untuk interaksi yang efektif antara klien dan layanan.
6. DCOM: Distributed Component Object Model (DCOM) adalah kumpulan konsep Microsoft dan antarmuka program di mana program klien dapat meminta layanan dari program server yang berjalan di komputer lain dalam jaringan. DCOM didasarkan pada Model Objek Komponen (COM).
7. WCF (Microsoft implementasi layanan web merupakan bagian dari WCF): Windows Communication Foundation (WCF) menyediakan satu set API di .NET Framework untuk membangun aplikasi berorientasi layanan yang terhubung.

**Persamaan dan Perbedaan antara SOA dan *Cloud computing***

Ada beberapa fitur umum tertentu yang SOA dan *cloud computing* bagikan sementara berbeda satu sama lain di area tertentu lainnya.

***Persamaan***

Baik komputasi awan dan SOA berbagi beberapa prinsip inti. Pertama, keduanya mengandalkan konsep layanan untuk mencapai tujuan. Layanan adalah fungsionalitas atau fitur yang ditawarkan oleh satu entitas dan digunakan oleh entitas

lain. Misalnya, layanan dapat mengambil detail rekening bank online pengguna. SOA dan komputasi awan menggunakan pendelegasian layanan karena tugas yang diperlukan didelegasikan ke penyedia layanan (dalam kasus komputasi awan) atau ke aplikasi lain atau komponen bisnis di perusahaan (dalam kasus SOA). Pendelegasian layanan membantu orang untuk menggunakan layanan tanpa khawatir tentang detail implementasi dan pemeliharaan. Layanan dapat dibagi oleh beberapa aplikasi dan pengguna, sehingga mencapai pemanfaatan sumber daya yang dioptimalkan. Kedua, komputasi awan dan SOA mempromosikan sambungan longgar di antara komponen atau layanan, yang memastikan ketergantungan minimum di antara berbagai bagian sistem. Fitur ini mengurangi dampak perubahan tunggal apa pun pada satu bagian sistem terhadap kinerja sistem secara keseluruhan. Kopling longgar membantu layanan yang diimplementasikan dipisahkan dan tidak mengetahui teknologi, topologi, siklus hidup, dan organisasi yang mendasarinya. Berbagai format dan protokol yang digunakan dalam komputasi terdistribusi, seperti XML, WSDL, Interface Description Language (IDL), dan Common Data Representation (CDR), membantu mencapai enkapsulasi perbedaan teknologi dan heterogenitas di antara berbagai komponen yang digunakan untuk menggabungkan - ing solusi bisnis untuk memecahkan masalah komputasi. Berbagai layanan harus lokasi dan teknologi independen dalam komputasi awan, dan SOA dapat digunakan untuk mencapai transparansi ini dalam domain awan.

***Perbedaan***

Ada juga beberapa perbedaan antara SOA dan paradigma *cloud computing*. Layanan dalam SOA terutama berfokus pada bisnis. Setiap layanan dalam SOA dapat mewakili salah satu aspek dari proses bisnis. Layanan dapat digabungkan bersama untuk menyediakan aplikasi bisnis lengkap atau solusi bisnis yang diperlukan. Karenanya, dalam pengertian ini, layanan bersifat horizontal. Pada saat yang sama, berbagai layanan dalam komputasi awan biasanya berlapis seperti infrastruktur, platform, atau perangkat lunak, dan layanan lapisan bawah mendukung layanan atas untuk mengirimkan aplikasi. Oleh karena itu, layanan dalam hal ini bersifat vertikal. SOA digunakan untuk mendefinisikan arsitektur aplikasi. Berbagai komponen atau layanan aplikasi dibagi berdasarkan perannya dalam aplikasi SOA. Itu berarti solusi untuk masalah bisnis dapat dicapai dengan menggabungkan berbagai layanan abstrak yang menjalankan fungsi yang diperlukan. Layanan dalam SOA dapat digunakan kembali oleh aplikasi lain. Komputasi awan adalah mekanisme untuk memberikan layanan TI. Berbagai layanan tersebut dapat dibagi atau dikelompokkan berdasarkan perannya seperti infrastruktur, platform, atau perangkat lunak. Dalam hal ini, untuk memanfaatkan layanan *cloud*, konsumen tidak memerlukan masalah sebelum menentukan layanan *cloud*. Layanan dalam hal ini juga dapat digunakan kembali oleh aplikasi lain.

**Bagaimana SOA Memenuhi *Cloud computing***

SOA secara luas dianggap sebagai teknologi yang memungkinkan untuk komputasi awan. Dalam kasus *cloud computing*, membutuhkan enkapsulasi tingkat tinggi. Seharusnya tidak ada ketergantungan yang kuat pada lokasi sumber daya untuk mencapai virtualisasi dan

elastisitas yang sebenarnya di *cloud*. Juga, utas eksekusi dari berbagai pengguna harus diisolasi dengan benar di *cloud*, karena kerentanan apa pun akan mengakibatkan informasi atau data dari satu pengguna bocor ke konsumen lain. Standar layanan web (WS*) yang digunakan dalam SOA juga digunakan dalam domain komputasi awan untuk menyelesaikan berbagai masalah, seperti perpesanan asinkron, pertukaran metadata, dan penanganan acara. SOA adalah gaya arsitektur yang benar-benar agnostik terhadap standar teknologi yang diadopsi dalam perakitan aplikasi komposit.

Orientasi layanan yang disediakan oleh SOA membantu dalam desain perangkat lunak menggunakan perangkat lunak yang berbeda, masing-masing menyediakan fungsionalitas aplikasi terpisah sebagai layanan untuk aplikasi lain. Fitur ini tidak bergantung pada platform, vendor, atau teknologi apa pun. Layanan dapat digabungkan dengan aplikasi perangkat lunak lain untuk menyediakan fungsionalitas lengkap dari aplikasi perangkat lunak besar. SOA membuat kerja sama komputer yang terhubung melalui jaringan menjadi mudah. Sejumlah layanan dapat dijalankan di komputer, dan setiap layanan dapat berkomunikasi dengan layanan lain di jaringan tanpa interaksi manusia dan juga tanpa perlu melakukan modifikasi apa pun pada program yang mendasarinya. Dalam SOA, layanan menggunakan protokol yang ditentukan untuk mentransfer dan menafsirkan pesan. WSDL digunakan untuk menggambarkan layanan. Protokol SOAP digunakan untuk menggambarkan protokol komunikasi.

SOA adalah sebuah arsitektur, dan *cloud computing* adalah turunan dari arsitektur atau pilihan arsitektur, bukan arsitektur itu sendiri. Ketika digunakan dengan komputasi awan, SOA membantu untuk memberikan sumber daya TI sebagai layanan melalui Internet, dan untuk mencampur dan mencocokkan sumber daya untuk memenuhi kebutuhan bisnis. Di perusahaan, database dapat dihosting dengan satu CSP, server proses dengan CSP lain, platform pengembangan aplikasi dengan CSP lain, dan server web dengan CSP lain. Itu berarti SOA dapat diperluas ke penyedia komputasi awan untuk memberikan solusi hemat biaya sedemikian rupa sehingga sumber daya berbasis *cloud* dan sumber daya lokal bekerja bersama-sama. SOA menggunakan arsitektur komputasi awan memberikan kelincahan sedemikian rupa sehingga dapat dengan mudah diubah untuk menggabungkan kebutuhan bisnis karena menggunakan layanan yang dikonfigurasi melalui lapisan konfigurasi atau proses.

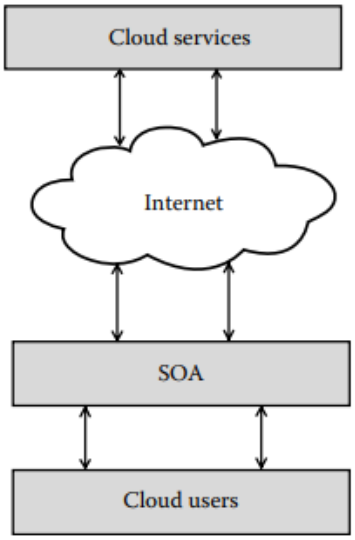
*Cloud* dan SOA dianggap saling melengkapi. SOA adalah gaya arsitektur untuk membangun aplikasi yang digabungkan secara longgar dan memungkinkan komposisi lebih lanjut. Ini juga membantu dalam menciptakan layanan yang dibagikan dan digunakan kembali. Komputasi awan menyediakan pengulangan dan akses mudah standar ke berbagai perangkat keras dan perangkat lunak bersama dengan biaya rendah.

Faktanya, ia menawarkan sejumlah kemampuan X sebagai Layanan. SOA dan *cloud* bersama-sama menyediakan solusi berbasis layanan lengkap yang diperlukan. Oleh karena itu, *cloud* dan SOA diperlukan untuk bekerja bersama untuk memberikan visibilitas layanan dan manajemen layanan. Visibilitas dan tata kelola layanan memberi pengguna fungsionalitas penemuan layanan di dalam *cloud*, dan manajemen layanan SOA membantu dalam mengelola siklus hidup layanan yang tersedia di *cloud*. Dengan demikian, melalui integrasi *cloud* dan SOA,

*cloud* dapat memanfaatkan pendekatan tata kelola SOA tanpa perlu menciptakan overhead tata kelola baru. Memiliki SOA dan orientasi layanan, perusahaan atau organisasi dapat mengadopsi layanan *cloud* dengan lebih mudah dan tidak terlalu rumit, karena lingkungan komputasi *cloud* juga didasarkan pada layanan. Baik *cloud* maupun SOA berfokus untuk memberikan layanan ke bisnis dengan peningkatan kelincahan, kecepatan, dan efektivitas biaya.

Multitenancy adalah fitur karakteristik dari sistem *cloud computing*. Perlu dicatat bahwa ini adalah fitur yang dimiliki oleh sistem berbasis SOA. Dalam aplikasi multitenant, CSP memiliki satu contoh program atau aplikasi yang berjalan di server, dan lebih dari satu pelanggan pada satu waktu menggunakan salinan aplikasi tersebut. Contohnya adalah program Gmail atau Hotmail. Multitenancy meningkatkan efisiensi sistem *cloud*. Dalam aplikasi penyewa tunggal, hanya satu pengguna pada satu waktu yang menggunakan aplikasi yang disediakan oleh penyedia layanan. Contohnya bisa berupa aplikasi editor teks yang digunakan oleh satu pengguna pada satu waktu. Di *cloud*, aplikasi multitenant lebih disukai karena membantu pemanfaatan sumber daya secara efektif. CSC harus memercayai CSP bahwa ia akan menjalankan tugas atau fungsi yang dimaksudkan tanpa gagal.

SOA dianggap sebagai arsitektur yang ideal untuk komputasi awan. Pindah ke lingkungan *cloud* membutuhkan SOA yang solid untuk menyediakan infrastruktur yang diperlukan untuk implementasi *cloud* yang sukses. Komputasi awan adalah arsitektur penyebaran, dan SOA adalah pendekatan arsitektur untuk bagaimana merancang TI perusahaan. Oleh karena itu, *cloud* memerlukan orientasi layanan yang disediakan oleh SOA. Dengan SOA yang telah diterapkan dan dijalankan dengan sukses, memanfaatkan sepenuhnya komputasi awan menjadi andal, lebih cepat, dan lebih aman. Gambar 6.2 menunjukkan bagaimana SOA dapat diperluas untuk menggunakan layanan *cloud*. Manfaat yang dirasakan dari integrasi ini dapat meningkatkan kolaborasi, kepuasan pelanggan, dan pertumbuhan bisnis. Dengan memanfaatkan SOA untuk menerapkan kemampuan bisnis ke lingkungan *cloud*, bisnis dapat sangat meningkatkan interaksi dengan mitra bisnis dan pelanggan yang sudah ada, sehingga meningkatkan pendapatan mereka.




**Gambar 6.2** Konvergensi SOA dan *cloud*.

**CCOA**

*Cloud computing Open Architecture* (CCOA) adalah arsitektur untuk lingkungan *cloud* yang menggabungkan SOA. Tujuan dari CCOA adalah sebagai berikut:

1. Untuk mengembangkan arsitektur yang dapat digunakan kembali dan dapat diskalakan. Artinya, di masa mendatang, arsitektur harus memasukkan perubahan lebih lanjut tanpa perlu mengganti keseluruhan arsitektur.
2. Mengembangkan platform yang seragam untuk pengembangan aplikasi *cloud*. Ini akan memungkinkan pengguna *cloud* untuk beralih di antara CSP tanpa perlu membuat perubahan signifikan dalam aplikasi.
3. Agar bisnis dapat berjalan secara efisien. Sasaran ini membantu CSP menghasilkan lebih banyak uang dengan memberikan layanan berkualitas dengan sukses.

Oleh karena itu, CCOA memberikan panduan dan instruksi umum yang diperlukan untuk desain dan pengembangan aplikasi *cloud* yang dapat diskalakan, dapat digunakan kembali, dan dapat dioperasikan dengan menggabungkan prinsip SOA ke dalamnya.

## 6.3 VIRTUALISASI

Virtualisasi adalah teknologi inti yang mendasari komputasi awan. Ini membantu dalam membuat model multitenant untuk lingkungan *cloud* dengan mengoptimalkan penggunaan sumber daya melalui. Manfaat virtualisasi termasuk biaya yang lebih rendah dan masa pakai teknologi yang lebih lama, yang menjadikannya pilihan populer dengan usaha kecil hingga menengah. Menggunakan virtualisasi, infrastruktur fisik yang dimiliki oleh penyedia layanan dibagi di antara banyak pengguna, meningkatkan pemanfaatan sumber daya. Virtualisasi memberikan pemanfaatan sumber daya yang efisien dan peningkatan laba atas investasi (ROI). Pada akhirnya, ini menghasilkan pengeluaran modal (CapEx) dan pengeluaran operasional (OpEx) yang rendah.

Beberapa manfaat virtualisasi termasuk tingkat pemanfaatan sumber daya penyedia layanan yang lebih baik, peningkatan ROI untuk penyedia layanan dan konsumen, dan mempromosikan TI ramah lingkungan dengan mengurangi pemborosan energi. Teknologi virtualisasi memiliki kelemahan berupa kemungkinan satu titik kegagalan perangkat lunak mencapai virtualisasi dan overhead kinerja seluruh sistem akibat virtualisasi.

**Pendekatan dalam Virtualisasi**

Ada banyak pendekatan yang diadopsi dalam implementasi teknologi virtualisasi. Beberapa pendekatan penting dibahas dalam subbagian berikut.

**Virtualisasi Penuh**

Virtualisasi penuh menggunakan jenis perangkat lunak khusus yang disebut hypervisor. Hypervisor berinteraksi langsung dengan sumber daya perangkat keras server fisik, seperti CPU dan ruang penyimpanan, dan bertindak sebagai platform untuk OS server virtual. Ini membantu menjaga setiap server virtual benar-benar independen dan tidak mengetahui server virtual lain yang berjalan di mesin fisik. Setiap server tamu atau mesin virtual (VM) dapat menjalankan OS-nya sendiri. Itu berarti satu server virtual dapat berjalan di Linux dan yang lainnya dapat berjalan di Windows. Contohnya termasuk VMWare ESX dan VirtualBox. Dalam virtualisasi penuh, OS tamu

tidak mengetahui infrastruktur perangkat keras yang mendasarinya. Itu berarti OS tamu tidak menyadari fakta bahwa itu berjalan pada platform virtual dan perasaan bahwa itu berjalan pada perangkat keras yang sebenarnya. Dalam hal ini, OS tamu tidak dapat berkomunikasi langsung dengan infrastruktur fisik yang mendasarinya. OS membutuhkan bantuan hypervisor perangkat lunak virtualisasi untuk berkomunikasi dengan infrastruktur yang mendasarinya. Keuntungan dari virtualisasi penuh termasuk isolasi di antara berbagai VM, isolasi antara VM dan hypervisor, eksekusi beberapa OS secara bersamaan, dan tidak ada perubahan yang diperlukan di OS tamu. Kerugiannya adalah kinerja sistem secara keseluruhan dapat terpengaruh karena terjemahan biner.

**Paravirtualisasi**

Dalam hal ini, VM tidak mensimulasikan perangkat keras yang mendasarinya, dan ini menggunakan API khusus yang harus digunakan oleh OS tamu yang dimodifikasi. Contohnya termasuk server Xen dan VMWare ESX. Dalam jenis virtualisasi ini, simulasi parsial infrastruktur perangkat keras yang mendasari tercapai. Ini juga dikenal sebagai virtualisasi parsial atau virtualisasi yang dibantu OS. Virtualisasi ini berbeda dari virtualisasi penuh, di sini, OS tamu menyadari fakta bahwa ia berjalan di lingkungan virtual. Dalam hal ini, hypercall digunakan untuk komunikasi langsung antara OS tamu dan hypervisor. Dalam paravirtualisasi, OS tamu yang dimodifikasi atau diparavirtualisasi diperlukan.

Keuntungan dari pendekatan ini adalah meningkatkan kinerja sistem secara keseluruhan dengan menghilangkan overhead terjemahan biner. Kerugiannya adalah modifikasi OS tamu diperlukan.

**Virtualisasi Berbantuan Perangkat Keras**

Dalam jenis virtualisasi ini, produk perangkat keras yang mendukung virtualisasi digunakan. Vendor perangkat keras seperti Intel dan AMD telah mengembangkan prosesor yang mendukung virtualisasi melalui ekstensi perangkat keras. Intel telah merilis prosesornya dengan teknologi virtualisasi VT-x, dan AMD telah merilis prosesornya dengan teknologi virtualisasi AMD-v untuk mendukung virtualisasi tersebut. Keuntungan dari pendekatan ini adalah menghilangkan overhead terjemahan biner dan paravirtualisasi. Sebuah dis-keuntungan termasuk kurangnya dukungan dari semua vendor.

**Hypervisor dan Perannya**

Konsep penggunaan VM meningkatkan pemanfaatan sumber daya di lingkungan komputasi awan. Hypervisor adalah alat perangkat lunak yang digunakan untuk membuat VM, dan mereka menghasilkan virtualisasi berbagai sumber daya perangkat keras seperti CPU, penyimpanan, dan perangkat jaringan. Mereka juga disebut monitor mesin virtual (VMM) atau manajer virtualisasi. Mereka membantu dalam virtualisasi pusat data *cloud* (DC). Berbagai hypervisor yang digunakan adalah VMware, Xen, Hyper-V, KVM, dll. Hypervisor membantu menjalankan banyak OS secara bersamaan pada sistem fisik yang berbagi perangkat kerasnya. Dengan demikian, hypervisor memungkinkan banyak OS untuk berbagi satu host perangkat keras. Dalam hal ini, setiap OS tampaknya memiliki prosesor host, memori, dan sumber daya lain yang dialokasikan hanya untuk itu. Namun, hypervisor sebenarnya mengendalikan

prosesor dan sumber daya host dan pada gilirannya mengalokasikan apa yang dibutuhkan untuk setiap OS. Hypervisor juga memastikan bahwa OS tamu (disebut VM) tidak mengganggu satu sama lain. Dalam teknologi virtualisasi, hypervisor mengelola beberapa OS atau beberapa instance dari OS yang sama pada satu sistem komputer fisik. Hypervisor dirancang agar sesuai dengan prosesor tertentu, dan mereka juga disebut manajer virtualisasi. Hypervisor terutama terdiri dari dua jenis:

1. Hypervisor tipe 1: Hypervisor jenis ini berjalan langsung di perangkat keras komputer host untuk mengontrol sumber daya perangkat keras dan juga mengelola OS tamu. Ini juga dikenal sebagai hypervisor asli atau bare-metal. Contohnya termasuk VMware ESXi, Citrix XenServer, dan hypervisor Microsoft Hyper-V.
2. Hypervisor tipe 2: Hypervisor jenis ini berjalan dalam lingkungan OS formal. Dalam tipe ini, hypervisor berjalan sebagai lapisan kedua yang berbeda sementara OS tamu berjalan sebagai lapisan ketiga di atas perangkat keras. Ini juga dikenal sebagai hypervisor yang dihosting. Contohnya termasuk VMware Workstation dan VirtualBox.

**Jenis Virtualisasi**

Bergantung pada sumber daya yang divirtualisasi, proses virtualisasi dapat diklasifikasikan ke dalam jenis berikut.

**Virtualisasi OS**

Dalam virtualisasi OS, OS utama desktop dipindahkan ke lingkungan virtual. Komputer yang digunakan oleh konsumen layanan tetap berada di meja mereka, tetapi OS dihosting di server di tempat lain. Biasanya, ada satu versi OS di server, dan salinan dari OS individual tersebut diberikan kepada pengguna individual. Berbagai pengguna kemudian dapat memodifikasi OS sesuai keinginan, tanpa mempengaruhi pengguna lain.

**Virtualisasi Server**

Dalam virtualisasi server, server fisik yang ada dipindahkan ke lingkungan virtual, yang kemudian dihosting di server fisik. Server modern dapat menghosting lebih dari satu server secara bersamaan, yang memungkinkan pengguna mengurangi jumlah server yang akan dipesan untuk berbagai keperluan. Oleh karena itu, pengeluaran TI dan administrasi berkurang. Virtualisasi server dapat menggunakan prosesor virtual yang dibuat dari prosesor perangkat keras nyata yang ada di sistem host. Prosesor fisik dapat diabstraksi menjadi kumpulan prosesor virtual yang dapat digunakan bersama oleh VM yang dibuat.

**Virtualisasi Memori**

Dalam virtualisasi memori utama, memori utama virtual yang dipisahkan dari memori fisik dialokasikan ke berbagai VM untuk memenuhi kebutuhan memorinya. Pemetaan memori fisik ke virtual dilakukan oleh perangkat lunak hypervisor. Dukungan virtualisasi memori utama disediakan dengan prosesor x86 modern. Juga, konsolidasi memori utama di *cloud* DC tervirtualisasi dapat dilakukan oleh hypervisor dengan menggabungkan segmen memori bebas dari berbagai server untuk membuat kumpulan memori virtual yang dapat digunakan oleh VM.

**Virtualisasi Penyimpanan**

Dalam virtualisasi penyimpanan, beberapa hard drive fisik digabungkan menjadi satu lingkungan penyimpanan virtual. Untuk berbagai pengguna, ini hanya disebut penyimpanan *cloud*, dan ini bisa berupa penyimpanan pribadi, seperti yang dihosting oleh perusahaan, atau penyimpanan publik, yang dihosting di luar perusahaan seperti DropBox, atau pendekatan campuran dari keduanya. Dalam kasus virtualisasi penyimpanan, disk penyimpanan fisik diabstraksi ke media penyimpanan virtual. Di *cloud* DC, ketersediaan tinggi dan pencadangan data pengguna dicapai melalui teknologi virtualisasi penyimpanan. Hypervisor modern membantu mencapai virtualisasi penyimpanan. Konsep virtualisasi penyimpanan diimplementasikan dalam teknik penyimpanan tingkat lanjut seperti jaringan area penyimpanan (SAN) dan penyimpanan yang terpasang di jaringan (NAS).

**Virtualisasi Jaringan**

Dalam virtualisasi jaringan (NV), jaringan virtual logis dibuat dari jaringan fisik yang mendasarinya. Komponen jaringan fisik seperti router, switch, atau kartu antarmuka jaringan dapat divirtualisasi oleh hypervisor untuk membuat komponen ekuivalen logis. Beberapa jaringan virtual dapat dibuat dengan menggunakan komponen jaringan fisik yang sama yang dapat digunakan untuk berbagai tujuan. NV juga dapat dicapai dengan menggabungkan berbagai komponen jaringan dari beberapa jaringan.

**Virtualisasi Aplikasi**

Dalam virtualisasi aplikasi, satu aplikasi yang diinstal pada server pusat divirtualisasi dan berbagai komponen aplikasi yang divirtualisasikan akan diberikan kepada pengguna yang meminta layanan. Dalam hal ini, aplikasi diberikan salinan komponennya sendiri seperti file registri sendiri dan objek global yang tidak dibagikan dengan orang lain. Lingkungan virtual mencegah konflik dalam penggunaan sumber daya. Contohnya adalah Java Virtual Machine (JVM). Di lingkungan komputasi awan, CSP menghadirkan model SaaS melalui teknologi virtualisasi aplikasi. Dalam kasus virtualisasi aplikasi, pengguna *cloud* tidak diharuskan untuk menginstal aplikasi yang diperlukan pada sistem masing-masing. Mereka bisa, pada gilirannya, mendapatkan salinan aplikasi virtual, dan menyesuaikan serta menggunakannya untuk tujuan mereka sendiri.

## 6.4 TEKNOLOGI MULTICORE

Dalam teknologi multicore, dua atau lebih CPU bekerja bersama pada chip yang sama. Dalam jenis arsitektur ini, satu prosesor fisik berisi logika inti dari dua atau lebih prosesor. Prosesor ini dikemas ke dalam satu sirkuit terpadu (IC). IC tunggal ini disebut die. Teknologi multicore juga dapat merujuk pada beberapa cetakan yang dikemas bersama. Teknologi ini memungkinkan sistem untuk melakukan lebih banyak tugas dengan kinerja sistem keseluruhan yang lebih besar. Ini juga membantu dalam mengurangi konsumsi daya dan mencapai pemrosesan beberapa tugas yang lebih efisien dan simultan. Teknologi multicore dapat digunakan di desktop, komputer pribadi seluler (PC), server, dan workstation. Oleh

karena itu, teknologi ini digunakan untuk mempercepat pemrosesan di lingkungan *cloud* multitenant. Arsitektur multicore telah menjadi tren terkini dari prosesor berperforma tinggi, dan berbagai hasil teoretis dan studi kasus menggambarkan bahwa arsitektur multicore dapat diskalakan dengan jumlah core.

**Prosesor Multicore dan Skalabilitas VM**

Dalam sistem berbasis prosesor multicore untuk *cloud*, arsitektur komputer canggih digunakan untuk memungkinkan beberapa VM untuk menskalakan selama batas cache, memori, bus, dan bandwidth jaringan tidak tercapai. Dengan demikian, dalam domain *cloud computing*, CPU dan beban kerja tervirtualisasi intensif-memori harus ditingkatkan hingga batas maksimum yang ditentukan oleh arsitektur memori.

**Teknologi Multicore dan Paralelisme di *Cloud***

Dalam chip multicore, beberapa prosesor yang lebih sederhana dikerahkan alih-alih satu prosesor besar, dan paralelisme menjadi terbuka bagi pemrogram. Dalam desain prosesor, pengembangan prosesor multicore telah menjadi perkembangan arsitektur yang signifikan akhir-akhir ini. Untuk mengeksploitasi desain arsitektur ini sepenuhnya, perangkat lunak yang berjalan pada perangkat keras ini perlu menunjukkan perilaku bersamaan. Fakta ini lebih menekankan pada prinsip, teknik, dan teknologi konkurensi. Arsitektur multicore mendominasi *cloud* server saat ini karena meningkatkan kecepatan dan efisiensi pemrosesan di *cloud*. Dalam hal ini, masalah utamanya adalah seberapa efisien chip multi-core digunakan di *cloud* server saat ini untuk mendapatkan paralelisme nyata dalam hal peningkatan kinerja di lingkungan *cloud* multitenant melalui pemrograman yang efisien.

**Studi Kasus**

Pembuat chip Intel telah meluncurkan keluarga sistem pada chip (SoC) generasi kedua untuk mikroserver. Produk SoC Intel Atom C2000 64-bit dirancang untuk server mikro serta platform penyimpanan dan jaringan. Ini cocok untuk jaringan yang ditentukan perangkat lunak (SDN). Server mikro adalah mesin kecil dan hemat daya yang dirancang untuk menangani beban kerja ringan seperti hosting web tingkat pemula. Mereka cocok untuk pekerjaan kecil atau sementara tanpa menyediakan sumber daya dari server kelas atas kontemporer. Atom C2000 didasarkan pada mikroarsitektur Silvermont dan ditujukan untuk meningkatkan kinerja dan efisiensi energi. Ini fitur hingga delapan core, hingga 20 W TDP (Thermal Design Power), Ethernet terintegrasi dan mendukung hingga 64 GB memori. Ini memenuhi kebutuhan khusus untuk mengamankan dan merutekan lalu lintas Internet dengan lebih efisien. Perusahaan layanan hosting web 1&1 telah menguji Atom C2000 sebagai percontohan dan berencana untuk menerapkan chip tersebut dalam layanan hosting khusus tingkat awal. Penyedia telekomunikasi Ericsson juga akan menambahkan Atom C2000 SoC ke platform layanan *cloud*-nya.

## 6.5 TEKNOLOGI MEMORI DAN PENYIMPANAN

Pada domain penyimpanan terlihat bahwa pertumbuhan data berbasis file lebih cepat dibandingkan data berbasis blok. Selain itu, pertumbuhan data tidak terstruktur lebih cepat daripada data terstruktur. Sebagian besar data organisasi tidak terstruktur, dan lebih dari 50% kebutuhan penyimpanan baru dikonsumsi oleh data tidak terstruktur seperti email, pesan

instan, pemutar MP3 identifikasi frekuensi radio (RFID), foto, pencitraan medis, citra satelit, dan GPS. Oleh karena itu, solusi memori atau penyimpanan yang digunakan di lingkungan *cloud* harus mendukung persyaratan *cloud*. Penyimpanan *cloud* harus berurusan dengan berbagai jenis data seperti gambar medis, MP3, foto, pencitraan definisi tinggi 3D, streaming video, tangkapan kamera pengintai, dan animasi film.

**Persyaratan Penyimpanan Awan**

Teknologi atau solusi penyimpanan yang digunakan di lingkungan *cloud* harus memenuhi persyaratan berikut.

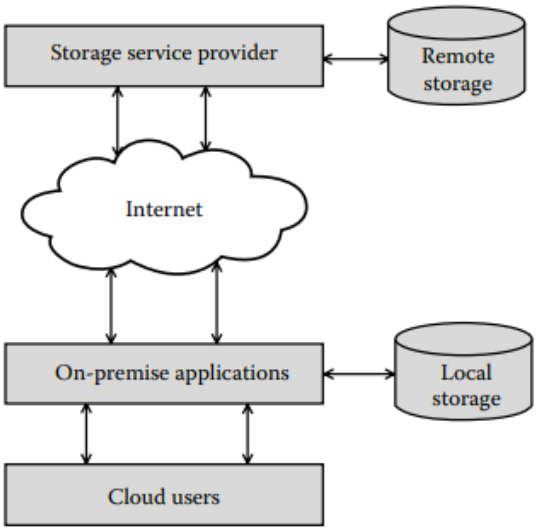
1. Skalabilitas: Sistem penyimpanan harus mendukung skalabilitas data pengguna.
2. Ketersediaan tinggi: Tingkat ketersediaan solusi penyimpanan yang diterapkan di *cloud* harus sangat tinggi.
3. Bandwidth tinggi: Sistem penyimpanan *cloud* harus mendukung kecepatan transfer data cepat yang diperlukan.
4. Performa konstan: Seharusnya tidak ada masalah performa yang terkait dengan sistem penyimpanan *cloud*, dan performa harus konsisten selama periode kontrak.
5. Load balancing (LB): Untuk mencapai penggunaan sumber daya yang efektif, sistem penyimpanan yang diterapkan di *cloud* harus cukup cerdas untuk mendukung LB otomatis dari data pengguna.

**Dukungan Virtualisasi**

Dalam virtualisasi penyimpanan, beberapa perangkat penyimpanan jaringan digabung menjadi satu unit penyimpanan. Jenis virtualisasi ini sering digunakan di SAN, yang merupakan jaringan perangkat penyimpanan bersama berkecepatan tinggi, dan teknologi SAN membuat tugas seperti pengarsipan, pencadangan, dan proses pemulihan menjadi lebih mudah dan lebih cepat. Aplikasi perangkat lunak digunakan untuk mengimplementasikan virtualisasi penyimpanan. Virtualisasi penyimpanan melibatkan penyatuan penyimpanan fisik dari berbagai perangkat penyimpanan jaringan menjadi satu perangkat penyimpanan logis, dan dikelola dari konsol terpusat. Virtualisasi penyimpanan membantu dalam mencapai proses pencadangan, pengarsipan, dan pemulihan yang mudah dan efisien.

**Penyimpanan sebagai Layanan (STAaS)**

Penyimpanan *cloud* dapat bersifat internal untuk organisasi atau dapat berupa penyimpanan eksternal di mana penyimpanan tersebut disediakan oleh CSP yang terletak di luar DC organisasi, yang juga dikenal sebagai Storage as a Service (STaaS). STaaS adalah model bisnis *cloud* di mana penyedia layanan menyewakan ruang dalam infrastruktur penyimpanannya ke berbagai pengguna *cloud*. Gambar 6.3 menunjukkan model *cloud* SaaS. Pelanggan STaaS dapat memiliki penghematan biaya yang signifikan dalam perangkat keras, pemeliharaan, dll. Penyedia STaaS setuju dalam perjanjian tingkat layanan (SLA) untuk menyewa ruang penyimpanan dengan biaya per gigabyte yang disimpan dan biaya per data dasar pengalihan. STaaS juga membantu dalam pencadangan, pemulihan bencana, kelangsungan bisnis, dan ketersediaan. Keuntungan lain dari STaaS adalah kemampuannya untuk mengakses data yang disimpan di *cloud* dari mana saja. Di sini, penyimpanan dikirimkan sesuai permintaan.




**Gambar 6.3** Penyimpanan sebagai Layanan.

**Tren dan Teknologi yang Muncul di *Cloud Storage***

Teknologi memori dan penyimpanan telah berkembang dengan sangat cepat, dan teknologi yang muncul memungkinkan *cloud* memiliki sistem penyimpanan yang andal, aman, dan dapat diskalakan. Berikut ini adalah beberapa perkembangan dalam teknologi memori dan penyimpanan yang membantu sistem *cloud* mencapai efisiensinya:

- HDD hibrid dengan memori magnetik dan flash yang memiliki cache tingkat kedua
- Perkembangan teknologi RAID seperti RAID 6, RAID triple parity, erasure coding + RAIN
- Penyatuan data blok, file, dan konten dalam satu subsistem penyimpanan
- Deduplikasi tertanam, pengurangan penyimpanan utama di unit penyimpanan
- Penggunaan perangkat penyimpanan berbasis objek (OSD)
- D-RAM SSD dan flash HDD memiliki fitur pemanfaatan server yang lebih baik, konsumsi energi yang jauh lebih sedikit, kurang sensitif terhadap getaran, dll.
- Virtualisasi file atau clustered NAS, yang mendukung namespace tunggal untuk melihat semua file, skala hampir linier dengan menambahkan node, ketersediaan dan kinerja yang lebih baik, dan LB. Pendekatan ini paling cocok untuk pemrosesan seismik, perenderan video, simulasi, desain otomatis/aero/elektronik, dll.

Model terbaru peralatan penyimpanan awan seperti peralatan penyimpanan awan Whitewater dapat mendukung hingga 14,4 petabyte data logis. Ini memberikan kapasitas yang lebih besar, kecepatan lebih cepat, dan lebih banyak opsi replikasi. Arsitektur backup disk-to-disk telah menjadi sangat populer dalam beberapa tahun terakhir. Menambahkan kemampuan untuk mengintegrasikan penyimpanan *cloud* publik ke dalam arsitektur ini menawarkan pengembalian langsung ke operasi di situs pemulihan bencana sambil menangkap keuntungan biaya dari layanan penyimpanan *cloud* yang sangat agresif seperti Amazon Glacier.

## 6.6 TEKNOLOGI JARINGAN

Di *cloud*, fitur jaringan harus mendukung interaksi yang efektif antara CSP dan CSC.

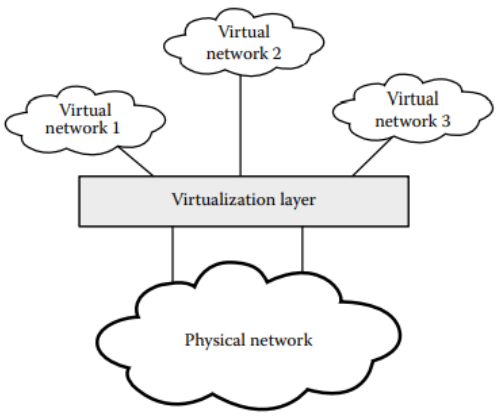
**Persyaratan Jaringan untuk *Cloud***

Berbagai persyaratan jaringan di lingkungan *cloud* adalah sebagai berikut:

1. Mengkonsolidasikan beban kerja dan menyediakan Infrastruktur sebagai Layanan (IaaS) ke berbagai penyewa: Teknologi jaringan harus memungkinkan konsolidasi beban kerja perusahaan, yang mengurangi biaya pengelolaan dan menyediakan lebih banyak fleksibilitas dan skalabilitas untuk pengelolaan VM.
2. Menyediakan konektivitas VM ke jaringan fisik dan virtual: Sistem jaringan *cloud* harus mendukung fitur yang dapat diperluas dan dikelola secara terprogram untuk menghubungkan berbagai VM ke jaringan virtual dan jaringan fisik. Itu juga harus menegakkan penegakan kebijakan untuk keamanan, isolasi, dan tingkat layanan.
3. Pastikan konektivitas dan kelola bandwidth jaringan: Sistem jaringan *cloud* harus memiliki fitur *load balancing and failover* (LBFO) yang memungkinkan agregasi bandwidth server dan *failover* lalu lintas.
4. Kecepatan aplikasi dan kinerja server: Jaringan dan kinerja OS harus ditingkatkan dengan fitur jaringan berkecepatan tinggi. Berbagai fitur dan teknologi seperti teknologi *low-latency*, Data Center Bridging, dan Data Center Transmission Control Protocol (DCTCP) dapat dimanfaatkan untuk meningkatkan kinerja infrastruktur jaringan *cloud*.

**Dukungan Virtualisasi**

Di jaringan NV, sumber daya digunakan melalui segmentasi logis dari satu jaringan fisik. NV dapat dicapai dengan menggunakan perangkat lunak dan layanan untuk mengelola berbagi penyimpanan, siklus komputasi, dan berbagai aplikasi. Ini menganggap semua server dan layanan dalam jaringan sebagai satu kumpulan sumber daya logis yang dapat diakses tanpa mempertimbangkan komponen fisik. Oleh karena itu, NV menciptakan logis, jaringan virtual yang dipisahkan atau dipisahkan dari perangkat keras jaringan fisik yang mendasarinya untuk memastikan bahwa jaringan terintegrasi lebih baik dengan lingkungan virtual. Dalam hal ini, perangkat jaringan fisik hanya bertanggung jawab untuk meneruskan paket data, sedangkan jaringan virtual yang dikontrol perangkat lunak menyediakan abstraksi yang diperlukan yang memudahkan penerapan dan pengelolaan layanan jaringan dan sumber daya jaringan yang mendasarinya. Gambar 6.4 menunjukkan dukungan virtualisasi dalam jaringan.




**Gambar 6.4** Virtualisasi dalam jaringan.

NV membantu dalam pemanfaatan sumber daya jaringan secara efisien melalui segmentasi logis dari satu jaringan fisik. Jenis virtualisasi ini dapat digunakan oleh unit organisasi atau departemen yang berbeda dalam sebuah perusahaan untuk berbagi jaringan fisik perusahaan sebagai jaringan logis yang terpisah. Di sini, total biaya belanja modal dan biaya operasional dikurangi dengan berbagi berbagai sumber daya jaringan. Di NV, pemisahan logis yang aman antara organisasi atau grup dipertahankan. Segmentasi logis dari jaringan fisik menjadi jaringan virtual yang aman dapat dilakukan dengan melapisi mekanisme jaringan pribadi virtual (VPN) seperti 802.1x, kontrol penerimaan jaringan (NAC), terowongan enkapsulasi perutean generik (GRE), perutean virtual dan forwarding (VRF)-lite, dan multiprotocol label switching (MPLS) VPN ke jaringan area lokal (LAN) yang ada.

**Penggunaan Jaringan Virtual**

NV dapat digunakan untuk membuat jaringan virtual dalam infrastruktur virtual. Ini membantu NV mendukung kebutuhan sumber daya yang kompleks di lingkungan *cloud* multitenant. NV dapat memfasilitasi jaringan virtual dalam lingkungan virtual yang dipisahkan dari sumber daya jaringan lainnya. Dalam lingkungan virtual ini, NV dapat memisahkan lalu lintas data ke dalam berbagai zona atau kontainer untuk memastikan bahwa lalu lintas tersebut tidak tercampur dengan sumber daya lain atau transfer data lain.

**DC dan VPLS**

DC yang benar-benar tervirtualisasi harus mendukung migrasi VM yang efektif. Dalam lingkungan DC tervirtualisasi, VM dapat dipindahkan dari satu server ke server lain untuk meningkatkan tingkat pemanfaatan server, dan juga untuk mencapai LB yang efektif. Layanan LAN pribadi virtual (VPLS) menyediakan jaringan datar dan jangkauan panjang yang dapat digunakan untuk menghubungkan server yang terpisah secara geografis, sehingga mencapai migrasi VM dan LB.

**SDN**

SDN adalah pendekatan jaringan di mana kontrol dipisahkan dari perangkat keras jaringan dan diberikan ke aplikasi perangkat lunak yang disebut pengontrol. Dalam jaringan konvensional, ketika paket data tiba di sebuah switch, aturan yang dibangun di dalam firmware switch memandu switch dalam meneruskan paket data. Dalam jaringan yang ditentukan perangkat lunak, administrator jaringan dapat mengontrol lalu lintas dari konsol kontrol terpusat tanpa harus berurusan dengan sakelar individu. Administrator dapat mengubah aturan sakelar jaringan bila diperlukan seperti memblokir jenis paket tertentu. Hal ini sangat membantu dalam lingkungan komputasi awan yang memiliki arsitektur multitenant karena memberikan fleksibilitas dan efisiensi yang diperlukan bagi administrator untuk mengelola beban lalu lintas. Saat ini, jaringan yang ditentukan perangkat lunak dapat dibuat menggunakan spesifikasi standar terbuka seperti OpenFlow, yang memungkinkan administrator jaringan untuk mengontrol tabel perutean dari jarak jauh. Ini dapat digunakan di lingkungan *cloud* untuk aplikasi yang memerlukan jaminan bandwidth dan penundaan. Intel telah meluncurkan silikon Ethernet Switch FM5224 yang dirancang untuk memfasilitasi SDN dan untuk meningkatkan kepadatan komputasi dan efisiensi daya.

**MPLS**

MPLS adalah mekanisme switching dalam jaringan telekomunikasi berkinerja tinggi yang mengarahkan data dari satu node jaringan ke yang lain berdasarkan label jalur pendek daripada alamat jaringan yang panjang. MPLS mengatur jalur khusus untuk paket data tertentu yang diidentifikasi oleh label yang diletakkan di setiap paket. Berbagai label mengidentifikasi tautan atau jalur virtual antara node yang jauh daripada titik akhir. Ini mengurangi waktu yang dibutuhkan router untuk mencari alamat ke node berikutnya dalam penerusan data. MPLS disebut multiprotocol karena dapat digunakan untuk mengenkapsulasi paket data dari berbagai protokol dan teknologi jaringan seperti Internet Protocol (IP), Asynchronous Transport Mode (ATM), dan protokol jaringan frame relay. Dengan demikian, MPLS meningkatkan kecepatan jaringan dan pengelolaannya.

**Tren dan Teknologi Jaringan Lain yang Muncul di *Cloud***

Untuk bandwidth tinggi dan latensi sangat rendah dalam komputasi awan, kepadatan gelombang multiplexing divisi (DWDM) tampaknya sangat menjanjikan sebagai teknologi transportasi WAN berkinerja tinggi di masa depan. DWDM adalah protokol dan kecepatan bit independen, dan juga dapat melipatgandakan beberapa sinyal optik. Para peneliti bekerja dalam teknologi jaringan wide area network (WAN) baru seperti jaringan Lambda yang menjanjikan sirkuit berbiaya rendah dan berkapasitas tinggi dalam jaringan. Di dalam *cloud* DC, berbagai server jaringan, sistem penyimpanan, node jaringan, dan elemen lainnya saling terhubung. Tiga teknologi jaringan LAN, yaitu Ethernet, FiberChannel, dan InifiBand, digunakan di DC. Sepuluh GB Ethernet (10GBE) peralatan dan jaringan banyak digunakan. FiberChannel cocok untuk komputasi ilmiah dan SAN, dan InifiBand hampir secara eksklusif digunakan dalam jaringan simulasi ilmiah dan teknik, misalnya, menggunakan server berkerumun.

*Cloud* DC besar mendukung puluhan ribu server, penyimpanan exabyte, lalu lintas terabit per detik, dan puluhan ribu penyewa. Di DC, server dan sumber daya penyimpanan saling terhubung dengan sakelar paket dan router yang menyediakan bandwidth dan kebutuhan jaringan virtual multitenant. DC saling terhubung di WAN melalui perutean dan teknologi transportasi untuk menyediakan kumpulan sumber daya, yang dikenal sebagai *cloud*. Antarmuka optik berkecepatan tinggi dan transpor optik DWDM digunakan untuk menyediakan transpor intra dan antar DC berkapasitas tinggi.

DC pribadi saling terhubung melalui jaringan pribadi atau VPN khusus perusahaan. DC publik terhubung melalui Internet, dan mereka menawarkan layanan berbasis Internet multitenant. DC pribadi virtual dapat dibangun menggunakan infrastruktur DC umum yang disediakan oleh penyedia IaaS. Jaringan yang mendasari untuk DC pribadi virtual menyediakan isolasi jaringan penyewa dan fitur privasi. Layanan *cloud* dapat dibangun dengan menghubungkan DC dan memanfaatkan kumpulan sumber daya kolektif di DC ini sebagai kumpulan sumber daya.

Teknologi virtualisasi digunakan untuk membuat VM di server dengan siklus CPU khusus, memori, penyimpanan, dan bandwidth input/output (I/O). Beberapa VM untuk berbagai penyewa dapat dibuat di server fisik yang sama. Seorang penyewa di lingkungan *cloud* juga dapat menyediakan satu set VM yang berada di server yang didistribusikan di

seluruh DC atau bahkan di berbagai DC. NV digunakan sebagai teknologi yang berkembang untuk membuat jaringan intra dan antar-DC dari jaringan area lokal virtual dasar (VLAN) dan arsitektur perutean IP. Ini membantu mendukung sejumlah besar penyewa dan memungkinkan jaringan di antara sumber daya virtual mereka. Dalam kasus kebutuhan bandwidth yang besar seperti berurusan dengan blok data dan video yang besar di antara DC, interkoneksi DC dapat menggunakan mekanisme transportasi optik. Berbagai layanan jaringan seperti firewall (FW), layanan LB, dan terjemahan alamat jaringan (NAT) disediakan sebagai bagian dari jaringan virtual penyewa.

Teknologi VPN digunakan untuk membawa lalu lintas pelanggan Ethernet dan IP melintasi jaringan penyedia layanan IP/MPLS menggunakan teknologi tunneling seperti IP atau MPLS. Ini membantu mencapai isolasi di antara pelanggan, dan teknologi ini dapat digunakan di domain *cloud*. Pada shared IP/ MPLS packet-switched network (PSN), VPLS dapat dimanfaatkan untuk menyediakan layanan LAN yang transparan. Border Gateway Protocol (BGP) dan MPLS IP VPN digunakan untuk menyediakan perutean IP pelanggan pribadi melalui PSN IP/MPLS bersama. Ethernet VPN (EVPN) adalah teknologi baru yang bertujuan untuk menyediakan layanan Ethernet LAN melalui IP/MPLS PSN. Dalam lingkungan komputasi awan, sakelar yang lebih besar, antarmuka Ethernet berkecepatan lebih tinggi, dan transportasi DWDM mengatasi kebutuhan bandwidth. Juga, teknologi paket yang berkembang seperti VXLAN serta Ethernet dan IP VPN digunakan untuk menciptakan paradigma jaringan DC generasi berikutnya. Berbagai protokol, standar, dan teknologi jaringan berkembang pesat untuk memenuhi persyaratan jaringan antar dan intra-DC di domain *cloud*.

## 6.7 WEB 2.0

Web 2.0 (atau Web 2) adalah istilah populer yang diberikan untuk teknologi dan aplikasi Internet canggih yang mencakup blog, wiki, really simple syndication (RSS), dan bookmark sosial. Dua kontributor utama Web 2.0 adalah kemajuan teknologi yang dimungkinkan oleh Ajax dan aplikasi lain seperti RSS dan Eclipse yang mendukung interaksi pengguna dan pemberdayaan mereka dalam menangani web. Istilah ini diciptakan oleh Tim O'Reilly, setelah konferensi yang membahas konsep dan isu web generasi mendatang yang diadakan oleh O'Reilly Media dan MediaLive International pada tahun 2004. Salah satu perbedaan paling signifikan antara Web 2.0 dan World Wide Web tradisional Web (disebut sebagai Web 1.0) adalah bahwa Web 2.0 memfasilitasi kolaborasi yang lebih besar dan berbagi informasi di antara pengguna Internet, penyedia konten, dan perusahaan. Oleh karena itu, dalam hal ini, ini dapat dianggap sebagai migrasi dari web hanya-baca ke web baca/tulis.

Sebagai contoh untuk paradigma ini, outlet buku online multi-vendor seperti BookFinder4U memungkinkan pengguna mengunggah resensi buku ke situs dan juga membantu pengguna menemukan buku langka dan tidak dicetak lagi dengan harga yang wajar. Dalam contoh lain, ensiklopedia dinamis seperti Wikipedia memungkinkan pengguna tidak hanya membaca informasi yang tersimpan tetapi juga membuat dan mengedit isi database informasi dalam berbagai bahasa. Forum internet seperti blogging telah menjadi lebih populer dan luas dan telah menyebabkan proliferasi dan berbagi informasi dan

pandangan. Juga, umpan RSS telah digunakan untuk penyebaran berita di seluruh pengguna dan situs web.

Fokus utama Web 2.0 adalah untuk menyediakan pengguna web kemampuan untuk berbagi dan mendistribusikan informasi secara online dengan pengguna dan situs lain. Ini mengacu pada transisi dari halaman web HTML statis ke web yang lebih dinamis untuk melayani aplikasi web kepada pengguna secara efektif. Situs Web 2.0 seperti situs jejaring sosial memungkinkan penggunanya untuk berinteraksi satu sama lain dalam dialog media sosial, berbeda dengan situs web di mana orang dibatasi untuk melihat informasi secara pasif. Contoh umum Web 2.0 termasuk situs jejaring sosial, blog, wiki, situs berbagi video, dan layanan host atau aplikasi web lainnya yang memungkinkan berbagi informasi secara dinamis di antara pengguna. Selain itu, teknologi Web 2.0 dapat digunakan sebagai alat interaktif untuk memberikan umpan balik tentang konten atau informasi yang diberikan di halaman web seperti praktik terbaik dan pembaruan terkini. Umpan balik ini akan membantu penyedia layanan untuk meningkatkan kualitas layanan mereka dan dengan demikian nilai bisnis.

Komputasi awan terkait erat dengan SOA Web 2.0 dan virtualisasi sumber daya perangkat keras dan perangkat lunak. Komputasi awan memungkinkan untuk membangun aplikasi dan layanan yang dapat dijalankan/dieksekusi menggunakan sumber daya (perangkat keras dan perangkat lunak) yang disediakan oleh penyedia layanan, tanpa membatasi pengembang aplikasi atau konsumen pada sumber daya yang tersedia di tempat.

**Karakteristik Web 2.0**

Di situs web Web 2.0, sebagaimana telah disebutkan, alih-alih hanya membaca konten dari halaman web, pengguna diizinkan untuk menulis atau berkontribusi pada konten yang tersedia untuk semua orang dengan cara yang efektif dan mudah digunakan. Web 2.0 juga disebut jaringan sebagai platform komputasi karena menyediakan perangkat lunak, komputasi, dan fasilitas penyimpanan kepada pengguna di seluruh browser. Aplikasi utama Web 2.0 meliputi situs jejaring sosial, platform penerbitan sendiri, penandaan, dan bookmark sosial.

Fitur utama dari Web 2.0 meliputi berikut ini:

- Folksonomi
- Pengalaman pengguna yang kaya
- Pengguna sebagai kontributor
- Partisipasi pengguna
- Dispersi

Folksonomy memungkinkan klasifikasi informasi gratis yang tersedia di web, yang membantu pengguna untuk mengklasifikasikan dan menemukan informasi secara kolektif menggunakan pendekatan seperti penandaan. Pengalaman pengguna yang kaya disediakan karena konten dinamis yang ditawarkan di web yang responsif terhadap masukan pengguna dengan cara yang mudah digunakan. Web 2.0 memungkinkan pengguna web untuk berperan sebagai kontributor informasi sebagai aliran informasi dalam dua cara, yaitu antara pemilik situs dan pengguna situs melalui evaluasi, tinjauan, dan umpan balik. Paradigma ini juga memfasilitasi partisipasi pengguna karena pengguna situs diizinkan untuk menambahkan konten untuk dilihat orang lain (misalnya, crowdsourcing). Kontribusi yang dibuat oleh masing-masing

pengguna tersedia untuk digunakan dan digunakan kembali oleh pengguna lain sebagai konten web. Juga, banyak saluran digunakan untuk pengiriman konten di antara pengguna.
Beberapa fitur karakteristik Web 2.0 adalah sebagai berikut:

1. Blogging: Blogging memungkinkan pengguna membuat posting ke web log atau blog. Blog adalah jurnal, buku harian, atau situs web pribadi yang dikelola di Internet, dan sering diperbarui oleh pengguna. Blog meningkatkan interaktivitas pengguna dengan menyertakan fitur seperti komentar dan tautan.
2. Penggunaan Ajax dan teknologi baru lainnya: Ajax adalah cara mengembangkan aplikasi web yang menggabungkan presentasi berbasis standar XHTML dan CSS. Ini memungkinkan interaksi dengan halaman web melalui DOM dan pertukaran data dengan XML dan XSLT.
3. Sindikasi yang dihasilkan RSS: RSS adalah format untuk mensindikasikan konten web. Ini memungkinkan untuk memberi makan konten web yang baru diterbitkan kepada pengguna melalui pembaca/agregator RSS.
4. Bookmark sosial: Bookmark sosial adalah sistem taksonomi yang ditentukan pengguna untuk menyimpan tag ke konten web. Taksonomi ini juga disebut peopleonomy, dan bookmark disebut sebagai tag. Alih-alih menyimpan bookmark dalam folder di komputer pengguna, halaman yang diberi tag disimpan di web meningkatkan aksesibilitas dari komputer mana pun yang terhubung ke Internet.
5. Mash-up: Mash-up adalah halaman web atau aplikasi yang dapat mengintegrasikan informasi dari dua sumber atau lebih. Metodologi pengembangan menggunakan Ajax dapat digunakan untuk membuat mash-up. Ini membantu untuk membuat web yang lebih interaktif dan partisipatif dengan konten dan layanan yang ditentukan pengguna terintegrasi.

**Perbedaan antara Web 1.0 dan Web 2.0**

Beberapa perbedaan antara Web 1.0 dan Web 2.0 ditunjukkan pada Tabel 6.1.

**Tabel 6.1** Perbedaan antara Web 1.0 dan Web 2.0

| **Fitur** | **Web 1.0** | **Web 2.0** |
|---|---|---|
| Mekanisme penulisan | Situs web pribadi | Blogging |
| Sumber informasi | Inggris | Wikipedia daring |
| Pembuatan dan pemeliharaan konten | Melalui CMS | Melalui wiki |
| Penyimpanan data | Disk lokal | Disk daring |
| Iklan online | Spanduk | Google AdSense |
| Pembayaran daring | akun bank | PayPal |

**Aplikasi Web 2.0**

Web 2.0 menemukan aplikasi di berbagai bidang. Beberapa aplikasi Web 2.0 dibahas dalam subbagian berikut.

**Media Sosial**

Web sosial adalah aplikasi penting dari Web 2.0 karena memberikan perubahan mendasar dalam cara orang berkomunikasi dan berbagi informasi. Web

sosial menawarkan sejumlah alat dan platform online yang dapat digunakan oleh pengguna untuk berbagi data, perspektif, dan pendapat mereka di antara komunitas pengguna lainnya.

**Pemasaran**

Web 2.0 menawarkan peluang bagus untuk pemasaran dengan melibatkan pelanggan dalam berbagai tahap siklus pengembangan produk. Ini memungkinkan pemasar untuk berkolaborasi dengan konsumen dalam berbagai aspek seperti pengembangan produk, peningkatan layanan, dan promosi. Kolaborasi dengan mitra bisnis dan konsumen dapat ditingkatkan oleh perusahaan dengan memanfaatkan alat yang disediakan oleh paradigma Web 2.0. Perusahaan yang berorientasi konsumen menggunakan jaringan seperti Twitter, Yelp, dan Facebook sebagai elemen umum dari promosi multisaluran produk mereka. Jejaring sosial menjadi lebih intuitif dan user friendly dan dapat dimanfaatkan untuk menyebarluaskan informasi produk sehingga dapat menjangkau sebanyak mungkin calon konsumen produk secara efisien.

**Pendidikan**

Teknologi Web 2.0 dapat membantu skenario pendidikan dengan memberikan siswa dan fakultas lebih banyak kesempatan untuk berinteraksi dan berkolaborasi dengan rekan-rekan mereka. Penemuan pengetahuan yang efektif dimungkinkan dengan fitur yang ditawarkan oleh Web 2.0 seperti kustomisasi dan pilihan topik yang lebih besar, dan lebih sedikit gangguan dari rekan-rekan mereka. Dengan memanfaatkan alat-alat Web 2.0, para siswa mendapat kesempatan untuk berbagi apa yang mereka pelajari dengan rekan-rekan lain dengan cara berkolaborasi dengan mereka.

**Web 2.0 dan Komputasi Awan**

Di Web 2.0, metadata yang mendeskripsikan konten web ditulis dalam bahasa seperti XML, yang dapat dibaca dan diproses oleh komputer secara otomatis. Berbagai protokol web berbasis XML seperti SOAP, WSDL, dan UDDI membantu mengintegrasikan aplikasi yang dikembangkan menggunakan bahasa pemrograman yang berbeda menggunakan platform komputasi dan OS yang heterogen. Mengandalkan kemampuan integrasi data dan pertukaran data lintas aplikasi yang heterogen, model bisnis baru pengembangan aplikasi, penyebaran, dan pengiriman melalui Internet telah dikonseptualisasikan dan diimplementasikan. Itu berarti aplikasi dapat dihosting di web dan diakses oleh klien yang terpisah secara geografis melalui Internet. Layanan web adalah aplikasi atau layanan yang dapat dioperasikan yang dihosting di web untuk penggunaan jarak jauh oleh banyak klien dengan platform heterogen, dan mereka bahkan dapat ditemukan secara dinamis dengan cepat tanpa mengetahui keberadaannya sebelumnya.

Dalam model bisnis komputasi awan, infrastruktur pengembangan aplikasi seperti prosesor, penyimpanan, memori, OS, dan alat pengembangan aplikasi serta perangkat lunak dapat diakses oleh klien sebagai layanan melalui Internet dalam model bayar per penggunaan. Dalam model pengiriman layanan ini, kumpulan besar sumber daya fisik yang dihosting di web oleh penyedia layanan akan digunakan bersama oleh banyak klien jika diperlukan. Komputasi

awan didasarkan pada SOA Web 2.0 dan virtualisasi sumber daya perangkat keras dan perangkat lunak yang disimpan oleh penyedia layanan. Oleh karena itu, komputasi awan dianggap sebagai masa depan komputasi Internet karena keuntungan yang ditawarkan oleh model bisnis ini seperti tidak ada pengeluaran modal, kecepatan penerapan aplikasi, waktu pemasaran yang lebih singkat, biaya operasi yang lebih rendah, dan pemeliharaan sumber daya yang lebih mudah untuk klien.

Keberhasilan jejaring sosial online dan fungsionalitas Web 2.0 lainnya mendorong banyak aplikasi SaaS untuk menawarkan fitur yang memungkinkan penggunanya bekerja sama, serta mendistribusikan dan berbagi data dan informasi. Komputasi awan adalah platform yang membantu individu dan perusahaan untuk mengakses perangkat keras, perangkat lunak, dan sumber daya data menggunakan Internet untuk sebagian besar kebutuhan komputasi mereka.

## 6.8 WEB 3.0

Nama Web 3.0 diberikan oleh John Markoff dari The New York Times kepada web generasi ketiga ini. Dua generasi pertama web disebut Web 1.0 dan Web 2.0. Ketiga teknologi tersebut dapat dijelaskan secara singkat sebagai berikut:

***Web 1.0:*** Web 1.0 adalah generasi pertama Web di mana fokus utamanya adalah membangun web, membuatnya dapat diakses, dan juga mengkomersialkannya. Bidang minat utama di Web 1.0 termasuk protokol seperti HTTP, bahasa markup standar terbuka seperti HTML dan XML, akses Internet melalui ISP, browser web pertama, platform dan alat untuk pengembangan web, bahasa perangkat lunak pengembangan web seperti Java dan Javascript, dan komersialisasi web.

***Web 2.0:*** Ungkapan Web 2.0 diciptakan oleh O'Reilly dan mengacu pada layanan berbasis Internet generasi kedua, seperti situs jejaring sosial, wiki, dan alat komunikasi, yang memfasilitasi kolaborasi online dan berbagi di antara berbagai pengguna.

***Web 3.0:*** John Markoff dari The New York Times menciptakan istilah Web 3.0 dan mengacu pada generasi ketiga dari layanan berbasis Internet yang secara kolektif disebut web cerdas. Web 3.0 mencakup layanan di Internet yang menggunakan teknologi seperti web semantik, pencarian bahasa alami, pembelajaran mesin, agen rekomendasi, dan kecerdasan buatan untuk mencapai pemahaman informasi yang difasilitasi mesin untuk memberikan pengalaman yang lebih produktif dan intuitif kepada pengguna web.

Teknologi web 2.0 memungkinkan penggunaan web baca/tulis, blog, aplikasi web interaktif, rich media, tagging atau folksonomy sambil berbagi konten, dan juga situs jejaring sosial yang berfokus pada komunitas. Pada saat yang sama, standar Web 3.0 menggunakan teknologi web semantik, drag and drop mash-up, widget, perilaku pengguna, keterlibatan pengguna, dan konsolidasi konten web dinamis tergantung pada minat masing-masing pengguna. Web 3.0 menggunakan teknologi Web Data, yang menampilkan rekaman data yang dapat diterbitkan dan digunakan kembali di web melalui format yang dapat dikueri seperti Resource Description Framework (RDF), XML, dan mikroformat. Ini adalah komponen penting yang memfasilitasi web semantik, yang memungkinkan tingkat baru interoperabilitas aplikasi dan integrasi data

di antara berbagai aplikasi dan layanan, dan juga membuat data dapat ditautkan dan diakses secara dinamis dalam bentuk halaman web. Tahap web semantik lengkap memperluas cakupan konten terstruktur dan tidak terstruktur melalui penggunaan Web Ontology Language (OWL) dan semantik RDF.

Standar Web 3.0 juga menggabungkan penelitian terbaru di bidang kecerdasan buatan. Penggunaan teknologi yang luas segera terlihat dalam kasus aplikasi yang membuat prediksi lagu hit berdasarkan umpan balik pengguna dari situs web musik yang dihosting oleh berbagai perguruan tinggi di Internet. Web 3.0 mencapai kecerdasan secara organik melalui interaksi pengguna web. Dengan demikian, Web 3.0 memungkinkan aplikasi untuk berpikir sendiri dengan data yang tersedia untuk membuat keputusan tertentu, dan juga memungkinkan untuk menghubungkan satu aplikasi ke aplikasi lainnya secara dinamis tergantung pada konteks penggunaan. Contoh aplikasi Web 3.0 tipikal adalah yang menggunakan sistem manajemen konten bersama dengan kecerdasan buatan. Sistem ini mampu menjawab pertanyaan yang diajukan oleh pengguna, karena aplikasi mampu berpikir sendiri dan menemukan jawaban yang paling mungkin, tergantung pada konteksnya, atas permintaan yang diajukan oleh pengguna. Dengan cara ini, Web 3.0 juga dapat digambarkan sebagai standar mesin untuk pengguna di Internet.

**Komponen Web 3.0**

Istilah Web 3.0, juga dikenal sebagai web semantik, menggambarkan situs di mana komputer akan menghasilkan data mentah sendiri tanpa interaksi pengguna langsung. Web 3.0 dianggap sebagai langkah logis berikutnya dalam evolusi Internet dan teknologi web. Untuk Web 1.0 dan Web 2.0, Internet dibatasi dalam dinding fisik komputer, tetapi karena semakin banyak perangkat seperti telepon pintar, mobil, dan peralatan rumah tangga lainnya yang terhubung ke web, Internet akan ada di mana-mana dan dapat dapat dimanfaatkan dengan cara yang paling efisien. Dalam hal ini, berbagai perangkat akan dapat bertukar data satu sama lain dan mereka bahkan akan menghasilkan informasi baru dari data mentah (misalnya, situs musik, Last.fm, akan dapat mengantisipasi jenis musik yang disukai pengguna tergantung pada pilihan lagu sebelumnya). Oleh karena itu, Internet akan dapat melakukan tugas-tugas pengguna dengan cara yang lebih cepat dan lebih efisien, seperti kasus mesin pencari yang dapat mencari minat sebenarnya dari pengguna individu dan tidak hanya berdasarkan kata kunci yang diketikkan. mesin pencari. Web 3.0 menyematkan kecerdasan di seluruh domain web. Ini menyebarkan robot web yang cukup pintar dalam mengambil keputusan tanpa adanya campur tangan pengguna. Jika Web 2.0 bisa disebut web baca/tulis, Web 3.0 pasti akan disebut web baca/tulis/eksekusi. Dua komponen utama yang membentuk dasar. Web 3.0 adalah sebagai berikut:

1. Web semantik
2. Layanan web

***Web Semantik***

Web semantik menyediakan pengguna web kerangka umum yang dapat digunakan untuk berbagi dan menggunakan kembali data di berbagai aplikasi, perusahaan, dan batasan komunitas. Web semantik adalah visi TI yang memungkinkan data dan informasi mudah ditafsirkan oleh mesin, sehingga mesin dapat mengambil

keputusan kontekstual sendiri dengan menemukan, menggabungkan, dan bertindak berdasarkan informasi yang relevan di web. Jaringan semantik, seperti yang dibayangkan semula, adalah sistem yang memungkinkan mesin memahami konteks dan makna permintaan manusia yang kompleks dan meresponsnya dengan tepat. Juga, web semantik dianggap sebagai integrator konten atau informasi di berbagai aplikasi dan sistem.

Web 1.0 merupakan implementasi pertama dari web, yang menurut Berners-Lee, dapat dianggap sebagai web read-only. Ini berarti bahwa web awal memungkinkan pengguna untuk mencari informasi yang diperlukan dan membacanya. Ada sangat sedikit interaksi pengguna atau kontribusi konten dalam kasus ini. Pemilik situs web mencapai tujuan mereka untuk membangun kehadiran online dan membuat informasi mereka tersedia untuk semua orang kapan saja.

Salah satu tantangan terbesar dalam menyajikan informasi di web adalah bahwa aplikasi web tidak dapat menghubungkan informasi konteks ke data, dan akibatnya, mereka tidak dapat benar-benar memahami apa yang relevan dan apa yang tidak. Melalui penggunaan semacam markup semantik, atau format pertukaran data, data dapat direpresentasikan dalam bentuk yang tidak hanya dapat diakses oleh manusia melalui bahasa alami tetapi juga dapat dipahami dan diinterpretasikan oleh aplikasi perangkat lunak. Memformat data agar dipahami oleh agen perangkat lunak ditekankan oleh bagian eksekusi dari definisi baca/tulis/eksekusi Web 3.0.

***Layanan Web***

Layanan web adalah sistem perangkat lunak yang mendukung interaksi komputer-ke-komputer melalui Internet. Layanan web biasanya direpresentasikan sebagai API. Misalnya, situs web berbagi fotografi populer Flickr menyediakan layanan web yang dapat digunakan oleh pengembang untuk secara terprogram berinteraksi dengan Flickr untuk mencari gambar. Saat ini, ribuan layanan web tersedia untuk pengguna, dan mereka merupakan komponen penting dalam konteks Web 3.0. Dengan kombinasi markup semantik dan layanan web, paradigma Web 3.0 menjanjikan potensi aplikasi yang dapat berkomunikasi satu sama lain secara langsung dan juga memfasilitasi pencarian informasi yang lebih luas melalui antarmuka yang lebih sederhana.

**Karakteristik Web 3.0**

Web 3.0 dapat dianggap sebagai generasi ketiga web dan dimungkinkan oleh konvergensi beberapa tren teknologi utama yang muncul seperti yang dibahas sebelumnya. Karakteristik utama dari paradigma ini adalah sebagai berikut:

- Konektivitas di mana-mana
- Komputasi jaringan
- Buka teknologi
- Buka identitas
- Jaring cerdas

Teknologi Web 3.0 memungkinkan konektivitas berkelanjutan dari pengguna yang meminta layanan dengan berbagai layanan yang tersedia melalui penggunaan perangkat seluler dan akses Internet seluler dengan adopsi broadband. Teknologi Web 3.0 memungkinkan model bisnis Software-as-a-Service (SaaS) dalam komputasi awan. Web 3.0 juga membuat berbagai layanan web dapat dioperasikan dengan menyediakan standar terbuka. Web 3.0 membantu dalam pembuatan dan penggunaan API terbuka dan protokol untuk komposisi dan komunikasi layanan. Format data terbuka dan platform perangkat lunak sumber terbuka didukung untuk pengembangan dan penggunaan berbagai aplikasi dan layanan. Web 3.0 juga memungkinkan penggunaan protokol identitas terbuka seperti OpenID, yang membantu memindahkan akun pengguna dari satu layanan ke layanan lainnya secara efektif.

Web 3.0 mendukung teknologi web semantik (RDF, OWL, SWRL, SPARQL, platform aplikasi semantik, dan penyimpanan data berbasis pernyataan seperti triplestore, tuplestore, dan basis data asosiatif) dan basis data terdistribusi (interoperabilitas basis data terdistribusi area luas yang diaktifkan oleh web semantik teknologi). Juga, aplikasi cerdas (menggunakan konsep pemrosesan bahasa alami, pembelajaran mesin, penalaran mesin, dan agen otonom) dibuat, dan komunikasi efektifnya dimungkinkan. Karenanya, Web 3.0 membantu mencapai web yang lebih terhubung, terbuka, dan cerdas dengan memanfaatkan teknologi yang disebutkan di atas.

**Konvergensi *Cloud* dan Web 3.0**

Konsep Web 2.0 dan Web 3.0 dapat digunakan untuk mengimplementasikan web sebagai platform, yang membentuk dasar pengiriman dan pemanfaatan layanan *cloud* yang efektif. Evolusi Web 3.0 dengan kemampuannya yang diperkaya seperti personalisasi, portabilitas data, dan identitas yang berpusat pada pengguna memberikan peluang yang cukup bagi perusahaan dan individu untuk menghasilkan dan menggunakan konten web dengan cara yang lebih efektif, yang pada gilirannya meningkatkan kinerja layanan *cloud computing*. Dengan demikian, paradigma Web 3.0 telah mengubah mode komputasi dengan memungkinkan konsumen layanan memanfaatkan perangkat lunak dan layanan yang terletak di DC penyedia layanan daripada membatasi layanan yang tersedia di PC pengguna. Oleh karena itu, dengan fitur Web 3.0, pengguna dapat menyimpan data dan aplikasi mereka di *cloud* dan berlangganan layanan *cloud* jika diperlukan. Selain itu, layanan *cloud* dapat diakses melalui berbagai perangkat selain PC, dan informasi atau layanan tersedia untuk berbagai pengguna saat berpindah dari satu perangkat ke perangkat lainnya.

Dengan diperkenalkannya Web 2.0, Internet telah digunakan lebih dari sekadar melakukan pembelian online. Itu telah menjadi tempat berkumpul di mana komunitas sosial terbentuk, informasi pribadi dibagikan, dan orang-orang berdialog dan berhubungan kembali dengan kenalan lama. Peluncuran situs jejaring sosial seperti Twitter dan Facebook telah sukses besar di kalangan pengguna akhir. Karena basis pelanggan mereka yang terus berkembang, situs jejaring sosial yang sangat interaktif ini digunakan sebagai alat e-marketing populer oleh vendor bisnis untuk menjangkau konsumen dan mempromosikan produk mereka melalui cara yang murah dan kreatif.

**Studi Kasus di *Cloud* dan Web 3.0**

Web 3.0 mewakili generasi berikutnya dari teknologi web yang membantu mencapai tingkat kecerdasan dan interaksi yang belum pernah terjadi sebelumnya dalam sistem dan aplikasi komputasi. Kemampuan komputasi yang lebih cerdas akan diperkenalkan ke dalam aplikasi web untuk melakukan tugas kompleks yang sebelumnya membutuhkan interaksi manusia untuk tujuan pemahaman dan penalaran. Dengan teknologi ini, berbagai tugas komputasi dapat diselesaikan dalam skala besar dan efisien. Saat ini, penyedia layanan seperti Facebook menggunakan teknologi ini dengan cara yang efektif. Mereka memperkenalkan berbagai kemampuan inovatif termasuk akses lebih cepat ke informasi yang disimpan di DC mereka dengan memanfaatkan aplikasi pengambilan keputusan yang sangat cerdas. Beberapa contoh diberikan dalam subbagian berikut.

***Menghubungkan Informasi: Facebook***

Grafik Terbuka Facebook adalah contoh yang bagus untuk fitur skalabilitas yang ditawarkan oleh Web 3.0. Grafik Terbuka menyertakan format untuk menandai halaman web berdasarkan RDF dan model data web semantik sehingga siapa pun yang menggunakan situs web dapat memasukkan markup Facebook untuk menentukan isi situs tersebut. Ini melampaui pemrosesan metadata dasar karena deskripsi memungkinkan pengguna web menemukan dan terhubung dengan teman-teman mereka yang memiliki minat yang sama. Tombol Suka yang disediakan oleh Facebook dapat dianggap sebagai manifestasi sederhana dari semua ini karena satu klik dapat menawarkan kepada analis sejumlah informasi yang tak ternilai yang nantinya dapat digunakan untuk komunikasi lebih lanjut dengan teman-teman dan juga untuk membuat rekomendasi dan penemuan.

***Pengoptimalan Pencarian dan Perdagangan Web: Best Buy***

Salah satu manfaat utama Web 3.0 adalah hasil pencarian yang lebih relevan. Dalam kasus perdagangan Web, ini berarti bahwa informasi tambahan dapat dimasukkan ke dalam deskripsi produk dan iklan online agar lebih mudah ditemukan oleh mesin pencari. Best Buy adalah yang terdepan dalam menggunakan teknologi ini untuk memanfaatkan upaya e-niaganya. Itu menggunakan RDFa (RDF dalam atribut, yang menambahkan satu set atribut XHTML) markup dan kosa kata GoodRelations sehingga hasil pencarian yang lebih bertarget dihasilkan untuk pembeli yang mencari berbagai produk. Sejauh ini, beberapa langkah audit internal perusahaan menunjukkan bahwa pendekatan ini telah meningkatkan lalu lintas konsumen sebesar 30%.

***Pengertian Teks: Millward Brown***

Web 3.0 sangat cocok untuk pengelolaan dan analisis berbagai dokumen dan informasi karena memungkinkan sistem komputasi untuk memproses teks dalam jumlah besar dengan cepat dan mengekstrak makna darinya. Analisis sentimen dapat dianggap sebagai contoh yang baik untuk ini, yang melibatkan pengukuran tentang bagaimana perasaan berbagai pelanggan tentang suatu organisasi, seperti yang diungkapkan melalui survei, blog, forum online, dan jejaring sosial. Badan riset global Millward Brown, yang bekerja dengan perusahaan Fortune 500 untuk

mengembangkan strategi branding mereka, menggunakan teknologi Web 3.0 dari OpenAmplify untuk mengidentifikasi informasi bermakna strategis yang diambil dari umpan balik pelanggan. Informasi ini kemudian dapat digunakan untuk mengarahkan pesan pemasaran dan juga untuk meningkatkan upaya hubungan masyarakat, strategi penetapan harga, dan tanggapan layanan perusahaan.

## 6.9 MODEL PROSES PERANGKAT LUNAK UNTUK *CLOUD*

Keberhasilan atau kualitas proyek perangkat lunak diukur dengan apakah dikembangkan dalam waktu dan anggaran dan dengan efisiensi, kegunaan, ketergantungan, dan pemeliharaan. Seluruh proses pengembangan perangkat lunak mulai dari konseptualisasi hingga pengoperasian dan pensiun disebut siklus hidup pengembangan perangkat lunak (SDLC). SDLC melewati beberapa aktivitas kerangka kerja seperti pengumpulan persyaratan, perencanaan, desain, pengkodean, pengujian, penerapan, pemeliharaan, dan penghentian. Kegiatan ini disinkronkan sesuai dengan model proses yang diadopsi untuk pengembangan perangkat lunak tertentu.

Ada banyak model proses yang dapat dipilih, tergantung pada ukuran proyek, persyaratan waktu pengiriman, dan jenis proyek. Misalnya, model proses yang dipilih untuk pengembangan sistem tertanam avionik akan berbeda dari yang dipilih untuk pengembangan aplikasi web.

**Jenis Model Perangkat Lunak**

Ada berbagai model atau metodologi pengembangan perangkat lunak seperti waterfall, V, incremental, RAD, agile, iterative, dan spiral. Mereka dibahas sebagai berikut.

**Model Air Terjun**

Ini adalah model siklus hidup yang paling umum dan juga disebut sebagai model siklus hidup linier-sekuensial. Dalam model air terjun, setiap fase harus diselesaikan secara keseluruhan sebelum fase berikutnya dimulai. Di akhir setiap fase, dilakukan tinjauan untuk menentukan apakah proyek berada di jalur yang benar dan apakah akan melanjutkan proyek atau tidak.

**Model V**

Model V berarti model verifikasi dan validasi. Sama seperti model air terjun, model V adalah jalur eksekusi proses yang berurutan. Setiap fase harus diselesaikan sebelum fase berikutnya dimulai. Pengujian produk direncanakan secara paralel dengan fase pengembangan yang sesuai.

**Model Tambahan**

Model inkremental adalah pendekatan intuitif untuk model air terjun. Beberapa siklus pengembangan terjadi di sini, menjadikan siklus hidup sebagai siklus multi-air terjun. Siklus dibagi menjadi iterasi yang lebih kecil dan lebih mudah dikelola. Iterasi melewati fase persyaratan, desain, implementasi, dan pengujian, dan selama iterasi pertama, versi perangkat lunak yang berfungsi dihasilkan.

**Model RAD**

Model pengembangan aplikasi cepat (RAD) adalah jenis model inkremental. Dalam model RAD, fungsi atau komponen dihasilkan secara paralel, dan hasil yang

dihasilkan ini diberi batas waktu, dikirimkan, dan kemudian digabungkan menjadi prototipe yang berfungsi. Ini dapat dengan cepat memberi pelanggan sesuatu untuk dioperasikan dan untuk memberikan umpan balik tentang persyaratan.

**Model Tangkas**

Model tangkas juga merupakan model inkremental di mana perangkat lunak dikembangkan dalam siklus inkremental yang cepat. Hasil pengembangan dalam rilis tambahan kecil dan didasarkan pada fungsionalitas yang dibangun sebelumnya dan diuji secara hati-hati untuk memastikan kualitas perangkat lunak. Dalam aplikasi kritis waktu, model ini lebih disukai. Pemrograman ekstrem (XP) adalah salah satu contoh populer dari model siklus hidup perkembangan ini.

**Model Iteratif**

Itu tidak dimulai dengan spesifikasi lengkap persyaratan, tetapi dimulai dengan menentukan dan mengimplementasikan hanya sebagian dari perangkat lunak, yang kemudian dapat ditinjau untuk mengidentifikasi persyaratan lebih lanjut. Proses ini kemudian diulang, menghasilkan versi baru dari perangkat lunak untuk setiap siklus model.

**Model Spiral**

Model spiral mirip dengan model inkremental, dengan lebih menekankan pada analisis risiko. Model spiral memiliki empat fase: perencanaan, analisis risiko, rekayasa, dan evaluasi. Dalam model ini, proyek perangkat lunak berulang kali melewati fase-fase ini dalam iterasi yang disebut spiral. Spiral dasar dimulai pada fase perencanaan, persyaratan dikumpulkan, dan risiko dinilai. Setiap spiral berikutnya dibangun di atas spiral dasar.

**SDLC Tangkas untuk Komputasi Awan**

Dalam lingkungan komputasi yang berubah dengan cepat dalam layanan web dan platform *cloud*, pengembangan perangkat lunak akan menjadi sangat menantang. Proses pengembangan perangkat lunak akan melibatkan platform yang heterogen, layanan web terdistribusi, dan banyak perusahaan yang tersebar secara geografis di seluruh dunia. Model proses perangkat lunak dan aktivitas kerangka kerja yang ada tidak memadai kecuali interaksi dengan penyedia *cloud* disertakan. Fase pengumpulan persyaratan sejauh ini mencakup pelanggan, pengguna, dan insinyur perangkat lunak. Sekarang, itu harus menyertakan penyedia *cloud* juga, karena mereka akan menyediakan infrastruktur komputasi dan pemeliharaannya. Karena hanya penyedia *cloud* yang mengetahui ukuran, detail arsitektur, strategi virtualisasi, dan pemanfaatan sumber daya infrastruktur, mereka juga harus disertakan dalam fase perencanaan dan desain pengembangan perangkat lunak. Pengkodean dan pengujian dapat dilakukan pada platform *cloud*, yang merupakan keuntungan besar karena setiap orang akan memiliki akses mudah ke perangkat lunak yang sedang dibangun. Ini akan mengurangi biaya dan waktu untuk pengujian dan validasi.

Di lingkungan awan, pengembang perangkat lunak dapat menggunakan layanan web dan perangkat lunak sumber terbuka yang tersedia secara bebas dari awan alih-alih membelinya. Pengembang perangkat lunak membangun perangkat lunak dari komponen yang tersedia daripada menulis semuanya dan membangun aplikasi monolitik. Refactoring

aplikasi yang ada diperlukan untuk memanfaatkan arsitektur infrastruktur *cloud* dengan cara yang hemat biaya. Dalam teknologi perangkat keras terbaru, komputer adalah multicore dan jaringan dan insinyur perangkat lunak harus melatih diri mereka sendiri dalam komputasi paralel dan terdistribusi untuk melengkapi kemajuan teknologi perangkat keras dan jaringan ini. Penyedia *cloud* akan bersikeras bahwa perangkat lunak harus semodular mungkin untuk migrasi sesekali dari satu server ke server lain untuk LB seperti yang dipersyaratkan oleh penyedia *cloud*.

SDLC adalah kerangka kerja yang mendefinisikan tugas yang harus dilakukan pada setiap langkah dalam proses pengembangan perangkat lunak. Komputasi awan menyediakan akses yang hampir instan ke perangkat lunak dan lingkungan pengembangan, dengan menyediakan banyak server virtual dan infrastruktur TI lainnya. Secara khusus, Platform as a Service (PaaS), lingkungan platform pengembangan di *cloud*, mendorong penggunaan metodologi tangkas. Agile dan PaaS bersama-sama menambah nilai besar pada proses SDLC. Mereka membantu mengurangi biaya untuk perusahaan dalam jangka panjang dan membantu meningkatkan produktivitas pengembang pada saat yang bersamaan.

**Fitur *Cloud* SDLC**

SDLC untuk komputasi awan berbeda dari SDLC tradisional dalam hal berikut:

1. Kecenderungan terhadap metodologi tangkas: *Cloud* SDLC dapat memanfaatkan metodologi seperti SDLC tangkas. Ini dirancang untuk pendekatan berulang untuk pengembangan dan siklus hidup penyebaran cepat.
2. Kerangka kerja SDLC yang dapat disesuaikan untuk tahapan yang berbeda: *Cloud computing* SDLC harus memiliki kemampuan untuk disesuaikan sesuai dengan persyaratan proyek. Dengan kata lain, elastisitas dan ketangguhan lingkungan *cloud computing* dapat dimanfaatkan dengan baik jika SDLC untuk *cloud* dapat disesuaikan.
3. Panduan instalasi dan konfigurasi: SDLC untuk *cloud* harus menyediakan pendekatan implementasi dan panduan untuk instalasi dan konfigurasi *cloud* tergantung pada ukurannya. Pedoman tersebut harus memastikan bahwa instalasi dan konfigurasi infrastruktur dan lingkungan aplikasi diselesaikan dengan tepat untuk berbagai tahapan SDLC termasuk operasi dan pemeliharaan. Pedoman ini adalah kunci untuk membedakan SDLC untuk *cloud* dari SDLC tradisional.

**Proses Pengembangan Perangkat Lunak Agile**

Lebih dari 50% proyek perangkat lunak gagal karena berbagai alasan seperti jadwal dan slip anggaran, antarmuka perangkat lunak yang tidak ramah pengguna, dan tidak fleksibelnya pemeliharaan dan perubahan perangkat lunak Alasan dari semua masalah ini adalah kurangnya komunikasi dan koordinasi antara semua pihak yang terlibat. Perubahan kebutuhan perangkat lunak adalah penyebab utama meningkatnya kompleksitas, jadwal, dan slip anggaran. Memasukkan perubahan pada tahap selanjutnya dari SDLC meningkatkan biaya proyek secara eksponensial. Menambahkan lebih banyak pemrogram pada tahap selanjutnya tidak menyelesaikan masalah jadwal karena persyaratan koordinasi yang meningkat memperlambat proyek lebih lanjut. Sangat penting bahwa pengumpulan kebutuhan,

perencanaan, dan desain perangkat lunak dilakukan dengan melibatkan semua pihak terkait sejak awal proyek.

Itulah mengapa beberapa model proses tangkas seperti XP, Scrum, Crystal, dan Adaptive telah diperkenalkan pada pertengahan 1990-an untuk mengakomodasi perubahan kebutuhan yang terus-menerus selama pengembangan perangkat lunak. Model proses tangkas ini memiliki siklus pengembangan yang lebih pendek di mana potongan-potongan kecil pekerjaan diatur waktunya, dikembangkan, dan dirilis untuk umpan balik pelanggan, verifikasi, dan validasi secara iteratif. Satu kotak waktu memakan waktu beberapa minggu hingga satu bulan. Model proses tangkas intensif komunikasi. Ini menghilangkan peningkatan biaya secara eksponensial untuk memasukkan perubahan seperti dalam model air terjun, dengan membuat pelanggan tetap terlibat selama proses pengembangan dan memvalidasi pekerjaan kecil oleh mereka secara iteratif. Model proses tangkas ini bekerja lebih baik untuk sebagian besar proyek perangkat lunak karena perubahan tidak dapat dihindari dan menanggapi perubahan adalah kunci keberhasilan proyek.

Sejak agile development ditemukan pada pertengahan 1990-an, agile telah merevolusi bagaimana perangkat lunak dibuat dengan menekankan siklus pengembangan singkat berdasarkan umpan balik pelanggan yang cepat. Karena pengembang mencari periode waktu yang lebih singkat, rilis baru yang besar dikirimkan tepat waktu. Pengembang yang menggunakan metodologi ini menyebut proses peningkatan berkelanjutan. Tetapi untuk sebagian besar sejarahnya, pengembangan tangkas kehilangan komponen penting: platform pengembangan yang mendukung siklus pengembangan cepat yang membuat metodologi berfungsi. Dalam lingkungan perangkat lunak tradisional, distribusi perangkat lunak baru merupakan cobaan berat yang memerlukan tambalan, penginstalan ulang, dan bantuan dari tim pendukung. Dalam lingkungan seperti itu, berbulan-bulan atau bahkan bertahun-tahun diperlukan untuk mendapatkan distribusi baru ke tangan pengguna. Memasukkan umpan balik mereka ke rilis berikutnya membutuhkan waktu yang sebanding.

**Keuntungan Model Agile**

Proses perangkat lunak tangkas menawarkan keuntungan sebagai berikut dibandingkan dengan model pengembangan perangkat lunak tradisional:

1. Waktu pemasaran yang lebih cepat: Karena perangkat lunak dikembangkan menggunakan waktu yang lebih singkat dalam model proses yang gesit, ini mengurangi waktu yang dibutuhkan organisasi untuk meluncurkan produk ke pasar.
2. ROI Cepat: Karena organisasi dapat meluncurkan produk dalam waktu yang lebih singkat, ini menghasilkan ROI yang cepat.
3. Siklus rilis yang lebih pendek: Proses tangkas memastikan bahwa produk perangkat lunak dirilis dalam siklus yang lebih singkat dibandingkan dengan model pengembangan perangkat lunak tradisional.
4. Kualitas yang lebih baik: Karena model pengembangan tangkas memastikan interaksi maksimal antara pemangku kepentingan selama seluruh proses pengembangan, ini meningkatkan kualitas produk secara keseluruhan.
5. Adaptasi dan daya tanggap yang lebih baik terhadap persyaratan perubahan bisnis: Karena model proses tangkas adaptif untuk menggabungkan perubahan persyaratan

setiap saat selama proses pengembangan, ini meningkatkan daya tanggap terhadap perubahan persyaratan bisnis.

6. Deteksi dini kegagalan/proyek yang gagal: Model proses tangkas melibatkan interaksi maksimum di antara para pemangku kepentingan, dan fase pengujian tidak ditunda sampai seluruh proses pengembangan perangkat lunak selesai. Ini membantu dalam deteksi dini kegagalan/proyek yang gagal.

**Bagaimana *Cloud* Memenuhi Proses Agile?**

Kasus penggunaan pengembangan *cloud* mencakup aliran cacat/persyaratan melalui fase pengembangan/bangun/pengujian dan kembali ke pengajuan persyaratan atau cacat baru oleh berbagai pemangku kepentingan. Otomatisasi pada titik mana pun yang memungkinkan adalah kemampuan utama, termasuk kemampuan untuk menghidupkan dan mematikan sistem virtual atau fisik sesuai kebutuhan, di *cloud*. Integrasi berkelanjutan adalah konsep kunci untuk praktik tangkas. Itu didasarkan pada filosofi mengapa menunggu sampai akhir proyek untuk melihat apakah semua bagian dari sistem akan berfungsi? Setiap beberapa jam sistem harus terintegrasi penuh, tetapi diuji dengan semua perubahan terbaru, sehingga penyesuaian dapat dilakukan.

Di sinilah komputasi cloud membuat perbedaan besar, Komputasi *cloud* menghilangkan persyaratan distribusi yang rumit yang dapat membawa pengembangan yang gesit ke perayapan. Tidak ada tambalan untuk didistribusikan dan tidak diperlukan penginstalan ulang. Dengan komputasi awan, distribusi baru dipasang di server yang dihosting dan segera tersedia bagi pengguna. Akibatnya, ada kemungkinan aplikasi yang dijalankan hari ini telah dimodifikasi pada malam sebelumnya. Salah satu contoh terbaik dalam menyatukan pengembangan tangkas dan komputasi awan adalah pengalaman Salesforce.com di mana, pada akhir 2006, tim R&D pindah ke pengembangan tangkas.

**Enam Cara *Cloud* Meningkatkan Pengembangan Perangkat Lunak Agile**

Komputasi awan dan virtualisasi memungkinkan pembuatan VM dan penggunaan layanan berbasis awan untuk manajemen proyek, manajemen masalah, dan pembuatan perangkat lunak dengan pengujian otomatis. Ini, pada gilirannya, mendorong pengembangan tangkas dalam enam cara utama. Komputasi *cloud* dan virtualisasi memudahkan tim pengembangan yang gesit untuk menggabungkan berbagai lingkungan pengembangan, pengujian, dan produksi dengan mulus dengan layanan *cloud* lainnya. Berikut adalah enam cara penting di mana komputasi awan dan virtualisasi meningkatkan pengembangan perangkat lunak yang gesit:

1. *Cloud computing* menyediakan server pengujian dan pementasan dalam jumlah tak terbatas: Ketika pengembangan tangkas digunakan tanpa virtualisasi atau *cloud*, tim pengembangan dibatasi pada satu server fisik per pengembangan, pementasan, dan kebutuhan server produksi. Namun, saat VM atau instans *cloud* digunakan, tim pengembangan praktis memiliki jumlah server yang tidak terbatas yang tersedia untuk mereka. Mereka tidak perlu menunggu server fisik menjadi bebas untuk memulai atau melanjutkan pekerjaan mereka.
2. Ini mengubah pengembangan agile menjadi aktivitas yang benar-benar paralel: Bahkan dalam pengembangan agile, pengembang mungkin mengalami

penundaan dalam menyediakan instans server dan menginstal platform dasar yang diperlukan seperti perangkat lunak database. Tim pengembangan tangkas dapat menyediakan sendiri server yang mereka butuhkan dengan cepat, daripada menunggu operasi TI melakukannya untuk mereka.

3. Mendorong inovasi dan eksperimentasi: Mampu menelurkan contoh sebanyak yang diperlukan memungkinkan kelompok pengembangan yang gesit untuk berinovasi. Jika fitur atau cerita terlihat menarik, tim dapat menelurkan contoh pengembangan dengan cepat untuk membuat kode dan mengujinya. Tidak perlu menunggu build atau rilis berikutnya, seperti kasus ketika server fisik dalam jumlah terbatas tersedia. Saat menambahkan komputasi awan ke pengembangan yang gesit, build menjadi lebih cepat dan tidak terlalu menyakitkan, sehingga mendorong eksperimen.
4. Meningkatkan integrasi dan pengiriman yang berkelanjutan: Memiliki sejumlah besar VM yang tersedia untuk grup pengembangan tangkas di *cloud*-nya sendiri atau di *cloud* publik sangat meningkatkan kecepatan integrasi dan pengiriman yang berkelanjutan.
5. Itu membuat lebih banyak platform pengembangan dan layanan eksternal tersedia: Grup pengembangan tangkas mungkin perlu menggunakan berbagai manajemen proyek, manajemen masalah, dan, jika integrasi berkelanjutan digunakan, lingkungan pengujian otomatis. Sejumlah layanan ini tersedia sebagai penawaran Perangkat Lunak sebagai Layanan (SaaS) di *cloud*:
    a Pengembangan tangkas dapat menggunakan kombinasi virtualisasi, *cloud* pribadi, dan *cloud* publik di tingkat IaaS. Penawaran tersebut termasuk Amazon Web Services, GoGrid, OpSource, dan RackSpace *Cloud*.
    b Kemudian datanglah penggunaan instans PaaS seperti Oracle Database *Cloud* Service, Google App Engine, dan platform Salesforce.com (force.com), yang semuanya menyertakan database dan lingkungan bahasa sebagai layanan.
    c Terakhir, ada sejumlah layanan SaaS yang secara khusus membantu pengembangan tangkas, termasuk Saleforce.com, portal manajemen proyek Basecamp, dan TestFlight, yang menyediakan otomatisasi pengujian yang dihosting untuk perangkat Apple iOS.
6. Ini memudahkan percabangan dan penggabungan kode: Dalam upaya pemfaktoran ulang kode, rilis saat ini mungkin perlu ditingkatkan dengan peningkatan kecil dan digunakan dalam produksi, sementara desain ulang kode sedang berlangsung. Percabangan kode diperlukan dalam kasus ini. Percabangan dan penggabungan kode melibatkan menyulap banyak versi pengembangan dan pembuatan pementasan. Dengan virtualisasi dan komputasi awan, membeli atau menyewa server fisik tambahan untuk tujuan ini dapat dihindari.

**Studi Kasus Pengembangan Agile**

Sementara itu, R&D Salesforce.com memanfaatkan komputasi awan untuk mempercepat siklus rilis. Infrastruktur *cloud* perusahaan membantu mempertahankan satu basis kode terpadu yang dapat digunakan oleh tim pengembangan yang tersebar secara geografis. Tim-tim tersebut berhasil menggabungkan pengembangan yang gesit dan integrasi/pengiriman berkelanjutan dengan komputasi awan. Dalam referensi, Salesforce.com menemukan bahwa model proses tangkas bekerja lebih baik pada platform *cloud computing*. Sebelum pengenalan komputasi awan, ada jeda atau interval waktu antara rilis perangkat lunak dan mendapatkan umpan balik dari pelanggan dan sekarang rilis perangkat lunak baru dapat diunggah ke server dan digunakan oleh pelanggan secara bersamaan. Jadi, model pengembangan tangkas dapat melengkapi manfaat layanan perangkat lunak yang dihosting di Internet. Dalam lingkungan komputasi yang berubah dengan cepat dengan layanan web dan platform *cloud*, desain dan pengembangan perangkat lunak juga melibatkan berbagai platform, layanan web terdistribusi, dan perusahaan yang terdistribusi secara geografis.

Organisasi R&D Salesforce.com mendapat manfaat dalam beberapa cara dari transisinya ke pengembangan tangkas:

- Meningkatkan kecepatan pengiriman dan menciptakan proses yang membuat pelanggan dan Litbang senang
- Peningkatan waktu pemasaran rilis utama sebesar 61%
- Mencapai Net Promoter Score sebesar 94%, indikator kepuasan pelanggan yang baik
- Meyakinkan 90% tim R&D untuk merekomendasikan metodologi tersebut kepada kolega di dalam dan di luar perusahaan
- Peningkatan produktivitas di seluruh organisasi sebesar 38%, yang diukur dengan jumlah fitur yang dihasilkan per pengembang (manfaat sampingan yang tidak diantisipasi sebagai bagian dari tujuan awal)

## 6.10 MODEL PEMROGRAMAN

Model pemrograman untuk *cloud computing* telah menjadi fokus penelitian akhir-akhir ini. Komputasi awan berjanji untuk menyediakan layanan TI sesuai permintaan dan fleksibel, yang melampaui model pemrograman tradisional dan panggilan untuk yang baru. Platform *cloud* memungkinkan pemrogram untuk menulis aplikasi yang berjalan di *cloud*, atau menggunakan layanan dari *cloud*, atau keduanya sambil mengabstraksi esensi skalabilitas dan pemrosesan terdistribusi. Dengan munculnya *cloud* sebagai arsitektur yang baru lahir, diperlukan abstraksi yang mendukung model pemrograman yang muncul. Dalam beberapa tahun terakhir, komputasi awan telah mengarah pada desain dan pengembangan model pemrograman yang beragam untuk pemrosesan data yang masif dan aplikasi intensif komputasi.

Secara khusus, model pemrograman adalah abstraksi dari sistem komputer yang mendasari yang memungkinkan untuk ekspresi dari kedua algoritma dan struktur data. Sebagai perbandingan, bahasa dan API menyediakan implementasi abstraksi dan

memungkinkan algoritma dan struktur data untuk dipraktikkan. Model pemrograman ada secara independen dari pilihan bahasa pemrograman dan API pendukung. Model pemrograman biasanya difokuskan untuk mencapai peningkatan produktivitas, kinerja, dan portabilitas pengembang ke desain sistem lainnya. Sifat arsitektur prosesor yang berubah dengan cepat dan kompleksitas merancang platform memberikan tantangan yang signifikan untuk tujuan ini. Beberapa faktor lain cenderung mempengaruhi desain model pemrograman masa depan. Secara khusus, representasi dan pengelolaan tingkat paralelisme, konkurensi, dan hierarki memori yang meningkat, dikombinasikan dengan kemampuan untuk mempertahankan tingkat interoperabilitas yang progresif dengan aplikasi saat ini, menjadi perhatian yang signifikan. Selain itu, keberhasilan implementasi model pemrograman bergantung pada fitur yang diekspos dari lapisan perangkat lunak runtime dan fitur OS.

Selama bertahun-tahun, banyak organisasi telah membangun sistem berskala besar untuk memenuhi permintaan penyimpanan dan pemrosesan yang tinggi dari aplikasi intensif komputasi dan data. Dengan popularitas dan tuntutan pada DC, merupakan tantangan untuk menyediakan model pemrograman yang tepat yang mampu mendukung akses mudah ke data skala besar untuk melakukan komputasi sambil menyembunyikan semua detail lingkungan fisik tingkat rendah. Pemrograman *cloud* adalah tentang mengetahui apa dan bagaimana memprogram pada platform *cloud*. Platform *cloud* menyediakan fungsi lokal dasar yang diperlukan oleh program aplikasi. Ini dapat mencakup OS yang mendasari dan dukungan lokal seperti penyebaran, manajemen, dan pemantauan.

**Model Pemrograman di *Cloud***

Ada berbagai model pemrograman yang digunakan untuk memecahkan berbagai masalah komputasi atau intensif data di *cloud*. Model yang akan dipilih bergantung pada sifat masalahnya dan juga pada QoS yang diharapkan dari lingkungan *cloud*. Beberapa model pemrograman *cloud* dibahas dalam subbagian berikut.

**Model BSP**

Dengan keunggulan kinerja yang dapat diprediksi, pemrograman yang mudah, dan penghindaran kebuntuan, model paralel sinkron massal (BSP) telah diterapkan secara luas dalam database paralel, mesin pencari, dan komputasi ilmiah. Model BSP dapat diadaptasi ke dalam lingkungan *cloud*. Penjadwalan tugas komputasi dan alokasi sumber daya *cloud* diintegrasikan ke dalam model BSP. Baru-baru ini, penelitian tentang model pemrograman *cloud computing* telah membuat beberapa kemajuan yang signifikan, seperti MapReduce Google dan Dryad Microsoft. Model BSP awalnya diusulkan oleh Harvard's Valiant. Tujuan awalnya adalah untuk menjembatani perangkat lunak dan arsitektur komputasi paralel. Ini menawarkan keuntungan sebagai berikut: pertama, kinerjanya dapat diprediksi; kedua, tidak terjadi kebuntuan selama pengiriman pesan; dan ketiga, mudah diprogram. Model BSP dapat digunakan tidak hanya untuk aplikasi intensif data tetapi juga untuk aplikasi intensif komputasi dan intensif I/O.

**Model MapReduce**

Baru-baru ini, banyak sistem komputer skala besar dibangun untuk memenuhi permintaan penyimpanan dan pemrosesan yang tinggi dari aplikasi intensif komputasi

dan data. MapReduce adalah salah satu model pemrograman paling populer yang dirancang untuk mendukung pengembangan aplikasi semacam itu. Ini awalnya dibuat oleh Google untuk menyederhanakan pengembangan aplikasi pencarian web skala besar di DC dan telah diusulkan untuk membentuk dasar komputer pusat data. Dengan semakin populernya DC, merupakan tantangan untuk menyediakan model pemrograman yang tepat yang mampu mendukung akses mudah ke data berskala besar untuk melakukan komputasi sambil menyembunyikan semua detail lingkungan fisik tingkat rendah. Di antara semua kandidat, MapReduce adalah salah satu model pemrograman paling populer yang dirancang untuk tujuan ini.

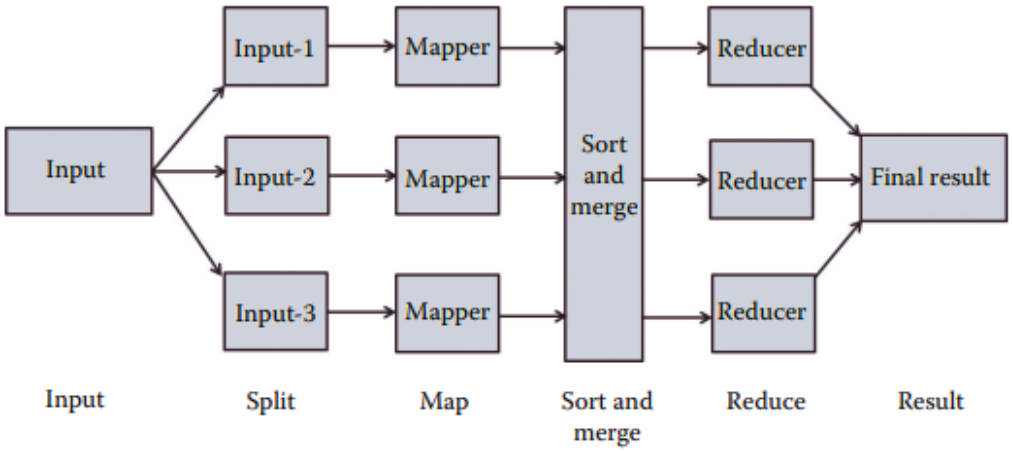
MapReduce dipicu oleh operasi peta dan pengurangan dalam bahasa fungsional, seperti Lisp. Model ini abstrak masalah perhitungan melalui dua fungsi: peta dan mengurangi. Semua masalah yang dirumuskan dengan cara ini dapat diparalelkan secara otomatis. Pada dasarnya, model MapReduce memungkinkan pengguna untuk menulis peta/mengurangi komponen dengan kode gaya fungsional. Komponen-komponen ini kemudian disusun sebagai grafik aliran data untuk secara eksplisit menentukan paralelismenya. Akhirnya, sistem runtime MapReduce menjadwalkan komponen-komponen ini ke sumber daya terdistribusi untuk dieksekusi sambil menangani banyak masalah sulit: paralelisasi, komunikasi jaringan, dan toleransi kesalahan.

Fungsi peta mengambil pasangan kunci/nilai sebagai masukan dan menghasilkan daftar pasangan kunci/nilai sebagai keluaran. Fungsi pengurangan mengambil kunci dan daftar nilai terkait sebagai input dan menghasilkan daftar nilai baru sebagai output. Aplikasi MapReduce dijalankan secara paralel melalui dua fase. Pada fase pertama, semua operasi peta dapat dijalankan secara independen satu sama lain. Pada fase kedua, setiap operasi pengurangan mungkin bergantung pada keluaran yang dihasilkan oleh sejumlah operasi peta. Semua operasi pengurangan juga dapat dijalankan secara independen mirip dengan operasi peta.

Eksekusi tugas dilakukan dalam empat tahap: memetakan, mengurutkan, menggabungkan, dan mengurangi. Fase peta diisi dengan sekumpulan pasangan kunci/nilai. Untuk setiap pasangan, modul mapper menghasilkan sebuah hasil. Fase urutkan dan gabungkan mengelompokkan data untuk menghasilkan larik, di mana setiap elemen adalah grup nilai untuk setiap kunci. Fase pengurangan bekerja pada data ini dan menerapkan fungsi pengurangan padanya. Fungsi hash yang digunakan dalam peta dan fungsi pengurangan ditentukan oleh pengguna dan bervariasi dengan penerapan model. Perhitungan keseluruhan digambarkan pada Gambar 6.5.

MapReduce telah muncul sebagai model pemrograman paralel data yang penting untuk komputasi intensif data. Namun, sebagian besar implementasi MapReduce terkait erat dengan infrastruktur. Ada model pemrograman yang diusulkan yang menyediakan antarmuka pemrograman tingkat tinggi, sehingga memberikan kemampuan untuk membuat aplikasi terdistribusi dengan cara yang tidak bergantung pada infrastruktur. Simple API for Grid Applications (SAGA) dan Transformer adalah contoh dari model tersebut, yang mencoba

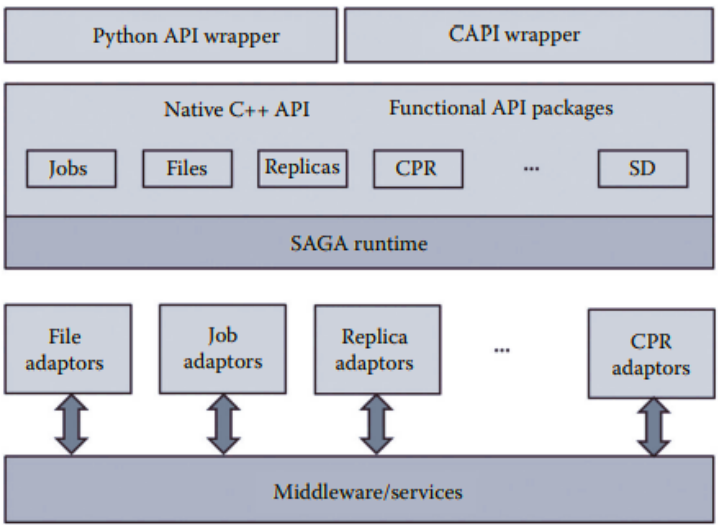
mengimplementasikan model paralel seperti MapReduce dan All-Pairs, mengambil beban yang cukup besar dari pengembang aplikasi.




**Gambar 6.5** Perhitungan MapReduce.

**SAGA**

Meskipun MapReduce telah muncul sebagai model pemrograman data-paralel yang penting untuk komputasi intensif data, sebagian besar, jika tidak semua, implementasi MapReduce terkait erat dengan infrastruktur tertentu. SAGA adalah antarmuka pemrograman tingkat tinggi yang menyediakan kemampuan untuk membuat aplikasi terdistribusi dengan cara yang tidak bergantung pada infrastruktur. SAGA mendukung berbagai model pemrograman dan berkonsentrasi pada interoperabilitas pada infrastruktur jaringan dan *cloud*. SAGA mendukung pengiriman pekerjaan di berbagai platform terdistribusi, akses/transfer file, dan file logis, serta pemulihan pos pemeriksaan dan penemuan layanan. SAGA API ditulis dalam C++ dan mendukung bahasa lain seperti Python, C, dan Java. Pengambilan keputusan lingkungan runtime didukung oleh mesin yang memuat adaptor yang relevan, seperti yang ditunjukkan pada Gambar 6.6.

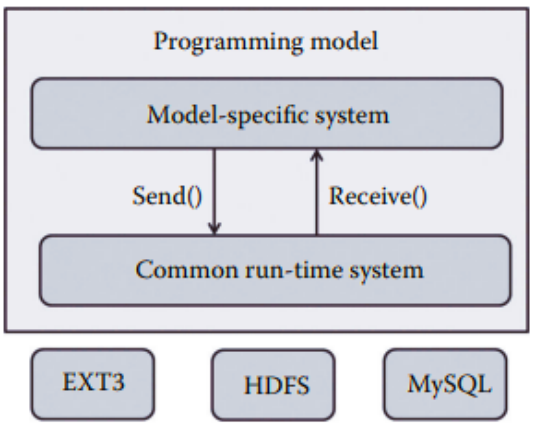



**Gambar 6.6** Model SAGA.

**Transformator**

Meskipun ada model pemrograman berbasis C ++ dan Java di pasar industri, mereka memiliki kekurangan tertentu. Pertama, programmer harus menguasai API yang besar dan kompleks untuk menggunakan model tersebut. Kedua, sebagian besar model pemrograman dirancang untuk abstraksi pemrograman tertentu dan dibuat untuk mengatasi satu jenis masalah tertentu. Tidak ada kerangka kerja perangkat lunak terdistribusi universal untuk memproses kumpulan data masif. Untuk mengatasi kekurangan yang disebutkan di atas, kerangka kerja baru yang disebut Transformer digunakan, yang mendukung beragam model pemrograman dan juga tidak spesifik masalah.

Transformator didasarkan pada dua operasi ringkas: kirim dan terima. Menggunakan model Transformer, berbagai model seperti MapReduce, Dryad, dan All-Pairs dapat dibangun. Arsitektur model Transformer dibagi menjadi dua lapisan: runtime umum dan sistem khusus model, seperti yang ditunjukkan pada Gambar 6.7. Hal ini dilakukan untuk mengurangi kopling. Sistem runtime menangani aliran data seperti tugas antara mesin dan mengeksekusi tugas pada sistem yang berbeda menggunakan fungsi kirim dan terima dari API runtime. Lapisan khusus model berurusan dengan tugas model tertentu seperti pemetaan, partisi data, dan ketergantungan data.




**Gambar 6.7** Arsitektur transformator.

Transformer memiliki arsitektur master/slave. Setiap node memiliki dua komponen komunikasi: pengirim pesan dan penerima pesan. Node master mengeluarkan perintah untuk eksekusi tugas pada node budak. Node budak mengembalikan status eksekusi saat selesai. Strategi toleransi kesalahan bersifat gesit. Kegagalan terdeteksi oleh sistem runtime sedangkan pemulihan kesalahan ditangani oleh lapisan khusus model. Ini melibatkan menjalankan kembali tugas atau mengirim ulang data. Sistem transformer dikodekan dengan Python. Komunikasi antar node dilakukan dengan menggunakan mekanisme *message-passing* yang berlawanan dengan semaphore dan kondisi dalam pendekatan threaded. Karena frekuensi

komunikasinya tinggi, pemrograman jaringan asinkron diadopsi, dan terlebih lagi pesannya diserialkan sebelum dikirim. Dengan menggunakan model Transformer, ketiga model pemrograman paralel yang dikenal, yaitu MapReduce, Dryad, dan All-Pairs, diimplementasikan.

**Kerangka Batch Grid**

Baru-baru ini, sebuah alternatif untuk model komputasi paralel telah disarankan yang memungkinkan pengguna untuk mempartisi data mereka dengan cara yang disederhanakan dengan efisiensi setinggi mungkin. Sistem Grid Batch memiliki dua tipe data fundamental: tabel dan tabel terindeks. Tabel adalah sekumpulan baris yang saling bebas. Tabel yang diindeks memiliki semua properti tabel selain memiliki indeks yang terkait dengan setiap catatan.

Dua komponen perangkat lunak utama dari sistem Batch Grid adalah Distributed File System (DFS) dan Job Scheduler. DFS bertanggung jawab untuk menyimpan dan mengelola file di semua node dalam sistem. Sebuah file dipecah menjadi banyak bagian dan masing-masing bagian ini disimpan pada node yang terpisah. Penjadwal Pekerjaan terdiri dari node master dan node slave terkait. Suatu pekerjaan dipecah menjadi banyak tugas yang lebih kecil oleh simpul master, dan masing-masing tugas ini didistribusikan di antara simpul budak. Peta dasar dan operator pengurangan dalam sistem MapReduce diperluas dalam model Grid Batch. Ini adalah operator peta, operator distribusi, operator join, operator Cartesian, operator recurse, dan operator tetangga.

## 6.11 KOMPUTASI PERVASIF

*Pervasive Computing* (komputasi pervasif) adalah kombinasi teknologi, seperti kemampuan Internet, pengenalan suara, jaringan, kecerdasan buatan, dan komputasi nirkabel, yang digunakan untuk memungkinkan komputasi di mana saja. Pervasif perangkat komputasi membuat aktivitas komputasi sehari-hari menjadi sangat mudah dilakukan. Teknologi ini bergerak melampaui PC ke perangkat sehari-hari dengan teknologi dan konektivitas tertanam. *Pervasive Computing* juga disebut ubiquitous computing, di mana hampir semua perangkat atau bahan seperti pakaian, peralatan, peralatan, kendaraan, rumah, tubuh manusia, atau bahkan cangkir kopi dapat ditanamkan dengan chip untuk menghubungkan objek tersebut ke jaringan tanpa batas. dari perangkat lain. Tujuan komputasi meresap, yang menggabungkan teknologi jaringan saat ini dengan komputasi nirkabel, pengenalan suara, kemampuan Internet, dan kecerdasan buatan, adalah untuk menciptakan lingkungan di mana konektivitas perangkat dicapai sedemikian rupa sehingga konektivitas tidak mengganggu dan selalu tersedia. Komputasi pervasif juga memiliki sejumlah aplikasi prospektif, mulai dari perawatan di rumah dan kesehatan, hingga pelacakan geografis dan sistem transportasi cerdas.

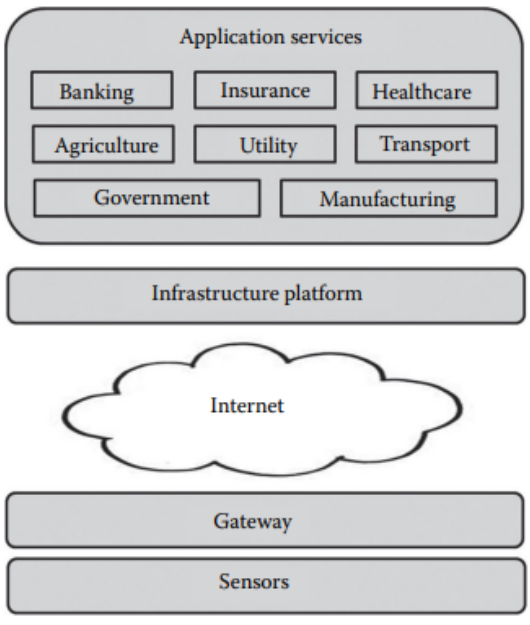
Kata meresap dan di mana-mana berarti ada di mana-mana. Perangkat komputasi yang meresap benar-benar terhubung dan selalu tersedia. Komputasi pervasif bergantung pada konvergensi teknologi nirkabel, elektronik canggih, dan Internet. Tujuan para peneliti yang bekerja dalam komputasi pervasif adalah untuk menciptakan produk pintar yang

berkomunikasi secara diam-diam. Produk terhubung ke Internet dan data yang mereka hasilkan tersedia dengan mudah. Contoh aplikasi praktis komputasi pervasif adalah penggantian meteran listrik lama dengan meteran pintar. Di masa lalu, meteran listrik harus dibaca secara manual oleh perwakilan perusahaan. Meter pintar melaporkan penggunaan listrik secara real time melalui Internet. Mereka juga akan memberi tahu perusahaan listrik saat terjadi pemadaman dan juga mengirim pesan untuk menampilkan unit di rumah dan mengatur pemanas air.

Oleh karena itu, dalam komputasi pervasif, komputasi dibuat muncul di mana saja dan di mana saja. Berbeda dengan komputasi desktop, komputasi pervasif dapat dilakukan dengan menggunakan perangkat pendukung apa pun, di lokasi mana pun. Teknologi yang mendasari untuk mendukung komputasi pervasif meliputi Internet, middleware canggih, OS, seluler, sensor, mikroprosesor, I/O baru dan antarmuka pengguna, jaringan, protokol seluler, dan layanan berbasis lokasi. Paradigma ini juga dinamai dengan nama yang berbeda seperti komputasi fisik, Internet of Things, dan hal-hal yang berpikir, dengan mempertimbangkan objek yang terlibat di dalamnya. Dalam hal ini, perangkat yang digunakan untuk mengakses aplikasi dan informasi hampir tidak relevan karena berbagai jenis perangkat atau platform dapat digunakan untuk melakukan operasi yang dimaksud.

**Bagaimana Komputasi Pervasif Bekerja?**

Keberhasilan komputasi di mana-mana terletak pada integrasi yang tepat dari berbagai komponen yang berbicara satu sama lain dan dengan demikian berperilaku sebagai satu sistem yang terhubung. Gambar 6.8 menunjukkan arsitektur tumpukan komputasi di mana-mana. Di bagian bawah tumpukan adalah lapisan fisik. Sensor kecil dipasang (dibawa, dipakai, atau disematkan) ke orang, hewan, mesin, rumah, mobil, gedung, kampus, dan lapangan. Sensor menangkap berbagai bit informasi dari lingkungan terdekat. Selain mikrofon dan kamera, beberapa sensor seperti GPS, akselerometer, dan kompas dapat diintegrasikan ke dalamnya.




**Gambar 6.8** Tumpukan komputasi meresap.

Di atas sensor terdapat infrastruktur komunikasi nirkabel, yang dapat disediakan oleh rangkaian jaringan 802.11. Bersama dengan jaringan mesh, standar tersebut memastikan konektivitas sensor dan perangkat.

Teknologi lain yang disebut ZigBee adalah alternatif berbiaya rendah untuk menjaga beberapa perangkat tetap terhubung, memungkinkan perangkat induk mengontrol sensor anak secara nirkabel. Near field communication (NFC) adalah standar teknologi lain yang memanfaatkan RFID dan dapat digunakan untuk komputasi di mana-mana, terutama dalam skenario di mana titik pasif yang dioperasikan dengan baterai tidak diperhatikan. Perangkat bertenaga NFC juga dapat berinteraksi satu sama lain.

Tingkat berikutnya mencakup berbagai layanan aplikasi. Data dari sensor dan perangkat genggam dikumpulkan, ditambang, dan dianalisis untuk polanya. Pola tersebut membantu menyediakan opsi untuk aplikasi cerdas yang secara proaktif membuat perubahan pada lingkungan melalui smartphone, tablet, notebook, atau perangkat genggam atau perangkat pintar lainnya. Contohnya adalah pasien jantung yang memakai monitor kecil yang terhubung ke perangkat seluler. EKG yang tidak teratur akan memicu ponsel untuk mengingatkan dokter pasien dan layanan darurat.

**Bagaimana Pervasive Computing Membantu *Cloud computing*?**

Saat ini, perusahaan TI mengadopsi komputasi awan untuk mengurangi total biaya yang terlibat dan juga untuk meningkatkan QoS yang dikirimkan ke pelanggan. Komputasi awan memberikan peluang untuk mengakses infrastruktur, platform, dan perangkat lunak dari penyedia layanan berdasarkan pembayaran per penggunaan. Pervasive Computing membantu *cloud computing* dengan menyediakan kemampuan untuk mengakses sumber daya *cloud* kapanpun, dimanapun dan juga melalui perangkat apapun. Komputasi pervasif menyediakan fitur yang diperlukan seperti komputasi di mana-mana, penyimpanan dan pengarsipan, aplikasi berbasis komunitas sosial, dan aplikasi bisnis serta non-bisnis agar komputasi awan mendapatkan potensi penuhnya. Komputasi awan biasanya merupakan arsitektur klien-server, di mana klien dapat berupa perangkat portabel apa pun seperti laptop, ponsel, browser, atau perangkat lain yang mendukung OS. Masalah utama dengan perangkat portabel ini adalah kendala yang mereka hadirkan dalam hal penyimpanan, memori, pemrosesan, dan masa pakai baterai. Dengan menyimpan data di *cloud*, dan berinteraksi dengan *cloud* melalui saluran komunikasi yang aman, semua batasan ini dapat dengan mudah dipenuhi.

Komunikasi mesin-ke-mesin (M2M) penting dalam komputasi awan. Skenario komunikasi seperti itu terbentang dari lantai toko, ke DC, ke ruang rapat, saat perangkat dibawa sepanjang melacak pergerakan dan aktivitas pengguna dan juga berinteraksi dengan sistem lain di sekitarnya. Misalnya, seorang karyawan sedang berada di New York dan dia ingin mendiskusikan sesuatu dengan dua rekannya. Dia meminta janji temu menggunakan perangkat selulernya, dan berdasarkan data lokasinya dan data rekannya, serta waktu rapat, sistem backend secara otomatis memesankan ruang konferensi untuknya dan mengatur tautan video ke rekan kerja di luar kota. Berdasarkan analitik dan judul rapat, dokumen yang relevan dimasukkan ke dalam ruang kolaborasi. Perangkat karyawan merekam rapat ke arsip

dan catatan yang telah hadir secara langsung. Dan, percakapan ini secara otomatis ditranskrip, diberi tag, dan diteruskan ke anggota tim untuk ditinjau.

Perangkat yang dapat dikenakan seperti Google Glass juga akan masuk ke tempat kerja baru. Kekuatan sebenarnya di balik aplikasi ini bukan pada perangkat itu sendiri tetapi pada sistem analitik yang mendukungnya. Sistem backend aplikasi menggabungkan data yang dikumpulkan dari berbagai jenis perangkat komputasi seperti Google Glass, telepon pintar, perangkat GPS tertanam di palet, atau sensor di mesin mobil. Sistem tersebut memproses data dan kemudian mengubahnya menjadi informasi berguna yang digunakan untuk memicu tindakan yang diperlukan. Sistem komputasi berbeda yang melakukan berbagai aktivitas disebarkan dengan API sehingga pengguna dapat membangun aplikasi yang mengekstraksi informasi dari banyak sistem ini.

## 6.12 SISTEM OPERASI

OS adalah kumpulan perangkat lunak yang mengelola sumber daya perangkat keras komputer dan program lain dalam sistem komputasi. Ini menyediakan layanan umum yang dibutuhkan oleh program komputer untuk pelaksanaannya yang efektif dalam lingkungan komputasi. OS adalah komponen penting dari perangkat lunak sistem dalam sistem komputer karena program aplikasi biasanya memerlukan OS untuk antarmuka mereka dengan sumber daya perangkat keras dan program sistem lainnya. Untuk fungsi perangkat keras seperti input dan output, dan alokasi memori, OS bertindak sebagai perantara antara program dan perangkat keras komputer.

**Jenis Sistem Operasi**

Varian yang berbeda dari OS adalah sebagai berikut:

1. Network OSs: Sebuah sistem operasi jaringan (NOS) adalah OS komputer yang dirancang terutama untuk mendukung workstation, PC yang terhubung pada LAN. Sebuah NOS menyediakan fitur-fitur seperti berbagi printer, sistem file umum dan berbagi basis data, berbagi aplikasi, mekanisme keamanan, dan juga kemampuan untuk mengelola direktori nama jaringan dan fungsi pemeliharaan jaringan lainnya. NetWare Novell dan Manajer LAN Microsoft adalah contoh NOS. Selain itu, beberapa OS multiguna, seperti Windows NT dan Digital's OpenVMS, hadir dengan kemampuan yang memungkinkannya untuk dideskripsikan sebagai NOS.
2. OS Web: OS Web pada dasarnya adalah situs web yang mereplikasi lingkungan desktop OS modern, semuanya di dalam browser web. Mereka diinstal ke server web dan hidup di Internet. Dengan demikian, pengguna dapat mengakses desktop virtualnya dari perangkat apa saja, di mana saja, yang terhubung ke internet. Web OS juga disebut komputer dinamis. Dalam hal ini, aplikasi, hard disk, dan OS semuanya ada di server tempat mereka diakses. Penyedia layanan OS web mengelola akses aplikasi dan basis data dari berbagai pengguna. Pengguna diberikan antarmuka pengguna grafis yang serupa dengan yang tersedia di PC desktop, yang dapat digunakan untuk mengakses data dan aplikasi dari server. Google Chrome OS adalah contoh OS web.
3. OS Terdistribusi: OS terdistribusi adalah perangkat lunak yang hadir di atas kumpulan node komputasi independen, jaringan, berkomunikasi, dan terpisah secara fisik. Setiap node

individu memiliki perangkat lunak tertentu yang merupakan bagian dari OS agregat global. Setiap subset terdiri dari dua komponen berbeda dari OS terdistribusi. Yang pertama adalah kernel minimal, atau mikrokernel, yang secara langsung mengontrol perangkat keras node. Yang kedua adalah kumpulan komponen manajemen sistem tingkat tinggi yang mengoordinasikan aktivitas individu dan kolaboratif node. Mikrokernel dan komponen manajemen bekerja sama. Mereka mendukung tujuan sistem terdistribusi untuk mengintegrasikan banyak sumber daya dan memproses fungsionalitas menjadi sistem yang efisien dan stabil. Bagi pengguna, OS terdistribusi bekerja dengan cara yang mirip dengan OS monolitik single-node. Artinya, meskipun terdiri dari banyak node, tampaknya bagi pengguna dan aplikasi sebagai OS node tunggal.

4. Sistem Tertanam: Sistem Tertanam adalah OS yang ada di perangkat elektronik yang digunakan untuk berbagai keperluan agar menjadikannya pintar dan lebih efisien. Sistem tertanam hadir dalam perangkat seperti router, misalnya, biasanya menyertakan server web yang telah dikonfigurasi sebelumnya, server DHCP, dan beberapa utilitas untuk operasi jaringan yang efektif, dan mereka tidak mengizinkan penginstalan program baru di dalamnya. Contoh OS tersemat untuk router termasuk Cisco Internetwork Operating System (IOS), DD-WRT, dan Juniper Junos. OS yang disematkan juga dapat ditemukan di dalam semakin banyak gadget konsumen termasuk ponsel, bantuan digital pribadi (PDA), dan pemutar media digital untuk menyelesaikan tugas yang dimaksudkan dengan sukses.

**Peran OS dalam *Cloud computing***

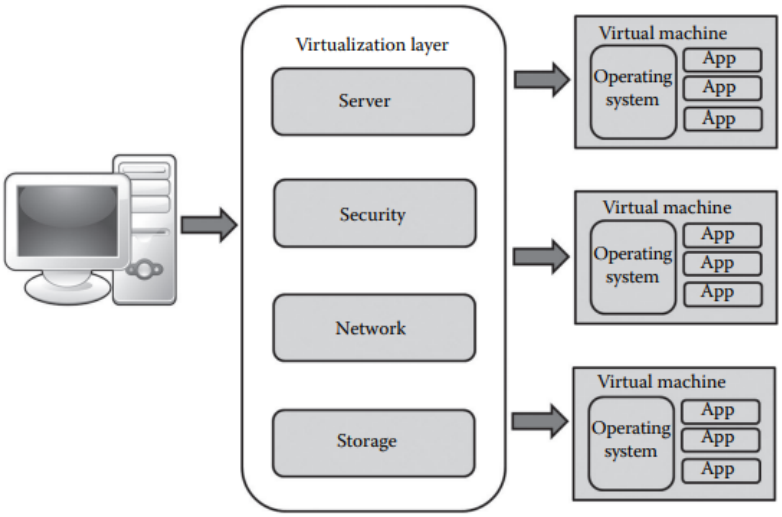
Pada tahun 1970-an, International Business Machines Corporation (IBM) merilis OS yang disebut VM yang memungkinkan sistem mainframe memiliki banyak sistem virtual, atau VM pada satu node fisik. VM OS mewujudkan akses bersama dari sistem mainframe ke tingkat berikutnya dengan memungkinkan beberapa lingkungan komputasi yang berbeda untuk hidup di lingkungan fisik yang sama. Sebagian besar fungsi dasar perangkat lunak virtualisasi saat ini diwariskan dari OS VM awal ini. Perangkat lunak virtualisasi sekarang diwakili dengan istilah hypervisor. Virtualisasi perangkat keras digunakan untuk berbagi sumber daya penyedia *cloud* secara efektif dengan pelanggan dengan menghasilkan ilusi komputasi khusus, penyimpanan, dan jaringan pada infrastruktur komputasi. Konsep virtualisasi diimplementasikan secara fisik menggunakan modul hypervisor, dan pengoperasian serta pemrosesan hypervisor diwujudkan oleh OS. Dengan kata lain, modul hypervisor dipasang di atas OS, yang berfungsi sebagai antarmuka antara unit perangkat keras dan paket hypervisor.

Dengan menggunakan virtualisasi, beberapa VM dihasilkan sesuai dengan kebutuhan, dan VM ini dioperasikan dengan menginstal OS secara individual pada setiap VM. Gambar 6.9 menunjukkan virtualisasi satu perangkat keras CSP untuk membuat VM yang berbeda, masing-masing diinstal dengan OS-nya sendiri. Setiap VM menjalankan OS khusus atau OS tamu yang memiliki memori, CPU, dan hard drive sendiri bersama dengan CD-ROM, keyboard, dan jaringan, terlepas dari kenyataan bahwa semua sumber daya tersebut dibagi di antara VM.

Selain itu, OS seperti Linux mendukung standar yang diperlukan yang meningkatkan portabilitas dan interoperabilitas di lingkungan *cloud*. Platform OS dirancang untuk menyembunyikan banyak kerumitan yang diperlukan untuk mendukung aplikasi yang berjalan di lingkungan yang kompleks dan federasi. Perlu bekerja secara efektif di latar belakang untuk

memastikan bahwa semua sumber daya yang tepat (seperti kekuatan pemrosesan, memori yang diperlukan, dan penyimpanan) dialokasikan bila diperlukan. Selain itu, OS mengimplementasikan tingkat keamanan dan QoS untuk memastikan bahwa aplikasi dapat mengakses sumber daya yang dibutuhkan untuk memberikan tingkat kinerja yang dapat diterima, dengan cara yang efisien dan aman.

Salah satu cara terpenting untuk mendukung kompleksitas yang mendasari sumber daya komputasi awan yang dikelola dengan baik adalah melalui OS. Salah satu persyaratan paling signifikan bagi perusahaan yang mengadopsi *cloud computing* adalah kebutuhan untuk mengadopsi pendekatan hybrid untuk komputasi. Untuk melakukannya, sebagian besar organisasi akan terus mempertahankan DC tradisional mereka untuk mendukung beban kerja campuran yang kompleks. Misalnya, organisasi dapat memilih lingkungan *cloud* publik untuk pengembangan dan pengujian beban kerja, *cloud* pribadi untuk lingkungan web yang berhadapan dengan pelanggan yang berhubungan dengan informasi pribadi, dan DC tradisional untuk beban kerja tagihan dan keuangan lama. Dianggap bahwa lingkungan komputasi awan hibrid akan menjadi norma di masa depan. Oleh karena itu, semakin penting bagi OS untuk mendukung dan menyatukan berbagai model penyebaran komputasi sehingga mereka tampak sebagai sistem tunggal dari pengalaman pelanggan dan perspektif manajemen sistem.




**Gambar 6.9** OS dan virtualisasi.

**Fitur *Cloud* OS**

Unsur-unsur yang diperlukan untuk menciptakan lingkungan komputasi awan hibrid yang canggih secara operasional adalah sebagai berikut:

1. Antarmuka yang terdefinisi dengan baik yang menyembunyikan detail implementasi
2. Layanan keamanan inti
3. Kemampuan mengelola virtualisasi
4. Pengelolaan beban kerja untuk menyediakan QoS dan kinerja Fitur-fitur ini dijelaskan pada sub-bagian berikut.

**Antarmuka yang Didefinisikan dengan Baik dan Abstrak**

*Cloud* OS harus menyediakan API yang memungkinkan interoperabilitas data dan layanan di seluruh lingkungan *cloud* terdistribusi. OS yang matang menyediakan serangkaian layanan yang kaya untuk aplikasi sehingga setiap aplikasi tidak harus menemukan fungsi penting seperti pemantauan VM, penjadwalan, keamanan, manajemen daya, dan manajemen memori. Selain itu, jika API dibuat dengan standar terbuka, ini akan membantu organisasi menghindari vendor lock-in dan dengan demikian menciptakan lingkungan yang lebih fleksibel. Misalnya, tautan akan diperlukan untuk menjembatani DC tradisional dan lingkungan *cloud* publik atau pribadi. Fleksibilitas pergerakan data atau informasi di seluruh sistem ini menuntut OS untuk menyediakan landasan yang aman dan konsisten untuk menuai keuntungan nyata yang ditawarkan oleh lingkungan komputasi awan. Selain itu, OS perlu memastikan sumber daya yang tepat dialokasikan untuk aplikasi yang meminta. Persyaratan ini bahkan lebih penting dalam lingkungan *cloud* hybrid. Oleh karena itu, setiap lingkungan *cloud* yang dirancang dengan baik harus memiliki API yang terdefinisi dengan baik yang memungkinkan aplikasi atau layanan terhubung ke *cloud* dengan mudah. Antarmuka ini harus didasarkan pada standar terbuka untuk melindungi pelanggan agar tidak terkunci dalam lingkungan *cloud* satu vendor.

**Dukungan untuk Keamanan pada Inti**

Baik organisasi sedang mempertimbangkan lingkungan *cloud* publik, privat, atau hybrid, keamanan adalah landasan paling penting untuk memastikan perlindungan asetnya. Masalah keamanan diperparah dalam lingkungan *cloud* karena sangat terdistribusi, dan juga melibatkan berbagai macam sistem internal dan eksternal yang ditambahkan atau dihapus ke/dari *cloud* secara dinamis. Oleh karena itu, lingkungan *cloud* harus melindungi identitas pengguna dan informasinya dari ancaman eksternal. Untuk mendukung kebutuhan organisasi, keamanan *cloud* memerlukan kemampuan manajemen terintegrasi dalam OS yang dapat melacak semua aset TI dalam konteks penggunaannya. Kemampuan ini perlu memastikan bahwa keamanan memenuhi persyaratan kepatuhan dan tata kelola organisasi.

Dukungan virtualisasi dan multitenancy harus diterapkan dengan cara yang aman. Karena virtualisasi dan multitenancy menjadi norma di lingkungan *cloud*, sangat penting untuk membangun keamanan sebagai intinya. Ketika server divirtualisasi, itu memungkinkan pembuatan gambar baru dengan sedikit usaha. Perluasan gambar virtual ini meningkatkan risiko serangan karena meningkatkan kemungkinan celah keamanan di hypervisor dapat dieksploitasi oleh instance tamu. Itu dapat mengekspos sistem yang ada dan mitra lain yang berinteraksi dengan sistem tersebut terhadap ancaman keamanan. Ketika keamanan diimplementasikan sebagai kerangka kerja di dalam OS, itu meningkatkan keamanan keseluruhan dari lingkungan virtual dan nonvirtual.

**Mengelola Beban Kerja Virtual**

Virtualisasi sangat penting untuk komputasi awan karena memutus hubungan tradisional antara server fisik dan aplikasi. Lingkungan tervirtualisasi ini dikendalikan

dan dikelola oleh hypervisor. Intinya, hypervisor adalah OS pada perangkat keras fisik dan menyajikan abstraksi perangkat keras inti dan instruksi I/O yang dibutuhkan oleh tamu di lingkungannya. Oleh karena itu, pengelolaan hypervisor dan lingkungan virtual yang efektif sangat penting untuk keberhasilan komputasi awan.

**Pengelolaan Beban Kerja**

Lingkungan *cloud* harus dirancang dengan cara yang melindungi beban kerja masing-masing pelanggan. Oleh karena itu, prasyarat untuk komputasi awan yang efektif adalah kemampuan untuk mengisolasi beban kerja pengguna satu sama lain. *Cloud* OS harus memastikan bahwa semua sumber daya yang diperlukan dikelola secara efektif di lingkungan *cloud* untuk masing-masing pengguna.

**Persyaratan *Cloud* OS**

Selain fitur *cloud* OS yang dijelaskan di bagian sebelumnya, persyaratan OS utama di lingkungan *cloud* diberikan sebagai berikut:

1. *Cloud* OS harus mengizinkan pengelolaan otonom atas sumber dayanya: *Cloud* OS harus menampilkan antarmuka yang konsisten dan terpadu yang menyembunyikan jika memungkinkan fakta bahwa masing-masing node terlibat dalam operasinya, dan apa saja operasi tingkat rendah tersebut. Itu harus mendukung manajemen otonom dari berbagai sumber daya *cloud* atas nama pengguna dan aplikasinya.
2. Operasi *Cloud* OS harus berlanjut meskipun terjadi kegagalan pada node, cluster, dan partisi jaringan: Menjamin kelanjutan pengoperasian proses manajemen *cloud* dalam kondisi ini melibatkan mekanisme untuk mendeteksi kegagalan dengan cepat dan melakukan tindakan yang tepat untuk memulihkan dari kegagalan. Beberapa pustaka *cloud* yang menerapkan toleransi kesalahan umum dan fitur pemulihan keadaan disediakan untuk digunakan pelanggan.
3. *Cloud* harus mendukung berbagai jenis aplikasi: Aplikasi dari berbagai jenis seperti komputasi performa tinggi, ketersediaan data tinggi, dan throughput jaringan tinggi idealnya harus berdampingan dalam lingkungan *cloud* dan memperoleh sumber daya yang paling sesuai dengan persyaratan aplikasi dari sistem.
4. Sistem manajemen *cloud* OS harus terdesentralisasi, dapat diskalakan, dan hemat biaya: Selain itu, selain penyebaran sumber daya awal, tidak diperlukan campur tangan manusia untuk memperluas sumber daya *cloud*. Demikian pula, manajemen pengguna seharusnya hanya memerlukan pembuatan kredensial pengguna sesuai permintaan, yang kemudian disebarkan secara otomatis ke seluruh *cloud*.
5. Sumber daya yang digunakan dalam arsitektur *cloud* harus dapat dipertanggungjawabkan: Penggunaan sumber daya dari berbagai pelanggan *cloud* harus dipantau secara efektif di lingkungan *cloud*. Aktivitas pemantauan ini dapat digunakan untuk menagih pelanggan atas akses sumber daya mereka, dan juga untuk audit keamanan (jika diperlukan). Selain itu, skema penagihan dinamis berdasarkan kemacetan sumber daya bisa menjadi cara yang efektif untuk alokasi sumber daya.

**OS Berbasis *Cloud***

Para peneliti sekarang bertujuan untuk melangkah lebih jauh dan membawa OS ke *cloud* dengan TransOS, OS lintas platform berbasis *cloud*. Kode sistem TransOS disimpan di server *cloud* dan kode dalam jumlah minimal diperlukan untuk mem-boot komputer dan menghubungkannya ke Internet. Menampilkan antarmuka pengguna grafis, TransOS mengunduh potongan kode tertentu untuk melakukan jenis tugas yang sama seperti OS konvensional, sehingga memungkinkan terminal telanjang untuk melakukan tugas di luar batasan perangkat kerasnya. Terminal akan memanggil kode TransOS yang relevan jika diperlukan, memastikan bahwa OS yang tidak aktif tidak memonopoli sumber daya sistem saat aplikasi sedang dijalankan. TransOS mengelola semua sumber daya perangkat keras dan perangkat lunak jaringan dan virtual dan memungkinkan pengguna untuk memilih dan menjalankan layanan apa pun sesuai permintaan. TransOS dapat diadaptasi ke platform selain PC seperti perangkat seluler, peralatan pabrik, dan bahkan peralatan rumah tangga.

Selain menjaga mesin tetap ramping, pengguna TransOS juga dapat menyimpan dokumen dan file mereka di *cloud*, seperti i*Cloud* milik Apple, menjaga penyimpanan lokal mereka tetap gratis. Dengan TransOS, pengguna tidak perlu khawatir menjalankan versi OS terbaru atau bahkan merawat komputer mereka sendiri. Dengan OS, data, file, dan pengaturan yang disimpan di *cloud*, komputer apa pun dengan koneksi Internet (misalnya, komputer yang tersedia untuk umum di perpustakaan dan perguruan tinggi) menjadi seperti mesin milik pengguna. File yang disimpan di dalam *cloud* juga dapat diakses kapan saja dari perangkat berinternet apa pun, termasuk smartphone dan tablet.

## 6.13 LINGKUNGAN APLIKASI

Lingkungan pengembangan aplikasi (ADE) adalah perangkat keras, perangkat lunak, dan sumber daya komputasi yang diperlukan untuk membangun aplikasi perangkat lunak. ADE adalah sekumpulan sumber daya komputasi yang menyediakan antarmuka untuk pengembangan aplikasi, pengujian, penerapan, integrasi, pemecahan masalah, dan layanan pemeliharaan. Ini digabungkan dengan sumber daya rekayasa perangkat lunak, seperti lingkungan pengembangan terintegrasi (IDE) bahasa pemrograman, perangkat lunak pelaporan dan analisis, alat pemecahan masalah, dan utilitas perangkat lunak evaluasi kinerja lainnya.

**Perlunya ADE yang Efektif**

Saat pasar aplikasi web seluler semakin matang, tekanan persaingan dan ekspektasi pengguna akan mendorong pengembang aplikasi untuk membedakan penawaran produk mereka dengan menyediakan fitur bernilai tambah. Lingkungan aplikasi berbasis standar harus memastikan interoperabilitas komponen pengembangan aplikasi. Secara khusus, itu harus mengaktifkan konten yang disesuaikan, fungsionalitas yang dapat diperluas, dan antarmuka pengguna tingkat lanjut.

Agar web ada di mana-mana, perangkat akses web harus ada hampir di semua tempat. Agar hal ini terjadi, perangkat akses web harus berukuran kecil dan portabel. Saat perangkat yang mendukung web berevolusi dari komputer desktop saat ini menjadi hal-hal seperti telepon seluler, radio mobil, dan pengatur pribadi, tantangannya adalah menyediakan

lingkungan pembuatan aplikasi umum di berbagai perangkat. Lingkungan aplikasi web standar yang ada terdiri dari HTML, JavaScript, dan koleksi ad hoc dari format file grafik standar, diproses oleh browser HTML. Untuk menggabungkan konten multimedia dan memperluas ekspresi atau fungsionalitas antarmuka pengguna, applet Java dan plug-in browser dapat digunakan. Mereka membutuhkan ekstensi yang seringkali khusus untuk perangkat dan memerlukan instalasi khusus.

Oleh karena itu, lingkungan aplikasi web harus menyediakan pengembang aplikasi alat yang mereka butuhkan untuk mengembangkan produk inovatif, tanpa mengorbankan interoperabilitas. Oleh karena itu, ADE yang efektif seharusnya

- Mendukung berbagai jenis konten, termasuk konten multimedia
- Mendukung interaksi pengguna multimodal
- Menyediakan kerangka kerja untuk integrasi teknologi baru saat tersedia

Lingkungan aplikasi harus mendukung kerangka kerja ekstensibilitas standar. Saat jenis media baru, agen pengguna, atau layanan tambahan muncul, mereka harus diintegrasikan ke dalam lingkungan dengan cara yang kompatibel ke belakang tanpa mempengaruhi kinerja aplikasi yang ada.

**Metodologi Pengembangan Aplikasi**

Saat ini, dua metodologi pengembangan banyak digunakan dalam pengembangan aplikasi: pengembangan terdistribusi dan agile.

**Pengembangan Terdistribusi**

Ini adalah produk sampingan alami dari Internet dan fenomena bahwa tidak semua jenius pengkodean tinggal dalam jarak perjalanan dari tempat kerja. Pembangunan terdistribusi adalah pembangunan global yang membawa tantangan tersendiri dengan kolaborasi dan manajemen kode. Ada aplikasi yang tersedia untuk manajemen kode terdistribusi seperti git dan Subversion. Mereka banyak digunakan di lingkungan terdistribusi.

**Pengembangan Agile**

Di sinilah pengembangan *cloud* benar-benar bisa lebih dari sekadar online. Karena lingkungan *cloud* dapat disediakan secara instan dan hampir semua konfigurasi dapat disalin dan diaktifkan, kemungkinan untuk pengembangan instan dan lingkungan pengujian sangat menarik bagi pengembang. Pengembangan *cloud* juga dapat meningkatkan pengembangan tangkas dengan mengoordinasikan kolaborasi, sprint terencana, dan perbaikan bug darurat. Men-deploy ke *cloud* juga sangat berguna untuk pengembangan yang gesit. Prarilis dapat diluncurkan ke mesin uji pelanggan di *cloud* mereka hampir secara instan. Bahkan jika pelanggan belum berada di lingkungan *cloud*, prarilis dapat diposting di *cloud* publik untuk diakses dan diuji oleh pelanggan dari jarak jauh sebelum menerima pengiriman rilis final aplikasi. Toolset yang dapat membantu manajemen tangkas di *cloud* termasuk *Code*2*Cloud*, bersama dengan Tasktop dan CollabNet.

**Kekuatan *Cloud computing* dalam Pengembangan Aplikasi**

Komputasi awan telah secara efektif memecahkan masalah keuangan dan infrastruktur yang terkait dengan pengembangan aplikasi khusus untuk perusahaan karena memudahkan investasi keuangan yang sebelumnya diperlukan untuk menyiapkan lingkungan pengembang canggih yang diperlukan untuk membangun, menguji, dan menggunakan aplikasi khusus di rumah. Akibatnya, pengenalan platform *cloud* telah memungkinkan pengembang untuk hanya berfokus pada pembuatan aplikasi modern yang sangat skalabel. Selanjutnya, proses pemasaran aplikasi kustom ini memakan waktu lebih sedikit dan lebih efektif sebagai hasil dari fleksibilitas yang disediakan oleh layanan *cloud computing*. Saat aplikasi dijalankan di *cloud*, aplikasi tersebut diakses sebagai layanan—ini dikenal sebagai Perangkat Lunak sebagai Layanan (SaaS). Dengan memanfaatkan SaaS, perusahaan dapat memberikan layanan dengan biaya yang efektif dan efisien. Proses ini memungkinkan bisnis bekerja sama dengan mitra untuk mengembangkan aplikasi dan mendistribusikannya dengan cepat di pasar.

Keuntungan menggunakan layanan komputasi awan dibandingkan perangkat lunak tradisional lebih dari sekadar penurunan biaya. Metode tradisional untuk mengembangkan aplikasi khusus yang seringkali membutuhkan waktu berbulan-bulan untuk diselesaikan kini telah turun menjadi hanya beberapa minggu. Dengan semua perangkat lunak dan alat yang diperlukan tersedia di *cloud*, pengembang dapat bekerja lebih efisien dan produktif daripada jika mereka menggunakan perangkat lunak tradisional, di mana, seringkali, komponen tambahan diperlukan untuk mengembangkan aplikasi yang lengkap. Pendekatan akses aplikasi online yang sangat disederhanakan saat ini memungkinkan pengembang untuk menghasilkan aplikasi tingkat perusahaan yang komprehensif hanya melalui browser web, tanpa kesulitan teknis yang terkait dengan solusi tradisional.

Manfaat utama lain menggunakan layanan komputasi awan untuk pengembangan aplikasi adalah penggunaan sumber daya yang efisien. Aplikasi yang memanfaatkan layanan TI tervirtualisasi umumnya lebih efisien dan dilengkapi dengan lebih baik untuk memenuhi permintaan pengguna. Model bayar-per-penggunaan layanan komputasi awan memberi klien fleksibilitas untuk membelanjakan sesuai dengan kebutuhan mereka dan dengan demikian menghilangkan biaya yang tidak perlu. Selain itu, layanan *cloud computing* memungkinkan pengiriman aplikasi pada beberapa perangkat; ini memungkinkan perusahaan untuk merancang aplikasi mereka sehingga kompatibel dengan berbagai perangkat.

**Kelemahan Pengembangan Desktop**

Lingkungan pengembangan desktop menjadi usang, lebih sering gagal, dan menyebabkan masalah produktivitas bagi pengembang. Masalah utama dengan lingkungan desktop adalah sebagai berikut:

1. Manajemen konfigurasi yang rumit: Proses manajemen konfigurasi yang substansial untuk ruang kerja pengembang mengubah pengembang menjadi administrator sistem paruh waktu, yang bertanggung jawab atas mini-DC mereka sendiri yang berjalan sepenuhnya di desktop. Ini memakan waktu, rawan kesalahan, dan menantang untuk diotomatisasi. Banyak pengembang memiliki banyak komputer dan terpaksa mengulangi tugas ini di setiap mesin. Tidak ada

cara untuk menyinkronkan konfigurasi komponen pada mesin yang berbeda, dan setiap mesin memerlukan perangkat keras dan OS yang serupa untuk mengoperasikan komponen secara identik.

2. Produktivitas menurun: Banyak IDE yang memakan memori dan disk, dengan waktu booting yang signifikan. Mereka sangat haus sumber daya sehingga mereka dapat membuat aplikasi lain kelaparan dan efek bersihnya adalah produktivitas yang lebih rendah karena mesin yang lebih lambat.
3. Aksesibilitas terbatas: Biasanya, ruang kerja pengembang desktop tidak dapat diakses melalui perangkat seluler melalui Internet. Pengembang yang membutuhkan akses jarak jauh harus menggunakan beberapa solusi yang rumit dan lambat seperti GotoMyPC.
4. Kolaborasi yang buruk: Saat ini, sebagian besar pengembang bekerja sebagai bagian dari tim, jadi komunikasi dan kolaborasi di antara anggota tim sangat penting untuk keberhasilan proyek. Dalam kasus IDE desktop, mereka harus mengalihdayakan kolaborasi ke sistem komunikasi di luar alur kerja pengembang, memaksa pengembang untuk terus beralih antara mengembangkan dalam IDE dan berkomunikasi dengan tim mereka melalui cara lain. Untuk mengatasi masalah ini, seluruh ruang kerja pengembangan harus dipindahkan ke *cloud*. Lingkungan berbasis *cloud* terpusat, membuatnya mudah untuk dibagikan. Pengembang dapat mengundang orang lain ke ruang kerja mereka untuk mengedit bersama, membuat bersama, atau membuat kode dan dapat berkomunikasi satu sama lain di ruang kerja itu sendiri. *Cloud* dapat menawarkan peningkatan dalam efisiensi sistem, memberikan setiap ruang kerja individu bagian yang dapat dikonfigurasi dari memori yang tersedia dan sumber daya komputasi.

**Keunggulan Pengembangan Aplikasi di *Cloud***

Platform *cloud* mengurangi waktu pengembangan keseluruhan proyek perangkat lunak. Hal ini sebagian besar disebabkan oleh kemampuan platform *cloud* untuk merampingkan proses pengembangan, termasuk kemampuan untuk mendapatkan aset pengembangan secara online dengan cepat. Selain itu, platform *cloud* memberikan kemampuan untuk berkolaborasi secara efektif dalam upaya pengembangan. Platform pengembangan berbasis *cloud* di *cloud* publik PaaS dan IaaS seperti Google, Amazon Web Services, Microsoft, dan Salesforce.com menawarkan penghematan biaya dan QoS yang lebih baik.

Beberapa manfaat pengembangan aplikasi di *cloud* diberikan sebagai berikut:

- Kemampuan untuk menyediakan sendiri lingkungan pengembangan dan pengujian
- Kemampuan untuk memasukkan aplikasi ke dalam produksi dengan cepat dan menskalakan aplikasi tersebut sesuai kebutuhan
- Kemampuan untuk berkolaborasi dengan developer, arsitek, dan desainer lain dalam pengembangan aplikasi

**Platform Pengembangan Aplikasi *Cloud***

Pengembangan aplikasi, penerapan, dan manajemen runtime selalu bergantung pada platform pengembangan seperti Microsoft .NET, WebSphere, atau JBoss, yang telah diterapkan secara tradisional. Dalam konteks *cloud computing*, aplikasi umumnya digunakan oleh penyedia *cloud* untuk menyediakan layanan yang sangat skalabel dan elastis kepada sebanyak mungkin pengguna akhir. Infrastruktur *cloud computing* perlu mendukung banyak pengguna untuk mengakses dan memanfaatkan layanan aplikasi yang sama, dengan alokasi sumber daya yang elastis. Hal ini menyebabkan peningkatan dalam teknologi dan arsitektur platform pengembangan untuk menangani kinerja, keamanan, alokasi sumber daya, pemantauan aplikasi, penagihan, dan toleransi kesalahan. *Cloud* menyediakan ADE sebagai PaaS. Ada beberapa solusi yang tersedia di pasar PaaS, termasuk Google App Engine, Microsoft Windows Azure, Force.com, dan Manjrasoft Aneka.

**Windows Azure**

Windows Azure menyediakan beragam layanan berbasis Windows untuk mengembangkan dan menerapkan aplikasi berbasis Windows di *cloud*. Itu memanfaatkan infrastruktur yang disediakan oleh Microsoft untuk menghosting layanan ini dan menskalakannya dengan mulus. Platform Windows Azure terdiri dari SQL Azure dan layanan .NET. Layanan .NET terdiri dari layanan kontrol akses dan bus layanan .NET. Windows Azure adalah platform dengan perangkat keras multitenant bersama yang disediakan oleh Microsoft. Pengembangan aplikasi Windows Azure mengamanatkan penggunaan SQL Azure untuk fungsionalitas RDBMS, karena itu adalah satu-satunya fungsionalitas DBMS yang dapat diakses dalam konteks perangkat keras yang sama dengan aplikasi.

**Mesin Aplikasi Google**

Google App Engine menyediakan lingkungan waktu proses yang dapat diperluas untuk aplikasi berbasis web yang dikembangkan dengan Java atau Python, yang memanfaatkan infrastruktur TI Google yang sangat besar. Google App Engine ditawarkan oleh Google, Inc. Nilai utamanya adalah pengembang dapat dengan cepat membuat aplikasi berbasis web di mesin mereka dan menerapkannya di *cloud*. Google App Engine memberi pengembang lingkungan yang disimulasikan untuk membangun dan menguji aplikasi secara lokal dengan OS apa pun atau sistem apa pun yang menjalankan versi lingkungan bahasa Python dan Java yang sesuai. Google menggunakan JVM dengan mesin Jetty Servlet dan Java Data Objects.

**Force.com**

Force.com adalah lingkungan pengembangan dan eksekusi dan merupakan pendekatan terbaik untuk PaaS untuk mengembangkan aplikasi berbasis manajemen hubungan pelanggan (CRM). Berkenaan dengan desain platform dan runtime environment-nya, ini didasarkan pada teknologi Java. Platform ini menggunakan bahasa dan lingkungan pemrograman berpemilik yang disebut kode Apex, yang memiliki reputasi untuk kesederhanaan dalam pembelajaran dan pengembangan serta eksekusi yang cepat.

**Manjrasoft Aneka**

Aneka adalah platform aplikasi terdistribusi untuk mengembangkan aplikasi *cloud*. Aneka dapat menyatukan sejumlah desktop atau server fisik atau virtual berbasis Windows ke dalam jaringan node yang saling terhubung yang bertindak sebagai lapisan eksekusi aplikasi logis tunggal. Awan berbasis Aneka dapat digunakan pada berbagai perangkat keras dan OS termasuk beberapa varian dari keluarga OS Windows dan Linux. Aneka menyediakan model yang fleksibel untuk mengembangkan aplikasi terdistribusi dan menyediakan integrasi dengan *cloud* eksternal seperti Amazon EC2 dan GoGrid. Aneka menawarkan kemungkinan untuk memilih penyebaran infrastruktur yang paling tepat tanpa terikat dengan vendor tertentu, sehingga memungkinkan perusahaan untuk dengan nyaman menskalakan ke *cloud* jika diperlukan.

**API Komputasi Awan**

API disediakan oleh beberapa CSP untuk pengembangan aplikasi *cloud*. Detail dari beberapa API yang disediakan oleh CSP seperti Rackspace, IBM, dan Intel diberikan sebagai berikut.

**Ruang rak**

Pengembang memiliki akses ke dokumentasi API dan kit pengembangan perangkat lunak (SDK) di semua layanan Rackspace di situs pengembang mereka, http://developer.rackspace.com. Dengan demikian, Rackspace memberi pengembang alat dan sumber daya yang diperlukan untuk membuat aplikasi dan layanan baru di atas API mereka.

**IBM**

IBM memperkenalkan API baru, yang dapat ditemukan di situs pengembang IBM, www.ibm.com/developerworks/. Pengenalan API baru berfokus pada mempersenjatai pengembang dengan alat dan sumber daya untuk membangun produk, aplikasi, dan layanan baru.

**Intel**

Intel memiliki beberapa SDK yang ditujukan untuk pengembang *cloud computing*. Intel memiliki platform layanan *cloud* beta tempat pengembang dapat mengunduh SDK untuk layanan berbasis identitas dan lintas platform. Program Intel *Cloud* Builders menyatukan vendor sistem dan solusi perangkat lunak terkemuka untuk memberikan praktik terbaik dan panduan praktis tentang cara menerapkan, memelihara, dan mengoptimalkan infrastruktur *cloud* berdasarkan arsitektur Intel. Dan bagi pengembang yang ingin menggunakan layanan infrastruktur *cloud* publik, Intel *Cloud* Finder mempermudah pemilihan penyedia yang memenuhi persyaratan pengembang.

## 6.14 RINGKASAN

Komputasi awan mendominasi industri TI di seluruh dunia saat ini. Semakin banyak perusahaan dan organisasi yang mengadopsi model *cloud* akhir-akhir ini. Meskipun komputasi awan adalah model penyampaian layanan baru, teknologi yang mendasarinya telah ada sejak

lama. Komputasi awan menggunakan banyak dari teknologi tersebut untuk mencapai tujuan yang telah ditetapkan. Bab ini berfokus pada berbagai penggerak teknologi komputasi awan. Ini membahas tentang teknologi dasar yang memungkinkan komputasi awan seperti SOA, hypervisor dan virtualisasi, teknologi multicore, dan teknologi memori dan penyimpanan. Ini juga berbicara tentang perkembangan terbaru di Web 2.0 dan Web 3.0, kemajuan dalam model pemrograman, model pengembangan perangkat lunak, komputasi pervasif, OS, dan ADE. Ini juga menjelaskan bagaimana teknologi ini terkait dengan model *cloud*, membantu *cloud* dalam memberikan layanan berkualitas. Perkembangan terkini di masing-masing teknologi yang memungkinkan ini disorot dengan keunggulan dan fitur karakteristiknya. Bab ini menjelaskan bagaimana teknologi dasar ini memberdayakan paradigma komputasi awan saat ini untuk memberikan layanannya secara efektif. Juga, bab ini menyajikan bagaimana berbagai pemangku kepentingan seperti penyedia layanan dan konsumen layanan diuntungkan dari fitur yang diperluas oleh teknologi ini.

**Tinjau Poin**

- SOA: Arsitektur berorientasi layanan adalah sekumpulan prinsip dan standar desain yang fleksibel yang digunakan untuk pengembangan dan integrasi sistem. Sistem berbasis SOA yang diimplementasikan dengan benar menyediakan serangkaian layanan yang digabungkan secara longgar yang dapat digunakan oleh konsumen layanan untuk memenuhi persyaratan layanan mereka dalam berbagai domain bisnis.
- Hypervisor: Hypervisor adalah perangkat lunak yang digunakan untuk membuat mesin virtual, dan menghasilkan virtualisasi berbagai sumber daya perangkat keras seperti CPU, penyimpanan, dan perangkat jaringan. Mereka juga disebut monitor mesin virtual (VMM) atau manajer virtualisasi.
- Teknologi multicore: Dalam teknologi multicore, dua atau lebih CPU bekerja bersama pada chip yang sama. Dalam jenis arsitektur ini, satu prosesor fisik berisi logika inti dari dua atau lebih prosesor.
- Penyimpanan sebagai Layanan: Penyimpanan sebagai Layanan (STAaS) adalah model bisnis *cloud* di mana penyedia layanan menyewakan ruang dalam infrastruktur penyimpanannya ke berbagai pengguna *cloud*.
- Jaringan yang ditentukan perangkat lunak: Jaringan yang ditentukan perangkat lunak (SDN) adalah pendekatan untuk jaringan di mana kontrol dipisahkan dari perangkat keras jaringan dan diberikan ke aplikasi perangkat lunak yang disebut.
- Web 2.0: Web 2.0 (atau Web 2) adalah istilah populer yang diberikan untuk teknologi dan aplikasi Internet canggih yang meliputi blog, wiki, RSS, dan bookmark social.
- Semantic web: Semantic web adalah visi TI yang memungkinkan data dan informasi mudah diinterpretasikan oleh mesin, sehingga mesin dapat mengambil keputusan kontekstual sendiri dengan menemukan, menggabungkan, dan bertindak berdasarkan informasi yang relevan di web.
- Model pengembangan tangkas: Model tangkas adalah model pengembangan perangkat lunak di mana perangkat lunak dikembangkan dalam siklus inkremental yang cepat. Hasil pengembangan dalam rilis inkremental kecil dan didasarkan pada

fungsionalitas yang dibangun sebelumnya dan diuji secara hati-hati untuk memastikan kualitas perangkat lunak.

- MapReduce: MapReduce adalah model pemrograman populer yang dirancang untuk mendukung pengembangan aplikasi intensif komputasi dan data, yang memerlukan permintaan penyimpanan dan pemrosesan yang tinggi.
- Komputasi pervasif: Komputasi pervasif adalah kombinasi teknologi seperti kemampuan Internet, pengenalan suara, jaringan, kecerdasan buatan, dan komputasi nirkabel yang digunakan untuk memungkinkan komputasi di mana saja.
- OS Web: Sistem operasi web pada dasarnya adalah situs web yang mereplikasi lingkungan desktop OS modern, semuanya di dalam browser web.
- *Cloud* API: *Cloud* API disediakan oleh penyedia layanan *cloud* untuk pengembangan aplikasi *cloud*.

**Latihan Soal**

1. Apa fitur karakteristik SOA yang digunakan dalam penyebaran komputasi awan yang berhasil?
2. Apa saja macam-macam pendekatan dalam virtualisasi? Apa peran yang dimainkan oleh hypervisor dan virtualisasi di lingkungan *cloud*?
3. Bagaimana teknologi multicore dapat digunakan untuk mencapai paralelisme di *cloud*?
4. Apa perkembangan teknologi terkini untuk memenuhi kebutuhan penyimpanan di *cloud*?
5. Bagaimana hubungan SDN dengan skenario *cloud computing*?
6. Bagaimana cara *cloud computing* bergantung pada konsep Web 2.0 untuk keberhasilan operasinya?
7. Bagaimana web semantik dan layanan web berkontribusi pada evolusi komputasi awan?
8. Membenarkan keputusan untuk mengadopsi model pengembangan tangkas untuk pengembangan perangkat lunak. Bagaimana paradigma komputasi awan membuat proses tangkas menjadi efektif?
9. Apa model pemrograman yang digunakan di *cloud*? Justifikasi jawabannya dengan menjelaskan ciri-ciri karakteristik model.
10. Jelaskan cara komputasi pervasif memengaruhi model *cloud*.
11. Jelaskan perbedaan antara OS web dan OS *cloud*.
12. Bagaimana paradigma komputasi awan membantu pengembangan aplikasi yang efektif?

# BAB 7
# VIRTUALISASI

**Tujuan pembelajaran**

Tujuan utama dari bab ini adalah untuk memperkenalkan konsep virtualisasi dan bagaimana hal itu digunakan sebagai teknologi yang memungkinkan untuk komputasi awan. Setelah membaca bab ini, Anda akan

- Memahami dasar-dasar virtualisasi
- Memahami bagaimana berbagai sumber daya seperti prosesor, memori, penyimpanan, dan jaringan dapat divirtualisasi
- Memahami pro dan kontra dari berbagai pendekatan virtualisasi
- Memahami dasar-dasar hypervisor dan masalah keamanannya
- Memahami bagaimana komputasi awan berbeda dari virtualisasi
- Memahami bagaimana komputasi awan memanfaatkan virtualisasi untuk berbagai model layanannya

**Pengantar**

Virtualisasi adalah teknologi yang memungkinkan untuk berbagai layanan komputasi awan. Ini membantu untuk meningkatkan skalabilitas dan pemanfaatan sumber daya infrastruktur yang mendasarinya. Ini juga memungkinkan personel TI untuk melakukan tugas administrasi dengan lebih mudah. Dengan bantuan berbagi sumber daya, hypervisor mendukung layanan TI ramah lingkungan. Bab ini menjelaskan virtualisasi dan membahas manfaat virtualisasi dan berbagai sumber daya yang dapat divirtualisasi. Bab ini juga menjelaskan berbagai pendekatan untuk virtualisasi seperti virtualisasi penuh, virtualisasi dengan bantuan perangkat keras, dan paravirtualisasi. Berbagai jenis hypervisor dan masalah keamanannya juga dibahas. Di akhir bab, perbedaan antara komputasi awan dan virtualisasi dan bagaimana virtualisasi digunakan oleh komputasi awan untuk menyediakan layanan dibahas.

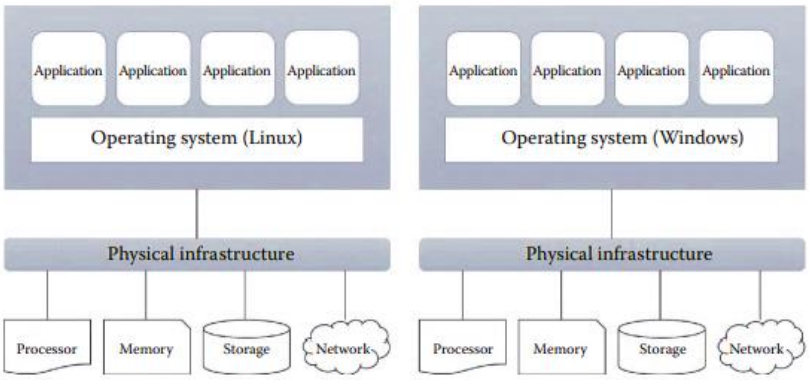
## 7.1 PENDAHULUAN

Dalam beberapa tahun terakhir, komputasi menjadi lebih kompleks dan membutuhkan infrastruktur yang besar. Organisasi menginvestasikan jumlah yang sangat besar untuk membeli infrastruktur fisik tambahan jika ada kebutuhan akan lebih banyak sumber daya komputasi. Jika Anda melihat belanja modal (CapEx) dan belanja operasional (OpEx) untuk membeli dan memelihara infrastruktur besar, itu sangat tinggi. Pada saat yang sama, pemanfaatan sumber daya dan laba atas investasi (ROI) untuk membeli infrastruktur tambahan sangat rendah. Untuk meningkatkan pemanfaatan sumber daya dan ROI, perusahaan mulai menggunakan teknologi yang disebut virtualisasi di mana satu infrastruktur fisik dapat digunakan untuk menjalankan beberapa sistem operasi (OS) dan aplikasi. Virtualisasi bukanlah kata baru bagi dunia komputasi; itu digunakan setidaknya selama empat dekade terakhir. Istilah virtualisasi menjadi kata kunci dalam beberapa tahun terakhir dan sebagian besar organisasi mulai mengadopsinya dengan cepat karena manfaatnya seperti

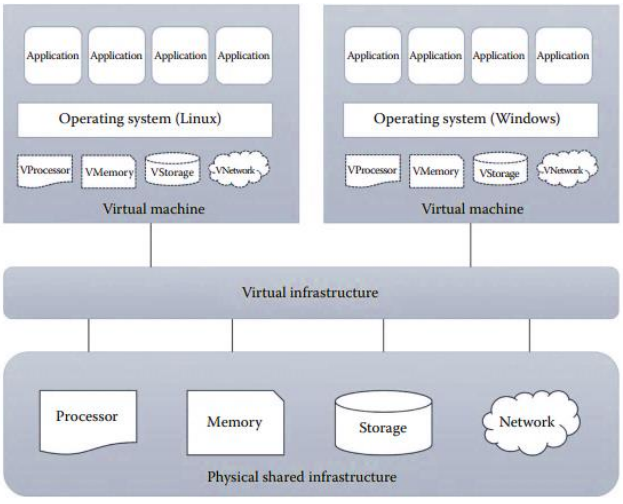
pemanfaatan sumber daya yang efisien dan peningkatan ROI, kemudahan administrasi, dan dukungan TI yang ramah lingkungan.

Virtualisasi adalah teknologi yang memungkinkan satu infrastruktur fisik berfungsi sebagai beberapa infrastruktur logis atau sumber daya. Virtualisasi tidak hanya terbatas pada perangkat keras, tetapi juga dalam berbagai bentuk seperti memori, prosesor, I/O, jaringan, OS, data, dan aplikasi. Berbagai bentuk virtualisasi akan dibahas pada bagian selanjutnya.

Sebelum virtualisasi, infrastruktur fisik tunggal digunakan untuk menjalankan satu OS dan aplikasinya, yang mengakibatkan penggunaan sumber daya yang kurang. Sifat nonshared dari perangkat keras memaksa organisasi untuk membeli perangkat keras baru untuk memenuhi kebutuhan komputasi tambahan mereka. Misalnya, jika ada organisasi yang ingin bereksperimen atau mensimulasikan ide baru mereka, mereka harus menggunakan sistem khusus yang terpisah untuk eksperimen yang berbeda. Jadi untuk menyelesaikan pekerjaan penelitian mereka dengan sukses, mereka cenderung membeli perangkat keras baru yang akan meningkatkan CapEx dan OpEx. Terkadang, jika organisasi tidak memiliki uang untuk berinvestasi lebih banyak pada sumber daya tambahan, mereka mungkin tidak dapat melakukan beberapa eksperimen berharga karena kekurangan sumber daya. Jadi, orang mulai berpikir untuk berbagi satu infrastruktur untuk berbagai keperluan dalam bentuk virtualisasi. Skenario komputasi sebelum dan sesudah virtualisasi masing-masing ditunjukkan pada Gambar 7.1 dan 7.2.




**Gambar 7.1** Sebelum virtualisasi.




**Gambar 7.2** Setelah virtualisasi.

Setelah virtualisasi diperkenalkan, berbagai OS dan aplikasi dapat berbagi satu infrastruktur fisik. Virtualisasi mengurangi jumlah besar yang diinvestasikan untuk membeli sumber daya tambahan. Virtualisasi menjadi pendorong utama dalam industri TI, khususnya dalam komputasi awan. Secara umum, istilah komputasi awan dan virtualisasi tidak sama. Ada perbedaan signifikan antara kedua teknologi ini, yang akan dibahas di bagian akhir bab ini.
Industri mengadopsi virtualisasi dalam organisasi mereka karena manfaat berikut:

- Pemanfaatan sumber daya yang lebih baik
- Meningkatkan ROI
- Pusat data dinamis
- Mendukung TI ramah lingkungan
- Memudahkan administrasi
- Meningkatkan pemulihan bencana

Meskipun virtualisasi menawarkan banyak manfaat, virtualisasi juga memiliki beberapa kelemahan:

- Titik kegagalan
- Menuntut infrastruktur kelas atas dan kuat
- Dapat menyebabkan kinerja yang lebih rendah
- Membutuhkan keahlian khusus

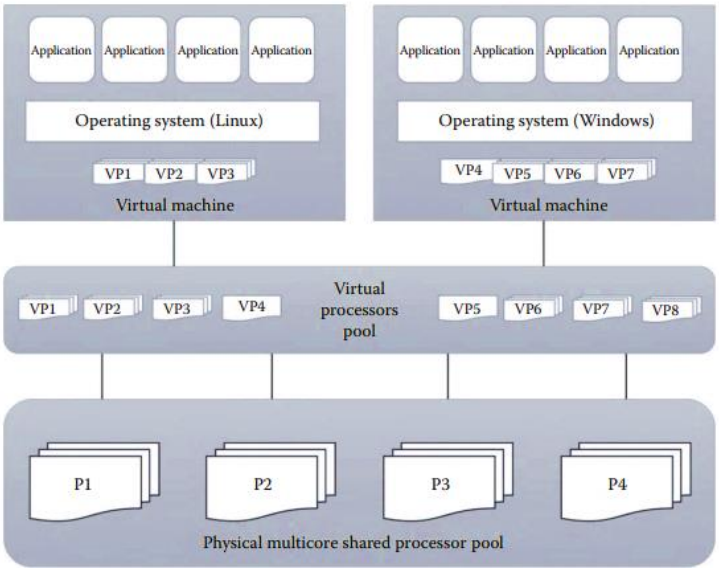
Bab ini berfokus pada peluang virtualisasi yang berbeda, pendekatan virtualisasi yang berbeda, peran hypervisor dalam virtualisasi, serangan yang menargetkan hypervisor, dan virtualisasi untuk komputasi awan.

## 7.2 PELUANG VIRTUALISASI

Virtualisasi adalah proses mengabstraksi sumber daya fisik ke kumpulan sumber daya virtual yang dapat diberikan ke mesin virtual (VM) apa pun. Sumber daya yang berbeda seperti memori, prosesor, penyimpanan, dan jaringan dapat divirtualisasikan menggunakan teknologi virtualisasi yang tepat. Pada bagian ini, kita akan membahas beberapa sumber daya yang dapat divirtualisasikan.

**Virtualisasi Prosesor**

Virtualisasi prosesor memungkinkan VM untuk berbagi prosesor virtual yang disarikan dari prosesor fisik yang tersedia di infrastruktur dasar. Lapisan virtualisasi memisahkan prosesor fisik ke kumpulan prosesor virtual yang digunakan bersama oleh VM. Lapisan virtualisasi biasanya berupa hypervisor apa saja. Virtualisasi prosesor dari satu perangkat keras diilustrasikan pada Gambar 7.3. Tetapi virtualisasi prosesor juga dapat dicapai dari server terdistribusi.

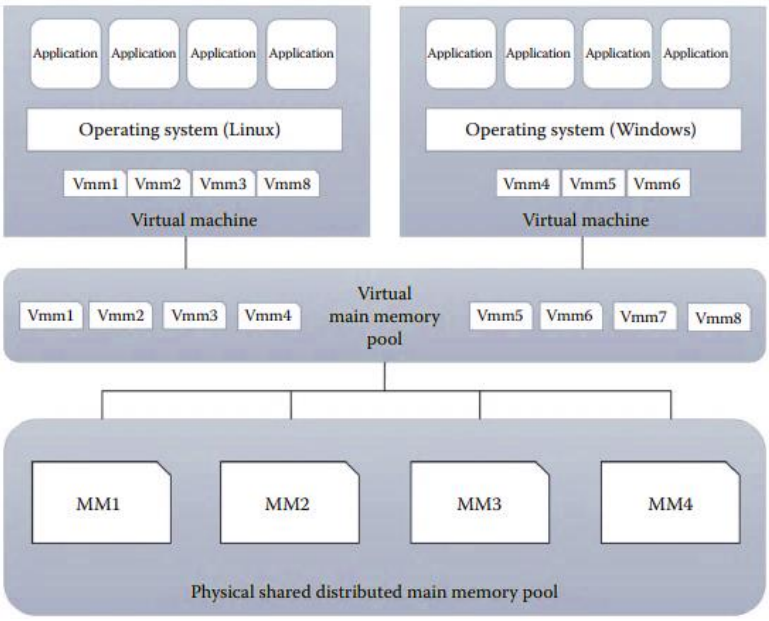



**Gambar 7.3** Virtualisasi prosesor.

**Virtualisasi Memori**

Teknik virtualisasi sumber daya penting lainnya adalah virtualisasi memori. Proses penyediaan memori utama virtual ke VM dikenal sebagai virtualisasi memori atau virtualisasi memori utama. Dalam virtualisasi memori utama, memori utama fisik dipetakan ke memori utama virtual seperti pada konsep memori virtual di sebagian besar OS. Ide utama virtualisasi memori utama adalah memetakan nomor halaman virtual ke nomor halaman fisik. Semua prosesor x86 modern mendukung virtualisasi memori utama.

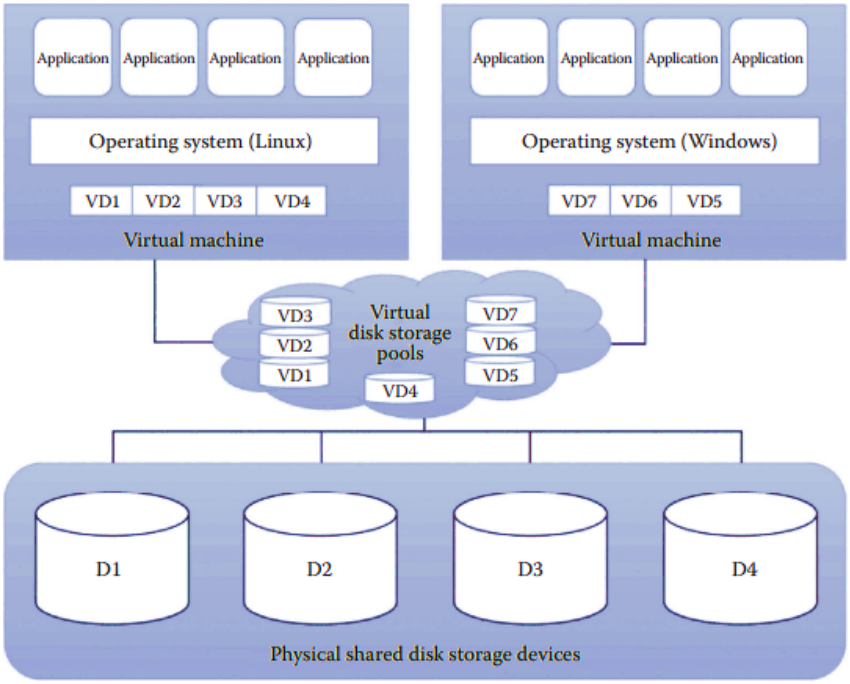
Virtualisasi memori utama juga dapat dicapai dengan menggunakan perangkat lunak hypervisor. Biasanya, di pusat data tervirtualisasi, memori utama yang tidak terpakai dari berbagai server akan dikonsolidasikan sebagai kumpulan memori utama virtual dan dapat diberikan ke VM. Konsep virtualisasi memori utama diilustrasikan pada Gambar 7.4.




**Gambar 7.4** Virtualisasi memori utama.

**Virtualisasi Penyimpanan**

Virtualisasi penyimpanan adalah bentuk virtualisasi sumber daya di mana beberapa disk penyimpanan fisik diabstraksikan sebagai kumpulan disk penyimpanan virtual ke VM. Biasanya, penyimpanan virtual akan disebut penyimpanan logis. Gambar 7.5 mengilustrasikan proses virtualisasi penyimpanan.

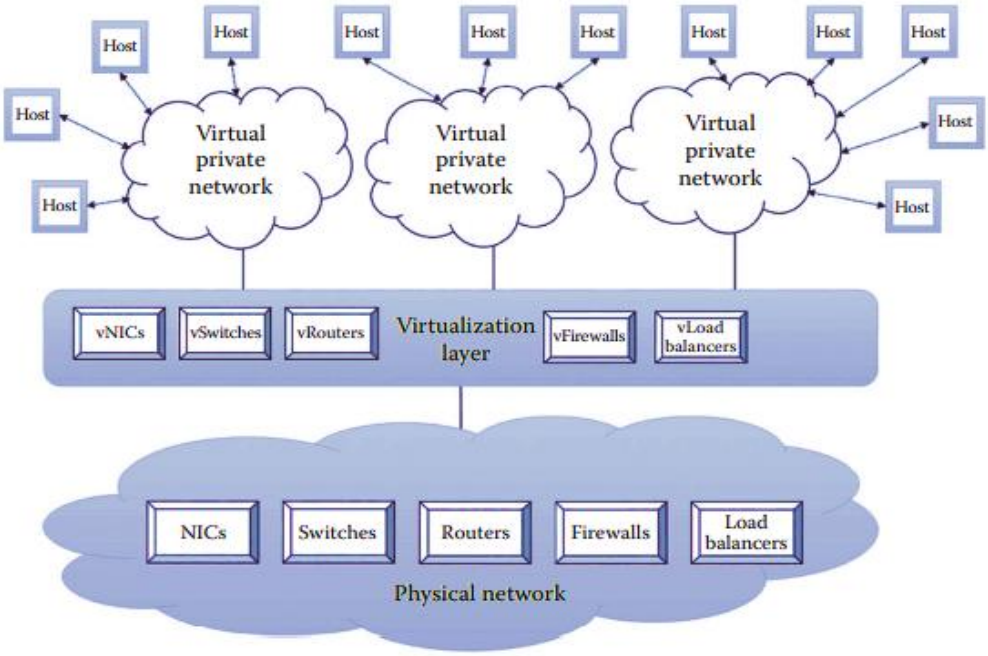



**Gambar 7.5** virtualisasi penyimpanan.

Virtualisasi penyimpanan terutama digunakan untuk memelihara cadangan atau replika data yang disimpan di VM. Ini dapat diperluas lebih lanjut untuk mendukung ketersediaan data yang tinggi. Itu juga dapat dicapai melalui hypervisor. Ini secara efisien menggunakan penyimpanan fisik yang mendasarinya. Teknik virtualisasi penyimpanan lanjutan lainnya adalah jaringan area penyimpanan (SAN) dan penyimpanan yang terpasang di jaringan (NAS).

**Virtualisasi Jaringan**

Virtualisasi jaringan adalah jenis virtualisasi sumber daya di mana jaringan fisik dapat diabstraksi untuk membuat jaringan virtual. Biasanya, komponen jaringan fisik seperti router, switch, dan Network Interface Card (NIC) akan dikendalikan oleh perangkat lunak virtualisasi untuk menyediakan komponen jaringan virtual. Jaringan virtual adalah entitas berbasis perangkat lunak tunggal yang berisi perangkat keras jaringan dan sumber daya perangkat lunak. Virtualisasi jaringan dapat dicapai dari jaringan internal atau dengan menggabungkan banyak jaringan eksternal. Keuntungan lain dari virtualisasi jaringan adalah memungkinkan komunikasi antara VM yang berbagi jaringan fisik. Ada berbagai jenis akses jaringan yang diberikan ke VM seperti jaringan yang dijembatani, terjemahan alamat jaringan (NAT), dan host saja. Konsep virtualisasi jaringan diilustrasikan pada Gambar 7.6.

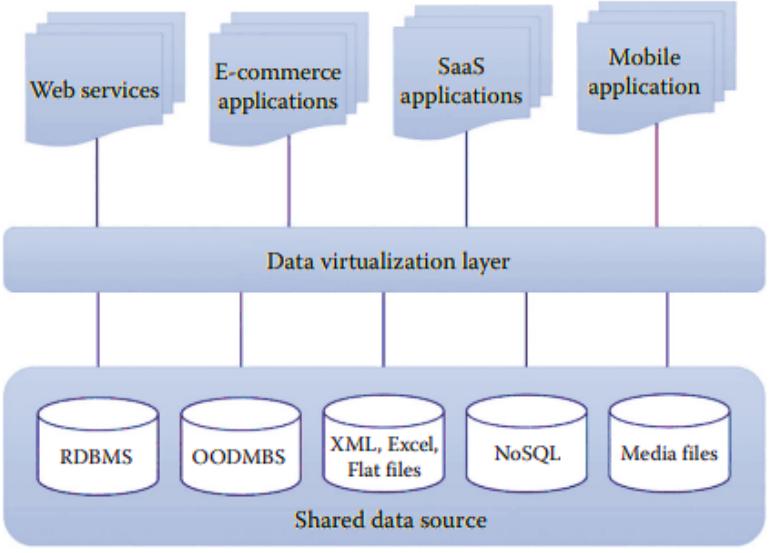



**Gambar 7.6** virtualisasi jaringan.

**Virtualisasi Data**

Virtualisasi data adalah kemampuan untuk mengambil data tanpa mengetahui jenisnya dan lokasi fisik tempat penyimpanannya. Ini mengumpulkan data heterogen dari sumber yang berbeda ke volume data logis/virtual tunggal. Data logis ini dapat diakses dari berbagai aplikasi seperti layanan web, aplikasi E-commerce, portal web, aplikasi Software as a Service (SaaS), dan aplikasi seluler.

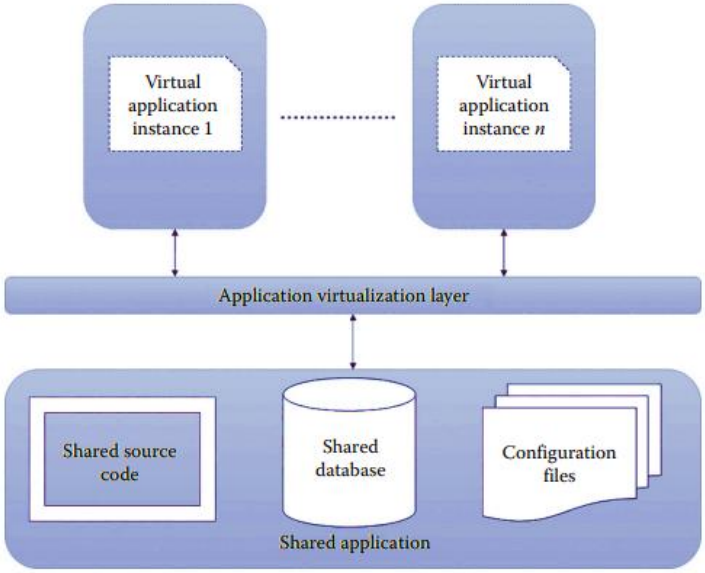
Virtualisasi data menyembunyikan jenis data dan lokasi data untuk aplikasi yang mengaksesnya. Ini juga memastikan akses titik tunggal ke data dengan menggabungkan data dari berbagai sumber. Ini terutama digunakan dalam integrasi data, intelijen bisnis, dan komputasi awan. Gambar 7.7 mewakili teknologi virtualisasi data.




**Gambar 7.7** virtualisasi data.

**Virtualisasi Aplikasi**

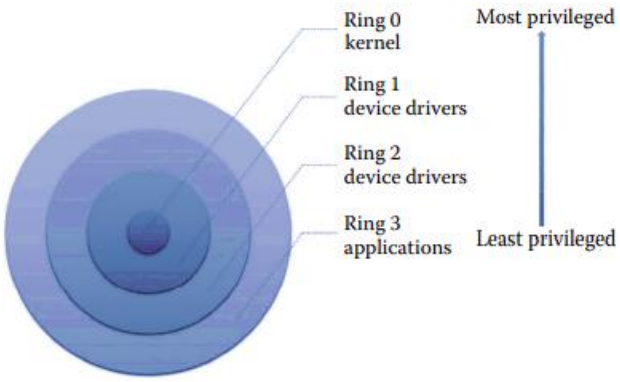
Virtualisasi aplikasi adalah teknologi yang memungkinkan untuk SaaS komputasi awan. Virtualisasi aplikasi menawarkan kemampuan kepada pengguna untuk menggunakan aplikasi tanpa perlu menginstal perangkat lunak atau alat apa pun di mesin. Di sini, kerumitan penginstalan alat klien atau perangkat lunak lain yang didukung berkurang. Biasanya, aplikasi akan dikembangkan dan dihosting di server pusat. Aplikasi yang dihosting akan divirtualisasikan lagi, dan pengguna akan diberikan salinan virtual yang terpisah/terisolasi untuk diakses. Konsep virtualisasi aplikasi diilustrasikan pada Gambar 7.8.




**Gambar 7.8** virtualisasi aplikasi.

## 7.3 PENDEKATAN VIRTUALISASI

Ada tiga pendekatan berbeda untuk virtualisasi. Sebelum membahasnya, penting untuk mengetahui tentang cincin perlindungan di OS. Cincin perlindungan digunakan untuk mengisolasi OS dari aplikasi pengguna yang tidak dipercaya. OS dapat dilindungi dengan tingkat hak istimewa yang berbeda. Dalam arsitektur ring proteksi, ring disusun dalam urutan hierarkis dari ring 0 hingga ring 3 seperti yang ditunjukkan pada Gambar 7.9. Dering 0 berisi program yang paling diistimewakan, dan dering 3 berisi program yang paling tidak diistimewakan. Biasanya, instruksi OS yang sangat tepercaya akan berjalan di ring 0, dan memiliki akses tidak terbatas ke sumber daya fisik. Dering 3 berisi aplikasi pengguna yang tidak dipercaya, dan membatasi akses ke sumber daya fisik. Dua dering lainnya (dering 1 dan dering 2) dialokasikan untuk driver perangkat. Arsitektur cincin perlindungan ini membatasi penyalahgunaan sumber daya dan perilaku berbahaya dari program tingkat pengguna yang tidak tepercaya. Misalnya, setiap aplikasi pengguna dari ring 3 tidak dapat secara langsung mengakses sumber daya fisik apa pun karena merupakan tingkat yang paling tidak diistimewakan. Tetapi kernel OS pada ring 0 dapat langsung mengakses sumber daya fisik karena ini adalah level yang paling diistimewakan.




**Gambar 7.9** Cincin perlindungan di OS.

Bergantung pada jenis virtualisasi, hypervisor dan OS tamu akan berjalan di tingkat hak istimewa yang berbeda. Biasanya, hypervisor akan berjalan dengan tingkat yang paling istimewa di ring 0, dan OS tamu akan berjalan di tingkat yang paling tidak diistimewakan daripada hypervisor. Ada tiga jenis pendekatan yang diikuti untuk virtualisasi:

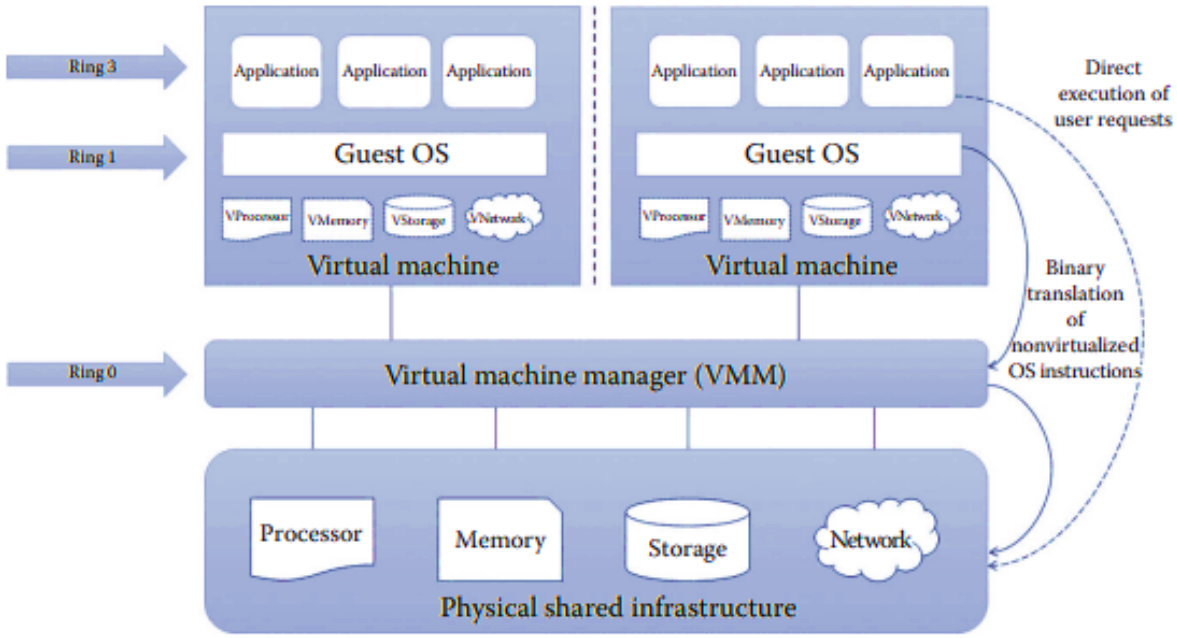
1. Virtualisasi penuh
2. Paravirtualisasi
3. Virtualisasi dengan bantuan perangkat keras

Setiap pendekatan virtualisasi dibahas secara rinci di bagian ini.

**Virtualisasi Penuh**

Dalam virtualisasi penuh, OS tamu benar-benar diabstraksikan dari infrastruktur yang mendasarinya. Lapisan virtualisasi atau manajer mesin virtual (VMM) sepenuhnya memisahkan OS tamu dari infrastruktur yang mendasarinya. OS tamu tidak menyadari bahwa itu divirtualisasi dan berpikir itu berjalan pada perangkat keras yang sebenarnya. Dalam pendekatan ini, hypervisor atau VMM berada di ring 0 dan menyediakan semua infrastruktur virtual yang diperlukan untuk VM.

OS tamu berada di ring 1 dan memiliki hak istimewa paling sedikit daripada hypervisor. Oleh karena itu, OS tidak dapat berkomunikasi dengan infrastruktur fisik secara langsung. Itu membutuhkan bantuan hypervisor untuk berkomunikasi dengan infrastruktur yang mendasarinya. Aplikasi pengguna berada di ring 3, seperti yang ditunjukkan pada Gambar 7.10. Pendekatan ini menggunakan terjemahan biner dan teknik eksekusi langsung. Terjemahan biner digunakan untuk menerjemahkan instruksi OS tamu nonvirtualisasi dengan urutan instruksi baru yang memiliki efek yang sama pada infrastruktur virtual. Di sisi lain, eksekusi langsung digunakan untuk permintaan aplikasi pengguna dimana aplikasi dapat langsung mengakses sumber daya fisik tanpa memodifikasi instruksi.




**Gambar 7.10** virtualisasi penuh.

*Pro*

- Pendekatan ini memberikan isolasi dan keamanan terbaik untuk VM.
- OS yang berbeda dapat berjalan secara bersamaan.
- OS tamu virtual dapat dengan mudah dipindahkan untuk bekerja di perangkat keras asli.
- Mudah diinstal dan digunakan dan tidak memerlukan perubahan apa pun di OS tamu.

*Kontra*

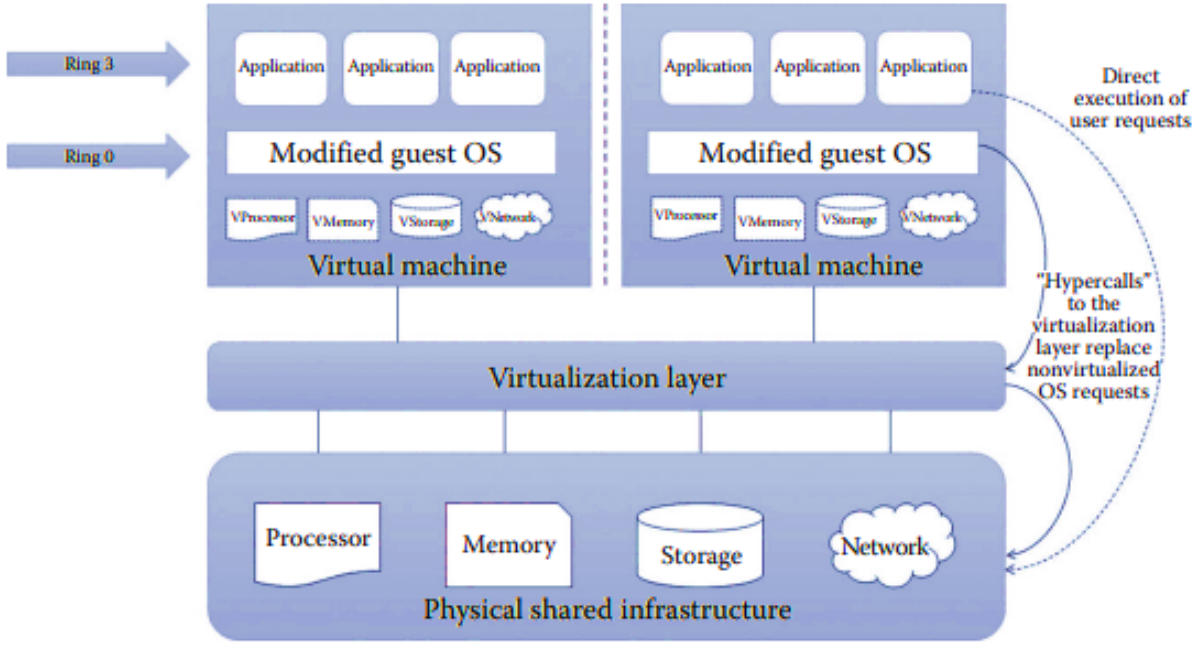
- Terjemahan biner adalah tambahan, overhead, dan mengurangi kinerja sistem secara keseluruhan.
- Ada kebutuhan untuk kombinasi perangkat keras dan perangkat lunak yang tepat.

**Paravirtualisasi**

Pendekatan ini juga dikenal sebagai virtualisasi parsial atau virtualisasi berbantuan OS dan memberikan simulasi parsial infrastruktur yang mendasarinya. Perbedaan utama antara virtualisasi penuh dan paravirtualisasi adalah OS tamu tahu bahwa itu berjalan di lingkungan virtualisasi dalam paravirtualisasi. Namun dalam virtualisasi penuh, informasi ini tidak diketahui oleh OS tamu. Perbedaan lainnya adalah bahwa paravirtualisasi menggantikan terjemahan permintaan OS nonvirtualisasi dengan hypercall. Hypercall mirip dengan panggilan sistem dan digunakan untuk komunikasi langsung antara OS dan hypervisor. Komunikasi langsung antara OS tamu dan hypervisor ini meningkatkan kinerja dan efisiensi. Dalam virtualisasi penuh, OS tamu akan digunakan tanpa modifikasi apa pun. Namun dalam paravirtualization, guest OS perlu dimodifikasi untuk menggantikan instruksi nonvirtualizable dengan hypercall.

Seperti yang ditunjukkan pada Gambar 7.11, OS tamu yang dimodifikasi berada di ring 0 dan aplikasi pengguna di ring 3. Karena OS tamu berada pada posisi istimewa, ia dapat berkomunikasi langsung ke lapisan virtualisasi tanpa terjemahan apa pun melalui hypercall.

Seperti dalam virtualisasi penuh, aplikasi pengguna diizinkan untuk mengakses infrastruktur yang mendasarinya secara langsung.




**Gambar 7.11** Paravirtualisasi.

*Pro*

- Ini menghilangkan overhead tambahan dari terjemahan biner dan karenanya meningkatkan efisiensi dan kinerja sistem secara keseluruhan.
- Lebih mudah diimplementasikan daripada virtualisasi penuh karena tidak memerlukan perangkat keras khusus.

*Kontra*

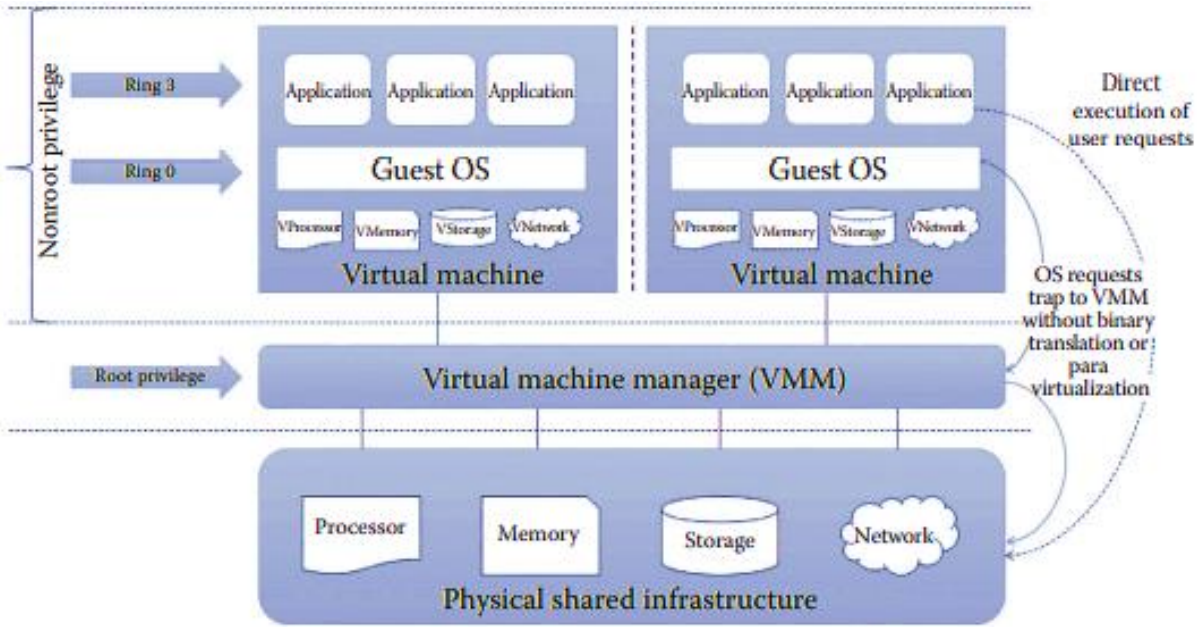
- Ada overhead modifikasi kernel OS tamu.
- OS tamu yang dimodifikasi tidak dapat dimigrasikan untuk berjalan di perangkat keras fisik.
- VM mengalami kekurangan kompatibilitas ke belakang dan sulit untuk bermigrasi ke host lain.

**Virtualisasi Berbantuan Perangkat Keras**

Dalam dua pendekatan sebelumnya, ada tambahan overhead terjemahan biner atau modifikasi OS tamu untuk mencapai virtualisasi. Namun dalam pendekatan ini, vendor perangkat keras itu sendiri, seperti Intel dan AMD, menawarkan dukungan untuk virtualisasi, yang menghilangkan banyak biaya tambahan yang terlibat dalam terjemahan biner dan modifikasi OS tamu. Vendor perangkat keras populer seperti Intel dan AMD telah memberikan ekstensi perangkat keras ke prosesor berbasis x86 mereka untuk mendukung virtualisasi.

Misalnya, Intel merilis Intel Virtualization Technology (VT-x) dan AMD merilis AMD-v untuk menyederhanakan teknik virtualisasi. Di VT-x, status tamu disimpan dalam struktur kontrol mesin virtual dan di AMD-v di blok kontrol mesin virtual. Dalam virtualisasi berbantuan perangkat keras, VMM memiliki tingkat hak istimewa (root privilege) tertinggi meskipun bekerja di bawah ring 0. OS berada di ring 0 dan aplikasi pengguna di ring 3, seperti yang

ditunjukkan pada Gambar 7.12. Berbeda dengan pendekatan virtualisasi lainnya, OS tamu dan aplikasi pengguna memiliki tingkat hak istimewa yang sama (tingkat hak istimewa nonroot). Seperti dibahas sebelumnya, teknik virtualisasi berbantuan perangkat keras menghilangkan terjemahan biner dan paravirtualisasi. Di sini, permintaan OS langsung menjebak hypervisor tanpa terjemahan apa pun. Seperti dalam pendekatan virtualisasi lainnya, permintaan pengguna langsung dieksekusi tanpa terjemahan apa pun.




**Gambar 7.12** Virtualisasi yang dibantu perangkat keras.

*Pro*

- Ini mengurangi overhead tambahan terjemahan biner dalam virtualisasi penuh.
- Ini menghilangkan modifikasi OS tamu di paravirtualization.

*Kontra*

- Hanya prosesor generasi baru yang memiliki kemampuan ini. Semua prosesor x86/x86_64 tidak mendukung fitur virtualisasi yang dibantu perangkat keras.
- Jumlah perangkap VM yang lebih banyak menghasilkan overhead CPU yang tinggi, skalabilitas yang terbatas, dan efisiensi yang lebih rendah dalam konsolidasi server.

Rangkuman dari pendekatan yang berbeda untuk virtualisasi diberikan pada Tabel 7.1.

**Tabel 7.1** Rangkuman Pendekatan Berbeda untuk Virtualisasi

| | **Virtualisasi Penuh** | **Paravirtualisasi** | **Virtualisasi Berbantuan Perangkat Keras** |
|---|---|---|---|
| Teknik | Terjemahan biner dan eksekusi langsung | Hypercall | Permintaan OS menjebak ke VMM tanpa terjemahan biner atau paravirtualisasi |
| Modifikasi OS tamu | Tidak | Ya | Tidak |
| Kesesuaian | Kompatibilitas yang sangat baik | Kompatibilitas buruk | Kompatibilitas yang sangat baik |

| | | | |
|---|---|---|---|
| Apakah hypervisor OS tamu independen? | Ya | Tidak | Ya |
| Pertunjukan | Bagus | Lebih baik dalam kasus tertentu | Adil |
| Posisi VMM dan tingkat keistimewaan | Dering 0 | Di bawah cincin 0 | Di bawah cincin 0 |
| | Hak istimewa root | | Hak istimewa root |
| Posisi tamu OS dan tingkat hak istimewa | Dering 1 | Dering 0 | Dering 0 |
| | Hak istimewa nonroot | Hak istimewa root | Hak istimewa nonroot |
| Vendor populer | VMware ESX | Xen | Microsoft, Besi Virtual, dan XenSorce |

## 7.4 HYPERVISOR

VM banyak digunakan sebagai pengganti mesin fisik di industri TI saat ini. VM mendukung solusi IT ramah lingkungan, dan penggunaannya meningkatkan pemanfaatan sumber daya, membuat tugas manajemen menjadi lebih mudah. Karena VM paling banyak digunakan, teknologi yang memungkinkan lingkungan virtual juga mendapat perhatian di industri dan akademisi. Lingkungan virtual dapat dibuat dengan bantuan perangkat lunak yang disebut hypervisor. Hypervisor adalah alat perangkat lunak yang berada di antara VM dan infrastruktur fisik dan menyediakan infrastruktur virtual yang diperlukan untuk VM. Umumnya, infrastruktur virtual berarti CPU virtual (vCPU), memori virtual, NIC virtual (vNIC), penyimpanan virtual, dan perangkat I/O virtual. Hypervisor juga disebut VMM. Mereka adalah pendorong utama dalam mengaktifkan virtualisasi di pusat data *cloud*. Ada berbagai hypervisor yang digunakan di industri TI. Beberapa contohnya adalah VMware, Xen, Hyper-V, KVM, dan OpenVZ. Berbagai jenis hypervisor, beberapa hypervisor populer di pasar, dan masalah keamanan dengan rekomendasi dibahas di bagian ini.

**Jenis Hypervisor**

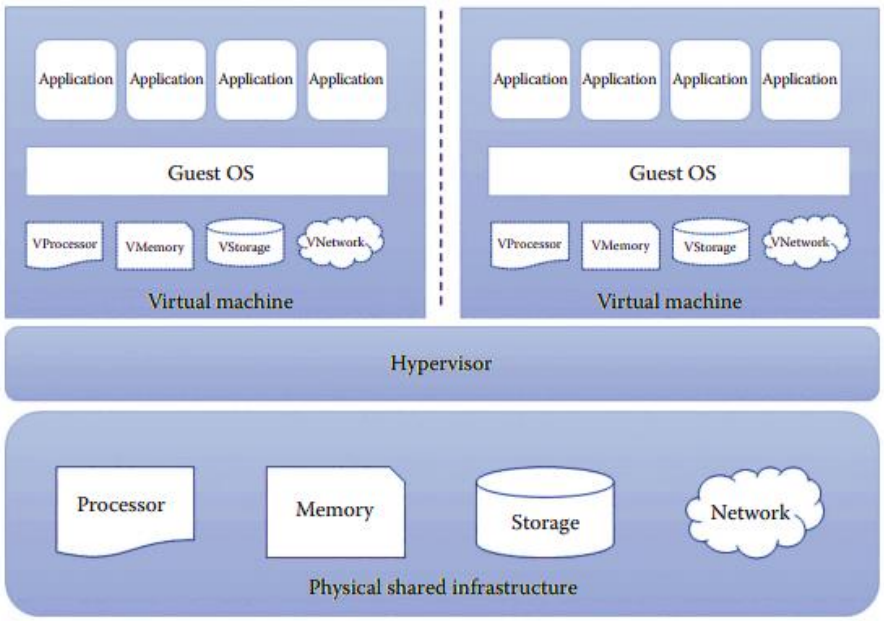
Sebelum hypervisor diperkenalkan, ada hubungan satu-ke-satu antara perangkat keras dan OS. Jenis komputasi ini menghasilkan sumber daya yang kurang dimanfaatkan. Setelah hypervisor diperkenalkan, itu menjadi hubungan satu-ke-banyak. Dengan bantuan hypervisor, banyak OS dapat berjalan dan berbagi satu perangkat keras. Hypervisor umumnya diklasifikasikan menjadi dua kategori:

1. Ketik 1 atau hypervisor logam polos
2. Tipe 2 atau hypervisor yang dihosting

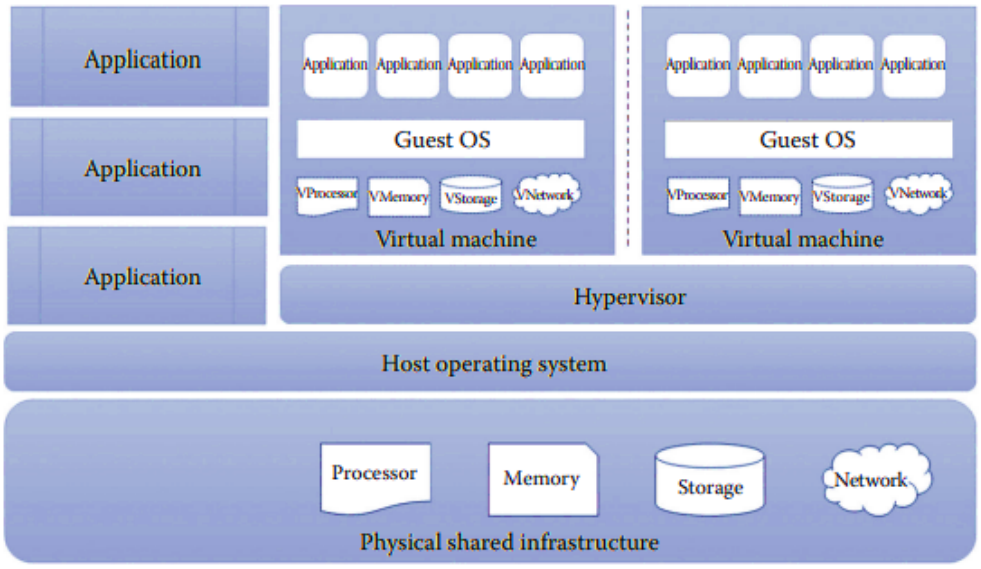
Perbedaan utama antara kedua jenis hypervisor ini adalah tipe 1 berjalan langsung di perangkat keras dan tipe 2 di OS host. Gambar 7.13 dan 7.14 masing-masing mengilustrasikan cara kerja hypervisor tipe 1 dan tipe 2.

Hypervisor tipe 1 juga dikenal sebagai bare metal atau hypervisor asli. Itu dapat menjalankan dan mengakses sumber daya fisik secara langsung tanpa bantuan OS host apa pun. Di sini, overhead komunikasi tambahan dengan OS host berkurang dan menawarkan

efisiensi yang lebih baik jika dibandingkan dengan hypervisor tipe 2. Jenis hypervisor ini digunakan untuk server yang menangani beban berat dan membutuhkan keamanan lebih. Beberapa contoh hypervisor tipe 1 termasuk Microsoft Hyper-V, Citrix XenServer, VMWare ESXi, dan Oracle VM Server for SPARC.




**Gambar 7.13** Ketik 1 atau hypervisor logam kosong.




**Gambar 7.14** Tipe 2 atau hypervisor yang dihosting.

Hypervisor tipe 2 juga dikenal sebagai hypervisor yang disematkan atau dihosting. Jenis hypervisor ini memerlukan OS host dan tidak memiliki akses langsung ke perangkat keras fisik. Jenis hypervisor ini diinstal pada OS host sebagai program perangkat lunak. OS host juga dikenal sebagai physical host, yang memiliki akses langsung ke perangkat keras yang mendasarinya. Kerugian utama dari pendekatan ini adalah jika OS host gagal atau macet, ini juga mengakibatkan VM mogok. Jadi, disarankan untuk menggunakan hypervisor tipe 2 hanya pada sistem klien yang efisiensinya kurang penting. Contoh hypervisor tipe 2 termasuk

VMWare Workstation dan Oracle Virtualbox. Ringkasan dari beberapa hypervisor populer yang ada di pasaran diberikan pada Tabel 7.2.

**Tabel 7.2** Ringkasan Hypervisor

| Hypervisor | Penjual | Jenis | Lisensi |
|---|---|---|---|
| Xen | Laboratorium Komputer Universitas Cambridge | Tipe 1 | GNU GPL v2 |
| VMWare ESXi | VMware, Inc. | Tipe 1 | Hak milik |
| Hiper-V | Microsoft | Tipe 1 | Hak milik |
| KVM | Buka aliansi virtualisasi | Tipe 2 | Lisensi publik umum GNU |
| stasiun kerja VMWare | VMware, Inc. | Tipe 2 | Shareware |
| Kotak Virtual Oracle | Perusahaan Oracle | Tipe 2 | Lisensi publik umum GNU versi 2 |

**Masalah dan Rekomendasi Keamanan**

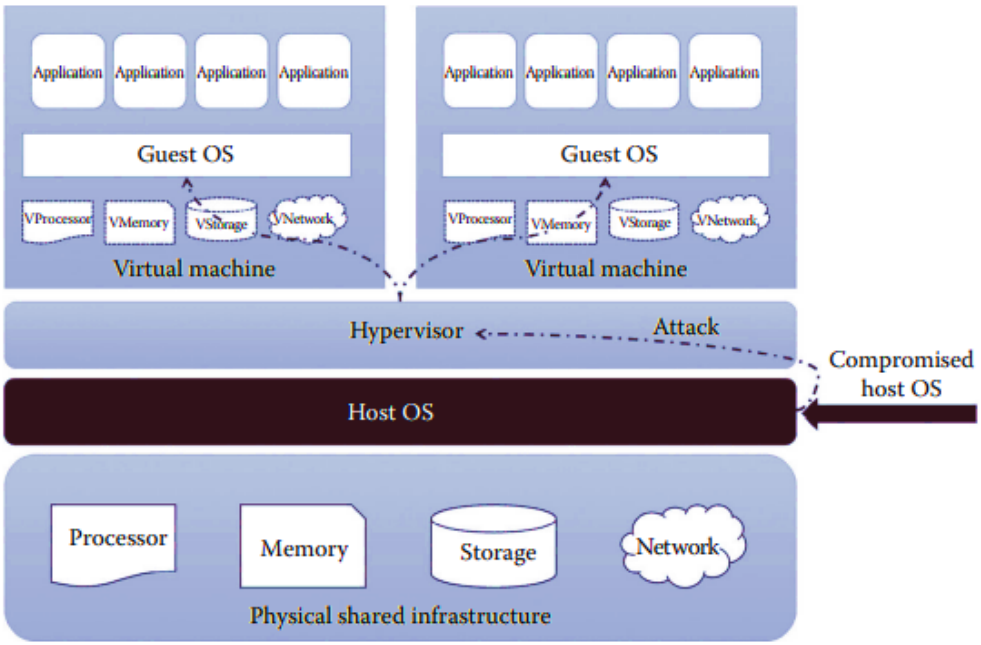
Hypervisor menciptakan lingkungan virtual di pusat data. Jadi, cara yang lebih baik untuk menyerang sumber daya adalah dengan menyerang hypervisor. Serangan hypervisor umumnya mengkompromikan hypervisor melalui kode berbahaya yang ditulis oleh penyerang mana pun untuk mengganggu atau merusak seluruh server. Dalam lingkungan tervirtualisasi, hypervisor adalah entitas dengan otoritas lebih tinggi yang memiliki akses langsung ke perangkat keras. Jadi, sebagian besar penyerang akan mengincar hypervisor sebagai entry point untuk menyerang sistem. Di hypervisor bare metal (tipe 1), sangat sulit untuk melakukan serangan karena diterapkan langsung pada perangkat keras. Tetapi hypervisor yang dihosting (tipe 2) lebih rentan terhadap serangan karena hypervisor berjalan di atas OS host. Ada dua kemungkinan menyerang hypervisor:

1. Melalui OS host
2. Melalui OS tamu

Menyerang melalui OS host: Serangan dari OS host dapat dilakukan dengan mengeksploitasi kerentanan OS host. Diketahui bahwa OS modern pun juga rentan terhadap serangan. Setelah OS dikompromikan, penyerang memiliki kendali penuh atas aplikasi yang berjalan di atas OS. Karena hypervisor (tipe 2) juga merupakan aplikasi yang berjalan di atas OS, ada kemungkinan untuk menyerang hypervisor melalui OS host yang disusupi. Ide di balik menyerang hypervisor melalui OS host diilustrasikan pada Gambar 7.15. Setelah penyerang mendapatkan kontrol penuh atas hypervisor melalui OS yang dikompromikan, penyerang akan dapat menjalankan semua instruksi istimewa yang dapat mengontrol perangkat keras yang sebenarnya. Penyerang dapat melakukan aktivitas berbahaya berikut:

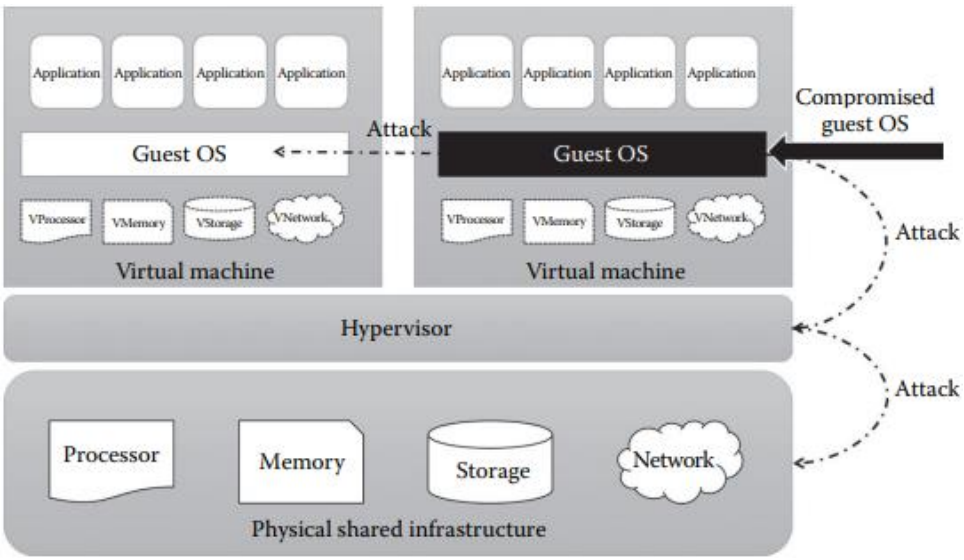
- Denial of service attack, dimana penyerang dapat menolak sumber daya virtual ketika ada permintaan dari VM baru
- Mencuri informasi rahasia yang disimpan di VM

Serang melalui OS tamu: Hypervisor juga dapat disusupi atau diserang dari skrip berbahaya dari OS tamu yang disusupi.




**Gambar 7.15** Serang melalui OS host.

Karena OS tamu berkomunikasi dengan hypervisor untuk mendapatkan sumber daya virtual, kode jahat apa pun dari OS tamu atau VM dapat membahayakan hypervisor. Biasanya, serangan dari OS tamu akan mencoba menyalahgunakan sumber daya yang mendasarinya. Ide di balik serangan melalui guest OS diilustrasikan pada Gambar 7.16. Seperti yang ditunjukkan pada gambar, penyerang akan mencoba menyerang atau mengkompromikan hypervisor dari VM berbahaya. Setelah hypervisor dikompromikan dari OS tamu atau VM jahat, hypervisor dapat menyalahgunakan hak tinggi hypervisor pada perangkat keras. Jenis serangan ini dimungkinkan pada hypervisor tipe 1 dan tipe 2.




**Gambar 7.16** Serang melalui OS tamu.

Setelah hypervisor dikompromikan, penyerang dapat melakukan aktivitas berbahaya berikut:

- Dapatkan akses tidak sah ke VM lain yang berbagi perangkat keras fisik.
- Penyerang dapat menggunakan sumber daya perangkat keras sepenuhnya untuk melancarkan serangan kehabisan sumber daya, dll.

Rekomendasi untuk menghindari serangan hypervisor: Sebagian besar serangan pada hypervisor dilakukan melalui OS host atau OS tamu. Jadi, OS harus dilindungi dari penyerang jahat atau hypervisor harus dilindungi dari OS yang disusupi. Ada beberapa praktik terbaik untuk menjaga agar hypervisor tetap aman:

- Perbarui perangkat lunak hypervisor dan OS host secara teratur.
- Putuskan sambungan sumber daya fisik yang tidak terpakai dari sistem host atau hypervisor.
- Aktifkan sedikit hak istimewa untuk hypervisor dan OS tamu untuk menghindari serangan melalui akses tidak sah.
- Terapkan alat pemantauan di hypervisor untuk mendeteksi/mencegah aktivitas jahat.
- Isolasi tamu yang kuat.
- Terapkan kebijakan kontrol akses wajib.

## 7.5 DARI VIRTUALISASI KE *CLOUD COMPUTING*

Banyak pengguna solusi TI saat ini menganggap teknologi virtualisasi dan komputasi awan sebagai hal yang sama. Namun kedua teknologi tersebut sebenarnya berbeda, atau dengan kata lain virtualisasi bukanlah *cloud computing*. Kami dapat membuktikan klaim ini dengan parameter berikut:

1. Jenis layanan: Umumnya, virtualisasi menawarkan lebih banyak layanan infrastruktur daripada layanan platform dan aplikasi. Tetapi komputasi awan menawarkan semua layanan infrastruktur (IaaS), platform (PaaS), dan perangkat lunak (SaaS).
2. Pengiriman layanan: Pengiriman layanan dalam komputasi awan sesuai permintaan dan memungkinkan pengguna akhir untuk menggunakan layanan awan sesuai kebutuhan. Tetapi virtualisasi tidak dibuat untuk layanan sesuai permintaan.
3. Penyediaan layanan: Dalam komputasi awan, penyediaan otomatis dan swalayan dimungkinkan untuk pengguna akhir, sedangkan dalam virtualisasi, tidak mungkin dan banyak pekerjaan manual diperlukan dari penyedia atau administrator sistem untuk menyediakan layanan kepada pengguna akhir.
4. Orkestrasi layanan: *Cloud computing* memungkinkan orkestrasi layanan dan komposisi layanan untuk memenuhi kebutuhan pengguna akhir. Beberapa penyedia juga menyediakan orkestrasi layanan otomatis kepada pengguna akhir. Namun dalam virtualisasi, mengatur layanan yang berbeda untuk mendapatkan layanan komposit tidak dimungkinkan.
5. Elastisitas: Salah satu karakteristik penting yang membedakan komputasi awan dari virtualisasi adalah elastisitas. Dalam *cloud computing*, kita dapat menambah atau menghapus infrastruktur secara dinamis sesuai dengan kebutuhan, dan menambah atau

menghapus infrastruktur secara otomatis. Tetapi virtualisasi gagal memberikan elastisitas karena menghentikan dan memulai VM bersifat manual dan juga sulit.

6. Pemirsa yang ditargetkan: Pemirsa yang ditargetkan dari kedua teknologi ini juga berbeda. Komputasi awan menargetkan penyedia layanan untuk pemanfaatan sumber daya yang tinggi dan ROI yang lebih baik. Pada saat yang sama, ini juga memfasilitasi pengguna akhir untuk menghemat uang dengan menggunakan layanan sesuai permintaan. Dalam kasus virtualisasi, audiens yang ditargetkan hanyalah penyedia layanan atau pemilik TI, bukan pengguna akhir.

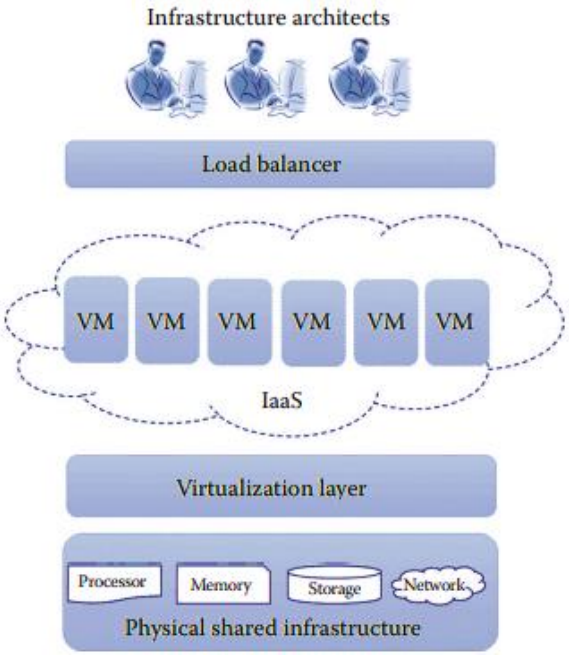
Dengan pembahasan singkat ini, dapat disimpulkan bahwa *cloud computing* dan virtualisasi itu berbeda. Namun mungkin ada beberapa pertanyaan yang akan muncul dari para pemilik TI: "Saya sudah berinvestasi lebih banyak dalam teknologi virtualisasi, apakah saya perlu mengubah segalanya untuk mendapatkan manfaat komputasi awan?" Jawaban atas pertanyaan ini adalah tidak karena *cloud computing* menggunakan virtualisasi untuk penyampaian layanannya. Komputasi awan dapat berjalan di lingkungan virtual apa pun karena virtualisasi adalah salah satu teknologi yang memungkinkan untuk komputasi awan. Tentu saja, tanpa virtualisasi, komputasi awan mungkin tidak ada. Komputasi awan menggunakan virtualisasi untuk pemanfaatan sumber daya yang lebih baik dan digabungkan dengan komputasi utilitas untuk menguntungkan penyedia layanan, pengembang, dan pengguna akhir. Dengan kata lain, kita dapat mengatakan bahwa komputasi awan membawa virtualisasi ke langkah selanjutnya. Komputasi awan dan virtualisasi menyatu dalam pemanfaatan sumber daya yang lebih baik, dan virtualisasi berhenti di sana sedangkan komputasi awan bergerak selangkah lebih maju dan bergabung dengan komputasi utilitas untuk menyediakan TI sebagai layanan. Ada berbagai model layanan *cloud* yang tersedia, yaitu *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), dan SaaS. Pada bagian ini, kita akan membahas bagaimana *cloud computing* menggunakan teknologi virtualisasi untuk menyediakan layanan *cloud* yang berbeda.

**IaaS**

Model pengiriman layanan *cloud computing* yang memungkinkan pelanggan untuk mengakses sumber daya sebagai layanan dari pusat data penyedia layanan dikenal sebagai model Infrastructure as a Service (IaaS). Konsep virtualisasi sepenuhnya digunakan dalam lapisan infrastruktur komputasi awan. Layanan IaaS menawarkan memori virtual, prosesor virtual, penyimpanan virtual, dan jaringan virtual untuk menjalankan VM.

Layanan IaaS menggunakan memori, prosesor, penyimpanan, dan virtualisasi jaringan infrastruktur yang mendasarinya. Lapisan IaaS menggunakan hypervisor untuk abstrak sumber daya yang mendasari untuk VM. Pusat data virtual tidak akan hanya disebut sebagai pusat data *cloud*. Pusat data tervirtualisasi akan disebut sebagai pusat data *cloud* jika memberikan layanan berdasarkan pembayaran per penggunaan. Biasanya, untuk mencapai layanan IaaS, hypervisor tipe 1 akan dipilih daripada hypervisor tipe 2 karena hypervisor tipe 1 secara langsung mengakses perangkat keras yang mendasarinya. IaaS umumnya disediakan dalam bentuk VM yang menggunakan sumber daya virtual yang disarikan dari sumber daya fisik.

Biasanya, pusat data *cloud* sebenarnya akan berisi mesin server jaringan untuk menyediakan layanan infrastruktur besar-besaran. Jadi, setiap kali satu server kelebihan beban dengan banyak VM, permintaan tambahan akan dipindahkan ke server fisik gratis lainnya dengan menggunakan penyeimbang beban. Ada banyak penyedia IaaS yang tersedia di pasar, termasuk Amazon, Microsoft, OpenStack, Eucalyptus, dan *Cloud*Stack. Mekanisme penyediaan layanan umum layanan IaaS diilustrasikan pada Gambar 7.17.



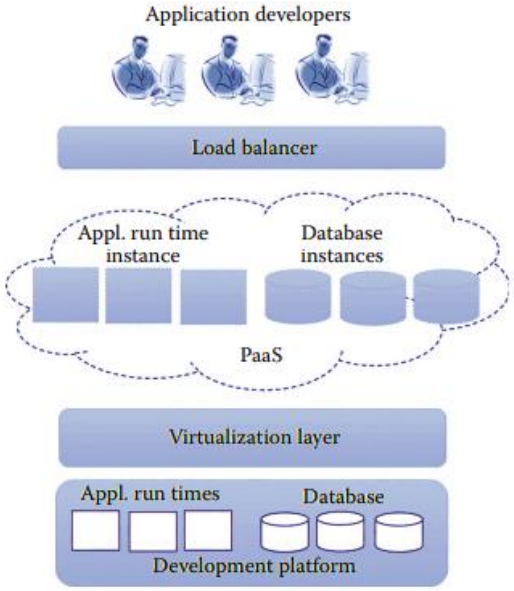

**Gambar 7.17** IaaS.

**PaaS**

Platform as a Service (PaaS) memungkinkan pengguna akhir untuk mengembangkan dan menyebarkan aplikasi secara online dengan menggunakan platform pengembangan virtual yang disediakan oleh penyedia layanan. Umumnya, penyedia layanan akan menyediakan semua alat pengembangan sebagai layanan kepada pengguna akhir melalui Internet. Pengguna akhir tidak perlu menginstal lingkungan pengembangan terintegrasi (IDE), bahasa pemrograman, dan pustaka komponen apa pun di mesin mereka untuk mengakses layanan.

Bahasa pemrograman, basis data, runtime bahasa, middleware, dan pustaka komponen akan diberikan kepada pelanggan dengan mengabstraksi platform sebenarnya yang berjalan di pusat data penyedia. Secara umum, penerapan aplikasi yang dikembangkan menggunakan layanan PaaS bergantung pada jenis model penerapan *cloud*. Jika pengguna akhir memilih model penerapan *cloud* publik apa pun, aplikasi akan dihosting sebagai aplikasi di luar lokasi. Jika pengguna memilih model private deployment, aplikasi akan diakses sebagai aplikasi on-premise. Umumnya, layanan PaaS menggunakan virtualisasi tingkat OS, tingkat database, tingkat bahasa pemrograman untuk menyediakan platform pengembangan virtual kepada pengguna akhir. Umumnya, penyedia PaaS akan menyediakan berbagai alat klien seperti WebCLI, REST API, dan UI Web kepada pengembang untuk mengakses platform virtual.

Beberapa penyedia PaaS mengizinkan pengembangan offline dengan mengintegrasikan dengan IDE seperti gerhana untuk membuat lingkungan pengembangan tersedia. Pengembang tidak perlu online untuk menggunakan layanan mereka. Mereka dapat bekerja secara offline dan mendorong aplikasi secara online kapan pun siap untuk diterapkan. Di sini, skalabilitas aplikasi merupakan faktor yang sangat penting. Skalabilitas aplikasi dapat dicapai dengan penyeimbang beban yang tepat yang mentransfer beban ekstra ke server baru. Contoh penyedia layanan PaaS termasuk Google App Engine, Microsoft Windows Azure, Redhat OpenShift, dan force.com. Gambaran umum dari layanan PaaS diilustrasikan pada Gambar 7.18.

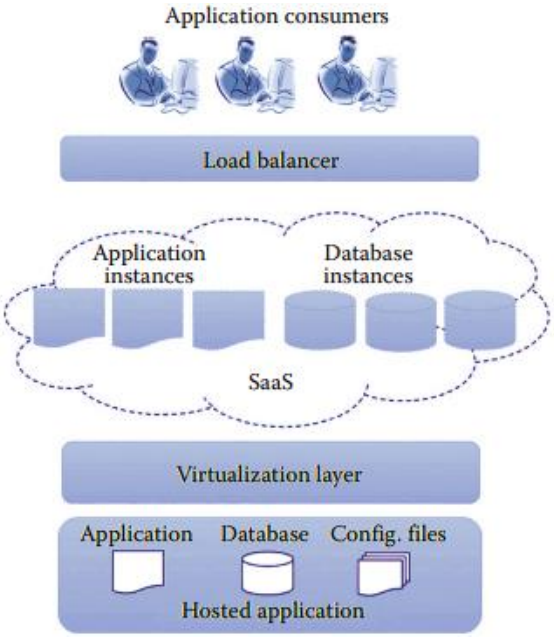



**Gambar 7.18** PaaS.

**SaaS**

Seperti infrastruktur dan platform, aplikasi perangkat lunak juga dapat divirtualisasikan. Model pengiriman perangkat lunak yang memungkinkan pelanggan untuk mengakses perangkat lunak yang dihosting di pusat data penyedia layanan melalui Internet dikenal sebagai Perangkat Lunak sebagai Layanan (SaaS).

Secara umum, SaaS adalah aplikasi berbasis langganan daripada aplikasi berlisensi. Untuk mengakses aplikasi SaaS, pelanggan tidak perlu menginstalnya di mesin mereka. Dengan web browser sederhana, mereka dapat mengakses aplikasi dari pusat data penyedia layanan melalui Internet. SaaS menggunakan virtualisasi tingkat aplikasi untuk menyebarkan aplikasi. Aplikasi SaaS memungkinkan banyak pelanggan untuk berbagi instance aplikasi yang sama. Teknologi ini dikenal sebagai multitenancy. Karena banyak pengguna yang berbagi aplikasi, beban aplikasi akan lebih banyak dan tidak dapat diprediksi. Kemampuan menangani beban ekstra menentukan skalabilitas aplikasi. Skalabilitas aplikasi akan ditingkatkan oleh penyeimbang beban perangkat lunak, yang akan mentransfer beban tambahan ke server aplikasi/database baru. Di sini, beberapa instance aplikasi dan instance database akan dibuat untuk memastikan skalabilitas tinggi. Beberapa aplikasi SaaS yang populer adalah Google Docs, Google Drive, dan Microsoft Office 360. Gambaran umum aplikasi SaaS diilustrasikan pada Gambar 7.19.




**Gambar 7.19** SaaS.

Virtualisasi digunakan sebagai teknologi yang memungkinkan untuk menyediakan infrastruktur multitenant, platform pengembangan, dan SaaS. Selain itu, ada layanan *cloud* lain yang menggunakan virtualisasi seperti *Network as a Service* menggunakan virtualisasi jaringan, Penyimpanan sebagai Layanan menggunakan virtualisasi penyimpanan, dan Database sebagai Layanan menggunakan virtualisasi basis data.

## 7.6 RINGKASAN

Virtualisasi adalah teknologi yang banyak digunakan di industri TI untuk meningkatkan pemanfaatan sumber daya dan ROI. Ini memungkinkan infrastruktur fisik yang sama untuk dibagi antara beberapa OS dan aplikasi. Manfaat lain dari virtualisasi meliputi pusat data yang dinamis, dukungan TI yang ramah lingkungan, kemudahan administrasi, dan pemulihan bencana yang lebih baik. Ada tiga jenis pendekatan yang digunakan untuk mencapai virtualisasi, yaitu full virtualization, paravirtualization, dan hardware-assisted virtualization. Virtualisasi penuh sepenuhnya memisahkan OS tamu dari infrastruktur yang mendasarinya. Paravirtualization memberikan abstraksi parsial OS tamu dari infrastruktur yang mendasarinya dengan sedikit modifikasi OS tamu. Dalam virtualisasi berbantuan perangkat keras, vendor perangkat keras itu sendiri menawarkan dukungan untuk virtualisasi. Hypervisor adalah pendorong utama dalam mengaktifkan virtualisasi di pusat data *cloud* berskala besar.

Ada dua jenis hypervisor yang tersedia, yaitu hypervisor tipe 1 atau bare metal dan hypervisor tipe 2 atau host. Hypervisor tipe 1 dapat langsung berinteraksi dengan infrastruktur yang mendasarinya tanpa bantuan OS host. Hypervisor tipe 2 membutuhkan OS host untuk berinteraksi dengan infrastruktur yang mendasarinya. Karena hypervisor digunakan sebagai teknologi yang memungkinkan di pusat data tervirtualisasi, ada berbagai jenis serangan yang ditargetkan pada hypervisor untuk mengganggu server. Biasanya, serangan dilakukan oleh kode berbahaya untuk mengkompromikan hypervisor. Serangan dapat menargetkan OS tamu

atau OS host. Serangan dapat dikurangi dengan isolasi tamu yang kuat, pembaruan yang sering, mengaktifkan kebijakan hak istimewa, alat pemantauan, dll. Virtualisasi membantu dalam menciptakan lingkungan *cloud* multitenant, di mana satu instance sumber daya dapat digunakan bersama oleh banyak pengguna. *Cloud computing* dan virtualisasi berbeda. Komputasi awan menggunakan virtualisasi dengan komputasi utilitas untuk menyediakan berbagai layanan seperti IaaS, PaaS, dan SaaS.

**Tinjau Poin**

- Virtualisasi adalah teknologi yang mengubah komputasi dari infrastruktur fisik menjadi infrastruktur logis.
- Prosesor virtualisasi adalah proses pengabstraksian prosesor fisik ke kumpulan prosesor virtual.
- Virtualisasi memori adalah proses penyediaan memori utama virtual ke VM yang dipisahkan dari memori utama fisik.
- Virtualisasi penyimpanan adalah bentuk virtualisasi sumber daya di mana beberapa penyimpanan fisik diabstraksikan sebagai penyimpanan logis ganda.
- Virtualisasi jaringan adalah proses pengabstraksian komponen jaringan fisik untuk membentuk jaringan virtuaL.
- Virtualisasi data menggabungkan data heterogen dari sumber yang berbeda ke satu volume data logis atau virtual.
- Virtualisasi aplikasi memungkinkan pengguna untuk mengakses contoh virtual dari aplikasi yang dihosting secara terpusat tanpa instalasi.
- Cincin perlindungan digunakan untuk mengisolasi OS dari aplikasi pengguna yang tidak dipercaya.
- Virtualisasi penuh adalah proses mengabstraksi sepenuhnya infrastruktur fisik yang mendasarinya dengan terjemahan biner dan eksekusi langsung.
- Paravirtualisasi atau virtualisasi berbantuan OS mengabstraksi sebagian infrastruktur yang mendasarinya dengan hypercall.
- Virtualisasi berbantuan perangkat keras menghilangkan overhead terjemahan biner dan hypercall, di mana vendor perangkat keras itu sendiri mendukung virtualisasi.
- Hypervisor atau VMM adalah alat perangkat lunak yang memungkinkan virtualisasi.
- Hypervisor Bare metal atau hypervisor tipe 1 dapat berjalan pada infrastruktur fisik tanpa bantuan apa pun dari OS host.
- Hypervisor yang dihosting atau hypervisor tipe 2 memerlukan bantuan OS host untuk berkomunikasi dengan infrastruktur yang mendasarinya.
- Serangan hypervisor sebagian besar menargetkan VMM dengan kode berbahaya baik dari OS tamu atau OS host.
- Komputasi awan berbeda dari virtualisasi melalui jenis layanan, pemberian layanan, elastisitas, dll.
- IaaS menggunakan prosesor, memori, penyimpanan, dan virtualisasi jaringan untuk menyediakan layanan infrastruktur.
- PaaS memvirtualisasikan platform pengembangan dan menyediakannya sebagai layanan bagi para pengembang.

- SaaS memungkinkan beberapa pengguna akhir untuk berbagi contoh tunggal perangkat lunak yang dihosting secara terpusat.

**Latihan Soal**

1. Apa itu virtualisasi? Sebutkan keuntungan dan kerugiannya.
2. Jelaskan bagaimana virtualisasi mengubah komputasi di industri TI.
3. Jelaskan secara singkat bagaimana sumber daya perangkat keras seperti prosesor, memori, penyimpanan, dan jaringan dapat divirtualisasi.
4. Tulis catatan singkat tentang virtualisasi data dan virtualisasi aplikasi.
5. Apa itu cincin pelindung? Jelaskan bagaimana ini digunakan dalam virtualisasi.
6. Jelaskan berbagai pendekatan yang digunakan untuk mencapai virtualisasi dengan diagram yang rapi.
7. Membedakan teknik full virtualization, paravirtualization, dan hardware-assisted virtualization.
8. Apa peran hypervisor dalam virtualisasi? Jelaskan secara singkat berbagai jenis hypervisor dengan diagram yang rapi.
9. Bedakan hypervisor tipe 1 dan tipe 2.
10. Jelaskan berbagai serangan yang ditargetkan pada hypervisor dengan diagram yang rapi.
11. Merekomendasikan beberapa praktik terbaik untuk menghindari/mencegah serangan terhadap hypervisor.
12. Apakah virtualisasi dan komputasi awan itu sama? Benarkan jawaban Anda.
13. Jelaskan perbedaan *cloud computing* dengan virtualisasi.
14. Bandingkan dan kontraskan komputasi awan dan virtualisasi.
15. Jelaskan bagaimana virtualisasi digunakan sebagai teknologi yang memungkinkan dalam memberikan layanan *cloud* seperti IaaS, PaaS dan SaaS.

# BAB 8
# MODEL PEMROGRAMAN UNTUK *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan dari bab ini adalah untuk

- Berikan penjelasan singkat tentang model pemrograman yang tersedia di *cloud*
- Tunjukkan kelebihan dan kekurangan masing-masing model pemrograman
- Tunjukkan karakteristik dari masing-masing model pemrograman
- Jelaskan dasar kerja dari masing-masing model pemrograman
- Tunjukkan fitur utama dari model pemrograman
- Mendiskusikan kesesuaian setiap model pemrograman dengan jenis aplikasi yang berbeda

**Pengantar**

Komputasi awan adalah area luas yang telah menarik perhatian beberapa orang dan sekarang lebih dari sekadar kata kunci. Teknologi ini adalah salah satu dari sedikit yang secara langsung mempengaruhi manusia. Secara sederhana, *cloud* menawarkan layanan kepada pelanggannya dalam beberapa cara. Salah satu cara penting adalah melalui *Software as a Service* (SaaS). Di SaaS, aplikasi perangkat lunak diberikan sebagai layanan kepada pengguna. Ada beberapa properti yang terkait dengan aplikasi *cloud*: skalabilitas, elastisitas, dan multitenancy. Setiap aplikasi *cloud* harus memiliki properti yang disebutkan di atas.

Inilah yang membedakan *cloud* dari aplikasi konvensional. Ini adalah fakta yang diketahui bahwa aplikasi apa pun melibatkan strategi pengembangan tertentu. Secara umum, semua proses pengembangan ditandai dengan penggunaan model pemrograman tertentu yang dipilih berdasarkan jenis aplikasi yang kami kembangkan. Properti aplikasi memainkan peran yang sangat penting dalam memilih model pemrograman. Dengan demikian, pengembangan aplikasi *cloud* ini menghadapi tantangan tertentu.

Komputasi awan adalah teknologi baru, dan tidak banyak bahasa pemrograman yang dikembangkan untuk itu. Bab ini secara luas membahas model pemrograman yang tersedia di *cloud*. Berdasarkan pendekatan atau metodenya, model pemrograman dapat diklasifikasikan menjadi dua. Yang pertama menggunakan model pemrograman yang ada dan memperluas atau memodifikasinya agar sesuai dengan platform *cloud*, dan yang kedua adalah dengan mengembangkan model baru yang sesuai untuk *cloud*. Ini akan dibahas dalam dua bagian terpisah. Bagian pertama dari bab ini memberikan pengantar singkat tentang model pemrograman. Ini juga membahas tentang komputasi awan, propertinya, dan dampak bidang ini terhadap dunia teknologi dan pasar. Bagian ini juga membahas perbedaan antara aplikasi biasa dan aplikasi *cloud*, serta alasan utama di balik pencarian model pemrograman baru. Model dan bahasa pemrograman yang ada akan dibahas terlebih dahulu, diikuti oleh model pemrograman baru yang dirancang khusus untuk *cloud*. Bab ini diakhiri dengan ringkasan dan beberapa pertanyaan ulasan.

## 8.1 PENDAHULUAN

Model pemrograman adalah cara tertentu atau metode atau pendekatan yang diikuti untuk pemrograman. Ada beberapa model pemrograman yang tersedia secara umum, masing-masing memiliki kelebihan dan kekurangannya sendiri. Model pemrograman membentuk dasar untuk setiap perangkat lunak atau pendekatan pengembangan aplikasi. Properti model pemrograman ini menentukan penggunaan dan dampaknya. Sejauh menyangkut model pemrograman umum, ini terus berkembang. Setiap langkah evolusioner ditandai dengan perubahan tertentu pada metodologi yang ada, dan pada setiap level, fungsionalitas baru ditambahkan, yang memfasilitasi pemrogram dalam satu atau beberapa cara. Seiring berkembangnya teknologi, begitu pula masalah yang dihadapi. Satu model pemrograman tidak bisa menjadi solusi untuk semua masalah, bahkan dengan banyak kemajuan. Dalam kasus ini, model pemrograman baru dikembangkan untuk serangkaian masalah tertentu atau untuk penggunaan umum. Untuk mengatasi masalah, salah satu dari dua pendekatan dapat digunakan.

Saat ini, komputasi awan adalah salah satu teknologi yang paling populer dan banyak digunakan. Sebagai teknologi, *cloud* dianggap baik dan populer karena memungkinkan pengguna untuk terutama menggunakan layanan komputasi, penyimpanan, dan jaringan sesuai permintaan tanpa mengeluarkan biaya untuk infrastruktur (belanja modal). Penelitian ekstensif sedang berlangsung di segmen ini. Banyak organisasi bisnis sudah mulai bergantung pada *cloud*. Perkembangan *cloud* yang cepat didorong oleh sifatnya yang berorientasi pasar.

Beberapa perusahaan besar seperti Amazon, Google, dan Microsoft sudah mulai menggunakan *cloud* secara luas, dan mereka berniat melanjutkan tren ini.

Dengan munculnya komputasi awan, muncullah pergeseran baru di era komputasi. Semua aplikasi dimigrasikan ke *cloud* karena masa depan tampaknya sepenuhnya bergantung padanya. Arsitektur layanan *cloud* memiliki tiga komponen dasar yang disebut model layanan, yaitu *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), dan *Software as a Service* (SaaS). Di antara ketiganya, SaaS adalah model terpenting di mana perangkat lunak diberikan sebagai layanan dari *cloud*. Perangkat lunak ini dirancang khusus untuk platform *cloud*, dan untuk merancang perangkat lunak atau aplikasi, Anda harus menggunakan model pemrograman tertentu. Oleh karena itu, kajian tentang hal tersebut menjadi penting.

*Cloud* memiliki beberapa properti yang membuatnya sangat berbeda dari aplikasi perangkat lunak konvensional lainnya, seperti skalabilitas, konkurensi, multitenansi, dan toleransi kesalahan. Biasanya ada dua jenis praktik: satu memperluas model atau bahasa pemrograman yang ada ke *cloud*, dan yang lainnya adalah memiliki model yang benar-benar baru yang dirancang khusus untuk *cloud*. Model pemrograman di bawah dua area ini dibahas secara luas.

## 8.2 MODEL PEMROGRAMAN YANG DIPERLUAS UNTUK *CLOUD*

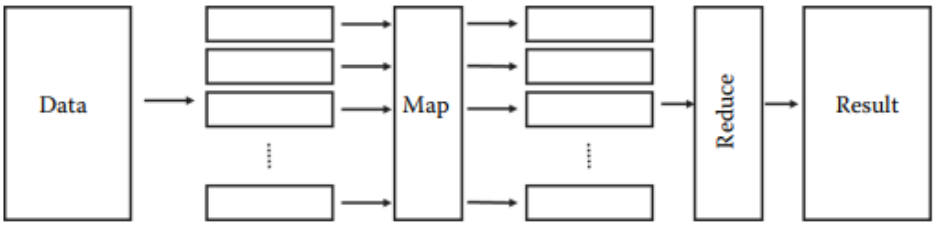
Bagian ini membahas model pemrograman yang ada yang dapat diperluas ke platform *cloud*. Tidak semua model pemrograman dapat dimigrasikan ke *cloud*. Seperti yang telah dibahas, ada beberapa sifat *cloud* yang membuatnya menjadi teknologi yang berbeda dari yang lain. Dengan demikian, perubahan tertentu harus dilakukan untuk mengatasi masalah

*cloud*. Hanya model pemrograman tertentu yang memiliki kemiripan dengan *cloud* atau memiliki struktur yang pada akhirnya dapat mendukung parameter tertentu seperti skalabilitas, konkurensi, dan multitenansi yang dapat diperluas. Dari sifat-sifat ini, skalabilitas dan konkurensi dianggap sangat penting untuk *cloud*. Skalabilitas adalah kemampuan sistem untuk meningkatkan atau menurunkan dirinya sendiri sesuai dengan jumlah pengguna. Ini adalah salah satu alasan di balik popularitas *cloud*. Dengan demikian, properti ini harus dijaga oleh model pemrograman apa pun. Beberapa model dan framework pemrograman yang cocok untuk *cloud*, termasuk bahasa pemrograman tertentu, dibahas dalam subbagian berikut.

**Pengurangan Peta**

MapReduce adalah model pemrograman paralel yang dikembangkan oleh Google. Ini adalah salah satu teknologi yang memiliki dampak khusus pada bidang komputasi paralel dan komputasi awan. MapReduce menggunakan paradigma pemrograman paralel untuk perhitungan sejumlah besar data dengan mempartisi atau memisahkannya menjadi beberapa bagian, memprosesnya secara paralel, dan kemudian menggabungkannya untuk menghasilkan satu hasil. Proses ini masing-masing disebut map dan reduce. PaaS populer bernama Aneka juga menggunakan Map Reduce dengan platform .NET.

Langkah pertama dalam fungsi peta dan pengurangan melibatkan pemisahan data besar dan mengirimkannya ke fungsi peta. Fungsi peta melakukan operasi peta pada data. Selanjutnya, data dikirim untuk operasi pengurangan di mana hasil sebenarnya diperoleh. Ini dibahas secara rinci dalam sub-bagian berikut dan ditunjukkan pada Gambar 8.1.




**Gambar 8.1** Pengurangan Peta

**Fungsi Peta**

Pertama, data besar dilacak dan dipisahkan sebagai pasangan kunci/nilai. Fungsi peta menerima pasangan kunci/nilai dan mengembalikan pasangan kunci/nilai perantara. Biasanya, fungsi peta bekerja pada satu dokumen. Rumus berikut menggambarkan operasi yang tepat:

$$Map(Key1, Val1) \rightarrow List(Key2, Val2)$$

Daftar pasangan kunci/nilai yang diperoleh selanjutnya dikirim ke fungsi pengurangan.

**Mengurangi Fungsi**

Setelah daftar pasangan kunci/nilai disiapkan setelah peta berfungsi, operasi pengurutan dan penggabungan diterapkan pada pasangan kunci/nilai ini. Biasanya,

pengurangan fungsi bekerja pada satu kata. Masukan pasangan kunci/nilai yang berbeda diberikan dan sebagai gantinya diperoleh grup dengan pasangan kunci/nilai yang sama, seperti yang ditunjukkan dalam persamaan berikut:

$$Reduce(Key1, List(Val2)) \rightarrow List(Val3)$$

Nilai yang dihasilkan memberikan rangkaian hasil akhir.

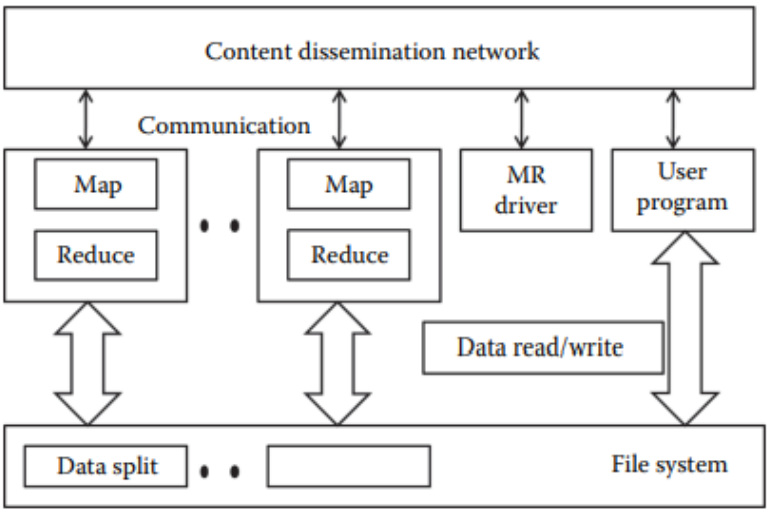
Contoh paling populer adalah menghitung kata dalam sebuah file. Pada langkah pertama, kata (kunci) diberi nilai. Kata mewakili kunci, dan hitungan mewakili nilainya. Setiap kali kata ditemukan, pasangan kunci/nilai dihasilkan. Kumpulan pasangan <word, count> yang diperoleh dari fungsi peta ini dikirim ke fungsi pengurangan. Hasilnya disortir, dan fungsi pengurangan menggabungkan semua kata yang memiliki kunci yang sama dan penghitungan (nilai) secara otomatis bertambah (ditambahkan) untuk setiap pasangan yang diperoleh. Jadi, hasilnya memberikan setiap kata dengan hitungannya.

Fitur utama

- Mendukung pemrograman paralel
- Cepat
- Dapat menangani data dalam jumlah besar

**CGL-MapReduce**

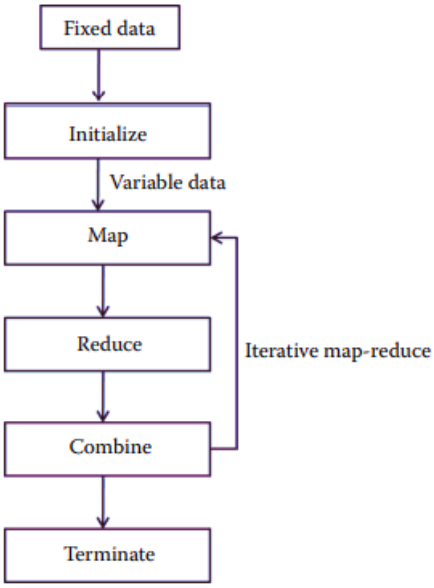
CGL-MapReduce adalah tipe lain dari runtime MapReduce dan dikembangkan oleh Ekanayake et al. Ada perbedaan spesifik antara CGL-MapReduce dan MapReduce. Tidak seperti MapReduce, CGL-MapReduce menggunakan streaming alih-alih sistem file untuk semua komunikasi. Ini menghilangkan overhead yang terkait dengan komunikasi file, dan hasil antara dari fungsi peta dikirim secara langsung. Aplikasi ini terutama dibuat dan diuji untuk sejumlah besar data ilmiah dan dibandingkan dengan MapReduce.




**Gambar 8.2** CGL-MapReduce

Gambar 8.2 menunjukkan arsitektur kerja CGL-MapReduce dimana beberapa komponennya digambarkan.

1. Pekerja peta: Pekerja peta bertanggung jawab untuk melakukan operasi peta.
2. Kurangi pekerja: Pekerja kurangi bertanggung jawab untuk melakukan operasi kurangi.
3. Jaringan Penyebaran Konten: Jaringan penyebaran konten menangani semua komunikasi antar komponen menggunakan NaradaBroker, yang dikembangkan oleh Ekanayake et al.
4. MRDriver: MRDriver adalah pekerja utama dan mengontrol pekerja lain berdasarkan instruksi dari program pengguna. CGL- MapReduce berbeda dengan MapReduce, perbedaan utamanya adalah penghindaran sistem file dan penggunaan streaming. Beberapa tahapan dalam CGL-MapReduce digambarkan pada Gambar 8.3 dan dijelaskan sebagai berikut:
    a Tahap inisialisasi: Langkah pertama melibatkan memulai node pekerja MapReduce dan konfigurasi tugas MapReduce. Konfigurasi MapReduce adalah proses sekali pakai dan beberapa salinannya dapat digunakan kembali. Ini adalah salah satu peningkatan CGL-MapReduce, yang memfasilitasi komputasi MapReduce iteratif yang efisien. MapReduce yang dikonfigurasi ini disimpan dan dijalankan atas permintaan pengguna. Di sini, data tetap diberikan sebagai input.
    b Tahap peta: Setelah langkah inisialisasi, MRDriver memulai komputasi peta atas instruksi pemrogram. Ini dilakukan dengan meneruskan data variabel ke tugas peta. Ini diteruskan ke pekerja untuk menjalankan tugas peta yang dikonfigurasi. Ini juga memungkinkan meneruskan hasil dari satu iterasi ke iterasi lainnya. Akhirnya, tugas peta ditransfer langsung untuk mengurangi pekerja menggunakan jaringan diseminasi.
    c Reduce stage: Segera setelah semua tugas peta selesai, mereka dipindahkan ke pekerja yang dikurangi, dan pekerja ini mulai menjalankan tugas setelah mereka diinisialisasi oleh MRDriver. Output dari fungsi pengurangan dikirim langsung ke aplikasi pengguna.
    d Tahap Combine: Pada tahap ini, semua hasil yang diperoleh pada tahap pengurangan digabungkan. Ada dua cara untuk melakukannya: jika itu adalah perhitungan MapReduce single-pass, maka hasilnya digabungkan secara langsung, dan jika itu adalah operasi iteratif, maka kombinasi yang sesuai diperoleh sedemikian rupa sehingga iterasi berlanjut dengan sukses.
    e Tahap penghentian: Ini adalah tahap akhir, dan program pengguna memberikan perintah untuk penghentian. Pada tahap ini, semua pekerja diberhentikan.




**Gambar 8.3** Langkah-langkah yang terlibat dalam CGL-MapReduce.

Fitur utama

- Menggunakan streaming untuk komunikasi
- Mendukung paralelisasi
- Iteratif di alam
- Dapat menangani data dalam jumlah besar

***Cloud* Haskell: Pemrograman Fungsional**

*Cloud* Haskell didasarkan pada bahasa pemrograman fungsional Haskell. Bahasa pemrograman fungsional ini berbasis fungsi dan bekerja seperti fungsi matematika. Biasanya, keluaran di sini hanya berdasarkan nilai masukan. Pemrograman fungsional terdiri dari komponen yang seringkali tidak dapat diubah. Di sini, di *Cloud* Haskell, komponen yang tidak dapat diubah menjadi dasarnya. Komponen yang tidak dapat diubah adalah komponen yang statusnya tidak dapat dimodifikasi setelah dibuat. *Cloud* Haskell adalah bahasa khusus domain.

Model pemrograman sepenuhnya didasarkan pada antarmuka penyampaian pesan, yang dapat digunakan untuk aplikasi real-time yang sangat andal. Tujuan utamanya adalah menggunakan paradigma pemrograman konkuren. *Cloud* terdiri dari beberapa aplikasi dan pengguna yang independen dan banyak jumlahnya. Jadi, untuk mengatasi masalah ini, konkurensi harus dipastikan untuk sejumlah besar permintaan. *Cloud* Haskell menyediakan fitur konkurensi dengan kemandirian penuh untuk aplikasi. Data benar-benar terisolasi dan aplikasi tidak dapat menembus batas dan mengakses data lain. Seperti disebutkan sebelumnya, komunikasi sepenuhnya didasarkan pada model *message-passing.* Menurut pengembang, perlu ada model biaya tersendiri yang akan menentukan perpindahan data. Ada beberapa keunggulan Haskell selain dari penggunaan model toleransi kesalahan Erlang. Keuntungan utamanya adalah bahwa ini adalah bahasa fungsional murni. Semua data tidak dapat diubah secara default dan fungsinya idempoten. Idempotensi di sini mengacu pada kemampuan fungsi untuk memulai kembali dari titik mana pun dengan sendirinya tanpa

menggunakan entitas eksternal apa pun seperti database terdistribusi ketika fungsi dibatalkan karena kesalahan perangkat keras.

Model biaya telah diperkenalkan secara khusus di *Cloud* Haskell, yang tidak ada di Haskell. Di sini, model biaya menentukan biaya komunikasi antar proses. Model ini tidak dirancang untuk memiliki memori bersama. Karena model mendukung konkurensi dan proses ini diisolasi, toleransi kesalahan diharapkan sampai batas tertentu dan kegagalan satu proses tidak akan mempengaruhi yang lain. Ada beberapa fitur novel lain yang dimasukkan ke dalam model ini seperti serialisasi. Setiap kali seorang programmer mencoba menggunakan kode yang didistribusikan, dia harus menjalankan kode tersebut di sistem jarak jauh. Ini tidak mungkin dalam versi Haskell. Dengan demikian, dalam model ini, para pengembang memperkenalkan mekanisme yang secara otomatis menangani masalah ini tanpa memperluas kompiler.

Sederhananya, serialisasi diurus dengan abstraksi penuh sehingga programmer tidak mengetahuinya. Model ini juga mengatasi kegagalan, yang merupakan masalah penting di *cloud*. Ada banyak cara di mana kegagalan dapat mempengaruhi fungsi sistem. Karena jumlah sistem yang terhubung dan pengguna meningkat, tingkat kegagalan juga meningkat. Biasanya, untuk menghadapi kegagalan sebagian, seluruh sistem di-restart. Ini dianggap sebagai salah satu solusi yang tidak efisien karena jika menyangkut *cloud*, jumlah node terdistribusi tinggi, sehingga memulai ulang sistem terbukti sangat mahal. Dengan demikian, untuk menjaga toleransi kesalahan, metode Erlang digunakan. Bahkan jika suatu fungsi gagal karena alasan tertentu, itu dapat dimulai ulang secara terpisah tanpa mempengaruhi bagian dan proses lainnya. Di sini, komponennya tidak dapat diubah dan disebut fungsi murni.

Fitur utama

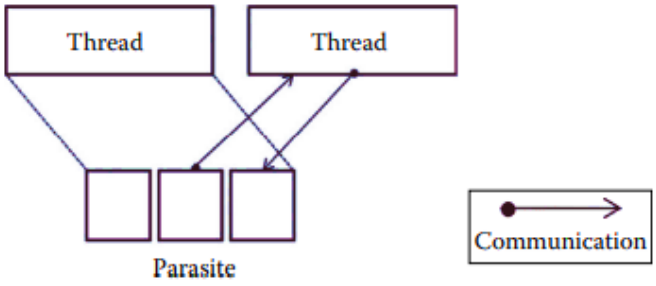
- Toleransi kesalahan
- Bahasa khusus domain
- Menggunakan paradigma pemrograman bersamaan
- Tidak ada memori bersama
- Idempoten
- Murni fungsional

**MultiMLton: Pemrograman Fungsional**

MultiMLton secara khusus digunakan untuk lingkungan multiprosesor dan merupakan perpanjangan dari MLton. MLton adalah kompiler ML standar sumber terbuka, seluruh program, pengoptimalan. Ini digunakan untuk mengekspresikan dan mengimplementasikan berbagai jenis paralelisme berbutir halus. Ini memiliki kemampuan untuk secara efisien menangani sejumlah besar benang ringan. Itu juga menggabungkan abstraksi bahasa baru dan analisis kompiler. Ada beberapa perbedaan antara MultiMLton dan mitra lainnya seperti Erlang dan ConcurrentML. Ini memberikan paralelisme tambahan bila memungkinkan dan kapan pun hasilnya menguntungkan dengan menggunakan konkurensi deterministik dalam utas. Ini meningkatkan kecepatan dan pada akhirnya akan memberikan kinerja yang baik. Hal ini juga memberikan tindakan spekulatif sadar penyampaian pesan yang dapat dikomposisi. Ini menyediakan isolasi antara kelompok utas yang berkomunikasi. Komunikasi antar utas

terjadi melalui teknik pengiriman pesan sederhana. Ini memungkinkan konstruksi kejadian asinkron yang mengintegrasikan protokol komunikasi sinkron abstrak.

Model MultiMLton terdiri dari komponen yang dikenal sebagai parasit. Parasit ini adalah utas mini yang bergantung pada utas utama atau master untuk pelaksanaannya. Seperti disebutkan sebelumnya, komunikasi antar utas dapat sinkron dan asinkron. Gambar 8.4 menunjukkan komunikasi sederhana antara parasit dari satu utas dan utas lainnya.




**Gambar 8.4** Pesan lewat.

Fitur utama

- Dapat secara efisien menangani sejumlah besar benang ringan.
- Cocok untuk lingkungan multiprosesor.
- Konkurensi deterministik digunakan jika diperlukan.

**Erlang: Pemrograman Fungsional**

Erlang adalah salah satu jenis penting dari bahasa pemrograman fungsional. Ini dikembangkan oleh Joe Armstrong dan merupakan salah satu dari sedikit bahasa yang sangat toleran terhadap kesalahan. Bahkan Haskell (bahasa pemrograman lain) menggunakan model toleransi kesalahan Erlang.

Erlang adalah bahasa pemrograman bersamaan yang menggunakan teknik pengiriman pesan asinkron untuk komunikasi. Di sini, komunikasi hanya dapat dilakukan melalui mekanisme pengiriman pesan. Membuat dan menghapus proses sangat mudah, dan prosesor yang mengirimkan pesan membutuhkan lebih sedikit memori; karenanya, ini dianggap ringan. Ini mengikuti proses berdasarkan model konkurensi.

Model pemrograman ini digunakan untuk mengembangkan sistem real-time dimana pengaturan waktu sangat kritis. Toleransi kesalahan adalah salah satu sifat penting dari Erlang. Itu tidak memiliki memori bersama seperti yang disebutkan sebelumnya; semua interaksi dilakukan melalui penyampaian pesan. Ini memiliki sistem manajemen memori yang baik dan mengalokasikan dan membatalkan alokasi memori secara otomatis dengan pengumpulan sampah dinamis (runtime). Dengan demikian, seorang programmer tidak perlu khawatir tentang manajemen memori, dan dengan demikian kesalahan terkait memori dapat dihindari. Salah satu properti Erlang yang paling mencolok adalah memungkinkan penggantian kode saat runtime. Seorang pengguna dapat mengubah kode saat program sedang berjalan dan dapat menggunakan kode baru yang sama secara bersamaan.

Erlang, menurut definisi, adalah bahasa deklaratif. Itu juga dapat dianggap sebagai bahasa pemrograman fungsional. Bahasa pemrograman fungsional adalah bahasa yang hanya bergantung pada input data. Properti ini biasanya disebut kekekalan. Ini adalah properti dasar

yang membuat pemrograman fungsional menjadi model pemrograman paling cocok untuk *cloud*. Properti ini memungkinkan pemrogram untuk menggunakan konkurensi dan menskalakan sistem, yang merupakan salah satu properti penting *cloud*.

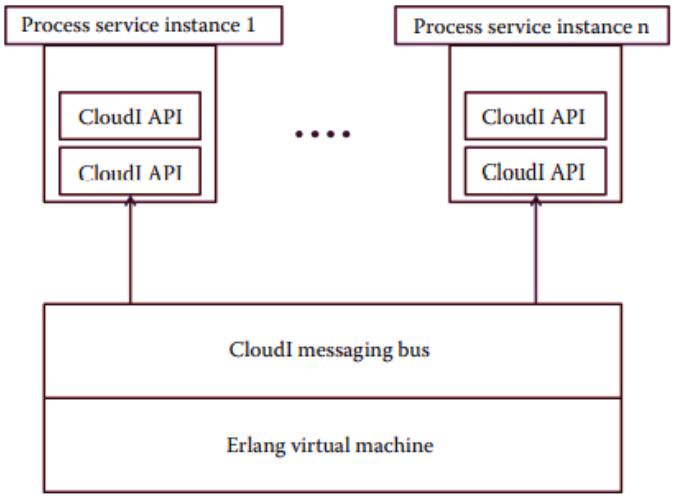
Fitur utama

- Toleransi kesalahan
- Kokoh
- Bersamaan
- Menggunakan model pemrograman fungsional

**CloudI**

CloudI adalah kerangka komputasi awan sumber terbuka yang dikembangkan oleh Michael Truog. Ini memungkinkan membangun platform *cloud* (awan pribadi) tanpa virtualisasi yang sebenarnya. Ini pada dasarnya untuk tugas pemrosesan server back-end. Tugas-tugas ini memerlukan pemrosesan transaksi real-time lunak di luar penggunaan basis data dalam bahasa apa pun seperti C, C++, Python, Java, Erlang, dan Ruby. CloudI dibangun di atas bahasa pemrograman Erlang. Ini bertujuan untuk menjadi toleran terhadap kesalahan dan memiliki arsitektur yang sangat skalabel (awan di tingkat terendah; membawa toleransi kesalahan Erlang ke pengembangan polyglot). Ini terdiri dari antarmuka pemrograman aplikasi perpesanan (API) yang memungkinkan layanan CloudI mengirim permintaan.

CloudI API mendukung pesan terbitkan/pasok dan pesan permintaan/balasan. Terlepas dari toleransi kesalahan, yang merupakan salah satu sifat pentingnya, *Cloud*I dapat dengan mudah mendukung bahasa lain atau lingkungan poliglot. Pengguna tidak perlu mengetahui Erlang sepenuhnya untuk bekerja dengan CloudI. Modul layanan apa pun di CloudI adalah miniatur dari modul pusat. Setiap modul layanan berisi CloudI API.




**Gambar 8.5** komunikasi CloudI.

Di sini, semua komunikasi dilakukan melalui CloudI API. Gambar 8.5 menggambarkan keterkaitan dan komunikasi antara contoh layanan CloudI dan proses contoh layanan. Bus perpesanan CloudI bertindak sebagai antarmuka antara API dan

VM Erlang. Setiap layanan proses terdiri dari CloudI API dan bertanggung jawab untuk komunikasi.

Fitur utama

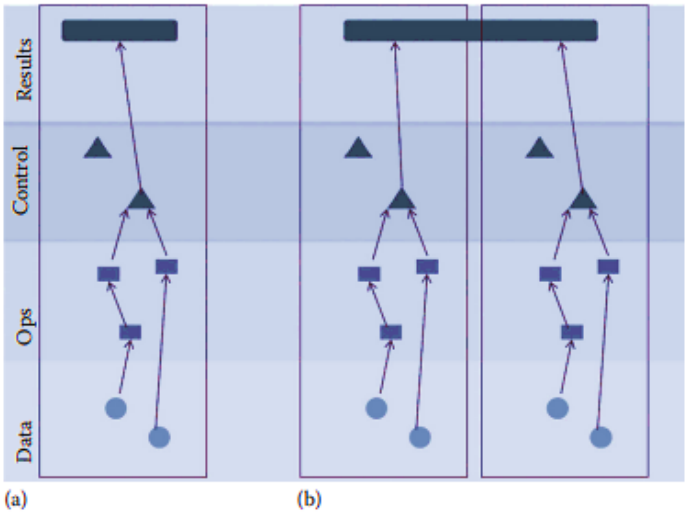
- Menggunakan model pemrograman fungsional
- Toleransi kesalahan
- Mendukung banyak bahasa
- Mendukung skalabilitas ekstensif

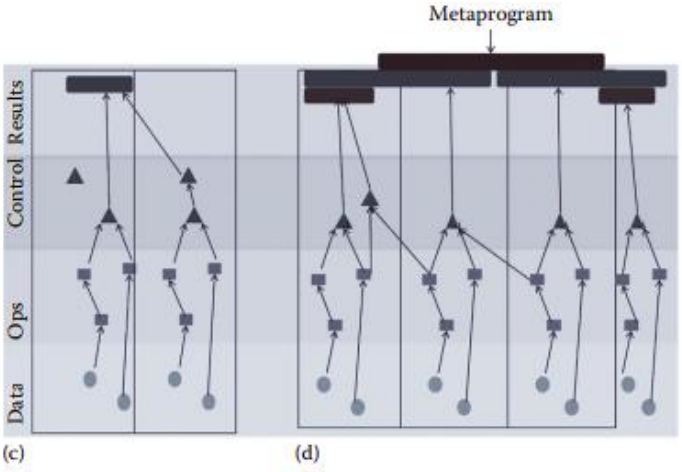
**SORCER: Pemrograman Berorientasi Objek**

Lingkungan Komputasi Berorientasi Layanan (SORCER) adalah platform *cloud* berorientasi objek untuk abstraksi layanan transdisipliner yang diusulkan oleh Michael Sobolewski, yang juga mendukung komputasi kinerja tinggi. Ini adalah lingkungan metacomputing *service-to-service* (S2S) federasi yang memperlakukan penyedia layanan sebagai peer jaringan dengan semantik yang terdefinisi dengan baik dari arsitektur berorientasi objek layanan.

Komputasi transdisipliner terdiri dari beberapa penyedia layanan. Semua penyedia layanan memiliki federasi kolaboratif, dan mereka menggunakan alur kerja yang diatur untuk layanan mereka. Segera setelah pekerjaan selesai, kolaborasi dibubarkan dan penyedia layanan mencari federasi lain untuk bergabung. Metode ini sepenuhnya berbasis jaringan. Di sini, setiap layanan adalah entitas independen dan mandiri yang disebut penyedia layanan jarak jauh, yang melakukan tugas jaringan tertentu. Dengan demikian, seluruh metode adalah jaringan sentris. Untuk mengontrol keseluruhan sistem, diperlukan sistem operasi (OS), yang memungkinkan penyedia memberikan instruksi terkait interaksi antar layanan.

Pengguna tidak menggunakan teknologi ini untuk mengelola orkestrasi dan interaksi antar layanan. Pengguna harus mengetahui kode dan skrip baris perintah, sehingga menjadikannya pekerjaan yang membosankan. Oleh karena itu, untuk mengatasi masalah ini, sistem operasi berorientasi layanan (SOOS) digunakan. Di sini, metaprogram dianggap sebagai ekspresi proses dari program komponen jarak jauh, dan SOOS membuat semua keputusan tentang tempat, waktu, dan detail lainnya terkait menjalankan program komponen jarak jauh. Metaprogram adalah program yang memanipulasi program lain dari jarak jauh sebagai datanya. Di sini, digunakan sebagai spesifikasi untuk kolaborasi layanan.



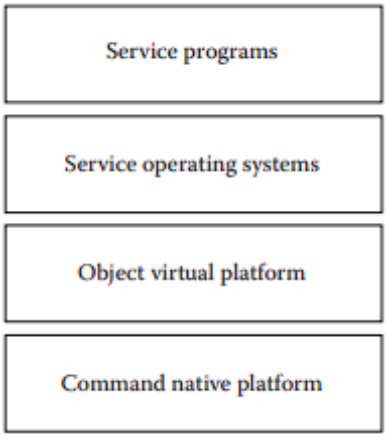



**Gambar 8.6** Jenis operasi: (a) disiplin, (b) multidisiplin, (c) interdisipliner, dan (d) transdisipliner.

Gambar 8.6 menunjukkan perbedaan antara berbagai jenis mekanisme komputasi. Ini termasuk komputasi multidisiplin, interdisipliner, dan transdisipliner. Metaprogram digunakan untuk komputasi transdisipliner. Diagram dengan jelas menunjukkan tempat di mana metaprogram dijalankan. Gambar 8.6a menunjukkan disiplin yang terdiri dari beberapa data, operasi, dan komponen kontrol yang dihubungkan bersama dalam alur kerja untuk memberikan hasil. Gambar 8.6b menunjukkan model multidisiplin yang melibatkan dua disiplin ilmu atau lebih tetapi bukan interkomunikasi antar disiplin ilmu. Demikian pula, Gambar 8.6c menunjukkan model interdisipliner yang melibatkan komunikasi antar disiplin ilmu. Akhirnya, Gambar 8.6d menunjukkan model transdisipliner di mana berbagai disiplin ilmu dengan komunikasi interdisipliner hadir, dan di sini komponen dari luar disiplin juga dapat berkontribusi pada hasil. Model transdisipliner ini menggunakan metaprogram. Seluruh konsep SORCER didasarkan pada pemrosesan jarak jauh. Di sini, komponen komputasi yang sesuai disediakan secara otonom dengan menggunakan metainstruksi yang dihasilkan oleh metaprosesor layanan. Ini dilakukan alih-alih mengirim file yang dapat dieksekusi langsung melalui Internet. Metaprogram dan metainstruction masing-masing adalah perintah layanan dan instruksi layanan.

Di sini, OS metacompute digunakan, yang mengelola seluruh federasi dinamis penyedia layanan dan sumber daya. Dengan strategi kontrolnya sendiri, memungkinkan semua penyedia layanan untuk berkolaborasi di antara mereka sendiri, yang dilakukan sesuai dengan definisi metaprogram yang diberikan. Metaprogram, dalam istilah metainstruction, dapat dikirimkan ke metacompute OS.

Ada beberapa lapisan dalam SORCER, seperti yang ditunjukkan pada Gambar 8.7, dan di antaranya OS SORCER adalah yang paling penting. Pada gambar, layer dikelompokkan menurut fungsinya. Platform virtual perintah dan platform virtual objek secara kolektif membentuk *cloud* platform.




**Gambar 8.7** arsitektur SORCER.

Node layanan dan penyedia layanan khusus domain (DS) secara kolektif melakukan operasi *cloud* layanan. Operasi yang disebutkan di atas dengan platform perintah asli dan filter evaluator dikendalikan oleh metaprosesor. Semuanya dikelola oleh sistem operasi layanan dan di atasnya program layanan atau pemohon layanan meminta layanan.

SORCER berfokus pada metamogramming. Metamogramming adalah proses di mana metamodeling atau lingkungan pemrograman digunakan untuk pembuatan ekspresi proses simbolik. Dengan demikian, SORCER memperkenalkan OS metacompute yang terdiri dari semua layanan sistem yang diperlukan, yang mencakup metaprogramming objek, sistem file federasi, dan manajemen sumber daya otonom. Properti ini untuk mendukung pemrograman berorientasi objek layanan. Ini pada dasarnya memberikan solusi untuk beberapa aplikasi transdisipliner yang membutuhkan solusi transdisipliner yang kompleks. SORCER digunakan melalui FIPER.

FIPER adalah lingkungan proyek cerdas federasi, dan tujuan utamanya adalah untuk menyediakan federasi objek layanan terdistribusi yang menyediakan data teknik, aplikasi, dan alat pada jaringan. Singkatnya, arsitektur metamodeling SORCER didasarkan pada domain/manajemen/carrier, atau triplet DMC.

Fitur utama

- Sifat berorientasi objek
- Arsitektur berorientasi layanan
- Transdisipliner

**Model Pemrograman di Aneka**

Aneka adalah PaaS yang dikembangkan oleh Manjrasoft, Inc. Ada tiga model pemrograman yang didefinisikan dan digunakan oleh Aneka, yang dibahas berikut ini.

**Model Eksekusi Tugas**

Dalam model ini, aplikasi dibagi menjadi beberapa tugas. Setiap tugas dijalankan secara mandiri. Seperti yang didefinisikan oleh pengembang Aneka, tugas adalah unit kerja yang dijalankan oleh penjadwal dalam urutan apa pun. Karena melibatkan sejumlah besar tugas, ini sangat cocok untuk aplikasi terdistribusi, khususnya aplikasi yang melibatkan beberapa hasil independen, dan hasil independen ini kemudian digabungkan untuk memberikan hasil akhir. Untuk jenis masalah ini, hasil

independen dapat dianggap sebagai keluaran dari tugas, dan nantinya hasil ini dapat digabungkan oleh pengguna. Ini juga mendukung pembuatan tugas yang dinamis.

Fitur utama

- Cocok untuk aplikasi terdistribusi
- Tugas mandiri

**Model Eksekusi Thread**

Dalam model utas, aplikasi dijalankan menggunakan proses. Setiap proses terdiri dari satu atau lebih utas. Seperti yang didefinisikan oleh pengembang Aneka, utas adalah rangkaian instruksi yang dapat dijalankan secara paralel dengan instruksi lainnya. Jadi, dalam model eksekusi thread, eksekusi dilakukan oleh sistem.

Fitur utama

- Mendukung paralelisasi berat.
- Pemrograman multi-utas tersedia.

**Model Pengurangan Peta**

Model ini mirip dengan model MapReduce sebenarnya yang dijelaskan di bagian pertama. Model Pengurangan Peta diimplementasikan di platform Aneka.

## 8.3 MODEL PEMROGRAMAN BARU DIUSULKAN UNTUK *CLOUD*

Bagian ini membahas tentang model pemrograman yang baru dirancang untuk *cloud*. Model pemrograman ini dirancang dari awal. Karena model ini dirancang khusus untuk *cloud*, properti penting *cloud*, seperti skalabilitas, sifat terdistribusi, dan multitenancy, membentuk dasar untuk merancang algoritma. Berbeda dengan metode sebelumnya, tidak ada keharusan untuk mengikuti spesifikasi bahasa apa pun. Terutama semua pendekatan berkonsentrasi pada sifat terdistribusi dari algoritma. Konsep model pemrograman baru muncul karena kelemahan tertentu yang ditemukan pada model pemrograman yang sudah ada. Kerugian utama adalah Anda tidak dapat menghilangkan sifat dasar tertentu dari pendekatan yang ada. Hanya parameter baru yang dapat ditambahkan atau sedikit modifikasi yang dapat dilakukan untuk mencapai tujuan yang diinginkan. Sedangkan ketika model pemrograman baru dirancang, perancang memiliki kebebasan penuh untuk memasukkan apapun tanpa batasan apapun. Bagian berikut juga menyertakan beberapa API dan toolkit yang dirancang khusus untuk *cloud*.

**Orleans**

Orleans adalah framework yang dikembangkan oleh Microsoft untuk membuat aplikasi *cloud*. Ini adalah upaya Microsoft untuk memberikan kerangka kerja perangkat lunak yang akan memfasilitasi pengembangan aplikasi *cloud* dari sisi klien dan server. Ini menggunakan paradigma pemrograman bersamaan. Karena aplikasi ini didesain untuk *cloud*, ada beberapa karakteristik yang dipertimbangkan. Fitur penting dari Orleans termasuk

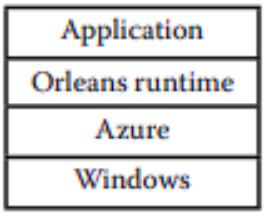
- Konkurensi
- Skalabilitas
- Mengurangi kesalahan
- Keamanan

Ide dasar di mana seluruh kerangka ini dikembangkan adalah penggunaan biji-bijian. Butir-butir ini adalah unit komputasi logis yang dianggap sebagai blok bangunan Orleans. Biji-bijian itu independen dan terisolasi, dan semua proses dijalankan dengan menggunakan biji-bijian. Butir dapat memiliki beberapa aktivasi. Aktivasi ini adalah instantiasi biji-bijian di server fisik. Aktivasi sangat membantu karena beberapa proses dapat dijalankan secara bersamaan dengan menggunakan beberapa aktivasi. Meskipun model mendukung konkurensi secara default, grain tidak menjalankan proses secara bersamaan. Butir hanya dapat melayani satu permintaan dalam satu waktu. Permintaan berikutnya hanya dapat dilayani setelah permintaan selesai.

Mungkin ada nol, satu, atau beberapa aktivasi. Jadi, untuk mendukung eksekusi paralel, beberapa aktivasi dari satu butir dapat dibuat dan beberapa permintaan dapat dilayani secara paralel. Beberapa keunggulan dari fitur ini, seperti dikutip dari pengembangnya, antara lain meningkatkan throughput dan mengurangi latensi. Fitur ini berdampak besar pada Orleans. Mereka berkomunikasi satu sama lain dengan menyampaikan pesan. Setiap butir memiliki keadaan. Status menentukan posisi butir saat ini. Ketika beberapa aktivasi dibuat untuk satu butir, ada kemungkinan bahwa keadaan butir dapat berubah. Ini dapat menyebabkan ketidakkonsistenan di antara semua aktivasi. Masalah ini juga ditangani oleh Orleans.

Skalabilitas adalah masalah lain yang perlu dipertimbangkan dalam pengembangan aplikasi *cloud*. Orleans pada bagiannya menganggap skalabilitas sebagai masalah penting dan mencoba memberikan solusi. Konsep biji-bijian dan aktivasi itu sendiri mempertimbangkan skalabilitas, dan selain itu, Orleans mengatasi masalah ini dengan menggunakan basis data yang dipecah. Sharded database dipartisi secara horizontal menjadi beberapa komponen dengan masing-masing komponen (shard) terdiri dari beberapa baris. Arsitektur database sharded ini digunakan di bagian belakang untuk mendukung biji-bijian. Dengan demikian, butir dapat sepenuhnya independen dan dapat memiliki komputasi independen. Ini disebut domain biji-bijian. Sifat komputasi terdistribusi dan independen sangat efektif karena semua aplikasi bergantung pada basis data. Jika ada satu database, maka itu menjadi hambatan. Jadi, semuanya akan bergantung pada satu basis data, dan jika basis data itu gagal, seluruh sistem runtuh. Tetapi ketika sistem didistribusikan di alam, tidak demikian.

Kegagalan satu aplikasi mengakibatkan kegagalan bagian dari basis data yang tidak akan memengaruhi potongan basis data lainnya dan dengan demikian tidak akan memengaruhi aplikasi lain. Dengan demikian, dengan cara tertentu, ini menjadi toleran terhadap kesalahan dan dapat digunakan secara luas serta dapat diskalakan karena isolasi dan kemandirian aplikasi dijamin. Tidak ada batasan dalam pembuatan jumlah butir dan jumlah aktivasi. Dengan demikian, seiring bertambahnya jumlah permintaan, begitu pula jumlah proses, maka kita dapat secara bersamaan meningkatkan butir dan aktivasi untuk melayani permintaan tersebut. Proses-proses ini tidak akan saling mempengaruhi karena mereka benar-benar terisolasi. Jadi, konsep butir dan aktivasi itu sendiri adalah solusi untuk skalabilitas. Gambar 8.8 menggambarkan lapisan tempat Orleans bekerja. Orleans diimplementasikan sebagai library di C# over .Net framework.




**Gambar 8.8** sistem orleans.

Terlepas dari masalah ini, model memperkenalkan transaksi, dan menurut pengembang ada beberapa alasan untuk menggunakan transaksi. Alasan pertama adalah aktivasi dapat mengubah keadaan butir; dengan demikian, semua perubahan yang dilakukan oleh proses perlu dipertahankan. Untuk menjaga konsistensi, diperlukan mekanisme terpisah untuk mempertahankannya karena beberapa aktivasi dapat mengubah data secara bersamaan. Alasan lainnya adalah replikasi data. Replikasi data, yang tidak dapat dihindari dalam kasus teknologi *cloud*, harus dilakukan dengan hati-hati. Ini karena beberapa aplikasi bergantung pada data, dan replikasi data dapat mengakibatkan ketidakkonsistenan. Dengan demikian, transaksi diperkenalkan. Terlepas dari skalabilitas, Orleans memperkenalkan konkurensi terbatas.

Fitur utama

- Toleransi kesalahan
- Basis data terfragmentasi
- Penanganan transaksi yang efisien
- Konkurensi terbatas
- Keamanan

**BOOM dan Mekar**

Berkeley order of magnitude (BOOM) adalah proyek yang dikerjakan oleh UC Berkeley. Ini bertujuan untuk membuat kerangka kerja pemrograman yang dirancang untuk *cloud*. Bloom adalah bahasa pemrograman yang dibuat untuk proyek BOOM. Bahasa pemrograman Bloom didistribusikan secara luas di alam. BOOM sangat terdistribusi dan tidak terstruktur. Seperti namanya, ini dirancang agar orang dapat merancang sistem besar sebagai *cloud*. Ini mencoba untuk mengikuti pendekatan nontradisional sejauh mungkin. Pendekatan yang dikembangkan adalah tidak teratur dan data sentris. BOOM bertujuan untuk mengurangi kode dan meningkatkan skalabilitas. Ini menggunakan pendekatan terdistribusi penuh. BOOM menggunakan BOOM FS, yang merupakan sistem file terdistribusi Hadoop (HDFS). Menurut analitik, BOOM FS 20% lebih baik daripada HDFS. Tujuan desain Bloom adalah sebagai berikut:

- Sintaks yang familier
- Terintegrasi dengan bahasa imperatif
- Modularitas, enkapsulasi, dan komposisi

BLOOM menggunakan prinsip CALM dan dibangun dengan menggunakan dedalus. Prinsip CALM digunakan untuk membangun alat analisis dan visualisasi program otomatis untuk penalaran tentang koordinasi dan konsistensi, sedangkan dedalus adalah bahasa logika temporal. Dedalus adalah bahasa pemrograman terdistribusi yang berorientasi waktu. Ini berfokus pada sifat berorientasi data berdasarkan waktu daripada ruang. Data yang terkait

dengan ruang tidak banyak digunakan. Programmer tidak perlu memahami perilaku interpreter atau compiler karena Dedalus adalah bahasa temporal murni.

Fitur utama

- Modularitas, enkapsulasi, dan komposisi
- Dapat didistribusikan secara luas
- Tidak terstruktur

**GridBatch**

GridBatch adalah sistem yang memungkinkan pemrogram untuk dengan mudah mengubah desain tingkat tinggi menjadi implementasi paralel aktual dalam infrastruktur mirip komputasi awan. Itu tidak memparalelkan secara otomatis, dan programmer harus melakukan ini setelah memahami aplikasi dan harus dapat memutuskan faktor paralelisasinya. GridBatch menyediakan beberapa operator yang memfasilitasi pemrogram untuk melakukan tugas paralelisasi. Biasanya, untuk mengimplementasikan aplikasi, itu harus dipecah menjadi 296 bagian tugas yang lebih kecil dan partisi harus sedemikian rupa sehingga memiliki kinerja maksimum. Untuk memfasilitasi hal tersebut, GridBatch menyediakan sekumpulan library. Perpustakaan ini memiliki segalanya di dalamnya. Pemrogram perlu mengetahui cara menggunakan pustaka, dan dengan menggunakan pustaka dan operator secara efektif, pemrogram dapat mencapai efisiensi tertinggi. GridBatch dirancang khusus untuk aplikasi *cloud*, tetapi karena sifat pemrograman paralelnya yang sederhana, GridBatch dapat digunakan secara efektif untuk komputasi grid dan juga komputasi cluster. Ini mendukung lingkungan yang disebutkan di atas lebih baik daripada rekan-rekan lainnya, dan memberikan produktivitas pemrogram yang lebih tinggi. Sejauh menyangkut GridBatch, aplikasi harus memiliki paralelisme data dalam jumlah besar untuk mendapatkan peningkatan kinerja yang besar. Framework GridBatch ini cocok untuk aplikasi analitik yang melibatkan sejumlah besar data statistik.

Fitur utama

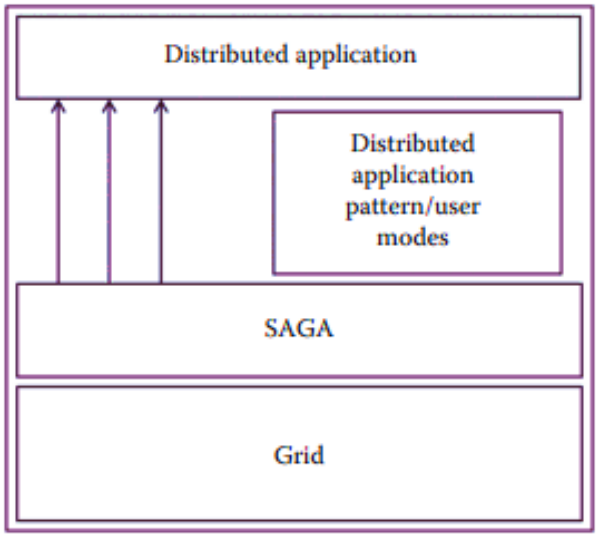
- Sederhana
- Mendukung paralelisasi yang efisien
- Peningkatan kinerja yang lebih baik

**API Sederhana untuk Aplikasi Grid**

API sederhana untuk aplikasi grid (SAGA) adalah API tingkat tinggi yang menyediakan antarmuka yang sederhana, standar, dan seragam untuk fungsionalitas terdistribusi yang paling umum dibutuhkan. Ini dikembangkan oleh Goodale et al. dan terutama digunakan untuk aplikasi terdistribusi. Ini juga menyediakan toolkit terpisah untuk bekerja dengan aplikasi terdistribusi dan memberikan bantuan untuk mengimplementasikan program yang umum digunakan. SAGA API ditulis dalam C++ dengan dukungan bahasa C dan Java. Pustaka yang disebut engine menyediakan dukungan untuk pengambilan keputusan lingkungan runtime dengan menggunakan adaptor yang relevan, dan ini adalah pustaka utama dan terpenting. SAGA menggunakan model tugas dan mekanisme pemberitahuan asinkron untuk menentukan model pemrograman bersamaan. Di sini, unit bersamaan diizinkan untuk berbagi status objek, yang merupakan salah satu properti yang diperlukan. Properti ini menciptakan overhead kontrol konkurensi tambahan. Dengan demikian, mekanisme konkurensi yang

ditegakkan perlu diadopsi. Namun, mekanisme konkurensi yang dipaksakan dapat menjadi masalah dalam beberapa kasus karena kadang-kadang mungkin tidak sesuai dengan domain masalah tertentu dan dapat mengakibatkan peningkatan kompleksitas dan overhead yang tidak perlu. Dengan demikian, mekanisme kontrol concurrency tidak ditegakkan. Pengguna harus menjaga konkurensi. SAGA bekerja langsung melalui toolkit grid seperti global dan condor yang digunakan untuk membuat grid. Tidak seperti open grid service architecture (OSGA), SAGA bukanlah sebuah middleware; sebaliknya, SAGA dapat bekerja melalui OSGA. Aplikasi terdistribusi disebarkan melalui SAGA.

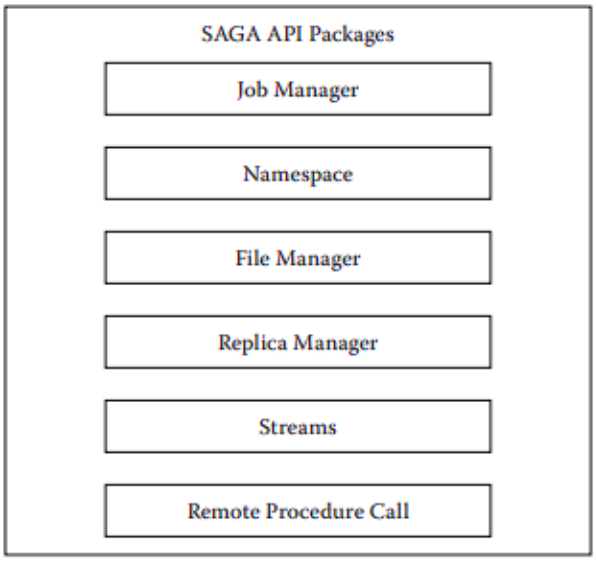
Aplikasi bekerja melalui pola aplikasi terdistribusi atau mode pengguna. Meskipun SAGA bekerja langsung melalui toolkit, ini bukanlah sebuah middleware. Ini adalah API seperti namanya. Gambar 8.9 menggambarkan arsitektur SAGA dengan semua lapisan yang terlibat dalam pengembangan aplikasi. Di sini, SAGA API bekerja pada beberapa komponen grid seperti condor dan globus. Beberapa jenis mode pengelolaan aplikasi terdistribusi disebarkan melalui SAGA. Ini melibatkan proses seperti MapReduce dan proses untuk mengelola pekerjaan hierarkis. Selama ini, aplikasi terdistribusi yang sebenarnya berfungsi. Aplikasi dapat bekerja secara langsung baik melalui SAGA atau melalui lapisan pola aplikasi terdistribusi. Biasanya, lapisan perantara antara SAGA dan aplikasi memfasilitasi pengelolaan sumber daya yang efisien.




**Gambar 8.9** arsitektur SAGA.

Lebih lanjut, SAGA dapat dijabarkan berdasarkan dua jenis paket yang tersedia. Yang pertama adalah paket yang terkait dengan tampilan dan nuansa. Paket API tampilan dan nuansa ini penting karena bertanggung jawab untuk menyediakan tampilan dan nuansa secara menyeluruh. Ini termasuk model tugas, keamanan, pemantauan, penanganan kesalahan, dan atribut. Paket lain disebut paket SAGA API. Ini adalah paket dasar yang diperlukan untuk SAGA.

Ini termasuk manajemen pekerjaan, ruang nama, manajemen file, aliran manajemen replika, dan panggilan prosedur jarak jauh. Gambar 8.10 menggambarkan paket-paket ini.




**Gambar 8.10** Paket API SAGA.

1. Manajemen pekerjaan: Karena SAGA melibatkan banyak pekerjaan untuk dikirim ke grid, fungsi manajemen pekerjaan digunakan. Ini digunakan untuk penyerahan, kontrol, dan pemantauan pekerjaan. Fungsi manajer pekerjaan adalah mengirimkan pekerjaan ke sumber daya jaringan, menjelaskan prosedur untuk mengendalikan pekerjaan ini, dan mengambil informasi tentang pekerjaan yang sedang berjalan dan selesai. Pengajuan pekerjaan dilakukan untuk dua mode: batch dan interaktif.
2. Manajemen namespace: Manajemen namespace bertanggung jawab untuk mengelola ruang nama, dan bekerja secara kolektif dengan paket manajemen file.
3. Manajemen file: File-file yang merupakan bagian dari SAGA ini dikelola oleh aplikasi. File-file tersebut diakses oleh aplikasi SAGA tanpa mengetahui lokasi file tersebut. Ini, bersama dengan namespace, bertanggung jawab untuk beberapa operasi pada file seperti membaca, mencari, dan menulis.
4. Manajemen replika: Manajemen replika menggambarkan interaksi dengan sistem replika. Replika adalah file yang terdaftar pada file logis. Ini digunakan untuk pembuatan dan pemeliharaan replika file logis dan untuk mencari file logis pada metadata.
5. Manajemen aliran: Fungsi utama pengelola aliran adalah menerapkan koneksi soket terautentikasi yang paling sederhana dengan kait untuk mendukung otorisasi tingkat aplikasi.
6. Panggilan prosedur jarak jauh: Ini mengatur semua panggilan prosedur jarak jauh dan menyediakan metode untuk komunikasi.

Fitur utama

- API sederhana
- Dukungan bahasa C dan Java

## 8.4 RINGKASAN

Pengguna yang menggunakan *cloud* meningkat pesat; dengan demikian, permintaan *cloud* sangat tinggi. Cara aplikasi dikembangkan di lingkungan biasa dan cara dikembangkan

di *cloud* berbeda. Bab ini membahas tentang model pemrograman yang diusulkan untuk komputasi awan. Karena komputasi awan melibatkan pengembangan beberapa aplikasi, perlu ada model pemrograman khusus yang akan memfasilitasi pengembangan aplikasi awan. Karena aplikasi *cloud* berbeda, model pemrograman yang ada tidak dapat digunakan seperti itu. Bab ini membahas kedua jenis model pemrograman untuk *cloud*: model pemrograman yang sudah ada yang dapat diperluas ke *cloud* dan model pemrograman baru yang diusulkan. Dalam kategori pertama, pemrograman fungsional, pemrograman paralel, dan pemrograman berorientasi objek diperluas ke *cloud*. Bahasa pemrograman yang ada yang sesuai dengan *cloud* juga dibahas. Demikian pula, sejauh menyangkut model-model baru, sifat awan terdistribusi terutama diperhitungkan. Beberapa model pemrograman baru yang diusulkan dibahas serta beberapa kerangka kerja yang dirancang khusus untuk pemrograman *cloud*.

**Tinjau Poin**

- Sharded database: Sharded database dipartisi secara horizontal menjadi beberapa komponen dengan masing-masing komponen (shard) terdiri dari beberapa.
- Grain: Unit komputasi logis dianggap sebagai blok bangunan Orleans.
- Parasit: Parasit adalah utas kecil (ringan) yang bergantung pada utas lain (utama).
- Komponen yang tidak dapat diubah: Ini adalah komponen yang statusnya tidak dapat diubah setelah dibuat.
- Idempotensi: Ini adalah kemampuan suatu fungsi untuk memulai ulang tanpa menggunakan entitas eksternal atau mekanisme pemulihan apa pun, bahkan setelah sistem mengalami kegagalan perangkat keras.
- Jaringan penyebaran konten: Jaringan penyebaran konten bertanggung jawab untuk mengelola komunikasi antara berbagai komponen CGL-MapReduce.
- Metamogramming: Sebuah proses di mana lingkungan metamodeling atau pemrograman digunakan untuk pembuatan ekspresi proses simbolik.
- Metaprogram: Ini adalah program yang memanipulasi program lain dari jarak jauh sebagai datanya.
- GridBatch: Ini adalah sistem yang memungkinkan pemrogram dengan mudah mengubah desain tingkat tinggi menjadi implementasi paralel aktual dalam infrastruktur mirip komputasi awan.
- Condor dan globus: Toolkit grid yang digunakan untuk membuat grid.
- Fault-tolerance: Mengacu pada kemampuan sistem untuk menahan kesalahan.

**Latihan Soal**

1. Apa perbedaan utama antara *Cloud* Haskell dan Erlang?
2. Apa pro dan kontra dari memiliki model pemrograman baru?
3. Bahasa pemrograman mana yang sangat toleran terhadap kesalahan dan untuk jenis aplikasi apa awalnya dirancang?
4. Tunjukkan perbedaan antara MapReduce dan CGL-MapReduce.
5. Di Orleans, kegagalan satu bagian aplikasi tidak akan mempengaruhi keseluruhan aplikasi. Membenarkan.

6. Apakah *Cloud* Haskell memiliki mekanisme toleransi kesalahannya sendiri? Jika tidak, lalu mekanisme kesalahan siapa yang mendasarinya?
7. Model (bahasa) pemrograman mana yang memungkinkan penggantian kode runtime?
8. Apa alasan utama menggunakan transaksi di Orleans?
9. Apa saja komponen SAGA?
10. Apakah GridBatch memparalelkan kode secara otomatis? Jika ya, lalu bagaimana? Selain itu, siapa yang bertanggung jawab atas paralelisasi kode?

# BAB 9
# PENGEMBANGAN PERANGKAT LUNAK DI *CLOUD*

**Tujuan pembelajaran**

Tujuan utama dari bab ini adalah untuk memperkenalkan konsep *Software as a Service* (SaaS) dan pengembangannya menggunakan teknologi *Platform-as-a-Service* (PaaS). Setelah membaca bab ini, Anda akan

- Memahami perbedaan aplikasi SaaS dari perangkat lunak/aplikasi tradisional
- Memahami bagaimana SaaS menguntungkan penyedia layanan dan pengguna akhir
- Memahami pro dan kontra dari berbagai model pengiriman SaaS
- Memahami tantangan yang diperkenalkan oleh aplikasi SaaS
- Memahami cara mengembangkan aplikasi SaaS berbasis *cloud* menggunakan teknologi PaaS

**Pengantar**

Bab ini memberikan wawasan tentang aplikasi SaaS yang berbeda dan menawarkan banyak keuntungan jika dibandingkan dengan aplikasi tradisional. Kami tidak dapat memilih opsi pengiriman SaaS untuk semua jenis aplikasi. Kesesuaian SaaS juga dibahas dalam bab ini serta banyak model penerapan dan pengiriman SaaS yang tersedia untuk pengembangan SaaS. Meskipun aplikasi SaaS menawarkan lebih banyak keuntungan bagi konsumen, aplikasi SaaS membawa banyak tantangan bagi pengembang sehubungan dengan pengembangan aplikasi SaaS, dan bab ini akan membahas tantangan ini lebih lanjut. Tinjauan model layanan *cloud* lainnya seperti *Infrastructure as a Service* (IaaS) dan *Platform as a Service* (PaaS) yang dapat dimanfaatkan untuk mengembangkan aplikasi SaaS yang sadar akan multitenant, dapat diskalakan, dan sangat tersedia dirinci dalam bab ini. Bab ini juga memberikan gambaran tentang mencapai multitenancy yang aman di tingkat basis data dengan model multitenancy yang berbeda. Terakhir, bab ini membahas tentang pemantauan dan pemeliharaan SLA aplikasi SaaS.

## 9.1 PENDAHULUAN

SaaS adalah pengiriman perangkat lunak dan model bisnis yang menjanjikan dalam industri teknologi informasi (TI). Aplikasi akan disebarkan oleh penyedia SaaS dalam infrastruktur yang dikelola atau dihosting. Pengguna akhir dapat mengakses aplikasi yang dihosting sebagai layanan kapan pun dibutuhkan. Untuk menggunakan SaaS, pengguna akhir harus membayar biaya lisensi penuh. Mereka dapat membayar untuk apa yang mereka gunakan atau konsumsi. Untuk mengakses aplikasi SaaS, pelanggan tidak perlu menginstal perangkat lunak di perangkat mereka. Itu dapat diakses dari infrastruktur penyedia layanan menggunakan browser web sederhana melalui Internet. Dalam model pengiriman perangkat lunak tradisional, hubungan antara pengguna akhir dan perangkat lunak adalah satu ke satu dan berbasis lisensi. Pengguna akhir harus membeli perangkat lunak dari vendor dengan membayar sejumlah besar lisensi.

Beberapa aplikasi kelas berat membutuhkan daya komputasi yang tinggi. Jadi, pengguna akhir juga harus membeli perangkat keras yang dibutuhkan. Jadi, investasi awal untuk menggunakan perangkat lunak ini tinggi, dan jika dilihat dari penggunaannya, akan sangat rendah. Untuk mengatasi kelemahan model pengiriman perangkat lunak tradisional ini, perusahaan mulai mengembangkan aplikasi SaaS, yang memiliki hubungan satu-ke-banyak dengan pengguna akhir dan perangkat lunak. SaaS mengikuti arsitektur multitenant, dan memungkinkan banyak pelanggan untuk berbagi satu contoh perangkat lunak, yang dikenal sebagai multitenancy. Karena sifatnya yang hemat biaya, banyak pelanggan mulai beralih ke aplikasi SaaS daripada perangkat lunak berbasis lisensi tradisional.

Aplikasi SaaS tidak hanya menguntungkan pengguna akhir tetapi juga penyedia layanan. Dalam model penyedia layanan aplikasi tradisional (ASP), penyedia layanan menghosting aplikasi di pusat data mereka sendiri untuk memberi manfaat bagi pengguna akhir. Karena pengiriman aplikasi adalah satu ke satu, ASP mengelola infrastruktur khusus untuk setiap pelanggan, yang memaksa mereka untuk berinvestasi dalam jumlah besar dalam mengelola infrastruktur, mengurangi laba atas investasi (ROI) mereka. Kemudian ASP dan vendor perangkat lunak independen (ISV) mulai menggunakan model pengiriman perangkat lunak alternatif (SaaS), yang meningkatkan ROI mereka dan pada saat yang sama menguntungkan pengguna. Sekarang, sebagian besar ASP dan ISV tradisional mulai menyadari manfaat bisnis SaaS dan memulai bisnis pengembangan SaaS.

**SaaS Berbeda dari Perangkat Lunak Tradisional**

Model pengiriman SaaS berbeda dari perangkat lunak tradisional berbasis lisensi tradisional. Berikut ini membahas banyak karakteristik yang membedakan aplikasi SaaS dari aplikasi tradisional:

- SaaS menyediakan akses web ke perangkat lunak komersial dengan basis bayar sesuai penggunaan.
- Aplikasi SaaS dikembangkan, diterapkan, dan dikelola dari lokasi terpusat oleh penyedia layanan.
- Ini memungkinkan contoh yang sama dari aplikasi SaaS untuk dibagikan oleh banyak pelanggan atau penyewa.
- Pembaruan dilakukan oleh penyedia layanan, bukan oleh pengguna.
- Aplikasi SaaS memungkinkan integrasi layanan dengan layanan pihak ketiga lainnya melalui antarmuka pemrograman aplikasi (API) yang disediakan oleh mereka.

**Manfaat SaaS**

Aplikasi SaaS memberikan manfaat berbasis biaya kepada pelanggan. Ini juga merupakan cara sesuai permintaan, mudah, dan terjangkau untuk menggunakan aplikasi tanpa perlu membelinya. Selain itu, solusi SaaS mudah diadopsi dan diintegrasikan dengan perangkat lunak lain. Beberapa manfaat penting dari SaaS disebutkan sebagai berikut:

1. Bayar per penggunaan: Aplikasi SaaS digunakan oleh pengguna akhir berdasarkan pembayaran per penggunaan. Aplikasi SaaS berbasis langganan dan memungkinkan pelanggan untuk menggunakan dan memutus layanan sesuai keinginan mereka. Dalam model pengiriman perangkat lunak tradisional, pelanggan harus membayar jumlah penuh meskipun mereka menggunakannya sangat sedikit.

2. Infrastruktur Nol: Untuk menginstal dan menjalankan perangkat lunak tradisional, pelanggan perlu membeli perangkat keras dan perangkat lunak yang dibutuhkan, meningkatkan pengeluaran modal. Pelanggan SaaS tidak perlu membeli dan memelihara infrastruktur, sistem operasi, platform pengembangan, dan pembaruan perangkat lunak.
3. Kemudahan akses: Aplikasi SaaS memerlukan web browser sederhana dan koneksi internet untuk mengaksesnya. Antarmuka pengguna responsif (UI) berbasis template akan beradaptasi secara otomatis ke perangkat pengguna akhir, meningkatkan pengalaman pengguna dan kemudahan akses.
4. Pembaruan otomatis: Dalam model pengiriman perangkat lunak tradisional, pelanggan perlu melakukan pembaruan massal, yang merupakan biaya tambahan. Namun di SaaS, penyedia layanan akan melakukan pembaruan otomatis. Jadi, pelanggan dapat mengakses aplikasi versi terbaru tanpa ada pembaruan dari pihak mereka.
5. Layanan komposit: Dengan menggunakan aplikasi SaaS, kami dapat mengintegrasikan layanan web pihak ketiga atau layanan *cloud* lain yang diperlukan melalui API mereka. Hal ini memungkinkan kita untuk membuat layanan komposit.
6. Penskalaan dinamis: Aplikasi SaaS digunakan oleh komunitas pengguna yang beragam. Beban pada aplikasi akan dinamis dan tidak dapat diprediksi. Tetapi dengan kemampuan penyeimbangan beban dinamis, aplikasi SaaS dapat menangani beban tambahan apa pun secara efektif tanpa memengaruhi perilaku normalnya.
7. Solusi Green IT: Aplikasi SaaS mendukung solusi Green IT. Karena aplikasi SaaS bersifat multitenant dan berbagi sumber daya dan instans aplikasi yang sama, membeli perangkat keras dan sumber daya tambahan dapat dihilangkan. Pemanfaatan sumber daya yang tinggi memungkinkan aplikasi mengkonsumsi lebih sedikit energi dan daya komputasi. Solusi SaaS menjadi lebih pintar dan memiliki fitur hemat energi di dalamnya yang tidak memakan banyak sumber daya untuk pengoperasiannya.

**Kesesuaian SaaS**

Aplikasi SaaS digunakan oleh banyak individu dan organisasi karena sifatnya yang hemat biaya. Adopsi mereka oleh perusahaan besar juga meningkat dalam jumlah yang wajar. Tapi kita tidak bisa menggunakan aplikasi SaaS di semua tempat. Berikut ini adalah beberapa aplikasi di mana SaaS mungkin bukan pilihan terbaik:

- Beberapa aplikasi real-time yang membutuhkan pemrosesan data yang cepat
- Aplikasi di mana data organisasi lebih rahasia dan organisasi tidak ingin menghosting data mereka secara eksternal
- Aplikasi dimana aplikasi on-premise yang ada memenuhi kebutuhan organisasi

Berikut ini adalah contoh di mana SaaS adalah opsi terbaik:

- Untuk aplikasi di mana pengguna akhir mencari perangkat lunak sesuai permintaan daripada perangkat lunak jangka panjang/berbasis lisensi
- Untuk perusahaan pemula yang tidak dapat menginvestasikan lebih banyak uang untuk membeli perangkat lunak berlisensi
- Untuk aplikasi yang memerlukan aksesibilitas dari perangkat genggam atau thin client

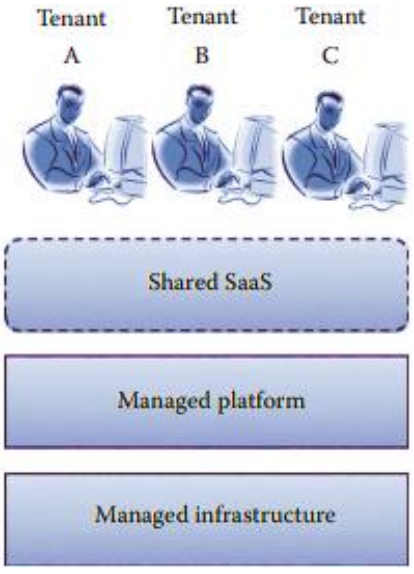
- Untuk aplikasi dengan beban yang tidak dapat diprediksi dan dinamis

## 9.2 PERSPEKTIF BERBEDA TENTANG PENGEMBANGAN SAAS

Model SaaS menyediakan akses web ke perangkat lunak komersial. Karena aplikasi SaaS dikerahkan dan dikelola oleh penyedia layanan, pelanggan mendapatkan akses ke layanan perangkat lunak tanpa biaya pemeliharaan infrastruktur dan platform yang mendasarinya. Penyedia SaaS juga dapat mengurangi biaya pemeliharaan dengan memilih layanan *cloud* lain yang sesuai seperti IaaS dan PaaS, mengurangi modal dan pengeluaran operasi untuk memelihara server. SaaS dapat digunakan dan dikirimkan dari infrastruktur tradisional atau infrastruktur *cloud*. Ada berbagai model penyebaran dan pengiriman SaaS yang tersedia untuk menguntungkan penyedia layanan dan pelanggan, yang dibahas dalam subbagian ini.

**SaaS dari Infrastruktur dan Platform Terkelola**

Model ini menggunakan infrastruktur tradisional dan platform untuk mengembangkan dan menerapkan aplikasi SaaS. Karakteristik komputasi awan hanya akan terpenuhi pada lapisan SaaS. Di dua lapisan lainnya (platform dan infrastruktur), karakteristik *cloud* tidak akan terpenuhi. Ini berarti bahwa infrastruktur dan platform yang mendasarinya tidak mendukung *cloud*. Tingkat multitenancy juga sangat rendah dalam model jenis ini karena multitenancy tidak tercapai pada level PaaS dan IaaS. Aplikasi SaaS yang dikembangkan akan dikirimkan ke pelanggan dalam model one-to-many. Gambar 9.1 mengilustrasikan konsep pengiriman SaaS dari infrastruktur dan platform yang dikelola sendiri.




**Gambar 9.1** Pengiriman SaaS dari infrastruktur dan platform terkelola.

*Pro*

- Model pengiriman SaaS jenis ini memastikan keamanan yang lebih besar untuk data pengguna karena infrastruktur dan platform dikelola oleh penyedia SaaS.
- Penyedia SaaS mendapatkan kendali penuh atas infrastruktur dan platform pengembangan.

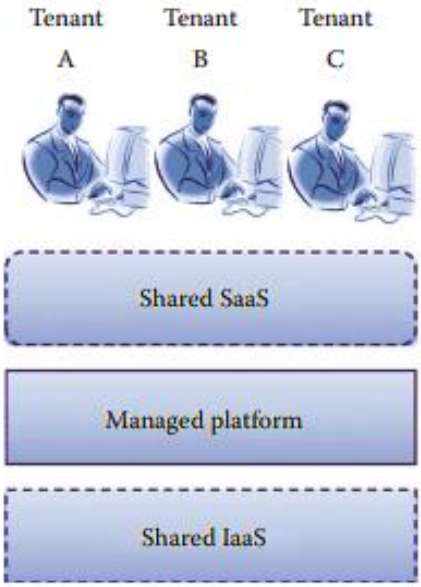
- Tidak ada masalah vendor lock-in. Aplikasi dapat dengan mudah dipindahkan ke infrastruktur lain tanpa modifikasi besar.

*Kontra*

- Lebih banyak biaya dalam memelihara infrastruktur dan platform yang mendasarinya.
- Penyedia layanan harus berinvestasi lebih banyak pada infrastruktur. Jadi, model ini tidak cocok untuk perusahaan pengembang SaaS yang tidak memiliki banyak investasi.
- Karena lingkungan pengembangan dikelola oleh penyedia layanan, ada biaya tambahan untuk memelihara skalabilitas dan ketersediaan aplikasi.
- Pemanfaatan sumber daya akan sangat rendah.

**SaaS dari IaaS dan Platform Terkelola**

Dalam model penyampaian layanan jenis ini, penyedia SaaS dapat menggunakan infrastruktur yang disediakan oleh penyedia IaaS mana pun. Penyedia infrastruktur dapat berupa penyedia IaaS publik atau swasta. Infrastruktur publik akan dipilih jika aplikasi SaaS tidak memerlukan keamanan lebih pada data. Jika aplikasi membutuhkan lebih banyak keamanan dan pada saat yang sama lebih banyak pemanfaatan sumber daya, mereka dapat memilih model IaaS pribadi. Di sini, multitenancy akan dicapai pada lapisan infrastruktur dan aplikasi. Penyedia layanan harus mengelola platform pengembangan mereka sendiri. Karena infrastruktur diberikan oleh pihak ketiga, ada kemungkinan vendor lock-in pada lapisan IaaS. Gambar 9.2 mengilustrasikan pengiriman SaaS dari IaaS dan platform yang dikelola sendiri.




**Gambar 9.2** Pengiriman SaaS dari IaaS bersama dan platform terkelola.

*Pro*

- Memastikan pemanfaatan sumber daya yang tinggi di tingkat infrastruktur.
- Mengurangi investasi modal pada infrastruktur.
- Investasi modal dapat dikurangi.

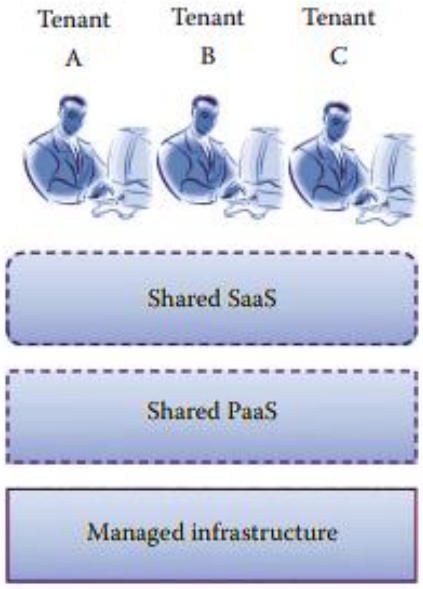
*Kontra*

- Masih ada biaya tambahan dalam pemeliharaan platform pengembangan.
- Harus mengaktifkan fitur yang sangat skalabel dan tersedia secara manual.

- Ada kemungkinan vendor lock-in pada lapisan infrastruktur.

**SaaS dari Managed Infrastructure dan PaaS**

SaaS dapat dikembangkan dan dikirimkan dari infrastruktur yang dikelola sendiri dan PaaS bersama. Biasanya, aplikasi yang dikembangkan menggunakan model ini akan di-deploy sebagai aplikasi on-premise. PaaS yang digunakan di sini umumnya adalah PaaS privat. Ada banyak penyedia PaaS yang mengizinkan pelanggan membangun PaaS pribadi mereka sendiri di pusat data terkelola mereka. Model ini paling sesuai dengan penerapan SaaS oleh komunitas. Dalam model penyebaran komunitas, infrastruktur akan dipelihara oleh sekelompok organisasi. Platform pengembangan (PaaS) dapat diakses sebagai layanan oleh berbagai organisasi. Jenis model ini akan mengurangi biaya overhead dalam pemeliharaan platform. Fitur khusus SaaS seperti skalabilitas tinggi dan ketersediaan akan ditangani oleh PaaS itu sendiri. Tapi overhead dalam pemeliharaan infrastruktur akan tetap tidak terpecahkan. Jenis model ini juga memberikan keamanan tinggi pada data. Penguncian vendor pada tingkat PaaS dapat dimungkinkan dalam model jenis ini karena penyedia akan menjadi penyedia pihak ketiga. Jenis masalah ini dapat dihindari dengan membangun platform PaaS kita sendiri di infrastruktur terkelola. Gambar 9.3 menggambarkan ide pengembangan dan pengiriman SaaS dari infrastruktur terkelola dan PaaS bersama.




**Gambar 9.3** Pengiriman SaaS dari infrastruktur terkelola dan PaaS bersama.

*Pro*

- Skalabilitas dan ketersediaan aplikasi akan disediakan oleh PaaS secara default. Jadi, penyedia SaaS bisa lebih berkonsentrasi pada pengembangan aplikasi.
- Keamanan akan dimoderasi, dan ada tata kelola penuh atas data pengguna.

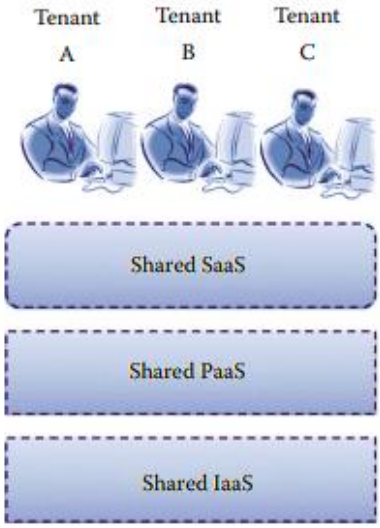
*Kontra*

- Meskipun biaya pemeliharaan platform pengembangan berkurang, biaya pemeliharaan infrastruktur masih belum terpecahkan.
- Jenis model ini hanya cocok untuk aplikasi SaaS pribadi/publik. Karena beban SaaS publik tinggi dan tidak dapat diprediksi, penyedia layanan mungkin harus membeli

infrastruktur tambahan baru untuk menangani beban tambahan. Ini tidak mungkin untuk perusahaan pengembangan SaaS kecil.

**SaaS dari IaaS dan PaaS**

Model pengembangan dan pengiriman SaaS jenis ini mendapatkan semua manfaat komputasi awan. Infrastruktur untuk mengembangkan dan menggunakan aplikasi SaaS akan disediakan oleh penyedia IaaS, mengurangi biaya tambahan dalam memelihara infrastruktur yang mendasarinya. Dengan cara yang sama, platform pengembangan juga dapat disediakan oleh penyedia PaaS. Penyedia IaaS dan PaaS mungkin bersifat publik atau pribadi. IaaS dan PaaS publik memberikan lebih banyak manfaat daripada IaaS dan PaaS privat. Biasanya, IaaS dan PaaS publik akan dipilih untuk mengurangi biaya pemeliharaan dan investasi awal. Multitenancy disediakan di semua lapisan, yaitu lapisan infrastruktur, platform, dan aplikasi. Jenis multitenansi ini disebut sebagai multitenansi tingkat tinggi, yang tidak tersedia dalam model pengembangan dan pengiriman SaaS lainnya. Gambar 9.4 mengilustrasikan ide mengembangkan dan mengirimkan SaaS dari IaaS dan PaaS.




**Gambar 9.4** Pengiriman SaaS dari IaaS dan PaaS bersama.

*Pro*

- Model pengiriman terbaik yang sesuai dengan aplikasi SaaS publik.
- Memastikan pemanfaatan sumber daya yang tinggi karena memungkinkan multitenancy di semua lapisan aplikasi. Ini juga mendukung aplikasi Green IT.
- Penskalaan dinamis penyedia IaaS dan PaaS memastikan skalabilitas aplikasi yang tinggi. Perusahaan pengembangan SaaS tidak perlu khawatir tentang skalabilitas aplikasi.
- Ketersediaan aplikasi yang tinggi dijamin oleh replika dan mekanisme pencadangan dan pemulihan yang disediakan oleh penyedia layanan.
- Tidak ada biaya tambahan dalam pemeliharaan infrastruktur dan platform pengembangan. Ini memungkinkan perusahaan pengembangan SaaS untuk mengembangkan lebih banyak aplikasi dalam rentang waktu singkat.

*Kontra*

- Karena jenis model ini sebagian besar menggunakan model IaaS dan PaaS publik, aplikasi akan dihosting sebagai aplikasi off-premise. Jadi, tidak ada tata kelola atas data pelanggan.
- Ada kemungkinan serangan lintas penyewa karena multitenancy diaktifkan di semua tingkat aplikasi.

Ada berbagai model pengembangan dan penyebaran yang dibahas dalam subbagian ini. Model penerapan SaaS (publik atau privat atau komunitas) akan dipilih berdasarkan persyaratan keamanan data pengguna. Investasi pembelian dan pengelolaan infrastruktur juga akan dipertimbangkan sebelum memilih model penyebaran.

## 9.3 TANTANGAN BARU

Banyak aplikasi web lokal digantikan oleh aplikasi SaaS karena keuntungan bisnis dari aplikasi SaaS. Adopsi SaaS oleh perusahaan besar semakin meningkat seiring dengan adopsi oleh pengguna individu. Ini karena masalah keamanan data yang disimpan di *cloud*. Perusahaan besar mengkhawatirkan keamanan data, tata kelola data, dan ketersediaan. Pada subbagian ini, kita akan membahas tantangan yang membuat pengembangan SaaS menjadi sulit.

**Multitenansi**

Mengaktifkan multitenancy dalam aplikasi SaaS merupakan tantangan yang dihadapi semua pengembang saat ini. Multitenancy dapat dicapai pada tingkat infrastruktur, platform, database, dan aplikasi untuk pemanfaatan sumber daya yang lebih baik. Multitenancy adalah model satu-ke-banyak yang memungkinkan banyak pelanggan berbagi satu contoh kode dan basis data aplikasi SaaS. Jadi, pengembang perlu mempelajari pengetahuan yang dibutuhkan untuk mengembangkan perangkat lunak multitenant. Misalnya, ada beberapa tingkat multitenancy yang tersedia sehubungan dengan database, yaitu database terpisah, database bersama dan skema terpisah, dan database bersama dan skema bersama. Pengembang harus memilih tingkat multitenancy yang tepat berdasarkan kebutuhan pelanggan sebelum mengembangkan aplikasi. Pengembang dapat menggunakan PaaS apa pun untuk mengurangi biaya tambahan dalam mengaktifkan multitenancy aplikasi SaaS.

**Keamanan**

Konsekuensi penting pertama dari model multitenancy dari aplikasi SaaS adalah keamanan data. Karena lingkungan dibagi, ada kemungkinan pelanggaran data. Satu penyewa dapat dengan mudah mendapatkan akses ke data penyewa lainnya, yang akan mengakibatkan masalah serius. Inilah alasan mengapa sebagian besar perusahaan tidak siap mengadopsi aplikasi SaaS untuk kebutuhan bisnis mereka dan mengisolasi data penyewa adalah salah satu cara untuk mengatasinya, yang merupakan tantangan terbesar bagi pengembang SaaS mana pun. Karena aplikasi SaaS berbasis web, ancaman keamanan yang berlaku untuk aplikasi tradisional juga berlaku untuk aplikasi SaaS. Serangan keamanan internal dan eksternal harus dideteksi dan dicegah dengan menggunakan sistem deteksi intrusi dan pencegahan intrusi yang tepat. Pengembang dapat memasukkan mekanisme keamanan lain seperti enkripsi yang kuat, audit, otentikasi, kontrol akses, dan otorisasi untuk mengamankan aplikasi SaaS.

**Skalabilitas**

Aplikasi SaaS yang dapat diskalakan harus menangani beban ekstra pada aplikasi secara efisien. Aplikasi SaaS dikatakan sangat terukur jika menangani beban tambahan dengan benar. Skalabilitas aplikasi dapat dipertahankan dengan memprediksi beban pada sistem dan menyediakan sumber daya yang cukup untuk menangani beban tersebut. Karena pengguna aplikasi SaaS beragam dan dapat terhubung dan terputus dari sistem kapan saja, memprediksi beban pada aplikasi merupakan tantangan besar bagi pengembang. Jadi, arsitek perangkat lunak harus merancang arsitektur yang dapat menangani segala jenis beban pada aplikasi. Arsitek harus menggunakan penskalaan vertikal, penskalaan horizontal, dan penyeimbang beban untuk mengembangkan aplikasi yang sangat dapat diskalakan. Skalabilitas pada tingkat platform dan infrastruktur juga dapat menentukan efisiensi aplikasi. Jadi, arsitektur aplikasi harus menggunakan IaaS dan PaaS untuk memanfaatkan keunggulan layanan *cloud*.

**Ketersediaan**

Karena pengguna SaaS menyimpan data mereka di pusat data penyedia layanan, memastikan ketersediaan data 99,99% merupakan tantangan bagi pengembang SaaS, dan cara yang lebih baik untuk mengatasi hal ini adalah dengan memelihara cadangan atau mekanisme replika yang tepat. Kami tidak dapat memprediksi kegagalan data yang disimpan sebelumnya dan tidak dapat mengambil tindakan pencegahan. Sebagian besar pengembang aplikasi tradisional mengandalkan alat pihak ketiga untuk memastikan pencadangan dan pemulihan bencana. Pengembang dapat menggunakan database NoSQL terdistribusi yang mendukung replikasi otomatis dan pemulihan bencana daripada database relasional.

**Kegunaan**

Tantangan baru yang diperkenalkan oleh aplikasi SaaS adalah dukungan banyak perangkat. Pengguna dapat mengakses aplikasi dari laptop, ponsel, tablet, dan perangkat genggam lainnya. Pengembang aplikasi web tradisional biasa mengembangkan aplikasi Internet kaya yang tidak dapat diadaptasi ke perangkat seluler.

Jadi, dengan menjaga keragaman perangkat pengguna akhir, pengembang harus mengembangkan UI web responsif yang beradaptasi secara otomatis ke semua perangkat pengguna. Pengembang tidak dapat mengembangkan versi terpisah dari aplikasi untuk perangkat yang berbeda. Tantangan lain bagi pengembang adalah mempertahankan pengaturan penyesuaian per penyewa. Setelan kustomisasi UI setiap pengguna harus dikelola dengan baik untuk setiap perangkat dan tidak memengaruhi yang lain.

**Pendaftaran Layanan Mandiri**

Banyak situs jejaring sosial tradisional menggunakan fitur pendaftaran swalayan untuk pengguna. Tapi itu tidak wajib untuk semua aplikasi web tradisional. Tetapi untuk aplikasi SaaS, itu adalah fitur wajib untuk dimasukkan, menciptakan tantangan bagi para pengembang. Aplikasi SaaS harus tersedia untuk pengguna segera setelah mereka mendaftar. Aplikasi tidak boleh mendorong persetujuan admin apa pun untuk mengizinkan pengguna mengakses layanan. Dengan cara yang sama, aplikasi harus mengizinkan pengguna untuk berhenti berlangganan menggunakan layanan kapan saja.

**Penagihan Otomatis**

Tantangan lain yang ditimbulkan oleh karakteristik unik SaaS adalah penagihan otomatis. Sebagian besar pengguna aplikasi SaaS yang ada menghadapi masalah penagihan yang kasar bahkan setelah terputus dari layanan. Jadi, pengembang harus menggabungkan mekanisme yang tepat untuk menghindari penagihan yang kasar dan menjaga riwayat penggunaan masing-masing penyewa. Penyewa dapat berlangganan layanan yang berbeda dari penyedia layanan yang sama, dan suatu mekanisme harus tersedia bagi pelanggan untuk melihat laporan dan tagihan total dan per penggunaan layanan mereka.

**Pembaruan Non-gangguan**

Pembaruan aplikasi yang sering dapat mengakibatkan tidak tersedianya aplikasi. Setiap kali pembaruan dilakukan, seharusnya tidak mempengaruhi perilaku normal pengguna. Konsekuensi lain dari pemutakhiran dapat menyebabkan pelanggaran perjanjian tingkat layanan (SLA). Operasi pembaruan memutuskan waktu henti aplikasi, yang merupakan parameter SLA yang penting. Pembaruan harus dilakukan sedemikian rupa sehingga tidak menambah waktu henti yang disebutkan dalam SLA. Jadi, menjadwalkan dan melakukan pembaruan aplikasi SaaS yang tidak mengganggu adalah tantangan terbesar yang dihadapi pengembang untuk menghindari pelanggaran SLA.

**Integrasi Layanan**

Aplikasi SaaS yang baik harus dapat berintegrasi dengan layanan pihak ketiga lainnya. Biasanya, kami tidak dapat mengintegrasikan layanan pihak ketiga secara langsung. Kami harus menggunakan API penyedia layanan untuk melakukan integrasi layanan. Jadi, arsitektur aplikasi SaaS harus memungkinkan integrasi layanan dari layanan lain melalui API mereka. Integrasi layanan dapat dicapai dengan mengikuti arsitektur berorientasi layanan (SOA) yang tepat. Manfaat lain dari integrasi layanan adalah kita dapat mengotomatiskan penerapan beberapa fungsi. Banyak penyedia SaaS gagal mendukung integrasi layanan, yang perlu mereka tingkatkan saat mengembangkan aplikasi SaaS.

**Penguncian Vendor**

Masalah utama layanan *cloud* publik adalah vendor lock-in. Tidak ada standar global yang diikuti di antara penyedia layanan. Setiap penyedia layanan mengikuti cara mereka sendiri dalam menyediakan infrastruktur dan layanan platform, dan migrasi aplikasi SaaS dari satu penyedia layanan ke penyedia layanan lainnya menjadi sulit dan menyebabkan penguncian vendor. Jadi, pengembang harus memilih PaaS atau IaaS yang dapat dioperasikan dengan penyedia layanan lainnya.

## 9.4 PENGEMBANGAN PERANGKAT LUNAK *CLOUD*-AWARE MENGGUNAKAN TEKNOLOGI PAAS

PaaS adalah model layanan *cloud* yang banyak digunakan yang memungkinkan pengembang untuk mengembangkan aplikasi secara online. PaaS menyediakan PaaS pengembangan berdasarkan permintaan 0. Pengembang tidak perlu menginstal perangkat lunak kelas berat apa pun di mesin mereka untuk menggunakan PaaS. Pengembang dapat mengembangkan dan menyebarkan aplikasi secara online melalui alat klien seperti web UI, Command Line Interface, web CLI, dan Representational State Transfer (REST) API yang

disediakan oleh penyedia layanan. Biasanya, pengembangan aplikasi SaaS menimbulkan banyak tantangan seperti yang telah dibahas di bagian sebelumnya. Dengan lingkungan pengembangan tradisional, sangat sulit untuk mengembangkan SaaS yang berhasil yang memenuhi semua persyaratan khusus SaaS. Untuk mengatasi tantangan ini, perusahaan pengembangan SaaS dapat menggunakan PaaS populer yang menyediakan banyak fitur khusus SaaS secara default. Dengan menggunakan PaaS, pengembang dapat lebih berkonsentrasi pada fungsionalitas aplikasi daripada berjuang untuk mengaktifkan fitur khusus SaaS. Dalam subbagian ini, kita akan membahas pengembangan aplikasi SaaS berbasis *cloud* menggunakan teknologi PaaS.

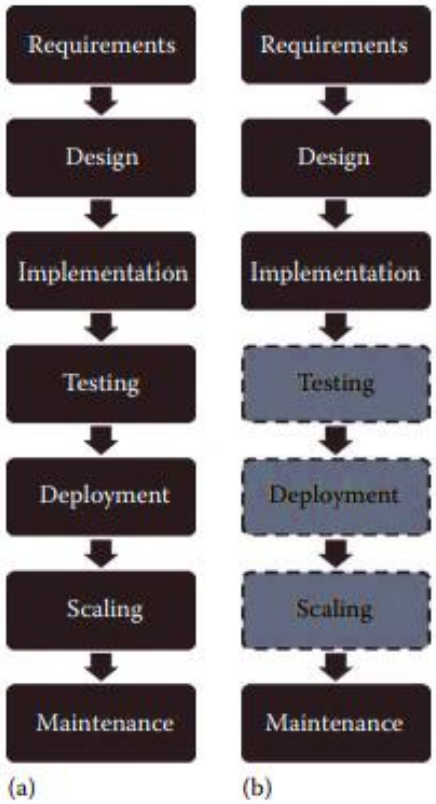
Manfaat PaaS: PaaS digunakan oleh banyak perusahaan pengembangan SaaS kecil dan ISV. Berikut ini membahas banyak karakteristik yang meningkatkan adopsi PaaS oleh suatu organisasi:

- PaaS menyediakan layanan untuk mengembangkan, menguji, menerapkan, menghosting, dan memelihara aplikasi di satu tempat.
- Sebagian besar penyedia layanan menawarkan polyglot PaaS di mana pengembang dapat menggunakan berbagai lingkungan pengembangan aplikasi dalam satu lingkungan pengembangan terintegrasi (IDE).
- Berbagai alat klien seperti web UI, web CLI, REST API, dan integrasi IDE meningkatkan kemudahan pengembangan dan penyebaran aplikasi.
- PaaS menawarkan arsitektur multitenant bawaan untuk aplikasi yang dikembangkan menggunakan PaaS.
- PaaS menyediakan penyeimbang beban perangkat lunak yang memastikan penskalaan dinamis aplikasi.
- Replika yang dikelola oleh penyedia PaaS memastikan ketersediaan aplikasi yang tinggi.
- Penyedia PaaS juga memungkinkan integrasi dengan layanan web atau *cloud* lainnya untuk mengembangkan layanan *cloud* komposit.
- PaaS meningkatkan kolaborasi antara tim pengembangan karena aplikasi akan diterapkan di tempat terpusat.
- Fitur khusus SaaS penting lainnya seperti alat pemantauan dan penagihan otomatis akan ditawarkan oleh PaaS itu sendiri.

Sebelum PaaS dan setelah PaaS: PaaS mengubah proses pengembangan perangkat lunak secara total jika dibandingkan dengan model pengembangan perangkat lunak tradisional. Dalam pengembangan perangkat lunak tradisional, proses pengembangan akan dimulai dari analisis kebutuhan yang melibatkan semua pemangku kepentingan sistem. Fase kedua adalah fase desain yang mencakup arsitektur perangkat lunak, UI, dan desain basis data. Tahap implementasi adalah pengembangan aktual, atau pengkodean aplikasi dengan platform pengembangan yang tersedia. Di sini, platform pengembangan akan menjadi perangkat lunak kelas berat berbasis lisensi yang membutuhkan mesin dengan daya komputasi tinggi, memaksa perusahaan untuk berinvestasi lebih banyak pada platform pengembangan dan perangkat keras. Pengujian adalah fase selanjutnya dari pengembangan perangkat lunak yang terutama memastikan persyaratan nonfungsional seperti keamanan, skalabilitas,

ketersediaan, dan portabilitas. Ada banyak tools yang tersedia dalam melakukan proses pengujian terhadap aplikasi yang dikembangkan. Setelah menguji suatu produk, produk tersebut dapat digunakan dalam infrastruktur yang sesuai dan dapat dikirimkan ke pengguna akhir. Biasanya, jika aplikasi tersebut adalah aplikasi yang berdiri sendiri, itu akan memberikan lisensi. Jika itu adalah aplikasi web, itu akan dikirimkan melalui Internet. Umumnya, setiap pelanggan aplikasi akan mendapatkan salinan aplikasi yang terpisah. Dalam kasus aplikasi web, aplikasi akan dihosting di infrastruktur penyedia layanan atau infrastruktur milik pelanggan. Langkah selanjutnya adalah menjaga skalabilitas dan ketersediaan aplikasi. Perusahaan harus menyimpan server tambahan yang cukup untuk menangani beban tambahan aplikasi. Namun secara real time, beban pada aplikasi akan bersifat dinamis dan tidak dapat diprediksi. Jadi, kami tidak dapat memutuskan kekuatan perangkat keras, yang perlu kami tambahkan nanti. Di sini, membeli dan memelihara perangkat keras tambahan akan menjadi biaya tambahan bagi perusahaan pengembang. Hal penting berikutnya adalah ketersediaan aplikasi. Untuk memastikan ketersediaan tinggi, perusahaan perlu menyimpan replika. Sekali lagi manajemen replika adalah masalah besar bagi perusahaan. Pada akhirnya, proses pemeliharaan aplikasi akan dilakukan oleh perusahaan. Biasanya, pembaruan akan dilakukan oleh pelanggan dari mesin mereka. Setiap salinan aplikasi harus diperbarui secara terpisah. Selain itu, terkadang pembaruan melalui koneksi Internet yang lambat akan mengganggu perilaku normal sistem.

Jadi, untuk menghindari semua masalah ini dalam metodologi pengembangan perangkat lunak tradisional, perusahaan mulai menggunakan teknologi PaaS untuk produktivitas yang lebih baik.




**Gambar 9.5** Pengembangan perangkat lunak (a) sebelum dan (b) setelah PaaS.

Aplikasi SaaS yang dikembangkan berbeda dengan aplikasi tradisional. Pengembang aplikasi SaaS harus mengaktifkan fitur berikut ke aplikasi: multitenancy, penskalaan dinamis, dan ketersediaan tinggi. Proses pengembangan aplikasi SaaS menggunakan teknologi PaaS dibahas berikut ini.

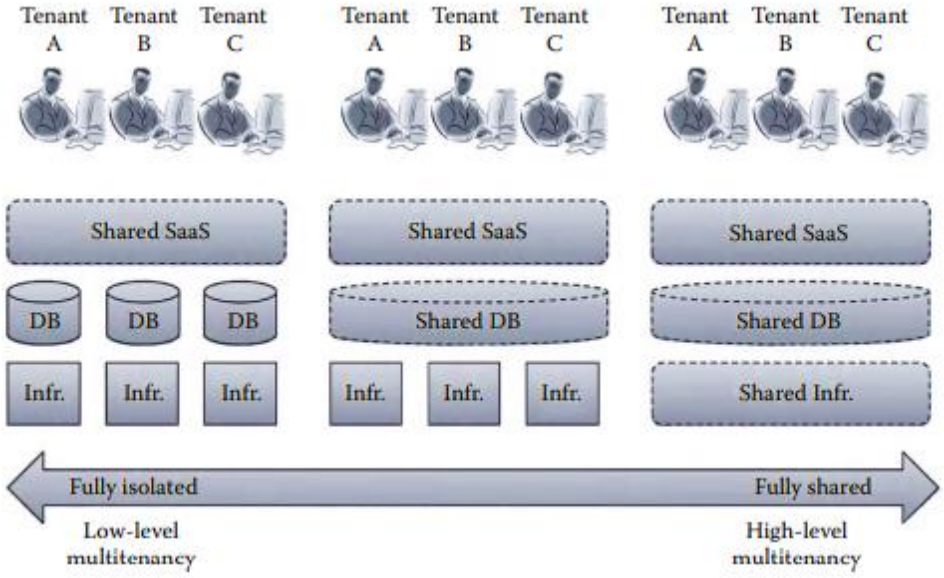
**Analisis Kebutuhan**

Tim pengembangan harus mengatasi kebutuhan aplikasi SaaS yang sering berubah. Umumnya, dalam analisis kebutuhan, hanya pelanggan dan tim pengembangan yang akan dilibatkan. Namun dalam pengembangan aplikasi SaaS, penyedia layanan untuk alat PaaS juga harus dilibatkan. Tim pengumpul persyaratan harus mengumpulkan persyaratan dari semua pemangku kepentingan. Dokumen persyaratan harus mencakup persyaratan fungsional, nonfungsional, dan persyaratan khusus SaaS lainnya. Sebelum mengembangkan aplikasi, tim pengumpul persyaratan harus menganalisis kesesuaian model pengiriman SaaS untuk kebutuhan pelanggan. Secara umum, model pengiriman SaaS akan dipilih berdasarkan keamanan dan ROI. Hal penting berikutnya dalam fase analisis kebutuhan adalah model penerapan aplikasi SaaS. Jika aplikasi tidak membutuhkan keamanan lebih, kami dapat mengembangkan aplikasi SaaS dari penyedia PaaS publik mana pun. Jika membutuhkan keamanan lebih, maka kami harus memilih penyedia PaaS swasta mana pun untuk mengembangkan aplikasi. Biasanya, dalam model penerapan publik, biaya pemeliharaan akan rendah dan ancaman keamanan akan tinggi. Namun dalam model penerapan pribadi, biaya pemeliharaan akan tinggi dan ancaman keamanan akan moderat. Karena aplikasi SaaS akan dikirimkan melalui Internet, perusahaan pengembangan SaaS harus menugaskan pakar keamanan dan arsitek perangkat lunak yang sangat terampil untuk berhasil dalam bisnis pengembangan SaaS. Pakar keamanan harus memastikan keamanan di semua lapisan aplikasi. Arsitek perangkat lunak harus memastikan skalabilitas dan ketersediaan aplikasi. Setelah analisis persyaratan dilakukan dengan benar, tim pengembang dapat mulai merancang arsitektur.

**Arsitektur Multipenyewa**

Karakteristik penting dari aplikasi SaaS adalah multitenancy dimana beberapa pelanggan diperbolehkan untuk berbagi aplikasi yang sama. Mencapai multitenansi tergantung pada arsitektur perangkat lunak. Arsitektur perangkat lunak harus memastikan multitenancy dari aplikasi SaaS. Multitenancy dapat dicapai pada tingkat infrastruktur, platform pengembangan, basis data, dan aplikasi. Arsitek dapat memilih berbagai tingkat multitenancy untuk memanfaatkan sumber daya secara efektif. Jika arsitek perangkat lunak memilih penyedia IaaS untuk kebutuhan infrastruktur, maka arsitek dibebaskan dari mengaktifkan multitenancy di tingkat infrastruktur. Jika arsitek memilih penyedia PaaS, tingkat platform dan multitenancy tingkat infrastruktur akan disediakan oleh penyedia PaaS itu sendiri. Jadi, pekerjaan arsitek perangkat lunak dikurangi untuk mengaktifkan fitur multitenancy hanya pada tingkat perangkat lunak. Bergantung pada tingkat multitenancy, keamanan isolasi data diputuskan. Jika multitenancy dicapai di semua level, maka disebut multitenancy level tinggi. Jika multitenancy dicapai hanya pada level aplikasi, maka disebut multitenancy level rendah. Ancaman keamanan terhadap data akan meningkat seiring dengan

meningkatnya tingkat multitenancy. Bergantung pada persyaratan keamanan pengguna, arsitek harus memilih tingkat multitenancy. Keuntungan lain dari multitenancy adalah pemanfaatan sumber daya. Jika perusahaan menginginkan pemanfaatan sumber daya yang tinggi, mereka dapat menggunakan multitenancy tingkat tinggi. Gambar 9.6 mengilustrasikan berbagai tingkat multitenancy dan isolasi yang tersedia untuk memastikan pemanfaatan sumber daya dan keamanan untuk arsitek perangkat lunak.



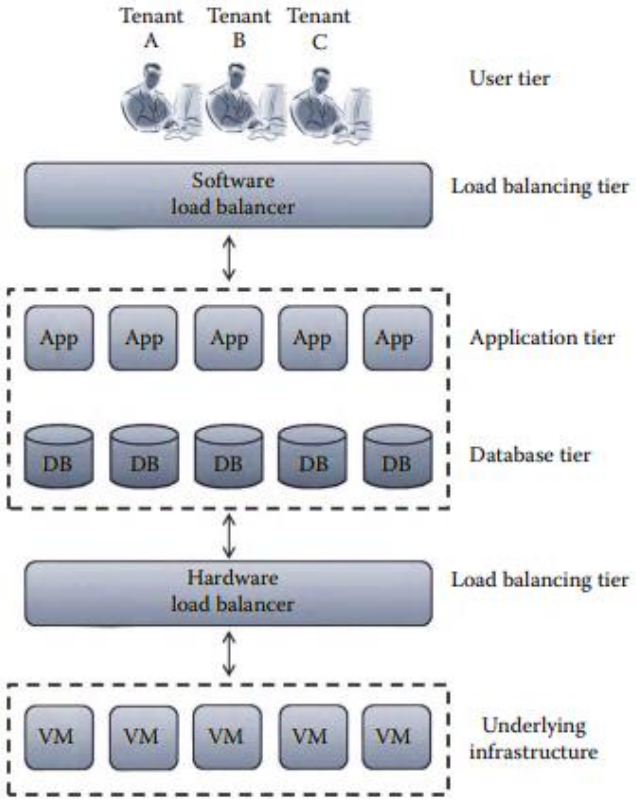

**Gambar 9.6** Tingkat multitenancy yang berbeda.

**Arsitektur yang Sangat Dapat Diskalakan dan Tersedia**

Karakteristik penting lainnya dari aplikasi SaaS adalah penskalaan dinamis. Fitur penskalaan dinamis tidak wajib dalam kasus aplikasi web tradisional. Namun dalam kasus aplikasi SaaS, penskalaan dinamis sangat penting. Seperti multitenancy, mencapai penskalaan dinamis juga bergantung pada arsitektur perangkat lunak. Karena beban pada aplikasi SaaS menjadi tidak dapat diprediksi dan bertambah atau berkurang setiap saat, arsitektur harus memastikan kinerja yang sama pada beban yang bervariasi. Skalabilitas aplikasi SaaS dapat dicapai dengan menggunakan penskalaan horizontal, penskalaan vertikal, penyeimbang beban perangkat lunak, dan penyeimbang beban perangkat keras. Dalam penskalaan horizontal, sumber daya identik (server aplikasi, server database, dan infrastruktur) akan ditambahkan ke aplikasi untuk menangani beban tambahan.

Dalam penskalaan vertikal, kapasitas server (aplikasi, database, dan infrastruktur) akan ditingkatkan seiring dengan peningkatan beban. Penyeimbang beban perangkat lunak juga dapat digunakan untuk memastikan skalabilitas dinamis aplikasi SaaS. Peran penyeimbang beban perangkat lunak adalah untuk mendistribusikan permintaan pengguna tambahan di berbagai aplikasi dan server basis data. Penyeimbang beban perangkat keras akan mendistribusikan beban ke berbagai mesin virtual saat ada kebutuhan akan daya komputasi yang lebih besar. Jika infrastruktur dan platform pengembangan dikonsumsi dari penyedia layanan apa pun (IaaS, PaaS), mereka akan menyediakan penyeimbang beban perangkat keras dan perangkat lunak untuk menyeimbangkan beban. Jika platform dan infrastruktur dikelola

sendiri, maka perusahaan pengembangan SaaS harus mengandalkan alat pihak ketiga atau mereka harus mengembangkannya sendiri. Gambar 9.7 mengilustrasikan arsitektur SaaS tipikal yang digunakan untuk mencapai skalabilitas tinggi dan ketersediaan aplikasi SaaS.




**Gambar 9.7** Arsitektur khas dari aplikasi SaaS.

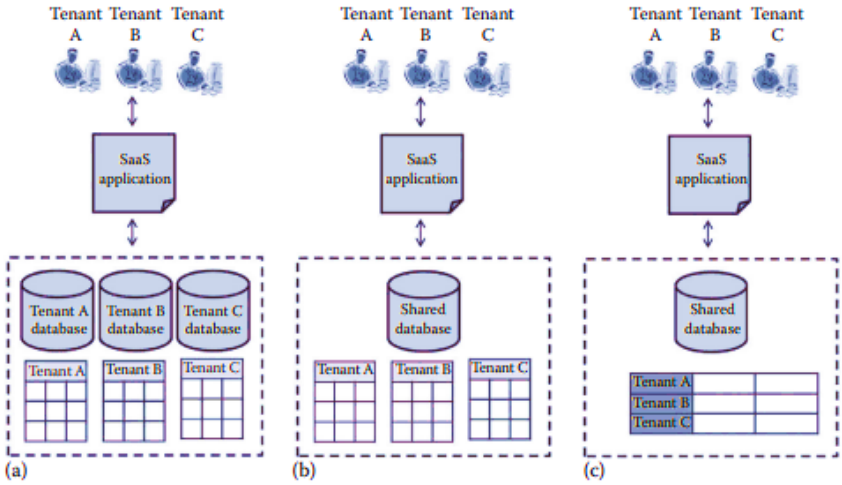
Seperti multitenancy dan skalabilitas, ketersediaan aplikasi juga merupakan karakteristik penting dari aplikasi SaaS. Ketersediaan aplikasi menentukan waktu aktif dan waktu hentinya. Ketersediaan aplikasi merupakan parameter penting dari SLA. Memastikan 99,99% ketersediaan aplikasi bergantung pada mekanisme replika yang ditentukan dalam arsitektur perangkat lunak. Seperti yang ditunjukkan pada Gambar 9.7, beberapa salinan dari mesin virtual dan aplikasi database harus dipelihara untuk ketersediaan yang tinggi. Sambil mempertahankan replika, aspek penting lainnya adalah waktu pemulihan setelah terjadi kegagalan. Aplikasi harus toleran terhadap kesalahan, dan waktu pemulihan harus minimal untuk menghindari pelanggaran SLA. Replika harus dipelihara di dekat lokasi pelanggan untuk mengurangi waktu pemulihan setelah terjadi kegagalan atau bencana.

**Desain Basis Data**

Mencapai multitenancy, skalabilitas, dan ketersediaan di tingkat database merupakan kriteria penting untuk pengembangan SaaS yang sukses. Desain database untuk multitenancy harus mempertimbangkan persyaratan keamanan data.

Multitenancy pada level database dapat dicapai dengan berbagi instance database, berbagi tabel database, dan berbagi skema database. Bergantung pada keamanan aplikasi, database harus dirancang untuk mengamankan multitenancy. Multitenancy tingkat database dapat dicapai dalam tiga cara berbeda seperti yang diilustrasikan pada Gambar 9.8. Jika

perancang basis data memilih basis data terpisah untuk penyewa yang berbeda seperti yang ditunjukkan pada Gambar 9.8b, keamanan akan terjamin. Jika database bersama dan skema terpisah dipilih seperti yang ditunjukkan pada Gambar 9.8a, keamanan data akan dimoderasi. Model multitenancy ketiga berbagi database dan skema untuk penyewa seperti yang ditunjukkan pada Gambar 9.8c.




**Gambar 9.8** Multitenancy tingkat basis data: (a) basis data terpisah, (b) basis data bersama dan skema terpisah, dan (c) basis data bersama dan skema bersama.

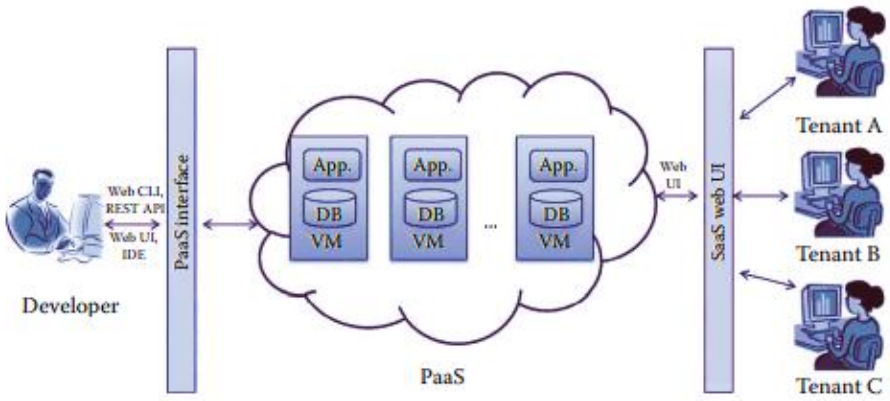
Skalabilitas dan ketersediaan database menentukan kinerja aplikasi. Sebagian besar aplikasi SaaS bersifat interaktif dan melibatkan sejumlah besar basis data permintaan baca dan tulis dari pengguna. Ketika jumlah permintaan melebihi kapasitas sebenarnya dari server basis data, permintaan tambahan harus dialihkan ke server lain. Saat permintaan dialihkan, perubahan data di satu server basis data juga harus tercermin di server basis data lainnya. Sebagian besar jenis data yang digunakan dalam aplikasi SaaS akan beragam dan mencakup data terstruktur, semi terstruktur, dan tidak terstruktur dalam jumlah besar. Jadi, untuk aplikasi SaaS, database Not Only Structured Query Language (NoSQL) akan menjadi pilihan yang lebih baik daripada database relasional berorientasi objek tradisional. Pengembang dapat memanfaatkan keunggulan database NoSQL untuk mencapai skalabilitas dan ketersediaan di tingkat database dengan cara yang efisien.

**Pengembangan SaaS**

Setelah merancang arsitektur dan database, pengembang perlu mengimplementasikan persyaratan fungsional yang diberikan oleh pelanggan. Seperti yang telah kita bahas sebelumnya, PaaS memfasilitasi pengembang dalam mengembangkan aplikasi SaaS yang sangat skalabel, tersedia, dan multitenant-aware.

Alat PaaS memungkinkan pengembang untuk mengembangkan aplikasi secara online, dan aplikasi tersebut akan diterapkan pada infrastruktur penyedia layanan segera setelah pengembang mendorong aplikasi tersebut secara online. Di sini, pengguna akhir atau konsumen SaaS dapat mengakses aplikasi secara online menggunakan UI web yang disediakan

oleh penyedia SaaS. Gambar 9.9 mengilustrasikan gambaran umum pengembangan SaaS menggunakan alat PaaS.




**Gambar 9.9** Gambaran umum pengembangan SaaS menggunakan PaaS.

Penyedia PaaS juga menyediakan alat pengujian di lingkungan pengembangan yang sama untuk memfasilitasi pengembang. Jadi, pengembang dapat menggunakan alat pengujian bawaan yang disediakan oleh penyedia PaaS untuk menguji aplikasi SaaS. Beberapa penyedia PaaS juga menawarkan pengujian aplikasi secara otomatis. Saat mengembangkan aplikasi, pengembang harus memasukkan yang berikut ini hal-hal untuk SaaS yang sukses:

- Desain UI responsif untuk mendukung banyak perangkat
- Mekanisme Role Based Access Control (RBAC), Access Control List (ACL) untuk mengidentifikasi pengguna dan penyewa secara unik
- Alat pemantauan yang akan memantau kinerja dan sering memberi tahu penyedia layanan
- Panel kontrol untuk penyewa dan admin untuk mengelola pengguna
- Panel kustomisasi yang berpusat pada pengguna yang tidak memengaruhi pengaturan penyewa atau pengguna lain
- Pendaftaran swalayan untuk pengguna
- Statistik penggunaan dan perhitungan tagihan
- Bantuan dokumentasi untuk menggunakan layanan ini
- Integrasi layanan

**Pemantauan dan Pemeliharaan SLA**

Segera setelah aplikasi SaaS dikembangkan dan digunakan menggunakan PaaS, pengguna akhir dapat mengakses aplikasi SaaS melalui Internet dari perangkat pengguna akhir seperti desktop, laptop, tablet, dan ponsel. Setelah mengirimkan aplikasi, setiap kesalahan perilaku, kegagalan, serangan keamanan, dan bencana aplikasi SaaS harus dipantau dan dicegah. Karena banyak pelanggan berbagi contoh yang sama dari satu aplikasi, setiap perilaku buruk dari satu penyewa akan memengaruhi penyewa lain dan sumber daya yang mendasarinya. Pembaruan harus dijadwalkan dan dilakukan sedemikian rupa sehingga tidak memengaruhi perilaku normal sistem, sehingga sering memperbarui aplikasi SaaS akan menawarkan versi terbarunya kepada pengguna akhir. Tetapi jika Anda sering memperbarui aplikasi dengan pembaruan massal, ini dapat menyebabkan tidak tersedianya aplikasi.

Pekerjaan penting lainnya dalam pemantauan SaaS adalah memantau pelanggaran SLA oleh penyedia layanan dan pelanggan. Jika ada pelanggaran SLA, alat pemantau harus memberi tahu pengembang untuk memperbaiki kesalahan yang menyebabkan pelanggaran SLA. Penyedia SaaS harus mendefinisikan SLA dengan jelas kepada pengguna akhir sebelum memberikan layanan apa pun. SLA harus mencakup ketersediaan, waktu respons, dan tingkat dukungan. Penyedia layanan juga harus memberikan dukungan 24 jam × 7 hari kepada pengguna akhir. Tim pengembangan harus sering menyelesaikan masalah segera setelah umpan balik diterima dari pengguna akhir. Ada banyak alat pemantauan pihak ketiga yang tersedia untuk memantau aplikasi SaaS. Perusahaan pengembangan SaaS dapat menggunakan alat pemantauan tersebut untuk mengurangi biaya overhead.

## 9.5 RINGKASAN

SaaS adalah salah satu layanan penting yang disediakan oleh komputasi awan. Tagihan berbasis penggunaan, skalabilitas tinggi, kemudahan akses, dan manfaat lainnya membuat sebagian besar pelanggan beralih dari aplikasi tradisional ke aplikasi SaaS. Perusahaan bisnis kecil/perusahaan pemula mulai menggunakan SaaS untuk mengurangi investasi mereka dalam membeli perangkat lunak yang kurang dimanfaatkan di organisasi mereka. Dengan menggunakan aplikasi SaaS sesuai permintaan, perusahaan mana pun dapat meningkatkan ROI mereka. Jika melihat penggunaan SaaS di perusahaan besar, sangat rendah dibandingkan dengan penggunaan individu dan perusahaan bisnis kecil. Perusahaan besar ragu untuk menggunakan aplikasi SaaS untuk organisasi mereka karena masalah keamanan. Aplikasi SaaS bersifat multitenant. Setiap kali pengguna berbagi aplikasi, ada kemungkinan serangan keamanan antara penyewa. Jika Anda menghapus fitur multitenant dari aplikasi SaaS di tingkat infrastruktur, platform, dan perangkat lunak, ini akan mengakibatkan biaya pengembangan yang tinggi. Jelas, pelanggan perlu membayar lebih untuk perangkat lunak yang mereka gunakan. Ketika seseorang memutuskan untuk menggunakan aplikasi SaaS, mereka harus mempertimbangkan persyaratan biaya dan keamanannya. Berdasarkan persyaratan biaya dan keamanan, penyedia layanan dapat mengikuti salah satu model pengembangan dan penerapan SaaS yang dibahas dalam bab ini. Selain masalah keamanan, aplikasi SaaS memperkenalkan banyak tantangan kepada pengembang seperti skalabilitas, ketersediaan, kegunaan, pendaftaran layanan mandiri, dan penagihan otomatis. Tantangan-tantangan ini dapat diatasi dengan menggabungkan praktik terbaik ke dalam rekayasa perangkat lunak dan teknologi PaaS. SaaS mengubah cara pengiriman perangkat lunak, dan PaaS mengubah cara perangkat lunak dikembangkan. PaaS mengotomatiskan proses penerapan, pengujian, dan penskalaan serta mengurangi pekerjaan manual dan biaya yang terlibat dalam pengembangan aplikasi. Penyedia SaaS juga dapat memanfaatkan komputasi awan IaaS untuk mengurangi investasi pembelian infrastruktur.

**Tinjau Poin**

- SaaS adalah salah satu model pengiriman perangkat lunak yang memungkinkan pengguna akhir berbagi aplikasi yang dihosting secara terpusat oleh penyedia.
- SaaS berisi karakteristik unik yang membedakannya dari perangkat lunak tradisional.

- Manfaat SaaS termasuk bayar per penggunaan, tanpa infrastruktur, kemudahan akses, pembaruan otomatis, dan layanan komposit.
- SaaS tidak sesuai dengan aplikasi yang memerlukan pemrosesan data yang cepat.
- Pengiriman SaaS dapat dalam berbagai bentuk: infrastruktur dan platform terkelola, IaaS dan platform terkelola, infrastruktur terkelola dan PaaS, serta IaaS dan PaaS.
- Tantangan SaaS seperti multitenancy, keamanan, skalabilitas, ketersediaan, dan kegunaan membuat pengembangan SaaS menjadi pekerjaan yang sulit bagi pengembang.
- Multitenancy adalah model one-to-many di mana satu instance aplikasi dapat digunakan bersama oleh banyak pengguna.
- Skalabilitas aplikasi SaaS bergantung pada seberapa baik aplikasi menangani beban tambahan.
- Ketersediaan aplikasi SaaS dapat ditingkatkan dengan mempertahankan mekanisme pencadangan dan pemulihan yang tepat.
- Kegunaan aplikasi SaaS bergantung pada desain UI adaptif dan responsif yang mendukung banyak perangkat.
- Fitur pendaftaran swalayan dari aplikasi SaaS memungkinkan pengguna akhir untuk berlangganan atau berhenti berlangganan dari layanan tanpa campur tangan penyedia.
- Fitur penagihan otomatis menyimpan riwayat penggunaan dan menyediakan tagihan berdasarkan penggunaan per penyewa atau per penggunaan layanan.
- Pembaruan tanpa gangguan memastikan waktu aktif aplikasi dan juga selama waktu pembaruan aplikasi.
- Integrasi layanan dari aplikasi SaaS memungkinkan setiap aplikasi SaaS untuk berintegrasi dengan layanan lain melalui API.
- Penguncian vendor tidak mengizinkan migrasi aplikasi ke penyedia layanan lain, yang merupakan masalah dengan sebagian besar penyedia *cloud* publik.
- PaaS mengubah cara perangkat lunak dikembangkan dengan menyediakan PaaS pengembangan.
- Pengembangan perangkat lunak *cloud*-aware memerlukan multitenant, arsitektur yang sangat dapat diskalakan.

**Latihan Soal**

1. Apa itu Perangkat Lunak sebagai Layanan (SaaS)? Apa bedanya dengan peranti lunak tradisional?
2. Jelaskan secara singkat manfaat aplikasi SaaS.
3. Apakah bijak memilih model pengiriman SaaS untuk semua jenis aplikasi? Benarkan jawaban Anda.
4. Jelaskan berbagai model pengembangan dan penyebaran SaaS dengan diagram yang rapi.
5. Sebutkan pro dan kontra dari berbagai model pengembangan dan penerapan SaaS.
6. Buat daftar tantangan yang membuat pengembangan SaaS menjadi tugas yang sulit. Juga, jelaskan lima tantangan secara rinci.

7. Tulis catatan singkat tentang manfaat yang diberikan oleh teknologi PaaS untuk mengembangkan aplikasi SaaS.
8. Jelaskan secara rinci bagaimana teknologi PaaS mengubah pengembangan perangkat lunak.
9. Jelaskan secara singkat analisis kebutuhan untuk aplikasi SaaS.
10. Jelaskan berbagai tingkat multitenancy dengan diagram yang rapi.
11. Gambarkan dan jelaskan arsitektur tipikal dari aplikasi SaaS, yang memastikan skalabilitas yang lebih baik dan ketersediaan yang tinggi.
12. Bagaimana multitenancy tingkat database dicapai? Jelaskan multitenancy tingkat database yang berbeda dengan diagram yang rapi.
13. Sebutkan fitur-fitur penting yang harus dimasukkan oleh pengembang SaaS saat mengembangkan aplikasi SaaS.
14. Tulis catatan singkat tentang pemantauan dan pemeliharaan SLA aplikasi SaaS.

# BAB 10
# JARINGAN UNTUK *CLOUD COMPUTING*

**Tujuan pembelajaran**

Setelah mempelajari bab ini, Anda harus bisa

- Memahami klasifikasi umum pusat data
- Menyajikan ikhtisar lingkungan pusat data
- Memahami masalah jaringan dasar di pusat data
- Menjelaskan tantangan kinerja yang dihadapi oleh TCP/IP dalam jaringan pusat data
- Jelaskan TCP yang baru dirancang untuk jaringan pusat data dan kebaruannya

**Pengantar**

Bab ini memberikan pengantar untuk jaringan di Pusat Data yang Diaktifkan *Cloud* (CEDC) dan masalahnya. Klasifikasi umum pusat data dan ikhtisar singkat tentang lingkungan pusat data disediakan untuk membiasakan pembaca dengan CEDC. Isu-isu utama yang terkait dengan jaringan di lingkungan *cloud* disajikan dengan penekanan pada isu-isu kinerja terkait TCP/IP. Protokol yang dirancang baru yang dirancang khusus untuk jaringan pusat data dijelaskan secara rinci, sambil menyebutkan kelebihan dan kekurangan masing-masing.

## 10.1 PENDAHULUAN

Internet selama beberapa tahun terakhir telah berubah dari sistem eksperimental menjadi sumber informasi yang sangat besar dan terdesentralisasi. Pusat data membentuk tulang punggung Internet dan menampung beragam aplikasi mulai dari jejaring sosial hingga pencarian web dan hosting web hingga iklan. Pusat data terutama diklasifikasikan menjadi dua jenis: yang bertujuan untuk menyediakan layanan online kepada pengguna, misalnya Google, Facebook, dan Yahoo, dan lainnya yang bertujuan untuk menyediakan sumber daya kepada pengguna, misalnya, Amazon Elastic Compute *Cloud* (EC2) dan Microsoft Azure.

Pusat data di masa lalu telah mengubah komputasi, dengan konsolidasi skala besar TI perusahaan menjadi hub pusat data dan dengan munculnya beberapa penyedia layanan komputasi awan. Dengan penerimaan komputasi awan yang meluas, pusat data telah menjadi kebutuhan. Sejak komputasi awan menjadi bagian penting di masa mendatang, mempelajari dan mengoptimalkan kinerja pusat data menjadi sangat penting.

Membangun pusat data yang efisien diperlukan untuk memperkuat pemrosesan data dan mengelola infrastruktur TI secara terpusat. Namun, sejak dimulainya pusat data, pengoperasian dan pemeliharaannya selalu menjadi tugas yang kompleks. Baru setelah tahun 1994 penggunaan pusat data meningkat secara luas. Berdasarkan kapasitas toleransi kesalahan dan waktu aktif layanan, pusat data saat ini diklasifikasikan menjadi empat tingkatan seperti yang ditunjukkan pada Tabel 10.1.

**Tabel 10.1** Klasifikasi Pusat Data

| Tingkatan | Fitur | Waktu aktif (%) |
|---|---|---|
| I | Komponen kapasitas nonredundan (uplink dan server tunggal) | 99.671 |
| II | Tingkat I + komponen kapasitas redundan | 99.741 |
| III | Tier I + Tier II + peralatan bertenaga ganda dan banyak tautan | 99.982 |
| IV | Tier I + Tier II + Tier III + semua komponen toleran terhadap kesalahan termasuk uplink, penyimpanan, sistem HVAC, server + semuanya bertenaga ganda | 99.995 |

Tier I–IV adalah metodologi standar yang digunakan untuk menentukan waktu aktif pusat data. Ini berguna untuk mengukur kinerja pusat data, investasi, dan laba atas investasi (ROI). Pusat data Tier IV dianggap paling kuat dan kurang rentan terhadap kegagalan. Ini dirancang untuk menjadi tuan rumah server penting misi dan sistem komputer dengan subsistem yang sepenuhnya redundan (pendinginan, daya, tautan jaringan, penyimpanan, dll.) dan zona keamanan terkotak yang dikendalikan oleh metode kontrol akses biometrik. Yang paling sederhana adalah data center Tier I, yang biasanya digunakan oleh toko-toko kecil.

Karena memelihara pusat data melibatkan banyak kerumitan dan biaya, perusahaan skala kecil dan menengah tidak dapat membangun pusat data mereka sendiri. Dengan demikian, perusahaan seperti Google, Amazon, Microsoft, Facebook, dan Yahoo berubah menjadi penyedia layanan komputasi awan dan mulai membangun pusat data Internet (IDC) untuk memenuhi permintaan penskalaan pengguna awan.

Saat ini, pusat data dari perusahaan yang disebutkan di atas menghosting beragam aplikasi seperti pencarian web, hosting web, jejaring sosial, penyimpanan, e-niaga, dan perhitungan skala besar. Seiring bertambahnya keragaman, kompleksitas, dan penetrasi layanan semacam itu, pusat data terus berkembang dan berkembang biak. Namun, sebagian besar pusat data menghadapi tantangan berat untuk mengurangi biaya server, biaya infrastruktur, dan konsumsi daya yang berlebihan serta mengoptimalkan kinerja jaringan. Tabel 10.2 menunjukkan ke mana perginya biaya di pusat data layanan *cloud* saat ini.

**Tabel 10.2** Panduan ke Mana Biaya Pergi di Pusat Data

| Diamortisasi (%) | Komponen Biaya | Subkomponen |
|---|---|---|
| 45 | Server | CPU, memori, sistem penyimpanan |
| 25 | Infrastruktur | Distribusi daya dan pendinginan |
| 15 | Menarik listrik | Biaya utilitas listrik |
| 15 | Jaringan | Tautan, transit, peralatan |

Bab ini menyoroti tantangan yang dihadapi dalam merancang jaringan yang cepat dan efisien untuk komunikasi dalam pusat data yang mendukung *cloud* (CEDC). Penekanan utama, bagaimanapun, adalah untuk memahami isu-isu lapisan transportasi di jaringan pusat data (DCN).

## 10.2 GAMBARAN UMUM LINGKUNGAN PUSAT DATA

Awalnya, organisasi digunakan untuk memelihara ruang server. Umumnya, ruang server digunakan untuk menampung server dan elektronik jaringan yang diperlukan untuk membangun LAN. Beberapa organisasi biasanya menyediakan satu ruang server utama dan satu ruang server siaga. Ruang server siaga dilengkapi untuk mempertahankan fungsi yang diperlukan jika ruang utama tidak berfungsi. Untuk toleransi kesalahan yang lebih baik, beberapa organisasi memilih untuk menempatkan ruangan ini di gedung yang berbeda.

Munculnya komputasi client-server dan Internet menyebabkan konsep pusat data. Perusahaan pusat data besar berkembang pesat di era dot-com ketika perusahaan Internet menghadapi pertumbuhan yang cepat. Pusat data difokuskan untuk menyediakan keandalan. Sejak saat itu, paradigma pusat data telah menjadi landasan bagi teknologi informasi yang menjalankan bisnis atau bisnis. Paradigma tersebut telah berevolusi selama beberapa dekade terakhir dan transformasional dalam 5-10 tahun terakhir.

Gambaran pusat data hampir selalu didahului dengan mission critical, karena itulah layanan yang disediakannya—perangkat keras dan perangkat lunak mission critical yang memerlukan waktu aktif maksimum. Pusat data adalah benteng yang didedikasikan untuk mencapai keandalan maksimum dengan biaya berapa pun. Meskipun keandalan masih menjadi faktor kunci, pusat data telah berkembang dan kemajuan dalam 5 tahun terakhir telah mempercepat laju inovasi.

**Arsitektur Pusat Data Klasik**

Pusat data adalah rumah bagi daya komputasi, penyimpanan, dan aplikasi yang diperlukan untuk mendukung bisnis perusahaan. Infrastruktur pusat data adalah inti dari arsitektur TI, dari mana semua konten bersumber atau melewatinya. Perencanaan yang tepat dari desain infrastruktur pusat data sangat penting, dan kinerja, ketahanan, dan skalabilitas perlu dipertimbangkan dengan cermat.

Model multitier adalah desain yang paling umum di perusahaan. Ini didasarkan pada web, aplikasi, dan desain berlapis basis data yang mendukung perdagangan dan solusi ERP bisnis perusahaan dan CRM. Jenis desain ini mendukung banyak arsitektur layanan web, seperti yang berbasis Microsoft. NET atau Java 2 Enterprise Edition. Lingkungan aplikasi layanan web ini digunakan oleh solusi ERP dan CRM dari Siebel dan Oracle, untuk beberapa nama. Model multitier bergantung pada keamanan dan layanan pengoptimalan aplikasi yang akan disediakan di jaringan.

Model cluster server telah tumbuh dari universitas dan komunitas ilmiah untuk muncul di vertikal bisnis perusahaan termasuk keuangan, manufaktur, dan hiburan. Model cluster server paling sering dikaitkan dengan komputasi kinerja tinggi (HPC), komputasi paralel, dan lingkungan komputasi throughput tinggi (HTC) tetapi juga dapat dikaitkan dengan komputasi grid/utilitas. Desain ini biasanya didasarkan pada arsitektur aplikasi yang disesuaikan, dan terkadang eksklusif, yang dibangun untuk melayani tujuan bisnis tertentu.

Model multitier: Model pusat data multitier didominasi oleh aplikasi berbasis HTTP dalam pendekatan multitier. Pendekatan multitier termasuk web, aplikasi, dan database tingkatan server. Saat ini, sebagian besar aplikasi berbasis web dibangun sebagai aplikasi multitier. Model multitier menggunakan perangkat lunak yang berjalan sebagai proses

terpisah pada mesin yang sama menggunakan komunikasi antarproses (IPC), atau pada mesin yang berbeda dengan komunikasi melalui jaringan. Biasanya, tiga tingkatan berikut digunakan:

1. Server web
2. Aplikasi
3. Basis data

Kumpulan server multitingkat yang dibangun dengan proses yang berjalan pada mesin terpisah dapat memberikan ketahanan dan keamanan yang lebih baik. Ketahanan ditingkatkan karena server dapat dikeluarkan dari layanan sementara fungsi yang sama masih disediakan oleh server lain yang termasuk dalam tingkat aplikasi yang sama. Keamanan ditingkatkan karena penyerang dapat membahayakan server web tanpa mendapatkan akses ke aplikasi atau server basis data. Server web dan aplikasi dapat hidup berdampingan di server fisik yang sama; database biasanya tetap terpisah.

**CEDC**

Tekanan teknologi dan ekonomi baru telah memaksa organisasi untuk mencari cara untuk mendapatkan lebih banyak dari infrastruktur TI mereka. Keadaan infrastruktur TI saat ini tegang, dan tuntutan baru membuat semakin sulit bagi bisnis untuk mempertahankan efisiensi dan efektivitas. Ini memiliki pengaruh pada kualitas layanan terlepas dari jumlah pengguna dan aplikasi. Solusinya terletak pada komputasi awan. *Cloud* memiliki kemampuan untuk mengurangi biaya dan meningkatkan fleksibilitas aplikasi dan layanan termasuk infrastruktur TI. CEDC membawa pusat data tervirtualisasi (dan bisnis) menjadi lebih gesit karena membutuhkan virtualisasi ke tingkat berikutnya. Lingkungan tervirtualisasi diubah menjadi lingkungan yang dioptimalkan dengan IaaS terintegrasi yang cerdas, dan PaaS yang mengelola beban kerja dinamis— memberikan beban kerja sumber daya yang dibutuhkan berdasarkan kebijakan bisnis. Ini juga menyediakan otomatisasi dan orkestrasi sumber daya di seluruh pusat data yang heterogen. Perkembangan dari manajemen virtualisasi ke CEDC ini penting karena menangani tujuan bisnis umum yang kami dengar berulang kali.

**Organisasi Fisik**

Pusat data umumnya terbentang di seluruh gedung, beberapa lantai gedung atau bahkan satu ruangan di dalam gedung. Gambar 10.1, dikumpulkan dari gambar Google, menunjukkan salah satu dari beberapa kemungkinan cara mengatur rak server di pusat data. Pusat data biasanya terdiri dari sejumlah besar server yang dipasang di lemari rak dan ditempatkan dalam baris tunggal membentuk koridor (disebut lorong) di antara mereka, sehingga memungkinkan akses ke depan dan belakang setiap lemari.

**Gambar 10.1** Organisasi fisik pusat data.

Selain itu, beberapa peralatan seperti perangkat penyimpanan seringkali sebesar rak. Peralatan seperti itu umumnya ditempatkan di samping rak. Pusat data besar menampung beberapa ribu server di mana terkadang kontainer pengiriman yang dikemas dengan 1000 atau lebih server masing-masing digunakan. Jika terjadi kegagalan atau saat peningkatan diperlukan, seluruh wadah diganti daripada mengganti server individual.

**Infrastruktur Penyimpanan dan Jaringan**

Biasanya, pusat data memerlukan empat jenis akses jaringan yang berbeda dan, karenanya, dapat menggunakan empat jenis jaringan fisik yang berbeda seperti yang ditunjukkan di bawah ini:

1. Jaringan klien-server: Untuk menyediakan konektivitas eksternal ke pusat data. Ethernet kabel tradisional atau teknologi LAN Nirkabel dapat digunakan.
2. Jaringan server-server: Untuk menyediakan komunikasi berkecepatan tinggi di antara server pusat data. Ethernet, InfiniBand (IBA), atau teknologi lainnya dapat digunakan. Gambar 10.2 menunjukkan contoh jaringan server-server.
3. Jaringan server-penyimpanan: Untuk menyediakan konektivitas berkecepatan tinggi antara server dan perangkat penyimpanan. Biasanya Fibre Channel digunakan, tetapi teknologi seperti Ethernet atau InfiniBand juga dapat digunakan.
4. Jaringan lain seperti jaringan yang dibutuhkan untuk mengelola pusat data. Umumnya, Ethernet digunakan tetapi pemasangan kabelnya mungkin berbeda dari jaringan utama.

Beberapa aplikasi berjalan di dalam satu pusat data, biasanya dengan setiap aplikasi dihosting di kumpulan mesin servernya sendiri (berpotensi virtual).

**Gambar 10.2** Infrastruktur jaringan di pusat data.

Setiap aplikasi dikaitkan dengan satu atau lebih alamat IP yang dapat dilihat secara publik dan dapat dirutekan ke mana klien di Internet mengirim permintaan mereka dan dari mana mereka menerima balasan. Di dalam pusat data, permintaan tersebar di antara kumpulan server frontend yang memproses permintaan. Penyebaran ini biasanya dilakukan oleh penyeimbang beban khusus.

Topologi jaringan dua tingkat sangat populer di DCN saat ini. Sakelar akses untuk konektivitas server runtuh dalam sakelar agregasi kepadatan tinggi yang menyediakan

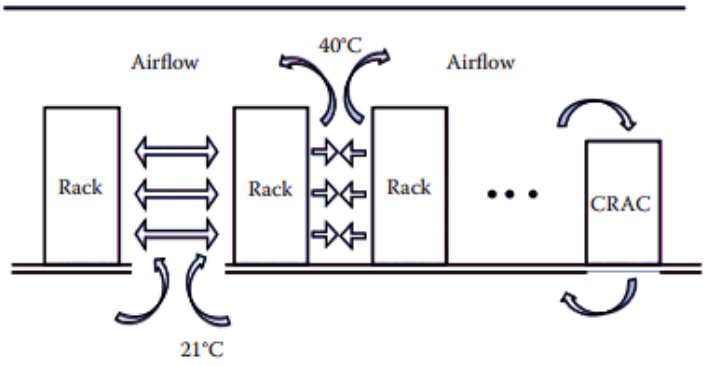
fungsionalitas peralihan dan perutean untuk interkoneksi peralihan akses dan berbagai server LAN. Ini memiliki beberapa manfaat:

- Kesederhanaan desain (saklar yang lebih sedikit dan node yang dikelola lebih sedikit)
- Mengurangi latensi jaringan (dengan mengurangi jumlah lompatan sakelar)
- Biasanya rasio oversubscription desain jaringan berkurang
- Konsumsi daya agregat lebih rendah

Namun, kerugian dari desain dua tingkat termasuk skalabilitas terbatas: ketika port pada pasangan sakelar agregasi digunakan sepenuhnya, maka penambahan pasangan sakelar/router agregasi lainnya menambah tingkat kerumitan yang tinggi. Sambungan antara pasangan sakelar agregasi harus sepenuhnya terhubung dengan bandwidth tinggi, sehingga tidak ada kemacetan yang terjadi dalam desain jaringan. Karena pasangan switch agregasi juga menjalankan protokol routing, lebih banyak pasangan switch berarti lebih banyak peering protokol routing dan lebih banyak antarmuka routing dan kompleksitas yang diperkenalkan oleh desain mesh penuh.

**Infrastruktur Pendinginan**

Karena pusat data biasanya tersebar di seluruh gedung dan menampung beberapa ribu server, infrastruktur pendingin yang canggih digunakan, yang mungkin melibatkan unit AC, kipas, dan sistem resirkulasi udara di tingkat gedung. Gambar 10.3 menunjukkan salah satu kemungkinan pengaturan gang dimana infrastruktur pendinginan disederhanakan.




**Gambar 10.3** Pendinginan di pusat data.

Rak server ditempatkan pada pleno yang ditinggikan dan diatur dalam lorong yang menghadap ke belakang dan menghadap ke depan secara bergantian. Udara dingin dipaksa masuk ke lorong yang menghadap ke depan, dan kipas server atau sasis menarik udara dingin melalui server ke belakang. Udara panas di bagian belakang kemudian naik dan diarahkan (terkadang dengan menggunakan beberapa deflektor) menuju pabrik chiller untuk pendinginan dan resirkulasi. Penulis dalam menyebutkan bahwa meskipun pengaturan seperti itu tidak mahal, namun dapat menimbulkan titik panas baik karena pendinginan yang tidak merata atau pencampuran udara panas dan dingin.

Dalam beberapa tahun terakhir, ada banyak inovasi dalam teknologi daya dan pendinginan serta pengelolaan fasilitas. Efisiensi telah diintegrasikan ke dalam setiap aspek pusat data dan desain bangunan, mencakup segala hal mulai dari bunker hingga desain

kandang ayam dan pusat data seluler hingga penggunaan gedung sebagai pengendali udara. Teknologi hijau dan kesadaran lingkungan juga telah menjadi bagian besar dari industri ini dalam 3 tahun terakhir. Tidak lagi hanya pilihan antara membangun dan menyewa, pusat data dapat dimiliki, ditempatkan di colocation, grosir, diletakkan di *cloud* publik atau pribadi, atau pilihan strategi hybrid.

Perubahan yang terjadi bahkan telah mengubah arti dari apa yang dimaksud dengan pusat data. Bagi Google, Microsoft, atau Yahoo, ini adalah fasilitas hyperscale dengan inovasi luar biasa yang direkayasa di dalamnya. Untuk proyek konsolidasi, itu berarti mengambil apa yang dulu dianggap sebagai pusat data dan membawanya ke sejumlah kecil fasilitas baru berskala besar. Bagi yang lain, definisi pusat data mereka diubah oleh kemajuan peralatan TI yang membutuhkan lebih banyak daya, lebih banyak pendinginan, dan fasilitas yang lebih canggih untuk mendukungnya.

Banyak segi pemilihan lokasi untuk pusat data dibentuk oleh proliferasi jaringan serat dan kebutuhan untuk menghindari bencana alam dan mencapai efisiensi daya dan pendinginan yang ekstrim. Di Amerika Serikat, beberapa wilayah menjadi hub pusat data, di mana perusahaan dan perusahaan Internet dipilih untuk membangun pusat data baru. Lembah Silikon terus makmur dan tumbuh, dan pusat regional berkembang di Quincy, Washington, Chicago, Dallas/Fort Worth, Carolina Utara, dan wilayah New York/New Jersey.

**Sifat Lalu Lintas di Pusat Data**

Lingkungan pusat data sangat berbeda dari Internet, misalnya, waktu perjalanan pulang pergi (RTT) di DCN bisa kurang dari 250 µs jika tidak ada antrian. Alasannya adalah DCN adalah jaringan milik pribadi yang dirancang untuk mencapai bandwidth tinggi dan latensi rendah. Selain itu, sifat lalu lintas di DCN sangat bervariasi dari lalu lintas Internet. Lalu lintas di DCN diklasifikasikan terutama menjadi tiga jenis:

1. Lalu lintas tikus: Kueri membentuk lalu lintas tikus (misalnya, pencarian Google dan pembaruan Facebook). Sebagian besar lalu lintas di DCN adalah lalu lintas kueri, dan volume transmisi datanya biasanya lebih sedikit.
2. Lalu lintas kucing: Status kontrol dan pesan koordinasi membentuk lalu lintas kucing (misalnya, unduhan file berukuran kecil dan menengah)
3. Lalu lintas gajah: Pembaruan besar membentuk lalu lintas gajah (mis., Pembaruan anti-virus dan unduhan film).

**Tabel 10.3** Lalu Lintas Pusat Data: Aplikasi dan Persyaratan Kinerja

| Jenis Lalu Lintas | Contoh | Persyaratan |
|---|---|---|
| Lalu lintas tikus (<100 kB) | Pencarian Google, Facebook | Waktu respons singkat |
| Lalu lintas kucing (100 kB hingga 5 MB) | Picasa, YouTube, foto Facebook | Latensi rendah |
| Lalu lintas gajah (>5 MB) | Pembaruan perangkat lunak, video sesuai permintaan | Hasil tinggi |

Berbagai jenis lalu lintas di DCN, aplikasinya, dan persyaratan kinerja dirangkum dalam Tabel 10.3. Dengan demikian, lalu lintas kueri bursty, lalu lintas kucing yang sensitif terhadap

penundaan, dan lalu lintas gajah yang sensitif terhadap throughput hidup berdampingan di DCN.

## 10.3 MASALAH JARINGAN DI PUSAT DATA

Jaringan *cloud* adalah tulang punggung fundamental untuk menyediakan layanan *cloud* dan alasan di balik perubahan dalam cara layanan TI diberikan kepada pengguna. Bagian ini berfokus pada beberapa masalah yang dihadapi dalam jaringan *cloud*.

**Ketersediaan**

Salah satu tantangan menakutkan yang dihadapi organisasi penyedia *cloud* adalah menyediakan waktu aktif maksimum untuk layanan yang ditawarkan kepada pengguna. Bahkan downtime beberapa detik dapat menyebabkan hilangnya reputasi organisasi dan mempengaruhi bisnis secara keseluruhan. Selain itu, downtime dapat menyebabkan pelanggaran perjanjian tingkat layanan (SLA) antara pengguna *cloud* dan penyedia *cloud*, sehingga sangat memengaruhi pendapatan penyedia *cloud*. Pendekatan yang paling banyak diadopsi untuk mencapai ketersediaan tinggi adalah mereplikasi data dan mengambil cadangan secara teratur.

**Performa Jaringan Buruk**

Tiga persyaratan kinerja dasar dari DCN adalah toleransi burst tinggi, latensi rendah, dan throughput tinggi. Ini karena lalu lintas di pusat data terdiri dari campuran ketiga jenis lalu lintas: lalu lintas tikus, lalu lintas kucing, dan lalu lintas gajah, masing-masing memiliki persyaratan kinerja yang berbeda dari yang lain. Tantangan utama adalah tumpukan Protokol Kontrol Transmisi/Protokol Internet (TCP/IP) tradisional, yang terutama dirancang untuk skenario mirip Internet, gagal memberikan kinerja optimal di DCN. Bagian selanjutnya memberikan deskripsi yang lebih mendetail tentang beberapa masalah TCP/IP di DCN.

**Keamanan**

Menjaga keamanan data pengguna *cloud* selama transit, atau saat tidak digunakan, masih menjadi perhatian penyedia layanan *cloud*. Penghapusan data yang tidak disengaja karena pemadaman listrik atau tidak berfungsinya program pencadangan reguler dapat menyebabkan hilangnya data pelanggan dan menimbulkan kerugian besar bagi organisasi hosting.

Terlepas dari kekhawatiran yang disebutkan di atas, penyedia *cloud* juga perlu memastikan keamanan fisik gedung pusat data dan infrastruktur jaringannya untuk mencegah serangan dari orang dalam yang jahat.

## 10.4 MASALAH TRANSPORT LAYER DI DCN

TCP adalah salah satu protokol transport yang paling dominan, banyak digunakan oleh berbagai macam aplikasi Internet, dan juga merupakan mayoritas lalu lintas di kedua jenis DCN [5]. Itu telah menjadi pekerja keras Internet sejak awal. Keberhasilan Internet, pada kenyataannya, sebagian dapat dikaitkan dengan mekanisme kontrol kemacetan yang diterapkan di TCP. Meskipun skala Internet dan penggunaannya meningkat secara eksponensial di masa lalu, TCP telah berevolusi untuk mengikuti perubahan kondisi jaringan dan terbukti dapat diskalakan dan kuat.

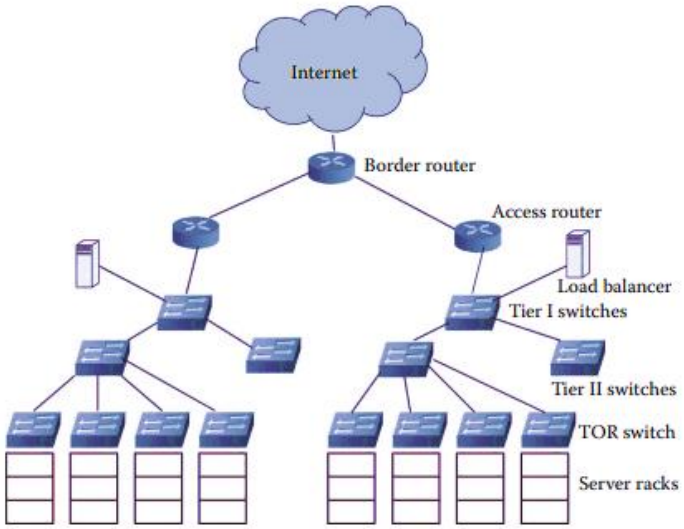
Namun, telah diamati bahwa TCP yang canggih gagal memenuhi tiga persyaratan dasar (disebutkan dalam subbagian sebelumnya) bersama-sama dalam batas waktu karena gangguan seperti TCP incast, TCP outcast, penumpukan antrian, tekanan buffer, dan efek pseudocongestion, yang dibahas lebih lanjut di bagian berikut.

**Kerusakan TCP di DCN**

Meskipun TCP terus berkembang selama tiga dekade, keragaman karakteristik jaringan sekarang dan generasi berikutnya dan berbagai persyaratan aplikasi telah menimbulkan beberapa tantangan untuk mekanisme kontrol kongesti TCP. Akibatnya, kekurangan dalam desain dasar TCP menjadi semakin jelas. Pada bagian ini, kami terutama berfokus pada tantangan yang dihadapi oleh TCP canggih di DCN.

**TCP Incast**

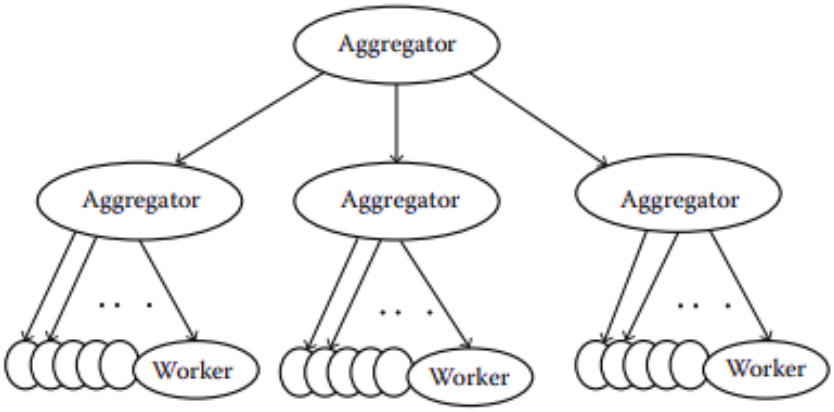
TCP incast telah didefinisikan sebagai perilaku patologis dari TCP yang menghasilkan underutilisasi besar dari kapasitas link dalam berbagai pola komunikasi *many-to-one,* misalnya, pola aplikasi partisi/agregat seperti yang ditunjukkan pada Gambar 10.4. Pola seperti itu adalah dasar dari banyak aplikasi berskala besar seperti pencarian web, MapReduce, komposisi konten jejaring sosial, dan pemilihan iklan. Akibatnya, masalah incast TCP secara luas ada di skenario pusat data saat ini seperti sistem penyimpanan terdistribusi, sistem komputasi skalabel intensif data, dan alur kerja partisi/agregat.




**Gambar 10.4** Struktur aplikasi partisi/agregat.

Dalam pola komunikasi many-to-one, agregator mengeluarkan permintaan data ke beberapa node pekerja. Node pekerja, setelah menerima permintaan, secara bersamaan mengirimkan sejumlah besar data ke agregator (lihat Gambar 10.5). Data dari semua node pekerja melewati link bottleneck dengan cara many-to-one. Probabilitas bahwa semua node pekerja mengirim balasan pada waktu yang sama tinggi karena batas waktu yang ketat. Oleh karena itu, paket-paket dari node-node ini meluap ke buffer switch *top-of-the-rack* (ToR) dan, dengan demikian, menyebabkan hilangnya paket. Fenomena ini dikenal sebagai tikus tersinkronisasi bertabrakan.

Selain itu, tidak ada node pekerja yang dapat mentransmisikan blok data berikutnya hingga semua node pekerja selesai mentransmisikan blok data saat ini. Transmisi seperti itu disebut sebagai transmisi tersinkronisasi penghalang.




**Gambar 10.5** incast TCP.

Di bawah kendala seperti itu, karena jumlah node pekerja bersamaan meningkat, throughput tingkat aplikasi yang dirasakan pada agregator menurun karena sejumlah besar paket yang hilang. Paket yang hilang ditransmisikan ulang hanya setelah batas waktu pengiriman ulang (RTO), yang umumnya dalam urutan beberapa milidetik. Seperti disebutkan sebelumnya, lalu lintas mouse membutuhkan waktu respons yang singkat dan sangat sensitif terhadap penundaan. Timeout yang sering dihasilkan dari incast secara signifikan menurunkan kinerja lalu lintas mouse karena paket yang hilang ditransmisikan ulang setelah beberapa milidetik.

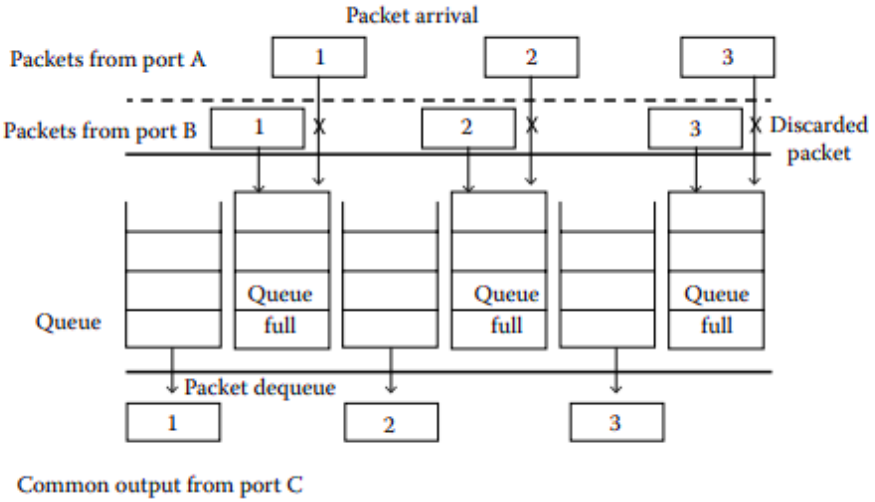
Perlu dicatat bahwa mekanisme pengiriman ulang yang cepat mungkin tidak dapat diterapkan pada aplikasi lalu lintas mouse karena volume transmisi data dari lalu lintas semacam itu sangat kecil, dan karenanya, hanya ada sedikit paket di seluruh aliran. Akibatnya, pengirim (atau node pekerja) mungkin tidak mendapatkan cukup duplikat pengakuan (dupack) untuk memicu pengiriman ulang yang cepat.

Mitigasi incast TCP: Banyak solusi, mulai dari solusi lapisan aplikasi hingga solusi lapisan transport dan solusi lapisan tautan, telah diusulkan baru-baru ini untuk mengatasi masalah incast TCP. Beberapa solusi menyarankan revisi TCP, yang lain merekomendasikan untuk mengganti TCP, sementara beberapa mencari solusi dari lapisan selain lapisan transport untuk mengatasi masalah ini.

**TCP Terbuang**

Ketika satu set besar aliran dan satu set kecil aliran tiba di dua port input yang berbeda dari sebuah switch dan bersaing untuk port output bottleneck yang sama, set kecil aliran kehilangan bagian throughput mereka secara signifikan. Fenomena ini disebut sebagai TCP outcast dan terutama terjadi pada switch pusat data yang menggunakan antrian drop-tail. Antrean drop-tail menyebabkan penurunan paket berturut-turut dari satu port dan, karenanya, sering menyebabkan waktu tunggu TCP. Sifat antrean drop-tail ini disebut sebagai port blackout, dan secara signifikan memengaruhi kinerja aliran kecil karena waktu tunggu yang sering menyebabkan

latensi tinggi dan, dengan demikian, waktu respons berkualitas rendah. Gambar 10.6 menunjukkan contoh skenario pemadaman port, di mana A dan B adalah port input dan C adalah port output yang umum. Gambar tersebut menunjukkan bahwa paket yang tiba di port B berhasil diantrikan sedangkan yang tiba di port A dijatuhkan secara berurutan.




**Gambar 10.6** Contoh skenario pemadaman port.

Telah diketahui bahwa throughput aliran TCP berbanding terbalik dengan RTT aliran tersebut. Perilaku TCP ini menyebabkan bias RTT, yaitu aliran dengan RTT rendah mencapai bagian bandwidth yang lebih besar daripada aliran dengan RTT tinggi. Namun, telah diamati bahwa karena masalah outcast TCP di DCN, TCP menunjukkan bias RTT terbalik, yaitu, aliran dengan RTT rendah dikeluarkan oleh aliran dengan RTT tinggi.

Dua faktor utama yang menyebabkan TCP terbuang adalah (1) penggunaan antrean drop-tail di switch dan (2) pola komunikasi many-to-one, yang mengarah ke sekumpulan besar aliran dan sekumpulan kecil aliran yang tiba di dua port input yang berbeda dan bersaing untuk port output bottleneck yang sama. Kedua faktor ini cukup umum di DCN karena sebagian besar sakelar menggunakan antrean drop-tail dan pola komunikasi banyak-ke-satu adalah dasar dari banyak aplikasi *cloud*.

Mitigasi TCP outcast: Pendekatan langsung untuk mengurangi TCP outcast adalah dengan menggunakan mekanisme antrian selain drop tail, misalnya, deteksi dini acak (RED) dan stochastic fair queue (SFQ). Pendekatan lain yang mungkin adalah meminimalkan hunian buffer di switch dengan merancang undang-undang kontrol kongesti TCP yang efisien di host akhir.

**Penumpukan Antrian**

Karena beragamnya aplikasi *cloud*, lalu lintas mouse, lalu lintas kucing, dan lalu lintas gajah hidup berdampingan dalam DCN. Sifat lalu lintas gajah yang tahan lama dan serakah mendorong jaringan ke titik kemacetan ekstrim dan meluap dari buffer leher botol. Jadi, ketika lalu lintas tikus dan lalu lintas gajah melintasi rute yang sama, kinerja lalu lintas tikus sangat terpengaruh karena adanya lalu lintas gajah.

Berikut adalah dua cara di mana kinerja lalu lintas tikus terdegradasi karena adanya lalu lintas gajah: (1) Karena sebagian besar buffer ditempati oleh lalu lintas gajah, ada kemungkinan besar paket tikus lalu lintas turun. Dampak dari situasi ini mirip dengan incast TCP karena kinerja lalu lintas mouse sangat dipengaruhi oleh seringnya kehilangan paket dan, karenanya, timeout. (2) Paket lalu lintas tikus, meskipun tidak ada yang hilang, mengalami penundaan antrian yang meningkat karena mereka berada dalam antrian di belakang paket lalu lintas gajah. Masalah ini disebut sebagai penumpukan antrian.

Mengurangi penumpukan antrian: Masalah penumpukan antrian dapat diselesaikan hanya dengan meminimalkan penempatan antrian di sakelar DCN. Sebagian besar varian TCP yang ada menggunakan pendekatan reaktif terhadap kontrol kemacetan, yaitu, mereka tidak mengurangi kecepatan pengiriman kecuali jika terjadi kehilangan paket dan, karenanya, gagal meminimalkan hunian antrian. Sebaliknya, pendekatan proaktif diinginkan untuk meminimalkan okupansi antrean dan mengatasi masalah penumpukan antrean.

**Tekanan Penyangga**

Tekanan penyangga adalah penurunan lain yang disebabkan oleh lalu lintas gajah yang tahan lama dan rakus. Ketika lalu lintas tikus dan lalu lintas gajah hidup berdampingan di rute yang sama, sebagian besar ruang penyangga ditempati oleh paket dari lalu lintas gajah. Ini menyisakan ruang yang sangat kecil untuk mengakomodasi semburan paket lalu lintas tikus yang muncul dari pola komunikasi banyak-ke-satu. Hasilnya adalah sejumlah besar paket dari lalu lintas mouse hilang, menyebabkan kinerja yang buruk. Selain itu, sebagian besar lalu lintas di DCN adalah bursty, dan karenanya, paket lalu lintas mouse sering dijatuhkan karena lalu lintas gajah berlangsung lebih lama dan membuat sebagian besar ruang buffer terisi.

Mengurangi tekanan buffer: Seperti penumpukan antrian, masalah tekanan buffer juga dapat diselesaikan dengan meminimalkan hunian buffer di sakelar.

**Efek Pseudokongesti**

Virtualisasi adalah salah satu teknologi kunci yang mendorong keberhasilan aplikasi *cloud computing*. Pusat data modern mengadopsi mesin virtual (VM) untuk menawarkan layanan *cloud* sesuai permintaan dan akses jarak jauh. Pusat data ini dikenal sebagai pusat data tervirtualisasi. Meskipun ada beberapa keuntungan dari virtualisasi seperti pemanfaatan server yang efisien, isolasi layanan, dan biaya pemeliharaan sistem yang rendah, ini secara signifikan memengaruhi lingkungan tempat protokol tradisional kami (misalnya, TCP dan UDP [protokol datagram pengguna]) bekerja. Studi terbaru dari pusat data Amazon EC2 mengungkapkan bahwa virtualisasi secara dramatis menurunkan kinerja TCP dan UDP dalam hal throughput dan penundaan end-to-end. Throughput menjadi tidak stabil, dan delay end-to-end menjadi cukup besar meskipun beban jaringan berkurang.

Ketika lebih banyak VM berjalan pada mesin fisik yang sama, latensi penjadwalan hypervisor meningkatkan waktu tunggu untuk setiap VM untuk mendapatkan akses ke prosesor. Latensi penjadwalan hypervisor bervariasi dari

mikrodetik hingga beberapa ratus milidetik, menyebabkan penundaan jaringan yang tidak dapat diprediksi (yaitu, RTT) dan, dengan demikian, memengaruhi stabilitas throughput dan sebagian besar meningkatkan penundaan end-to-end. Selain itu, latensi penjadwalan hypervisor bisa sangat tinggi sehingga dapat menyebabkan RTO di pengirim VM. Setelah RTO terjadi, pengirim VM menganggap bahwa jaringan sangat padat dan menurunkan kecepatan pengiriman secara signifikan. Kami menyebut efek ini sebagai efek pseudocongestion karena kemacetan yang dirasakan oleh pengirim VM sebenarnya adalah pseudocongestion.

Mengurangi efek pseudocongestion: Umumnya ada dua kemungkinan pendekatan untuk mengatasi masalah tersebut. Salah satunya adalah merancang penjadwal yang efisien untuk hypervisor sehingga latensi penjadwalan dapat diminimalkan. Pendekatan lain adalah memodifikasi TCP sedemikian rupa sehingga secara cerdas dapat mendeteksi pseudocongestion dan bereaksi sesuai dengan itu.

**Rangkuman: Gangguan dan Penyebab TCP**

Kami meringkas secara singkat kerusakan TCP yang dibahas pada sub-bagian sebelumnya dan menyebutkan penyebabnya pada Tabel 10.4.

**Tabel 10.4** Kerusakan TCP pada DCN dan Penyebabnya

| **Kerusakan TCP** | **Penyebab** |
|---|---|
| incast TCP | Buffer dangkal di sakelar dan semburan lalu lintas mouse yang dihasilkan dari pola komunikasi banyak-ke-satu |
| TCP terbuang | Penggunaan mekanisme tail-drop di sakelar |
| Penumpukan antrian | Antrean terus-menerus penuh di sakelar karena lalu lintas gajah |
| Tekanan penyangga | Antrean terus-menerus penuh di sakelar karena lalu lintas gajah dan sifat lalu lintas tikus yang meledak |
| Efek pseudokongesti | Latensi penjadwalan hypervisor |

## 10.5 PENINGKATAN TCP UNTUK DCN

Baru-baru ini, beberapa varian TCP, khususnya transportasi data, telah diusulkan dalam DCN. Tujuan utama varian TCP ini adalah untuk mengatasi gangguan yang disebutkan di atas dan meningkatkan kinerja TCP di DCN.

Bab ini menyajikan latar belakang dan penyebab dari masing-masing gangguan tersebut di atas, dilanjutkan dengan studi perbandingan varian TCP yang bertujuan untuk mengatasi gangguan tersebut. Meskipun beberapa protokol transport lain juga telah diusulkan untuk DCN, kami membatasi ruang lingkup bab ini untuk varian TCP karena TCP adalah protokol transport yang paling banyak digunakan dalam sistem operasi modern.

**TCP dengan RTO Berbutir Halus (FG-RTO)**

Nilai default RTO minimum di TCP umumnya dalam urutan milidetik (sekitar 200 ms). Nilai RTO ini cocok untuk skenario seperti Internet di mana RTT rata-rata berada di urutan ratusan milidetik. Namun, ini jauh lebih besar daripada RTT rata-rata di pusat data, yang berada di urutan beberapa mikrodetik. Sejumlah besar kehilangan paket karena incast TCP, outcast TCP, penumpukan antrian, tekanan buffer, dan efek pseudocongestion

mengakibatkan timeout yang sering dan, pada gilirannya, menyebabkan tenggat waktu yang terlewatkan dan penurunan kinerja TCP yang signifikan. Penelitian sebelumnya menunjukkan bahwa mengurangi RTO minimum dari 200 ms menjadi 200 µs secara signifikan mengurangi masalah TCP di DCN dan meningkatkan throughput keseluruhan dengan beberapa kali lipat.

Keuntungan: Keuntungan utama dari pendekatan ini adalah bahwa hal itu memerlukan modifikasi minimal untuk kerja tradisional TCP dan dengan demikian dapat dengan mudah digunakan.

Kekurangan: Penyebaran timer fine-grained secara real-time merupakan masalah yang menantang karena sistem operasi saat ini tidak memiliki timer beresolusi tinggi yang diperlukan untuk nilai RTO yang rendah. Selain itu, FG-RTO mungkin tidak cocok untuk server yang berkomunikasi dengan klien melalui Internet. Terlepas dari masalah implementasi pengatur waktu berbutir halus, harus dicatat bahwa pendekatan menghilangkan kelemahan TCP di DCN ini adalah pendekatan reaktif. Itu mencoba untuk mengurangi dampak dari kehilangan paket daripada menghindari kehilangan paket di tempat pertama. Jadi, meskipun pendekatan ini secara signifikan meningkatkan kinerja jaringan dengan mengurangi penundaan post-packet loss, itu tidak mengurangi masalah incast TCP untuk aplikasi yang sensitif terhadap kerugian.

**TCP dengan FG-RTO + Delayed ACK Dinonaktifkan**

Delayed ACK terutama digunakan untuk mengurangi overhead ACK pada jalur sebaliknya. Ketika ACK tertunda diaktifkan, penerima hanya mengirimkan satu ACK untuk setiap dua paket data yang diterima. Jika hanya satu paket yang diterima, penerima menunggu periode timeout ACK tertunda sebelum mengirim ACK. Periode waktu tunggu ini biasanya 40 ms. Skenario ini dapat menyebabkan pengiriman ulang palsu jika pengatur waktu FG-RTO (seperti yang dijelaskan di bagian sebelumnya) digunakan. Alasannya adalah penerima menunggu selama 40 ms sebelum mengirim ACK untuk paket yang diterima dan pada saat itu, FG-RTO, yang berada dalam urutan beberapa mikrodetik, kedaluwarsa dan memaksa pengirim untuk mengirim ulang paket.

Dengan demikian, periode batas waktu ACK yang tertunda harus dikurangi menjadi beberapa mikrodetik atau harus dinonaktifkan sepenuhnya saat menggunakan FG-RTO untuk menghindari transmisi ulang palsu tersebut. Pendekatan ini semakin meningkatkan throughput TCP di DCN.

Keuntungan: Telah ditunjukkan bahwa mengurangi periode timeout ACK yang tertunda menjadi 200 µs saat menggunakan FG-RTO mencapai throughput yang jauh lebih baik daripada throughput yang diperoleh saat ACK yang tertunda diaktifkan. Selain itu, menonaktifkan sepenuhnya ACK yang tertunda saat menggunakan FG-RTO semakin meningkatkan throughput TCP secara keseluruhan.

Kekurangan: Kekurangan dari pendekatan ini sama persis dengan TCP dengan FG-RTO karena pendekatan ini merupakan efek samping yang tidak diinginkan dari pendekatan sebelumnya.

**DCTCP**

Peningkatan aditif/penurunan perkalian (AIMD) adalah landasan dari algoritma kontrol kongesti TCP. Ketika pengakuan (ACK) diterima dalam fase AIMD, jendela kemacetan

(cwnd) meningkat seperti yang ditunjukkan pada persamaan berikut. Ini dikenal sebagai fase peningkatan aditif dari algoritma AIMD:

$$cwnd = cwnd + \frac{1}{cwnd}$$

Ketika kemacetan terdeteksi baik melalui dupacks atau selective acknowledgement (SACK), cwnd diperbarui seperti yang ditunjukkan pada persamaan berikut. Ini dikenal sebagai fase penurunan perkalian dari algoritma AIMD:

$$cwnd = \frac{cwnd}{2}$$

Pusat data TCP (DCTCP) menggunakan mekanisme pengurangan multiplikatif yang efisien yang mengurangi cwnd berdasarkan jumlah kemacetan di jaringan daripada mengurangi setengahnya. DCTCP memanfaatkan mekanisme pemberitahuan kemacetan eksplisit (ECN) untuk mengekstrak umpan balik multibit pada jumlah kemacetan di jaringan dari aliran bit tunggal tanda ECN. Subbagian selanjutnya menjelaskan cara kerja ECN secara singkat:

**ECN**

ECN adalah salah satu mekanisme pensinyalan kongesti paling populer dalam jaringan komunikasi. Ini digunakan secara luas di berbagai macam sistem operasi di host akhir, router Internet modern, dan digunakan oleh berbagai protokol transport. Selain itu, penggunaan ECN di Internet telah meningkat tiga kali lipat dalam beberapa tahun terakhir.

<table>
<tr><td colspan="6">Jenis bidang layanan 8-bit</td></tr>
<tr><td>versi 4-bit</td><td>panjang tajuk 4-bit</td><td>DSCP</td><td>ECT</td><td>CE</td><td>Panjang total 16-bit<br>(dalam byte)</td></tr>
<tr><td colspan="4">identifikasi 16-bit</td><td>bendera<br>3-bit</td><td>Offset fragmen 13-bit</td></tr>
<tr><td colspan="2">8-bit waktu untuk hidup (TTL)</td><td colspan="2">protokol 8-bit</td><td colspan="2">Checksum tajuk 16-bit</td></tr>
<tr><td colspan="6">Alamat IP sumber 32-bit</td></tr>
<tr><td colspan="6">Alamat IP tujuan 32-bit</td></tr>
</table>

**Gambar 10.7** Bit ECN di header IP.

<table>
<tr><td colspan="9">Alamat port sumber 16-bit</td><td>Alamat port tujuan 16-bit</td></tr>
<tr><td colspan="10">nomor urut 32-bit</td></tr>
<tr><td colspan="10">nomor pengakuan 32-bit</td></tr>
<tr><td>panjang<br>tajuk 4-bit</td><td>Disimpan</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>PST</td><td>SYN</td><td>FIN</td><td>Ukuran jendela iklan 16-bit</td></tr>
<tr><td colspan="9">Checksum TCP 16-bit</td><td>Penunjuk mendesak 16-bit</td></tr>
</table>

**Gambar 10.8** Bit ECN di header TCP.

Seperti ditunjukkan pada Gambar 10.7 dan 10.8, ECN menggunakan dua bit pada header IP, yaitu ECN-capable transport (ECT) dan kongesti yang dialami (CE), dan dua bit pada header TCP, yaitu Congestion Window Reduced (CWR) dan ECN echo (ECE), untuk memberi sinyal kemacetan ke host akhir. ECN adalah standar industri, dan mekanisme detailnya dijelaskan dalam RFC 3168. Tabel 10.5 dan 10.6 masing-masing menunjukkan titik kode ECN di header TCP dan header IP, dan Gambar 10.9 menunjukkan secara singkat langkah-langkah yang terlibat dalam kerja mekanisme ECN.

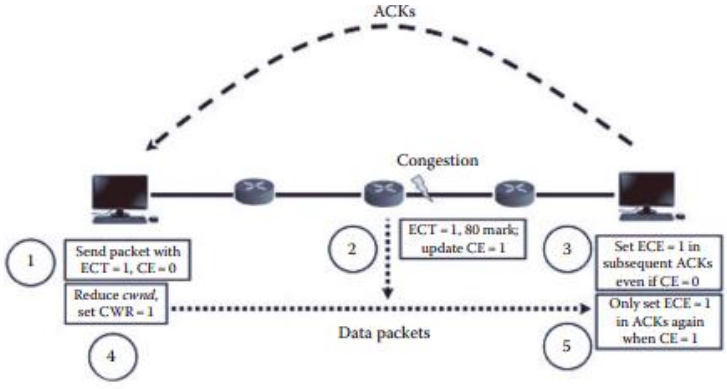
**Tabel 10.5** Codepoint ECN di Header TCP

| Titik kode | **Nilai Bit CWR** | **Nilai Bit ECE** |
|---|---|---|
| Pengaturan non-ECN | 0 | 0 |
| Gema ECN | 0 | 1 |
| CWR | 1 | 0 |
| Pengaturan ECN | 1 | 1 |

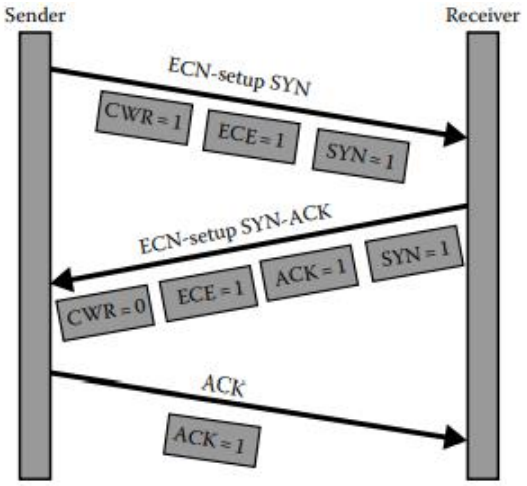
Seperti dijelaskan dalam RFC 3168, pengirim dan penerima harus menegosiasikan penggunaan ECN selama jabat tangan tiga arah (lihat Gambar 10.10). Jika keduanya mampu ECN, pengirim menandai setiap paket data keluar dengan ECT(1) codepoint atau ECT(0) codepoint. Ini berfungsi sebagai indikasi ke router bahwa pengirim dan penerima memiliki kemampuan ECN. Setiap kali terjadi kemacetan, router menandai paket data dengan mengganti titik kode ECT(1) atau ECT(0) dengan titik kode CE. Ketika penerima menerima paket yang ditandai dengan codepoint CE, ia menyimpulkan kongesti dan karenanya menandai serangkaian tanda terima keluar (ACK) dengan codepoint ECE sampai pengirim mengakui dengan codepoint CWR (lihat Gambar 10.9).

**Tabel 10.6** Codepoint ECN di IP Header

| Titik kode | **Nilai Bit ECT** | **Nilai Bit CE** |
|---|---|---|
| Non-ECT | 0 | 0 |
| ECT(1) | 0 | 1 |
| ECT(0) | 1 | 0 |
| CE | 1 | 1 |




**Gambar 10.9** ECN.




**Gambar 10.10** Negosiasi ECN.

Pengamatan utama di sini adalah bahwa, bahkan jika router menandai hanya satu paket data, penerima terus menandai ACK dengan ECE sampai menerima konfirmasi dari pengirim (lihat Langkah 3 dari Gambar 10.9). Ini untuk memastikan keandalan notifikasi kongesti, karena meskipun ACK bertanda pertama hilang, ACK bertanda lain akan memberi tahu pengirim tentang kongesti. Perhatikan bahwa cara kerja dasar ECN ini hanya bertujuan untuk memberi tahu pengirim tentang kemacetan. Itu tidak dirancang untuk memberikan informasi tambahan tentang jumlah kemacetan kepada pengirim.

Di penerima, menghitung jumlah paket yang ditandai oleh router memberikan informasi yang cukup akurat tentang jumlah kemacetan di jaringan. Namun, menyampaikan informasi ini kepada pengirim dengan menggunakan ECN merupakan tugas yang rumit. Salah satu cara yang mungkin adalah memungkinkan pengirim menghitung jumlah ACK bertanda yang diterimanya dari penerima. Keterbatasannya, bagaimanapun, adalah meskipun router menandai hanya satu paket data, penerima mengirimkan serangkaian ACK yang ditandai. Oleh karena itu, jumlah ACK yang ditandai yang dihitung oleh pengirim akan jauh lebih tinggi daripada jumlah paket yang sebenarnya ditandai oleh router. Hal ini akan menyebabkan kesalahan estimasi jumlah kemacetan di jaringan.

Untuk mengatasi keterbatasan ini, DCTCP memodifikasi mekanisme dasar ECN. Tidak seperti penerima TCP, yang mengirimkan serangkaian ACK yang ditandai, penerima DCTCP mengirimkan ACK yang ditandai hanya ketika menerima paket yang ditandai dari router, yaitu, menetapkan titik kode ECE di ACK keluar hanya ketika menerima paket dengan titik kode CE. Dengan demikian, pengirim DCTCP memperoleh jumlah paket yang akurat yang ditandai oleh router hanya dengan menghitung jumlah ACK yang ditandai yang diterimanya. Perhatikan bahwa modifikasi pada mekanisme ECN asli ini mengurangi keandalan karena jika ACK yang ditandai hilang, pengirim tetap tidak menyadari kemacetan dan tidak mengurangi kecepatan pengiriman. Namun, karena DCN adalah jaringan yang dikontrol secara pribadi, kemungkinan hilangnya ACK dapat diabaikan.

Saat menerima notifikasi kemacetan melalui ECN, cwnd di DCTCP berkurang seperti yang ditunjukkan berikut ini:

$$cwnd = cwnd \times \left(1 - \frac{\alpha}{2}\right)$$

di mana α(0≤ α ≤1) adalah perkiraan fraksi paket yang ditandai dan dihitung seperti yang ditunjukkan pada persamaan berikut ini. persamaan berikut ini adalah fraksi paket yang ditandai di cwnd sebelumnya dan g (0 < g < 1) adalah konstanta rata-rata bergerak tertimbang eksponensial. Jadi, ketika kemacetan rendah (α mendekati 0), cwnd berkurang sedikit dan ketika kemacetan tinggi (α mendekati 1), cwnd berkurang setengahnya, seperti TCP tradisional:

$$\alpha = (1 - \mathrm{g}) \times \alpha + \mathrm{g} \times F$$

Tujuan utama dari algoritma DCTCP adalah untuk mencapai latency rendah (diinginkan untuk lalu lintas mouse), throughput yang tinggi (diinginkan untuk lalu lintas gajah), dan toleransi ledakan tinggi (untuk menghindari kehilangan paket karena incast). DCTCP mencapai tujuan ini dengan bereaksi terhadap jumlah kemacetan daripada mengurangi separuh cwnd. DCTCP menggunakan skema penandaan pada sakelar yang menetapkan titik kode CE paket segera setelah hunian buffer melebihi ambang batas tetap yang telah ditentukan, K (17% seperti yang disebutkan dalam). Sumber DCTCP bereaksi dengan mengurangi jendela dengan faktor yang bergantung pada fraksi dari paket yang ditandai: semakin besar fraksi, semakin besar faktor penurunannya.

Keuntungan: DCTCP adalah varian TCP baru yang mengatasi masalah incast TCP, penumpukan antrean, dan tekanan buffer di DCN. Ini membutuhkan sedikit modifikasi pada desain asli TCP dan ECN untuk mencapai manfaat kinerja ini. DCTCP menggunakan perilaku proaktif, yaitu mencoba menghindari kehilangan paket. Telah ditunjukkan dalam bahwa ketika DCTCP menggunakan FG-RTO, ini semakin mengurangi dampak incast TCP dan juga meningkatkan skalabilitas DCTCP. Sifat stabilitas, konvergensi, dan kewajaran dari DCTCP menjadikannya solusi yang sesuai untuk implementasi di DCN. Apalagi, DCTCP sudah diimplementasikan di versi terbaru sistem operasi server Microsoft Windows.

Kekurangan: Kinerja DCTCP turun kembali ke TCP ketika tingkat incast meningkat melebihi 35, yaitu, jika ada lebih dari 35 node pekerja yang mengirim data ke agregator yang sama, DCTCP gagal menghindari incast dan kinerjanya turun kembali ke dari TCP tradisional. Namun, penulis menunjukkan bahwa alokasi buffer dinamis pada sakelar dan penggunaan FG-RTO dapat menskalakan kinerja DCTCP untuk menangani hingga 40 node pekerja secara paralel.

Meskipun DCTCP menggunakan mekanisme manajemen antrean sederhana di sakelar, masih diragukan apakah DCTCP dapat mengatasi masalah pengabaian TCP. Demikian pula, DCTCP tidak mengatasi masalah efek pseudocongestion di pusat data virtual. DCTCP menggunakan ruang penyangga minimum di sakelar, yang sebenarnya merupakan properti yang diinginkan untuk menghindari pengabaian TCP. Namun, studi eksperimental diperlukan untuk menyimpulkan apakah DCTCP dapat mengurangi masalah pengasingan TCP dan efek pseudocongestion.

**ICTCP**

Seperti DCTCP, ide utama kontrol kemacetan incast untuk TCP (ICTCP) adalah untuk menghindari kehilangan paket karena kemacetan daripada menghindari dari kehilangan paket. Diketahui bahwa pengirim TCP dapat mengirim minimal jendela yang diiklankan (rwnd) dan jendela kemacetan (cwnd) (yaitu, min (rwnd, cwnd)). ICTCP memanfaatkan properti ini dan secara efisien memvariasikan rwnd untuk menghindari TCP incast. Kontribusi utama dari ICTCP adalah sebagai berikut: (1) Bandwidth yang tersedia digunakan sebagai kuota untuk mengkoordinasikan peningkatan akhir dari semua koneksi, (2) kontrol kongesti per-aliran dilakukan secara independen, dan (3) rwnd disesuaikan berdasarkan rasio perbedaan antara throughput yang diharapkan dan throughput terukur di atas throughput yang diharapkan. Selain itu, live RTT digunakan untuk estimasi throughput.

Keuntungan: Tidak seperti DCTCP, ICTCP tidak memerlukan modifikasi di sisi pengirim (yaitu node pekerja) atau elemen jaringan seperti router dan switch. Sebaliknya, ICTCP membutuhkan modifikasi hanya pada sisi penerima (yaitu agregator). Pendekatan ini diadopsi untuk mempertahankan kompatibilitas ke belakang dan membuat algoritme cukup umum untuk menangani kongesti incast di jaringan bandwidth tinggi dan latensi rendah di masa mendatang.

Kekurangan: ICTCP mencapai timeout hampir nol dan throughput yang tinggi, skalabilitas ICTCP menjadi perhatian utama, yaitu kemampuan untuk menangani kemacetan incast ketika jumlah node pekerja sangat banyak sejak ICTCP menggunakan kontrol kemacetan per-aliran. Keterbatasan lain dari ICTCP adalah mengasumsikan bahwa pengirim dan penerima berada di bawah sakelar yang sama, yang mungkin tidak selalu demikian. Selain itu, tidak diketahui berapa banyak buffer space yang digunakan oleh ICTCP. Dengan demikian, sulit untuk menyimpulkan apakah ICTCP dapat mengurangi penumpukan antrian, tekanan buffer, dan masalah pengabaian TCP. Seperti DCTCP, ICTCP juga tidak mengatasi masalah efek pseudocongestion di pusat data virtual.

**IA-TCP**

Tidak seperti DCTCP dan ICTCP yang menggunakan kontrol kongesti berbasis jendela, algoritme penghindaran incast untuk TCP (IA-TCP) menggunakan algoritma kontrol kongesti berbasis kecepatan untuk mengontrol jumlah total paket yang diinjeksi dalam jaringan. Motivasi di balik pemilihan mekanisme kontrol kongesti berbasis laju adalah bahwa mekanisme kontrol kongesti berbasis jendela di DCN memiliki keterbatasan dalam hal skalabilitas, yaitu jumlah pengirim secara paralel.

Ide utama IA-TCP adalah untuk membatasi jumlah total paket data yang beredar di jaringan sehingga tidak melebihi bandwidth-delay product (BDP). IA-TCP menerapkan regulasi ACK pada penerima dan, seperti ICTCP, memanfaatkan field advertised window (rwnd) dari header TCP untuk mengatur cwnd dari setiap node pekerja. Rwnd minimum diatur ke 1 paket. Namun, jika sejumlah besar node pekerja mengirim paket sehubungan dengan minimal 1 paket, jumlah total paket yang beredar di jaringan dapat melebihi kapasitas link. Dalam skenario seperti itu, IA-TCP menambahkan penundaan, Δ, ke paket ACK untuk memastikan bahwa kecepatan data agregat tidak melebihi kapasitas tautan. Selain itu, IA-TCP juga

menggunakan delay, Δ, untuk menghindari sinkronisasi antar node pekerja saat mengirim data.

Keuntungan: Seperti ICTCP, IA-TCP juga memerlukan modifikasi hanya pada sisi penerima (yaitu agregator) dan tidak memerlukan modifikasi pada elemen pengirim atau jaringan. IA-TCP mencapai throughput yang tinggi dan secara signifikan meningkatkan waktu penyelesaian query. Selain itu, skalabilitas IA-TCP ditunjukkan dengan jelas dengan mengonfigurasi hingga 96 node pekerja yang mengirimkan data secara paralel.

Kekurangan: Serupa dengan masalah ICTCP, tidak jelas seberapa banyak buffer space yang digunakan oleh IA-TCP. Dengan demikian, studi eksperimental diperlukan untuk menyimpulkan apakah IA-TCP dapat mengurangi penumpukan antrian, tekanan buffer, dan masalah outcast TCP. Seperti DCTCP dan ICTCP, studi diperlukan di lingkungan pusat data virtual untuk menganalisis kinerja IA-TCP sehubungan dengan masalah efek pseudocongestion.

**$D^2$TCP**

Pusat data sadar batas waktu TCP (D2TCP) adalah protokol transport berbasis TCP baru yang dirancang khusus untuk menangani situasi ledakan tinggi. Tidak seperti varian TCP lainnya yang agnostik tenggat waktu, D2TCP menyadari tenggat waktu. D2TCP menggunakan pendekatan terdistribusi dan reaktif untuk alokasi bandwidth dan menggunakan algoritme penghindaran kemacetan sadar tenggat waktu baru yang menggunakan umpan balik ECN dan tenggat waktu untuk memvariasikan cwnd pengirim melalui fungsi koreksi gamma.

D2TCP tidak memelihara informasi per-aliran dan, sebagai gantinya, mewarisi sifat terdistribusi dan reaktif TCP sambil menambahkan kesadaran tenggat waktu ke dalamnya. Demikian pula, D2TCP menggunakan algoritme penghindaran kemacetan dengan menambahkan kesadaran tenggat waktu ke DCTCP. Ide utamanya, dengan demikian, adalah bahwa tenggat waktu jauh mengalir mundur secara agresif dan tenggat waktu dekat mengalir mundur hanya sedikit atau tidak sama sekali.

Keuntungan: Kebaruan D2TCP terletak pada kenyataan bahwa ia menyadari tenggat waktu dan mengurangi tenggat waktu yang terlewat hingga 75% dibandingkan dengan DCTCP. Selain itu, karena dirancang berdasarkan DCTCP, ini menghindari TCP incast dan penumpukan antrian serta memiliki toleransi burst yang tinggi.

Kekurangan: Kekurangan D2TCP persis sama dengan DCTCP: skalabilitas dan apakah kuat terhadap TCP outcast serta efek pseudocongestion. Namun, karena sudah tenggat waktu, akan menarik untuk menganalisis ketahanan D2TCP terhadap efek pseudocongestion di pusat data virtual.

**TCP-FITDC**

TCP-FITDC adalah mekanisme berbasis penundaan adaptif untuk mencegah masalah incast TCP. Seperti D2TCP, TCP-FITDC juga merupakan varian TCP berbasis DCTCP yang mendapat manfaat dari gagasan baru DCTCP. Selain memanfaatkan ECN sebagai indikator hunian buffer jaringan dan buffer overflow, TCP-FITDC juga memantau perubahan penundaan antrian untuk memperkirakan variasi bandwidth yang tersedia.

Jika tidak ada ACK bertanda yang diterima selama RTT, TCP-FITDC menunjukkan bahwa panjang antrian di switch berada di bawah ambang penandaan, dan karenanya, TCP-FITDC

meningkatkan cwnd untuk meningkatkan throughput. Jika tanda ACK diterima selama RTT, cwnd dikurangi untuk mengontrol panjang antrian. TCP- FITDC mempertahankan dua variabel terpisah yang disebut rtt1 dan rtt2 masing-masing untuk ACK yang tidak bertanda dan ACK yang bertanda. Dengan menganalisis perbedaan antara kedua tipe ACK ini, TCP-FITDC mendapatkan estimasi kondisi jaringan yang lebih akurat. Cwnd kemudian dikurangi secara wajar untuk mempertahankan panjang antrian yang rendah.

Keuntungan: TCP-FITDC mendapatkan perkiraan kondisi jaringan yang lebih baik dengan menggabungkan informasi yang diterima melalui ECN dan informasi yang diperoleh dengan memantau RTT. Dengan demikian, ini memiliki skalabilitas yang lebih baik daripada DCTCP dan menskalakan hingga 45 node pekerja secara paralel. Itu menghindari penumpukan TCP incast dan antrian dan memiliki toleransi burst yang tinggi karena dibangun di atas DCTCP.

Kekurangan: Kekurangan TCP-FITDC mirip dengan DCTCP, kecuali meningkatkan skalabilitas DCTCP. Tidak seperti D2TCP, TCP-FITDC adalah agnostik tenggat waktu, dan seperti semua varian TCP yang disebutkan di atas, ini tidak mengatasi masalah pengasingan TCP dan efek pseudocongestion.

**TDCTCP**

TDCTCP mencoba mengimprovisasi cara kerja DCTCP (jadi berbasis DCTCP) dengan membuat tiga modifikasi. Pertama, tidak seperti DCTCP, TDCTCP tidak hanya menurunkan tetapi juga meningkatkan cwnd berdasarkan jumlah kemacetan di jaringan, yaitu, alih-alih meningkatkan cwnd seperti yang ditunjukkan pada persamaan diatas, TDCTCP meningkatkan cwnd seperti yang ditunjukkan pada persamaan dibawah ini. Jadi, ketika jaringan dimuat ringan, kenaikan cwnd tinggi, dan sebaliknya:

$$cwnd = cwnd \times \left(1 + \frac{1}{1 + (\frac{\alpha}{2})}\right)$$

Kedua, TDCTCP menyetel ulang nilai α setelah setiap waktu tunggu ACK yang tertunda. Hal ini dilakukan untuk memastikan bahwa α tidak membawa informasi lama tentang kondisi jaringan, karena jika nilai lama α tinggi, ini akan membatasi kenaikan cwnd dan dengan demikian menurunkan throughput keseluruhan. Terakhir, TDCTCP menggunakan pendekatan yang efisien untuk secara dinamis menghitung batas waktu ACK yang tertunda dengan tujuan untuk mencapai keadilan yang lebih baik.

Keuntungan: TDCTCP mencapai throughput 26%–37% lebih baik dan keadilan 15%–20% lebih baik daripada DCTCP dalam berbagai skenario mulai dari topologi bottleneck tunggal hingga topologi multibottleneck dan berbagai ukuran buffer. Selain itu, ini mencapai throughput dan fairness yang lebih baik bahkan pada nilai K yang sangat rendah, yaitu ambang penandaan ECN pada sakelar. Namun, ada sedikit peningkatan dalam penundaan dan panjang antrean saat menggunakan TDCTCP dibandingkan dengan DCTCP.

Kekurangan: Meskipun TDCTCP meningkatkan throughput dan keadilan, ini tidak menjawab tantangan skalabilitas yang dihadapi oleh DCTCP. Seperti sebagian besar varian TCP

lainnya yang dibahas, TDCTCP juga agnostik tenggat waktu dan tidak mengurangi masalah pengasingan TCP dan efek pseudokongesti.

**TCP dengan Paket Penting Jaminan (GIP)**

TCP dengan GIP terutama bertujuan untuk meningkatkan kinerja jaringan dalam hal goodput dengan meminimalkan jumlah timeout. Timeout menyebabkan penurunan dramatis dalam kinerja jaringan dan mempengaruhi penundaan yang dirasakan pengguna. Penulis TCP dengan GIP berfokus pada menghindari terutama dua jenis batas waktu dalam jaringan: (1) batas waktu karena hilangnya jendela penuh paket, batas waktu hilangnya jendela penuh (FLoss-TOs), dan (2) timeouts karena kurangnya ACKs, kurangnya ACKs timeouts (LAck-TOs).

Floss-TO umumnya terjadi ketika total data yang dikirim oleh semua node pekerja melebihi bandwidth yang tersedia di jaringan, dan dengan demikian, beberapa aliran yang tidak beruntung akhirnya kehilangan semua paket jendela. Di sisi lain, LAck-TO terutama terjadi ketika transmisi adalah transmisi yang disinkronkan penghalang. Dalam transmisi seperti itu, aggregator tidak akan meminta node pekerja untuk mentransmisikan unit strip berikutnya sampai semua node pekerja selesai mengirimkan yang sekarang. Jika beberapa paket dijatuhkan di ujung unit garis, mereka tidak dapat dipulihkan sampai RTO menyala karena mungkin tidak ada cukup dupack untuk memicu pengiriman ulang cepat.

TCP dengan GIP memperkenalkan flag di antarmuka antara lapisan aplikasi dan lapisan transport. Bendera ini menunjukkan apakah aplikasi yang berjalan mengikuti pola komunikasi banyak-ke-satu atau tidak. Jika aplikasi yang berjalan mengikuti pola komunikasi seperti itu, TCP dengan GIP secara redundan mentransmisikan paket terakhir dari unit garis paling banyak tiga kali dan setiap node pekerja mengurangi cwnd awalnya di kepala unit garis. Sebaliknya, jika aplikasi yang berjalan tidak mengikuti pola komunikasi many-to-one, TCP dengan GIP bekerja seperti TCP standar.

Keuntungan: TCP dengan GIP mencapai waktu tunggu hampir nol dan goodput lebih tinggi dalam berbagai skenario termasuk dengan dan tanpa lalu lintas UDP latar belakang. Selain itu, skalabilitas TCP dengan GIP jauh lebih banyak daripada varian TCP lainnya yang dibahas sebelumnya, yaitu, skalanya hingga 150 node pekerja secara paralel.

Kekurangan: TCP dengan GIP tidak mengatasi masalah penempatan antrian akibat adanya lalu lintas gajah. Akibatnya, penumpukan antrian, tekanan buffer, dan masalah TCP outcast tetap tidak terpecahkan karena semua masalah ini muncul karena kurangnya ruang buffer di switch. Meskipun batas waktu dihilangkan oleh TCP dengan GIP, aliran mungkin melewatkan tenggat waktu yang ditentukan karena penundaan antrian. Selain itu, latensi penjadwalan hypervisor tidak dipertimbangkan, dan dengan demikian, masalah efek pseudocongestion juga tetap terbuka. Perhatikan bahwa latensi tinggi yang diperkenalkan oleh algoritme penjadwalan hypervisor juga dapat mencegah aliran memenuhi tenggat waktu yang ditentukan.

**PVTCP**

Paravirtualized TCP (PVTCP) mengusulkan solusi yang efisien untuk masalah efek pseudocongestion. Pendekatan ini tidak memerlukan perubahan apa pun yang harus dilakukan di hypervisor. Sebaliknya, kerja dasar TCP dimodifikasi untuk menerima latensi yang diperkenalkan oleh penjadwal hypervisor. Pendekatan yang efisien disarankan untuk

menangkap gambaran sebenarnya dari setiap transmisi paket yang melibatkan latensi yang diperkenalkan hypervisor dan kemudian menentukan RTO lebih akurat untuk menyaring efek pseudocongestion.

Setiap kali hypervisor memperkenalkan latensi penjadwalan, lonjakan tiba-tiba dapat diamati selama pengukuran reguler RTT. PVTCP mendeteksi lonjakan yang tiba-tiba ini dan menyaring dampak negatif dari lonjakan ini dengan pengukuran RTT dan manajemen RTO yang tepat. Saat menghitung RTT rata-rata, PVTCP mengabaikan pengukuran RTT tertentu jika lonjakan teramati dalam RTT tersebut.

Keuntungan: PVTCP memecahkan masalah efek pseudocongestion tanpa memerlukan perubahan apa pun pada hypervisor. Dengan mendeteksi lonjakan yang tidak biasa, mengukur RTT secara akurat, dan mengelola RTO dengan benar, PVTCP meningkatkan kinerja pusat data virtual.

Kekurangan: Skalabilitas PVTCP ambigu, dan dengan demikian, apakah dapat menyelesaikan TCP incast secara efektif atau tidak tidak jelas. Hunian antrian saat menggunakan PVTCP tidak dipertimbangkan, yang selanjutnya dapat menyebabkan masalah seperti penumpukan antrian, tekanan buffer, pengabaian TCP, dan tenggat waktu yang terlewat.

**Rangkuman: Penyempurnaan TCP untuk DCN**

Tabel 10.7 merangkum studi komparatif varian TCP yang diusulkan untuk DCN. Terlepas dari kebaruan dari varian TCP yang diusulkan, tabel ini juga menyoroti kompleksitas penerapan masing-masing protokol. Protokol yang memerlukan modifikasi pada pengirim, penerima, dan sakelar dianggap sulit untuk diterapkan. Yang membutuhkan modifikasi hanya pada pengirim atau penerima dianggap mudah untuk digunakan. DCN, bagaimanapun, adalah jaringan yang dikendalikan dan dikelola secara pribadi, dan dengan demikian, yang sebelumnya juga dapat dianggap mudah untuk diterapkan.

Terlepas dari parameter yang disebutkan di atas, ringkasan juga mencakup masalah mana di antara TCP incast, TCP outcast, penumpukan antrian, tekanan buffer, dan efek pseudocongestion yang dikurangi oleh masing-masing varian TCP. Rincian mengenai alat yang digunakan/pendekatan implementasi yang diadopsi oleh penulis juga dicantumkan.

**Tabel 10.7** Rangkuman Varian TCP Diusulkan untuk DCN

| Varian TCP | Modifies Diusulkan untuk DCN Sender | Modifies Receiver | Modifies Switch | Solves TCP | | Solves Queue Build Up | Solves Buffer Pressure | Is Deadline Aware? | Detects Pseudo congestion | Penerapan |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Incast | Outcast | | | | | |
| TCP dengan FG-RTO | √ | x | x | √ | x | X | x | x | x | Testbed dan ns-2 |
| TCP dengan FG-RTO + ACK tertunda dengan disabilitas | √ | x | x | √ | x | X | x | x | x | Testbed dan ns-2 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| DCTCP | √ | √ | √ | √ | x | √ | √ | x | x | Testbed dan ns-2 |
| ICTCP | x | √ | x | √ | x | x | X | x | x | Testbed |
| IA-TCP | x | √ | x | √ | x | x | x | x | x | ns-2 |
| D2TCP | √ | √ | √ | √ | x | √ | √ | √ | x | Testbed dan ns-3 |
| TCP-FITDC | √ | √ | √ | √ | x | √ | √ | x | x | Pemodelan dan ns-2 |
| TDCTCP | √ | √ | √ | √ | x | √ | √ | x | x | OMNeT++ |
| TCP dengan GIP | x | √ | x | √ | x | x | x | x | x | Testbed dan ns-2 |
| PVTCP | √ | √ | x | √ | x | x | x | x | √ | Testbed |

Meskipun beberapa modifikasi telah diusulkan untuk desain asli TCP, ada kebutuhan mendesak untuk lebih mengoptimalkan kinerja TCP di DCN. Beberapa masalah terbuka tercantum sebagai berikut:

1. Kecuali D2TCP, semua varian TCP lainnya adalah agnostik tenggat waktu. Memenuhi tenggat waktu adalah persyaratan terpenting dalam DCN. Tenggat waktu yang terlewat dapat menyebabkan pelanggaran SLA dan, dengan demikian, menimbulkan biaya tinggi bagi organisasi.
2. Sebagian besar pusat data saat ini menggunakan virtualisasi untuk penggunaan sumber daya yang efisien. Latensi penjadwalan hypervisor berkisar dari mikrodetik hingga ratusan milidetik dan, karenanya, dapat menghambat penyelesaian aliran yang berhasil dalam tenggat waktu yang ditentukan. Sementara membuat modifikasi pada hypervisor adalah salah satu solusi yang layak, merancang TCP yang efisien yang menyadari tenggat waktu dan secara otomatis mentolerir latensi penjadwalan hypervisor adalah solusi yang lebih disukai.
3. Solusi meyakinkan untuk masalah buangan TCP tidak tersedia. Solusi optimal untuk mengatasi TCP outcast harus memastikan hunian buffer minimal di switch. Karena RED diterapkan di sebagian besar sakelar modern, ini dapat digunakan untuk mengontrol hunian buffer. Sensitivitas parameter RED, bagaimanapun, menimbulkan tantangan lebih lanjut dan memperumit masalah.

## 10.6 RINGKASAN

Pusat data dalam skenario ini memiliki banyak sekali aplikasi Internet. Aplikasi ini beragam sifatnya dan memiliki berbagai persyaratan kinerja. Memahami arsitektur lengkap pusat data dari sudut infrastruktur fisik dan jaringannya sangat penting untuk keberhasilan komputasi awan.

Meskipun beberapa masalah yang terkait dengan jaringan masih ada di pusat data, salah satu yang paling penting adalah memenuhi beragam persyaratan dari berbagai jenis lalu lintas yang hidup berdampingan di lingkungan pusat data. Sebagian besar lalu lintas menggunakan pola komunikasi many-to-one untuk mendapatkan efisiensi kinerja. TCP, yang

telah menjadi protokol transport Internet yang matang sejak beberapa dekade terakhir, menderita gangguan kinerja seperti TCP incast, TCP outcast, penumpukan antrian, tekanan buffer, dan efek pseudocongestion di DCN. Kami menjelaskan masing-masing penurunan tersebut secara singkat bersama dengan penyebab dan kemungkinan pendekatan untuk memitigasinya. Sebuah studi komparatif varian TCP yang telah dirancang khusus untuk DCN disajikan, sambil menjelaskan kelebihan dan kekurangan masing-masing.

**Tinjau Poin**

- Pusat data terutama diklasifikasikan berdasarkan waktu aktif yang mereka berikan kepada pengguna *cloud*.
- Sekitar 15% dari biaya di pusat data digunakan untuk jaringan.
- Jenis lalu lintas yang berbeda dengan persyaratan kinerja yang berbeda ada di DCN.
- TCP/IP tradisional tidak memberikan kinerja jaringan yang optimal di DCN.
- Meskipun beberapa algoritma TCP baru telah dirancang untuk meningkatkan kinerja jaringan di pusat data, masih banyak yang harus dilakukan.

**Latihan Soal**

1. Apa sajakah cara yang berbeda untuk mengklasifikasikan pusat data?
2. Jelaskan berbagai jenis lalu lintas di DCN. Sebutkan persyaratan kinerja mereka dan berikan contoh untuk setiap jenis lalu lintas.
3. Bagaimana penumpukan antrian berbeda dari tekanan buffer?
4. Jelaskan setidaknya dua pendekatan untuk memecahkan masalah pseudocongestion di pusat data virtual.
5. Mengapa mekanisme ECN tradisional tidak dapat digunakan di DCTCP?

# BAB 11
# PENYEDIA LAYANAN *CLOUD*

**Tujuan pembelajaran**

Tujuan utama bab ini adalah untuk memberikan ikhtisar tentang penyedia layanan *cloud* yang berbeda. Setelah membaca bab ini, Anda akan

- Tahu tentang berbagai perusahaan yang mendukung komputasi awan
- Memahami alat *open source/proprietary* yang ditawarkan oleh perusahaan
- Ketahui layanan *cloud* yang ditawarkan oleh perusahaan
- Memahami fitur dan arsitektur yang tersedia dari berbagai alat

**Pengantar**

Bab ini memberikan ikhtisar tentang layanan *cloud* yang ditawarkan oleh berbagai perusahaan. Kami mulai dengan pengenalan layanan *cloud*. Bagian selanjutnya berbicara tentang perusahaan seperti Amazon, Microsoft, Google, EMC, Salesforce, dan IBM yang menyediakan berbagai alat dan layanan untuk memberikan dukungan *cloud*. Setiap bagian menjelaskan secara singkat fitur *cloud* yang didukung oleh perusahaan ini. Ini juga memberikan gambaran tentang alat dan teknologi yang diadaptasi oleh perusahaan untuk memberikan layanan kepada pengguna. Dalam bab ini, kami berfokus untuk memberikan informasi singkat kepada pembaca tentang berbagai alat dan teknologi yang disediakan oleh berbagai perusahaan. Setelah membaca bab ini, pembaca akan dapat membedakan berbagai layanan yang disediakan oleh berbagai perusahaan dan membuat pilihan yang tepat sesuai kebutuhan.

## 11.1 PENDAHULUAN

Komputasi awan adalah salah satu kata kunci paling populer yang digunakan saat ini. Ini adalah sumber daya penyediaan teknologi yang akan datang kepada konsumen dalam bentuk berbagai layanan seperti perangkat lunak, infrastruktur, platform, dan keamanan. Layanan disediakan untuk pengguna berdasarkan permintaan melalui Internet dari server penyedia komputasi awan, bukan disediakan dari server milik perusahaan sendiri. Layanan *cloud* dirancang untuk menyediakan akses yang mudah dan dapat diskalakan ke aplikasi, sumber daya, dan layanan serta dikelola sepenuhnya oleh penyedia layanan *cloud*. Layanan *cloud* dapat secara dinamis menyesuaikan untuk memenuhi kebutuhan penggunanya, dan karena penyedia layanan menyediakan perangkat keras dan perangkat lunak yang diperlukan untuk layanan tersebut, perusahaan tidak perlu menyediakan atau menyebarkan sumber dayanya sendiri atau mengalokasikan informasi. staf teknologi (TI) untuk mengelola layanan. Contoh layanan *cloud* termasuk penyimpanan data online dan solusi pencadangan, layanan email berbasis web, suite kantor yang dihosting dan layanan kolaborasi dokumen, pemrosesan basis data, dan layanan dukungan teknis terkelola.

Layanan *cloud* dapat diklasifikasikan secara luas menjadi tiga jenis: Perangkat Lunak sebagai Layanan (SaaS), Platform sebagai Layanan (PaaS), dan Infrastruktur sebagai Layanan (IaaS). Dengan berkembangnya teknologi, semakin banyak layanan bermunculan di bidang ini, seperti Security as a Service (SeaaS), Knowledge as a Service, dan Data Analytics as a Service.

Banyak perusahaan telah maju untuk mengadaptasi lingkungan *cloud* dan memastikan bahwa pengguna serta perusahaan mendapat manfaat dari ini. Amazon, Microsoft, Google, Yahoo, EMC, Salesforce, Oracle, IBM, dan banyak lagi perusahaan menyediakan berbagai alat dan layanan untuk memberikan dukungan *cloud* bagi pelanggan mereka.

## 11.2 EMC

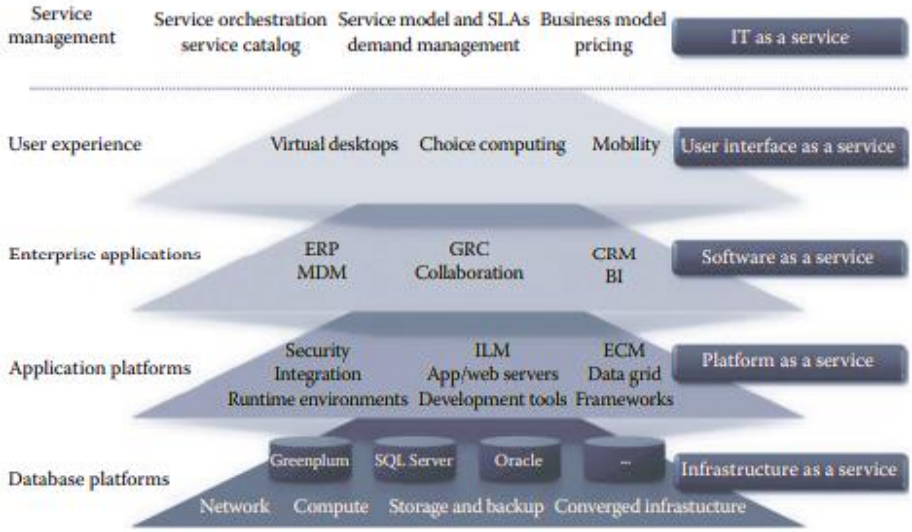
EMC adalah salah satu perusahaan global terkemuka yang membutuhkan skalabilitas dinamis dan kelincahan infrastruktur untuk memenuhi perubahan aplikasi serta kebutuhan bisnis. EMC memilih *cloud computing* sebagai solusi ideal untuk mengurangi kompleksitas dan mengoptimalkan infrastruktur. Menawarkan Teknologi Informasi sebagai Layanan (ITaaS) mengurangi konsumsi energi melalui pembagian sumber daya.

**TI EMC**

Virtualisasi adalah konsep utama di balik kesuksesan EMC IT. Dengan memvirtualisasikan infrastruktur, alokasi sumber daya sesuai permintaan dimungkinkan. Ini juga membantu meningkatkan efisiensi dan pemanfaatan sumber daya.

EMC IT menyediakan unit proses bisnisnya dengan IaaS, PaaS, dan SaaS. Gambar 11.1 memberikan gambaran tentang layanan yang ditawarkan oleh EMC, yang dijelaskan sebagai berikut:

1. IaaS menawarkan unit bisnis EMC kemampuan untuk menyediakan komponen infrastruktur seperti jaringan, penyimpanan, komputasi, dan sistem operasi secara individual atau sebagai layanan terintegrasi.
2. PaaS menyediakan kerangka kerja aplikasi dan informasi yang aman di atas server aplikasi, server web, basis data, manajemen konten tidak terstruktur, dan komponen keamanan sebagai layanan untuk unit bisnis untuk mengembangkan solusi. EMC IT menawarkan platform database (Database Oracle sebagai Layanan, SQL Server sebagai Layanan, Greenplum sebagai Layanan) dan platform aplikasi (pengembangan aplikasi, Manajemen Konten Perusahaan sebagai Layanan, Manajemen Siklus Informasi sebagai Layanan, PaaS Keamanan, Integrasi sebagai Service) untuk tujuan pengembangan.
3. SaaS menyediakan aplikasi dan alat dalam model layanan untuk pemberdayaan bisnis. EMC IT menyatukan beberapa solusi bisnis yang ada di bawah arsitektur terpadu yang disebut Business Intelligence as a Service. Ini juga menawarkan Perencanaan Sumber Daya Perusahaan (ERP) dan Manajemen Hubungan Pelanggan (CRM) sebagai Layanan.
4. Antarmuka Pengguna sebagai Layanan (UIaaS) menyediakan pengalaman pengguna dan antarmuka, bukan menyediakan perangkat aktual yang digunakan.




**Gambar 11.1** Layanan *cloud* oleh EMC. (Diadaptasi dari perjalanan TI EMC ke *cloud* pribadi, aplikasi, dan pengalaman *cloud*, White Paper-EMC.)

**Perangkat Captiva *Cloud***

EMC menawarkan alat bernama Captiva *Cloud* Toolkit untuk membantu pengembangan perangkat lunak. EMC Captiva *Cloud* Toolkit adalah Kit Pengembangan Perangkat Lunak (SDK) yang terdiri dari modul yang membantu pengembang aplikasi web untuk menambahkan fungsionalitas pemindaian dan pencitraan dengan cepat langsung ke aplikasi bisnis berbasis web mereka. Ini sangat ideal untuk vendor penangkap dokumen, pengembang perangkat lunak komersial, dan perusahaan yang ingin membuat aplikasi berbasis web khusus yang diaktifkan sepenuhnya untuk memindai, melengkapi penawaran solusi bisnis mereka.

Dengan menggunakan Captiva *Cloud* Toolkit, pengembang dapat dengan cepat membuat aplikasi bisnis berbasis web yang dapat dipindai dan berfungsi dalam waktu paling cepat 1 minggu. Akibatnya, waktu ke pasar dipersingkat dan biaya pengembangan, pengujian, dan dukungan sangat berkurang. Selain itu, laba atas investasi perusahaan dapat dicapai dengan cepat, dan kemampuannya untuk bersaing dalam pasar penangkapan dokumen terdistribusi yang semakin kompetitif dipercepat.

Ada beberapa modul yang umum digunakan di sebagian besar proses pengembangan. Ini adalah modul dasar yang mengimpor gambar dari berbagai sumber seperti faks, email, atau pemindai atau dari penyimpanan apa pun. Beberapa modul tersebut adalah sebagai berikut:

1. Scan: Scanning adalah kegiatan mengimport dokumen ke Captiva dari scanner. Pada dasarnya, pemindaian terjadi pada tingkat halaman untuk membawa gambar halaman demi halaman ke dalam Captiva. Pemindaian adalah titik masuk ke Captiva di mana seseorang dapat mengimpor segala jenis dokumen seperti pdf, tiff, dan jpg.
2. MDW: Multi Directory Watch adalah titik masuk lain ke Captiva. MDW dapat diarahkan ke folder/repositori mana pun dari mana Captiva dapat mengimpor dokumen secara langsung. MDW sangat berguna jika bisnis mendapatkan dokumen dalam bentuk soft

copy, misalnya file lampiran di email. MDW juga bertindak sebagai modul pindai kecuali ia tidak saling mengunci dengan pemindai.

3. IE: Image enhancement adalah sejenis filter atau alat perbaikan untuk gambar yang tidak jelas. Ini meningkatkan kualitas gambar, sehingga dapat diproses dengan mudah melalui Captiva. Satu dapat mengkonfigurasi IE sesuai kebutuhan bisnis dan gambar yang diterima. Fungsionalitas IE adalah deskew, penghilang kebisingan, dll.
4. Indeks: Pengindeksan adalah aktivitas pengambilan data di Captiva di mana seseorang dapat menangkap data kunci dari berbagai bidang. Misalnya, jika formulir bank sedang diproses, A/C no. dan kode urutkan bisa menjadi bidang pengindeksan. Pengindeksan dapat ditambahkan sesuai kebutuhan bisnis. Bidang validasi dapat ditambahkan untuk menghindari entri data yang tidak diinginkan saat mengindeks dokumen apa pun.
5. Ekspor: Ekspor adalah titik keluar dari Captiva dimana gambar/data dikirim ke berbagai repositori seperti file, net, dokumen, atau data. Data yang diekspor digunakan untuk kebutuhan bisnis dari berbagai divisi bisnis. Misalnya, jika kita menangkap no. A/C. dan mengurutkan kode untuk aplikasi bank, ini dapat dipetakan ke departemen mana pun yang membutuhkannya.
6. Multi: Multi adalah proses terakhir di Captiva untuk menghapus batch yang telah melewati semua modul dan berhasil mengekspor nilai. Multi dapat dikonfigurasi sesuai kebutuhan bisnis. Jika diperlukan untuk mengambil cadangan batch, modul ini dapat dihindari.

Modul yang disebutkan sebelumnya adalah modul Captiva yang sangat mendasar untuk pengindeksan dan ekspor. Tetapi untuk lebih banyak fleksibilitas dan otomatisasi, operator digunakan, yang lebih akurat untuk menangkap data.

## 11.3 GOOGLE

Google adalah salah satu penyedia *cloud* terkemuka yang menawarkan penyimpanan data pengguna yang aman. Ini menyediakan platform *cloud*, mesin aplikasi, *cloud* print, koneksi *cloud*, dan banyak lagi fitur yang dapat diskalakan, andal, dan juga aman. Google menawarkan banyak dari layanan ini secara gratis atau dengan biaya minimum sehingga ramah pengguna.

**Platform *Cloud***

Google *Cloud* Platform memungkinkan pengembang membangun, menguji, dan menerapkan aplikasi pada infrastruktur Google yang sangat skalabel dan andal. Google memiliki salah satu jaringan terbesar dan tercanggih di seluruh dunia. Infrastruktur perangkat lunak seperti MapReduce, BigTable, dan Dremel merupakan inovasi untuk pengembangan industri.

Google *Cloud* Platform mencakup mesin virtual, penyimpanan blok, penyimpanan data NoSQL, dan analitik data besar. Ini menyediakan berbagai layanan penyimpanan yang memungkinkan perawatan yang mudah dan akses cepat ke data pengguna. Platform *cloud* menawarkan platform yang dikelola sepenuhnya serta mesin virtual fleksibel yang memungkinkan pengguna untuk memilih sesuai kebutuhan. Google juga menyediakan integrasi aplikasi pengguna yang mudah di dalam platform *cloud*.

Aplikasi yang dihosting di platform *cloud* dapat ditingkatkan secara otomatis untuk menangani beban kerja yang paling menuntut dan menurunkan skala saat lalu lintas mereda. Platform *cloud* dirancang untuk skala seperti produk Google sendiri, bahkan ketika ada lonjakan lalu lintas yang besar. Layanan terkelola seperti App Engine atau *Cloud Datastore* menyediakan penskalaan otomatis yang memungkinkan aplikasi berkembang bersama pengguna. Pengguna harus membayar hanya untuk apa yang dia gunakan.

**Penyimpanan C*loud***

Google Cloud Storage adalah layanan web penyimpanan file online RESTful untuk menyimpan dan mengakses data seseorang di infrastruktur Google. Representational state transfer (REST) adalah gaya arsitektur yang terdiri dari serangkaian batasan arsitektural terkoordinasi yang diterapkan pada komponen, konektor, dan elemen data dalam sistem terdistribusi. Layanan ini menggabungkan kinerja dan skalabilitas *cloud* Google dengan keamanan tingkat lanjut dan kemampuan berbagi. Google *Cloud* Storage aman dan terjamin. Data dilindungi melalui penyimpanan redundan di beberapa lokasi fisik.

Berikut adalah beberapa alat untuk Google *Cloud* Storage:

- Google Developers Console adalah aplikasi web tempat seseorang dapat melakukan tugas manajemen penyimpanan sederhana di sistem Google *Cloud* Storage.
- gsutil adalah aplikasi Python yang memungkinkan pengguna mengakses Google *Cloud* Storage dari baris perintah.

**Google Cloud Connect**

Google Cloud Connect adalah fitur yang disediakan oleh Google *Cloud* dengan mengintegrasikan *cloud* dan application programming interface (API) untuk Microsoft Office. Setelah menginstal plug-in untuk rangkaian program Microsoft Office, seseorang dapat menyimpan file ke *cloud*. Salinan *cloud* dari file tersebut menjadi dokumen utama yang digunakan semua orang. Google Cloud Connect menetapkan setiap file URL unik yang dapat dibagikan agar orang lain dapat melihat dokumen tersebut.

Jika perubahan dilakukan pada dokumen, perubahan tersebut akan muncul untuk semua orang yang melihatnya. Saat beberapa orang melakukan perubahan pada bagian dokumen yang sama, *Cloud* Connect memberikan kesempatan kepada pengguna untuk memilih kumpulan perubahan mana yang akan disimpan.

Saat pengguna mengunggah dokumen ke Google Cloud Connect, layanan memasukkan beberapa metadata ke dalam file. Metadata adalah informasi tentang informasi lainnya. Dalam hal ini, metadata mengidentifikasi file sehingga perubahan akan dilacak di semua salinan. Bagian belakang mirip dengan Sistem File Google dan bergantung pada infrastruktur Google Docs. Saat dokumen disinkronkan ke file master, Google Cloud Connect mengirimkan data yang diperbarui ke semua salinan dokumen yang diunduh menggunakan metadata untuk memandu pembaruan ke file yang tepat.

**Google Cloud Print**

Google Cloud Print adalah layanan yang memperluas fungsi printer ke perangkat apa pun yang dapat terhubung ke Internet. Untuk menggunakan Google Cloud Print, pengguna harus memiliki profil Google gratis, aplikasi, program, atau situs web yang menggabungkan

fitur Google Cloud Print, printer siap pakai di gemawan atau printer yang terhubung ke komputer yang masuk ke Internet.

Saat Google Cloud Print digunakan melalui aplikasi atau situs web, permintaan cetak melewati server Google. Google merutekan permintaan ke printer yang sesuai yang dikaitkan dengan akun Google pengguna. Dengan asumsi masing-masing printer menyala dan memiliki koneksi Internet aktif, kertas, dan tinta, pekerjaan cetak harus dilakukan pada mesin. Printer dapat dibagikan dengan orang lain untuk menerima dokumen melalui Google Cloud Print.

Karena sebagian besar printer tidak siap untuk *cloud*, sebagian besar pengguna Google Cloud Print memerlukan komputer yang berfungsi sebagai penghubung. Google Cloud Print adalah ekstensi yang terpasang di Peramban Google Chrome, tetapi harus diaktifkan secara eksplisit. Setelah diaktifkan, layanan mengaktifkan sepotong kecil kode yang disebut konektor. Tugas konektor adalah menghubungkan antara printer dan dunia luar. Konektor menggunakan perangkat lunak printer komputer pengguna untuk mengirim perintah ke printer.

Jika seseorang memiliki printer cloud-ready, seseorang dapat menghubungkan printer ke Internet secara langsung tanpa memerlukan komputer khusus. Printer awan harus didaftarkan ke Google Cloud Print untuk memanfaatkan kemampuannya.

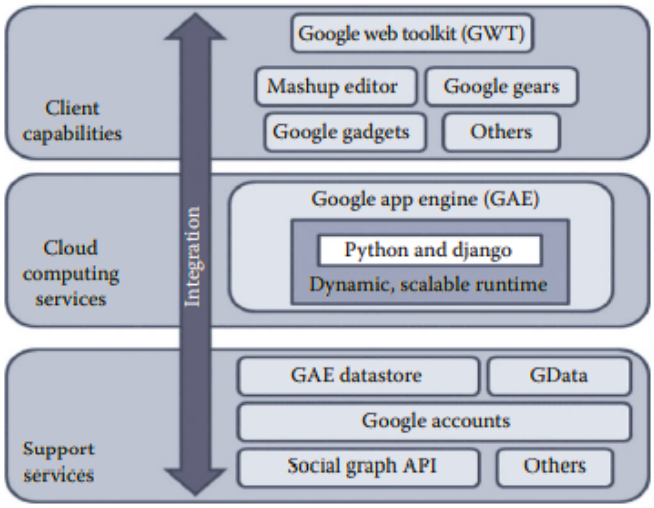
Karena Google mengizinkan pengembang aplikasi dan situs web untuk memasukkan Google Cloud Print ke dalam produk mereka sesuai keinginan mereka, tidak ada pendekatan standar untuk melaksanakan tugas pencetakan. Google Cloud Print bergantung pada pengembang yang memasukkan fitur tersebut ke dalam produk mereka. Tidak setiap aplikasi atau situs memiliki Google Cloud Print yang terpasang di dalamnya, yang membatasi fungsinya. Biasanya, Google membangun layanan ke dalam produknya sendiri, tetapi banyak orang mengandalkan layanan dari berbagai sumber dan mungkin menemukan bahwa Google Cloud Print tidak memiliki adopsi yang cukup luas untuk memenuhi semua kebutuhan mereka.

**Mesin Aplikasi Google**

Google App Engine memungkinkan pengguna menjalankan aplikasi web di infrastruktur Google. Aplikasi App Engine mudah dibuat, mudah dirawat, dan mudah diskalakan seiring meningkatnya kebutuhan lalu lintas dan penyimpanan data. Dengan App Engine, tidak ada server yang harus dipelihara: Cukup unggah aplikasi, dan siap melayani pengguna. Aplikasi dapat dilayani dari nama domain milik pengguna (seperti http://www.example.com/) menggunakan Google Apps. Jika tidak, dapat dilayani menggunakan nama gratis di domain appspot.com. Aplikasi dapat dibagikan dengan dunia atau membatasi akses ke anggota organisasi. Gambar 11.2 menunjukkan berbagai modul di Google App Engine. Integrasi layanan *cloud computing* dengan layanan dukungan dan kemampuan klien ditunjukkan dalam diagram.

Google App Engine mendukung aplikasi yang ditulis dalam beberapa bahasa pemrograman. Dengan lingkungan runtime Java App Engine, seseorang dapat membuat aplikasi menggunakan teknologi Java standar, termasuk JVM, servlet Java, dan bahasa pemrograman Java—atau bahasa lainnya. App Engine juga dilengkapi dengan lingkungan runtime Python, yang menyertakan juru bahasa Python cepat dan pustaka standar Python. App Engine juga menampilkan runtime PHP, dengan dukungan asli untuk Google *Cloud* SQL

dan Google Cloud Storage yang berfungsi seperti menggunakan instance MySQL lokal dan melakukan penulisan file lokal. Terakhir, App Engine menyediakan lingkungan runtime Go yang menjalankan kode Go yang dikompilasi secara native. Lingkungan runtime ini dibuat untuk memastikan bahwa aplikasi Anda berjalan dengan cepat, aman, dan tanpa gangguan dari aplikasi lain di sistem.




**Gambar 11.2** Mesin Aplikasi Google.

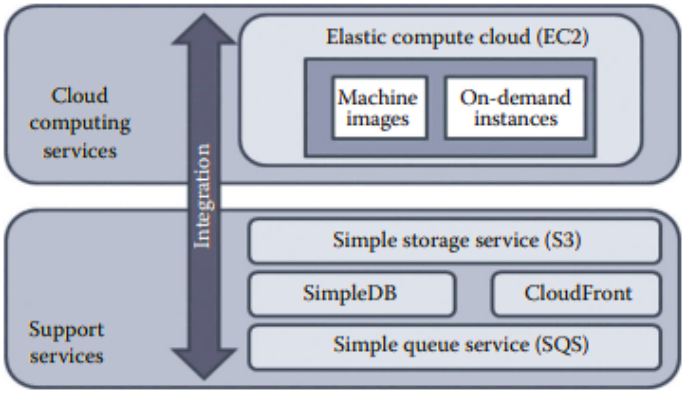
Dengan App Engine juga, pengguna hanya perlu membayar apa yang dia gunakan. Tidak ada biaya penyiapan dan tidak ada biaya berulang. Sumber daya yang digunakan oleh aplikasi seperti penyimpanan dan bandwidth diukur dalam gigabyte dan ditagih dengan tarif yang bersaing. Seseorang harus mengontrol jumlah maksimum sumber daya yang dapat digunakan oleh aplikasi, sehingga selalu sesuai dengan anggarannya.

App Engine tidak memerlukan biaya apa pun untuk memulai. Semua aplikasi dapat menggunakan penyimpanan hingga 1 GB dan CPU serta bandwidth yang cukup untuk mendukung aplikasi efisien yang melayani sekitar lima juta tampilan halaman per bulan, benar-benar gratis. Saat penagihan diaktifkan untuk aplikasi, batas gratis dinaikkan, dan seseorang hanya perlu membayar untuk sumber daya yang digunakan di atas level gratis.

## 11.4 LAYANAN WEB AMAZON

Amazon Web Services (AWS) adalah kumpulan layanan komputasi jarak jauh (juga disebut layanan web) yang bersama-sama membentuk platform komputasi awan, yang ditawarkan melalui Internet oleh Amazon.com. Layanan yang paling sentral dan terkenal adalah Amazon Elastic Compute *Cloud* (Amazon EC2), Amazon Simple Queue Service (Amazon SQS), dan Amazon S3 seperti yang ditunjukkan pada Gambar 11.3. Amazon EC2 adalah layanan komputasi, sedangkan Amazon SQS dan Amazon S3 adalah layanan dukungan. Layanan ini diiklankan sebagai menyediakan kapasitas komputasi yang besar (berpotensi banyak server) jauh lebih cepat dan lebih murah daripada membangun server farm fisik. Pusat data Amazon berlokasi di Ashburn, Virginia, Dallas/Fort Worth, Los Angeles, Miami, Newark, New Jersey,

Palo, Alto, California, Seattle, St. Louis, Amsterdam, Dublin, Frankfurt, London, Hong Kong, Singapura, Tokyo, dll.




**Gambar 11.3** AWS.

**Amazon Elastic Compute *Cloud***

Amazon EC2 adalah IaaS yang ditawarkan oleh AWS dan merupakan penyedia IaaS terkemuka di pasar saat ini. Didukung oleh infrastruktur besar yang telah dibangun perusahaan untuk menjalankan bisnis ritelnya, Amazon EC2 menyediakan lingkungan komputasi virtual yang sesungguhnya. Dengan menyediakan berbagai jenis mesin virtual atau instans, sistem operasi, dan paket perangkat lunak untuk dipilih, Amazon EC2 memungkinkan pengguna membuat mesin virtual pilihannya melalui antarmuka layanan web. Pengguna dapat mengubah kapasitas dan karakteristik mesin virtual dengan menggunakan antarmuka layanan web, maka dinamakan elastis. Kapasitas komputasi disediakan dalam bentuk mesin virtual atau instans server dengan mem-boot Amazon Machine Images (AMI), yang dapat dibuat instance-nya oleh pengguna. AMI berisi semua informasi yang diperlukan untuk membuat instance. Antarmuka Graphical User Interface (GUI) utama adalah AWS Management Console (tunjuk dan klik) dan API layanan web yang mendukung Protokol Akses Objek Sederhana dan Permintaan Kueri. API menyediakan pustaka dan sumber daya pemrograman untuk Java, PHP, Python, Ruby, Windows, dan .Net. Infrastruktur divirtualisasikan dengan menggunakan hypervisor Xen, dan jenis instans yang berbeda disediakan sebagai berikut:

- Instans standar—cocok untuk sebagian besar aplikasi
- Instans mikro—cocok untuk aplikasi throughput rendah
- Instans dengan memori tinggi—cocok untuk aplikasi throughput tinggi
- Instans CPU tinggi—cocok untuk aplikasi intensif komputasi
- Instans komputasi klaster—cocok untuk aplikasi komputasi kinerja tinggi (HPC).

Mesin virtual dapat diperoleh sesuai permintaan setiap jam, sehingga menghilangkan kebutuhan untuk memperkirakan kebutuhan komputasi lebih awal. Instans dapat dipesan lebih awal, dan tarif diskon dibebankan untuk instans tersebut. Pengguna juga dapat menawar kapasitas komputasi Amazon EC2 yang tidak terpakai dan mendapatkan instans. Instans semacam itu disebut sebagai Instans Spot. Tawaran yang melebihi Harga Spot saat ini

disediakan dengan instans, yang memungkinkan pengguna mengurangi biaya. Harga Spot bervariasi dan diputuskan oleh perusahaan.

Mesin virtual dapat ditempatkan di beberapa lokasi, yang ditentukan oleh wilayah dan zona ketersediaan. Availability zone adalah lokasi berbeda yang direkayasa untuk diisolasi dari kegagalan di availability zone lain dan menyediakan konektivitas jaringan latensi rendah yang murah ke availability zone lain di wilayah yang sama. Dengan demikian, menempatkan instans di beberapa lokasi memungkinkan toleransi kesalahan dan keandalan kegagalan. Instans Amazon EC2 dapat dipantau dan dikontrol oleh AWS Management Console dan API layanan web. Namun, AWS menyediakan Amazon *Cloud* Watch, layanan web yang menyediakan pemantauan untuk sumber daya *cloud* AWS, dimulai dengan Amazon EC2. Ini memberi pelanggan visibilitas ke dalam penggunaan sumber daya, kinerja operasional, dan pola permintaan keseluruhan—termasuk metrik seperti penggunaan CPU, pembacaan dan penulisan disk, dan lalu lintas jaringan.

Mesin virtual diautentikasi menggunakan protokol berbasis tanda tangan, yang menggunakan pasangan kunci. Fitur penting lainnya yang disediakan adalah Amazon Virtual Private *Cloud* (Amazon VPC). Infrastruktur TI yang ada dapat dihubungkan ke Amazon EC2 melalui jaringan pribadi virtual (VPN). Sumber daya komputasi terisolasi disediakan di Amazon VPC, dan kemampuan manajemen yang ada seperti layanan keamanan, firewall, dan sistem deteksi intrusi dapat diperluas ke sumber daya terisolasi Amazon EC2.

Elastic load balancing (ELB) memungkinkan pengguna untuk secara otomatis mendistribusikan dan menyeimbangkan lalu lintas aplikasi yang masuk di antara instans yang berjalan berdasarkan metrik seperti jumlah permintaan dan latensi permintaan. Toleransi kesalahan dan penskalaan otomatis dapat dilakukan dengan mengonfigurasi ELB sesuai kebutuhan spesifik. ELB memantau kesehatan instans yang berjalan dan merutekan lalu lintas dari instans yang gagal.

Instance disimpan selama ia beroperasi dan dihapus saat terminasi. Penyimpanan persisten dapat diaktifkan dengan menggunakan Elastic Block Storage (EBS) atau Amazon Simple Storage Service (S3). EBS menyediakan penyimpanan yang sangat andal dan aman, dan volume penyimpanan dapat digunakan untuk mem-boot instans Amazon EC2 atau dilampirkan ke instans sebagai perangkat blok standar. Amazon S3 menyediakan infrastruktur penyimpanan yang sangat tahan lama yang dirancang untuk penyimpanan data penting dan primer. Penyimpanan didasarkan pada unit yang disebut objek yang ukurannya dapat bervariasi dari satu byte hingga lima gigabyte data. Objek ini disimpan dalam keranjang dan diambil melalui kunci unik yang ditetapkan pengembang.

Itu dapat diakses melalui antarmuka layanan web dan menyediakan prosedur otentikasi untuk melindungi dari akses yang tidak sah.

**Layanan Penyimpanan Sederhana Amazon**

Amazon Simple Storage Service dikenal sebagai Amazon S3, adalah penyimpanan untuk Internet. Ini dirancang untuk membuat komputasi skala web lebih mudah bagi pengembang. Amazon S3 menyediakan antarmuka layanan web sederhana yang dapat digunakan untuk menyimpan dan mengambil sejumlah data, kapan pun, dari mana pun di web. Ini memberi setiap pengembang akses ke infrastruktur yang sangat dapat diskalakan,

andal, aman, cepat, dan murah yang digunakan Amazon untuk menjalankan jaringan situs web globalnya sendiri. Layanan ini bertujuan untuk memaksimalkan manfaat skala dan meneruskan manfaat tersebut kepada pengembang.

Seiring dengan kesederhanaannya, ini juga menangani fitur lain seperti keamanan, skalabilitas, keandalan, kinerja, dan biaya. Dengan demikian, Amazon S3 adalah layanan yang sangat dapat diskalakan, andal, murah, cepat, dan juga mudah digunakan yang memenuhi persyaratan dan ekspektasi desain.

Amazon S3 menyediakan penyimpanan yang sangat tahan lama dan tersedia untuk berbagai konten, mulai dari aplikasi web hingga file media. Hal ini memungkinkan pengguna untuk membongkar penyimpanan di mana seseorang dapat memanfaatkan skalabilitas dan harga bayar sesuai pemakaian. Untuk berbagi konten yang dapat direproduksi dengan mudah atau di mana seseorang perlu menyimpan salinan asli di tempat lain, fitur Reduced Redundancy Storage (RRS) Amazon S3 memberikan solusi yang menarik. Ini juga memberikan solusi yang lebih baik dalam hal penyimpanan untuk analitik data. Amazon S3 adalah solusi ideal untuk menyimpan data farmasi untuk analisis, data keuangan untuk komputasi, dan gambar untuk diubah ukurannya. Kemudian konten ini dapat dikirim ke Amazon EC2 untuk perhitungan, pengubahan ukuran, atau analitik berskala besar lainnya tanpa dikenakan biaya transfer data apa pun untuk memindahkan data antar layanan.

Amazon S3 menawarkan solusi yang dapat diskalakan, aman, dan sangat tahan lama untuk pencadangan dan pengarsipan data penting. Untuk data berukuran signifikan, fitur Impor/Ekspor AWS dapat digunakan untuk memindahkan data dalam jumlah besar ke dalam dan ke luar AWS dengan perangkat penyimpanan fisik. Ini ideal untuk memindahkan data dalam jumlah besar untuk pencadangan berkala, atau mengambil data dengan cepat untuk skenario pemulihan bencana. Fitur lain yang ditawarkan oleh Amazon S3 adalah Hosting Situs Web Statis, yang ideal untuk situs web dengan konten statis, termasuk file html, gambar, video, dan skrip sisi klien seperti JavaScript.

**Layanan Antrean Sederhana Amazon**

Layanan AWS lainnya adalah Amazon SQS. Ini adalah layanan antrean pesan yang cepat, andal, dapat diskalakan, dikelola sepenuhnya. SQS memudahkan dan menghemat biaya untuk memisahkan komponen aplikasi *cloud*. SQS dapat digunakan untuk mentransmisikan volume data apa pun, pada tingkat throughput apa pun, tanpa kehilangan pesan atau memerlukan layanan lain untuk selalu tersedia.

Amazon SQS adalah sistem antrean terdistribusi yang memungkinkan aplikasi layanan web mengantri pesan dengan cepat dan andal yang dihasilkan oleh satu komponen dalam aplikasi untuk dikonsumsi oleh komponen lain. Antrian adalah tempat penyimpanan sementara untuk pesan yang sedang menunggu untuk diproses.

Amazon SQS menawarkan berbagai fitur seperti memungkinkan beberapa pembaca dan penulis pada saat yang sama, menyediakan fasilitas kontrol akses, menjamin ketersediaan pengiriman yang tinggi, dan mengambil pesan karena infrastruktur yang berlebihan. Ini juga memberikan ketentuan untuk memiliki pesan panjang variabel serta pengaturan yang dapat dikonfigurasi untuk setiap antrian.

## 11.5 MICROSOFT

Komputasi awan menyediakan cara baru dalam memandang TI di Microsoft yang disebut Microsoft IT (MSIT). Komputasi awan sekarang menjadi lingkungan pilihan dan default untuk aplikasi baru dan yang dimigrasikan di Microsoft. MSIT telah mengembangkan metodologi dan serangkaian praktik terbaik untuk menganalisis portofolio aplikasi mereka saat ini untuk kandidat yang memungkinkan untuk bermigrasi ke komputasi awan. Analisis ini memungkinkan MSIT untuk memilih lingkungan berbasis komputasi awan yang ideal untuk setiap aplikasi. MSIT telah menangkap praktik terbaik ini dan mendokumentasikannya untuk pelanggan Microsoft lainnya yang ingin memigrasikan organisasi mereka ke komputasi awan.

**Windows Azure**

Windows Azure Cloud Services (peran web dan pekerja/PaaS) memungkinkan pengembang menyebarkan dan mengelola layanan aplikasi dengan mudah. Ini mendelegasikan pengelolaan contoh peran yang mendasari dan sistem operasi ke platform Windows Azure.

Migration Assessment Tool (MAT) untuk Windows Azure merangkum semua informasi yang harus diperhatikan sebelum mencoba migrasi aplikasi ke Windows Azure. Berdasarkan tanggapan terhadap serangkaian pertanyaan biner sederhana, alat tersebut menghasilkan laporan yang menguraikan jumlah upaya pengembangan yang terlibat untuk memigrasikan aplikasi, atau pertimbangan arsitektur untuk aplikasi baru.

Kalkulator Harga Windows Azure menganalisis potensi kebutuhan *cloud* publik aplikasi terhadap biaya infrastruktur aplikasi yang ada. Alat ini dapat membantu membandingkan biaya operasional saat ini untuk suatu aplikasi, dengan biaya pengoperasian pada Windows Azure dan SQL Azure.

Windows Azure Pack untuk Windows Server adalah kumpulan teknologi Windows Azure yang tersedia untuk pelanggan Microsoft tanpa biaya tambahan untuk penginstalan ke pusat data mereka. Ini berjalan di atas Windows Server 2012 R2 dan System Center 2012 R2 dan, melalui penggunaan teknologi Windows Azure, ini memungkinkan Anda untuk menawarkan *cloud* multitenant yang kaya, swalayan, konsisten dengan pengalaman publik Windows Azure.

**Perangkat Penilaian dan Perencanaan Microsoft**

Microsoft Assessment and Planning Toolkit (MAP) adalah alat perencanaan dan penilaian multiproduk tanpa agen, otomatis, untuk migrasi *cloud*. MAP memberikan laporan penilaian kesiapan terperinci, proposal eksekutif, dan informasi perangkat keras dan perangkat lunak. Ini juga memberikan rekomendasi untuk membantu organisasi mempercepat proses migrasi aplikasi untuk penilaian perencanaan *cloud* pribadi dan publik. MAP menganalisis data utilisasi server untuk virtualisasi server dan juga konsolidasi server dengan Hyper-V.

**SharePoint**

Microsoft menawarkan alat kolaborasi online sendiri yang disebut SharePoint. Microsoft SharePoint adalah platform aplikasi web yang terdiri dari serangkaian teknologi web serbaguna yang didukung oleh infrastruktur teknis umum. Secara default, SharePoint memiliki antarmuka mirip Microsoft Office, dan terintegrasi erat dengan suite Office. Alat web

dirancang agar dapat digunakan oleh pengguna nonteknis. SharePoint dapat digunakan untuk menyediakan portal intranet, manajemen dokumen dan file, kolaborasi, jejaring sosial, ekstranet, situs web, pencarian perusahaan, dan intelijen bisnis. Ini juga memiliki integrasi sistem, integrasi proses, dan kemampuan otomatisasi alur kerja. Tidak seperti Google *Cloud* Connect, Microsoft SharePoint bukanlah alat gratis. Tetapi memiliki fitur tambahan yang tidak dapat ditandingi oleh Google atau perusahaan lain.

## 11.6 IBM

IBM merupakan salah satu pemain di bidang *cloud computing* yang menawarkan berbagai layanan *cloud* kepada konsumen. Komputasi awan IBM terdiri dari solusi komputasi awan untuk perusahaan seperti yang ditawarkan oleh perusahaan IT global IBM. Semua tawaran dirancang untuk penggunaan bisnis, dipasarkan dengan nama IBM SmartCloud. Cloud IBM mencakup IaaS, SaaS, dan PaaS yang ditawarkan melalui model pengiriman *cloud* publik, pribadi, dan hibrid, selain komponen yang membentuk *cloud* tersebut.

IBM menawarkan titik masuk ke komputasi awan apakah klien merancang *cloud* pribadi virtual mereka sendiri, menerapkan layanan *cloud*, atau menggunakan aplikasi beban kerja *cloud*. Kerangka *cloud* IBM dimulai dengan perangkat keras fisik *cloud*. IBM menawarkan tiga platform perangkat keras untuk komputasi awan, yang menawarkan dukungan bawaan untuk virtualisasi. Lapisan berikutnya dari kerangka kerja IBM adalah virtualisasi. IBM menawarkan solusi infrastruktur aplikasi IBM Websphere yang mendukung model pemrograman dan standar terbuka untuk virtualisasi.

Lapisan manajemen kerangka *cloud* IBM mencakup perangkat menengah IBM Tivoli. Alat manajemen memberikan kemampuan untuk mengatur gambar dengan penyediaan dan deprovisioning otomatis, memantau operasi, dan penggunaan meter sambil melacak biaya dan mengalokasikan tagihan. Lapisan terakhir dari kerangka menyediakan alat beban kerja terintegrasi. Beban kerja untuk komputasi awan adalah layanan atau contoh kode yang dapat dijalankan untuk memenuhi kebutuhan bisnis tertentu. IBM menawarkan alat untuk kolaborasi, pengembangan dan pengujian berbasis *cloud*, pengembangan aplikasi, analitik, integrasi bisnis-ke-bisnis, dan keamanan.

**Model Awan**

IBM menawarkan spektrum opsi pengiriman *cloud* mulai dari *cloud* pribadi semata hingga *cloud* publik semata dan banyak variasi di antaranya. IBM memberikan opsi untuk membangun solusi *cloud* yang disesuaikan dari kombinasi elemen *cloud* publik dan *cloud* pribadi. Perusahaan yang lebih suka menyimpan semua data dan proses di belakang firewall mereka sendiri dapat memilih solusi *cloud* pribadi yang dikelola oleh staf TI mereka sendiri. Perusahaan juga dapat memilih harga bayar sesuai penggunaan yang memungkinkan mereka menjalankan aplikasi profil rendah pada model *cloud* publik yang aman. Opsi *cloud* hibrid memungkinkan beberapa proses dihosting dan dikelola oleh IBM, sementara yang lain disimpan di *cloud* pribadi atau di VPN atau Jaringan Area Lokal Virtual. IBM juga menawarkan perencanaan dan konsultasi selama proses penerapan. Komputasi awan adalah pilihan terbaik untuk perangkat lunak seluler. IBM menawarkan lima model penyediaan *cloud* yang berbeda:

1. *Cloud* pribadi, dimiliki dan dioperasikan oleh pelanggan

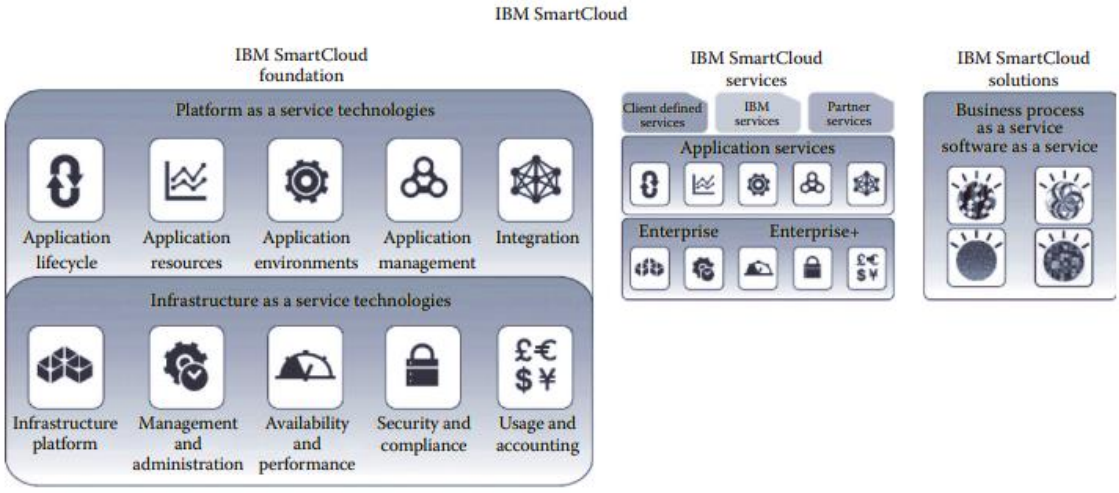
2. *Cloud* pribadi, dimiliki oleh pelanggan tetapi dioperasikan oleh IBM (atau penyedia lain)
3. *Cloud* pribadi, dimiliki dan dioperasikan oleh IBM (atau penyedia lainnya)
4. Layanan *cloud* pribadi virtual, berdasarkan dukungan multitenant untuk perusahaan individu
5. Layanan *cloud* publik, berdasarkan penyediaan fungsi kepada individu

Mayoritas pengguna *cloud* memilih model *cloud* hybrid, dengan beberapa beban kerja dilayani oleh sistem internal, beberapa dari penyedia *cloud* komersial, dan beberapa dari penyedia layanan *cloud* publik. Untuk pelanggan perusahaan yang merasa bahwa risiko keamanan adopsi komputasi awan terlalu tinggi, IBM berspesialisasi dalam penawaran *cloud* pribadi yang aman.

Untuk membangun *cloud* yang benar-benar pribadi, IBM menawarkan IBM Workload Deployer dan Cloudburst sebagai solusi *cloud in a box* yang siap diterapkan. Cloudburst menyediakan server blade, middleware, dan virtualisasi bagi perusahaan untuk membangun mesin virtual *cloud*-ready miliknya sendiri. Workload Deployer menghubungkan server perusahaan yang ada ke komponen virtualisasi dan middleware untuk membantu menyebarkan mesin virtual standar yang dirancang oleh IBM. Untuk pelanggan yang lebih memilih untuk melakukan integrasi *cloud* pribadi mereka sendiri, IBM menawarkan pilihan blok bangunan perangkat keras dan perangkat lunak, bersama dengan rekomendasi dan arsitektur referensi, yang memimpin dalam penerapan. Klien dapat memilih dari server, middleware, dan aplikasi SaaS yang mendukung virtualisasi IBM.

**IBM SmartCloud**

IBM SmartCloud adalah ekosistem produk dan solusi komputasi awan bermerek dari IBM. Ini mencakup IaaS, SaaS, dan PaaS yang ditawarkan melalui model pengiriman *cloud* publik, pribadi, dan hybrid. IBM menempatkan tawaran ini di bawah tiga payung: SmartCloud Foundation, Layanan SmartCloud, dan Solusi SmartCloud. Gambar 11.4 menjelaskan secara singkat arsitektur IBM SmartCloud.




**Gambar 11.4** Arsitektur IBM SmartCloud.

SmartCloud Foundation terdiri dari infrastruktur, perangkat keras, penyediaan, manajemen, integrasi, dan keamanan yang berfungsi sebagai dasar *cloud* pribadi atau hybrid. Dibangun menggunakan komponen dasar tersebut, PaaS, IaaS, dan layanan pencadangan membentuk Layanan SmartCloud. Berjalan di platform dan infrastruktur *cloud* ini, SmartCloud Solutions terdiri dari sejumlah aplikasi kolaborasi, analitik, dan pemasaran SaaS.

Bersamaan dengan IaaS, PaaS, dan SaaS, IBM juga menawarkan Proses Bisnis sebagai Layanan (BPaaS). Layanan *cloud* infrastruktur memberi konsumen penyediaan pemrosesan, penyimpanan, jaringan, dan sumber daya komputasi fundamental lainnya di mana konsumen dapat menyebarkan dan menjalankan perangkat lunak sewenang-wenang, yang dapat mencakup sistem operasi dan aplikasi. Dalam layanan *cloud* platform, konsumen dapat menyebarkan aplikasi yang dibuat konsumen atau yang diperoleh konsumen ke infrastruktur *cloud* yang dibuat menggunakan bahasa pemrograman dan alat yang didukung oleh penyedia. Layanan *cloud* aplikasi memungkinkan konsumen untuk menggunakan aplikasi penyedia yang berjalan di infrastruktur *cloud*. Aplikasi dapat diakses dari berbagai perangkat klien melalui antarmuka klien tipis seperti browser web (misalnya, email berbasis web). Layanan *cloud* proses bisnis adalah setiap proses bisnis (horizontal atau vertikal) yang disampaikan melalui model layanan *cloud* (multitenant, penyediaan layanan mandiri, penskalaan elastis, dan pengukuran penggunaan atau harga) melalui Internet dengan akses melalui antarmuka web-sentris dan mengeksploitasi arsitektur *cloud* berorientasi web. Penyedia BPaaS bertanggung jawab atas fungsi bisnis terkait.

## 11.7 SAP LAPS

SAP Labs membuat perangkat lunak perusahaan untuk mengelola operasi bisnis dan hubungan pelanggan. SAP adalah pemimpin di pasar aplikasi perusahaan dalam hal perangkat lunak dan layanan terkait perangkat lunak. Produk perangkat lunak perusahaan yang paling terkenal adalah sistem dan manajemen aplikasi perencanaan sumber daya perusahaan (SAP ERP), produk gudang data perusahaannya—SAP Business Warehouse (SAP BW), perangkat lunak SAP Business Objects, dan yang terbaru, produk seluler Sybase dan alat komputasi memori SAP HANA. SAP adalah salah satu perusahaan perangkat lunak terbesar di dunia.

**Platform *Cloud* SAP HANA**

SAP HANA *Cloud Platform* adalah PaaS modular berbasis Eclipse standar terbuka. Di SAP HANA *Cloud Platform*, aplikasi disebarkan melalui alat baris perintah ke *cloud* sebagai file arsip aplikasi web (WAR) atau bundel OSGi. Bundel OSGi adalah komponen jar normal dengan *header manifes* tambahan. Aplikasi berjalan dalam lingkungan runtime SAP HANA *Cloud Platform* berbasis Java. Didukung oleh SAP HANA dan dapat dikelola menggunakan alat manajemen berbasis web.

Fitur utama SAP HANA *Cloud Platform* adalah sebagai berikut:

- Platform perusahaan yang dibuat untuk pengembang
- Integrasi asli dengan perangkat lunak SAP dan non-SAP
- Persistensi dalam memori
- Platform data yang aman
- Kontainer runtime modular yang ringan untuk aplikasi

SAP HANA *Cloud Platform* memungkinkan pengguna dengan cepat membangun dan menyebarkan aplikasi bisnis dan konsumen yang memberikan fungsionalitas baru yang penting untuk memenuhi kebutuhan bisnis yang muncul. Ini juga membantu menghubungkan pengguna dengan pelanggan dalam pengalaman yang lebih menarik. Ini menyediakan konektivitas berdasarkan layanan konektivitas *cloud*. Akibatnya, platform merampingkan integrasi aplikasi baru dengan total biaya kepemilikan serendah mungkin. Dukungan untuk standar pemrograman terbuka memberikan hambatan masuk yang rendah bagi pengembang. Ini membuat mereka produktif sejak awal dalam membangun aplikasi perusahaan yang dapat diintegrasikan dengan solusi SAP atau non-SAP apa pun. Tidak diperlukan keterampilan pengkodean baru untuk bekerja dengan SAP HANA.

**Layanan Virtualisasi yang Disediakan oleh SAP**

Virtualisasi ERP meningkatkan laba atas investasi proyek dengan memaksimalkan pemanfaatan perangkat keras. Manfaat bisnis dari virtualisasi aplikasi ERP adalah siklus pengembangan yang lebih pendek, pengurangan biaya TI, peningkatan ketersediaan, dan penghematan energi. Layanan bersama dari SAP dan VMware membantu transisi ke platform *cloud* pribadi yang lebih terbuka dan fleksibel berdasarkan teknologi virtualisasi yang telah terbukti.

## 11.8 SALESFORCE.COM

Salesforce.com adalah penyedia komputasi awan dan SaaS perusahaan sosial yang berbasis di San Francisco. Dari platform *cloud* dan aplikasinya, perusahaan ini terkenal dengan produk Salesforce CRM-nya, yang terdiri dari *Sales Cloud, Service Cloud, Marketing Cloud*, Force.com, Chatter, dan Work.com. Selain produk dan platformnya, Salesforce.com membuat AppExchange, platform pembuatan dan berbagi aplikasi kustom. Perusahaan juga memiliki layanan konsultasi, penyebaran, dan pelatihan.

**Awan Penjualan**

*Sales Cloud* mengacu pada modul penjualan di Salesforce.com. Ini mencakup Prospek, Akun, Kontak, Kontrak, Peluang, Produk, Buku Harga, Kutipan Harga, dan Kampanye (batasan berlaku). Ini mencakup fitur seperti web-to-lead untuk mendukung pengambilan prospek online, dengan aturan respons otomatis. Ini dirancang untuk menjadi penyiapan awal hingga akhir untuk seluruh proses penjualan. Sales *Cloud* mengelola informasi kontak dan mengintegrasikan media sosial dan kolaborasi pelanggan real-time melalui Chatter. Sales *Cloud* memberikan platform untuk terhubung dengan pelanggan dari informasi akun yang lengkap dan terkini hingga wawasan sosial, semuanya di satu tempat dan tersedia kapan saja, di mana saja. Semuanya didorong secara otomatis dalam waktu nyata, mulai dari informasi kontak hingga pembaruan kesepakatan dan persetujuan diskon.

Salesforce.com menciptakan *Sales Cloud* agar mudah digunakan seperti situs web konsumen seperti Amazon dan membangunnya di *cloud* untuk menghilangkan risiko dan biaya yang terkait dengan perangkat lunak tradisional. Dengan arsitektur terbuka dan pembaruan otomatisnya, *Sales Cloud* menghilangkan biaya tersembunyi dan implementasi perangkat lunak CRM tradisional yang berlarut-larut. Dengan terus berinovasi dan memanfaatkan

teknologi seperti seluler, kolaborasi, dan kecerdasan sosial, *Sales Cloud* terus unggul dalam persaingan.

**Layanan *Cloud*: Pengetahuan sebagai Layanan**

*Service Cloud* mengacu pada modul layanan (seperti dalam layanan pelanggan) di Salesforce. com. Ini mencakup Akun, Kontak, Kasus, dan Solusi. Ini juga mencakup fitur-fitur seperti basis pengetahuan publik, *web-to-case*, call center, dan portal swalayan, serta otomatisasi layanan pelanggan. Service *Cloud* menyertakan fitur pelacakan kasus seperti pusat panggilan dan plug-in jejaring sosial untuk percakapan dan analitik.

*Service Cloud* memberikan basis pengetahuan tingkat perusahaan pertama di dunia untuk berjalan sepenuhnya pada platform *cloud* multitenant yang canggih. Itu berarti seseorang bisa mendapatkan semua manfaat komputasi awan yang dikenal oleh Salesforce.com tanpa pusat data atau perangkat lunak yang mahal. Hanya manajemen pengetahuan yang kuat, tanpa kerumitan perangkat lunak lokal, disediakan. Tidak seperti aplikasi yang berdiri sendiri, basis pengetahuan ini sepenuhnya terintegrasi dengan yang lainnya. *Service Cloud* menawarkan semua alat yang dibutuhkan untuk menjalankan seluruh operasi layanan. Ketika basis pengetahuan konsumen menjadi bagian inti dari solusi CRM, pengetahuan sebagai proses dapat dikelola. Seseorang dapat terus membuat, meninjau, menyampaikan, menganalisis, dan meningkatkan pengetahuan. Dan, karena dikirimkan oleh *Service Cloud*, pengetahuan pengguna tersedia di mana pun pelanggan lain membutuhkannya. Agen memiliki jawaban yang tepat di ujung jari mereka untuk berkomunikasi melalui telepon, mengirim melalui email, atau berbagi melalui klien obrolan. Basis pengetahuan yang sama menyajikan jawaban atas situs web layanan yang merupakan bagian dari situs publik perusahaan. Jika seseorang ingin memanfaatkan saluran sosial seperti Twitter atau Facebook, seseorang dapat dengan mudah berbagi pengetahuan yang memanfaatkan kebijaksanaan orang banyak untuk menangkap ide atau jawaban baru. Semua ini dilakukan dengan aman.

*Service Cloud* memberikan alat yang diperlukan untuk mengelola pengetahuan pada skala perusahaan. Tapi itu juga memberikan kemudahan penggunaan yang sama seperti Salesforce. com terkenal. Itu berarti pengguna akan mendapat manfaat terlepas dari ukuran atau seberapa rumit bisnisnya.

## 11.9 RACKSPACE

Rackspace *Cloud*, bagian dari Rackspace, adalah pemain lain di pasar *cloud computing*. Menawarkan IaaS kepada klien, telah digunakan oleh sejumlah besar perusahaan. Rackspace *Cloud* menawarkan tiga solusi komputasi awan—*Server Cloud, File Cloud*, dan Situs *Cloud*. Server *Cloud* menyediakan daya komputasi sesuai permintaan dalam hitungan menit; Situs *Cloud* untuk hosting web yang tangguh dan dapat diskalakan, dan File *Cloud* untuk penyimpanan file online yang elastis dan pengiriman konten. *Cloud* Servers merupakan implementasi IaaS dimana kapasitas komputasi disediakan sebagai mesin virtual yang dijalankan dalam sistem *Cloud Servers.* Instans mesin virtual dikonfigurasikan dengan jumlah kapasitas yang berbeda. Contoh datang dalam rasa dan gambar yang berbeda. Flavor adalah konfigurasi perangkat keras yang tersedia untuk server. Setiap jenis memiliki kombinasi ruang

disk, kapasitas memori, dan prioritas waktu CPU yang unik. Serangkaian instance yang bervariasi tersedia untuk dipilih pengguna.

Mesin virtual ini dibuat menggunakan gambar. Gambar adalah kumpulan file yang digunakan untuk membuat atau membangun kembali server. Berbagai image sistem operasi prebuilt disediakan oleh Rackspace *Cloud* (distribusi Linux 64-bit— Ubuntu, Debian, Gentoo, CentOS, Fedora, Arch, dan Red Hat Enterprise Linux) atau Windows Images (Windows Server 2008 dan Windows Server 2003). Gambar-gambar ini dapat disesuaikan dengan pilihan pengguna untuk membuat gambar khusus.

Sistem *Server Cloud* divirtualisasi menggunakan Xen Hypervisor untuk Linux dan Xen Server untuk Windows. Mesin virtual yang dihasilkan datang dalam berbagai ukuran dan diukur berdasarkan jumlah memori fisik yang dicadangkan. Saat ini, memori fisik dapat bervariasi dari 256 MB hingga 15,5 GB. Jika tersedia daya CPU ekstra, Rackspace *Cloud* mengklaim akan menyediakan daya pemrosesan ekstra untuk beban kerja yang sedang berjalan, tanpa biaya.

Jadwal cadangan dapat dibuat untuk menentukan kapan harus membuat gambar server. Ini adalah fitur yang berguna, yang memungkinkan pengguna untuk terus bekerja jika terjadi kegagalan dengan menggunakan gambar cadangan. Gambar khusus sangat membantu dalam membuat jadwal pencadangan. Jenis gambar, yang disebut sebagai gambar server emas, dapat dihasilkan jika server dari konfigurasi tersebut akan sering dibuat instance-nya. Server *Cloud* dapat dijalankan melalui Rackspace *Cloud* Control Panel (GUI) atau secara terprogram melalui *Cloud* Server API menggunakan antarmuka RESTful. Panel kontrol menyediakan fungsi penagihan dan pelaporan serta menyediakan akses ke materi pendukung termasuk sumber daya pengembang, basis pengetahuan, forum, dan obrolan langsung. *Cloud* Servers API merupakan sumber terbuka di bawah lisensi Creative Commons Attribution 3.0. Pengikatan bahasa melalui bahasa tingkat tinggi seperti C++, Java, Python, atau Ruby yang mematuhi spesifikasi Rackspace akan dianggap sebagai pengikatan yang disetujui Rackspace. Instance mesin virtual diautentikasi di API dengan protokol berbasis token yang menggunakan HTTP x-Header. Kunci privat/publik digunakan untuk memastikan Akses Shell Aman.

Server *Cloud* menskalakan secara otomatis untuk menyeimbangkan beban. Proses ini otomatis dan dimulai dari Rackspace *Cloud* Control Panel atau *Cloud* Server API. Jumlah untuk skala ditentukan; Server *Cloud* untuk sementara dimatikan; RAM, ruang disk, dan alokasi CPU disesuaikan; dan server dihidupkan ulang. Server *Cloud* dapat dibuat untuk bertindak sebagai penyeimbang muatan menggunakan paket-paket sederhana yang tersedia dari repositori distribusi mana pun. Rackspace *Cloud* sedang mengerjakan versi beta dari produk *Cloud* Load Balancing, yang menyediakan solusi load balancing yang lengkap. Server *Cloud* disediakan penyimpanan persisten melalui penyimpanan disk RAID10; dengan demikian, persistensi data diaktifkan mengarah ke fungsi yang lebih baik.

### 11.10 VMware

VMware, pemimpin dalam teknologi virtualisasi, telah menghadirkan solusi komputasi *cloud* perusahaan. Setelah menjadi pemain yang mendominasi dalam domain virtualisasi, VMware saat ini menyediakan berbagai produk untuk pengembangan *private* dan *public cloud*

dan untuk memanfaatkan layanan yang ditawarkan oleh keduanya sebagai hybrid *cloud*, seperti VMware vCloud Director, VMware vCloud Datacenter Layanan, VMware vSphere, dan VMware vShield untuk beberapa nama.

*Cloud* pribadi memungkinkan penggunaan dan pengelolaan infrastruktur TI internal yang lebih baik daripada metode tradisional. Efisiensi operasional yang lebih besar, lingkungan komputasi yang aman, toleran terhadap kesalahan, dan dikelola dengan baik dapat dimodelkan dan dioperasikan. Penawaran *cloud* pribadi VMware memberikan standarisasi yang lebih besar, penyediaan yang cepat, dan layanan mandiri untuk semua aplikasi dan penghematan biaya yang tak tertandingi dengan mengkonsolidasikan infrastruktur fisik mereka. Teknologi modular VMware memungkinkan pengguna untuk memilih dari berbagai perangkat keras, perangkat lunak, dan penyedia layanan bersertifikat untuk menghasilkan komputasi awan yang efisien. Dengan demikian, rangkaian produk yang ditawarkan oleh VMware mempromosikan kompatibilitas dan mempertahankan kebebasan pilihan bagi pengguna untuk mendapatkan layanan yang diinginkan.

*Cloud* pribadi dapat dibuat dengan menggunakan VMware vSphere dan VMware vCloud Director. VMware vSphere adalah platform virtualisasi tangguh yang digunakan untuk mengubah infrastruktur TI menjadi penyimpanan virtual, komputasi, dan sumber daya jaringan serta menyediakannya sebagai layanan dalam organisasi. VMware vSphere menyediakan layanan di tingkat infrastruktur dan aplikasi. Di tingkat infrastruktur, ini menyediakan opsi untuk menjalankan operasi dan manajemen sumber daya komputasi, penyimpanan, dan jaringan yang efisien. Pada level aplikasi, kontrol level layanan disediakan untuk aplikasi yang berjalan pada infrastruktur dasar, yang mengarah ke aplikasi yang tersedia, aman, dan dapat diskalakan.

VMware vCloud Director, digabungkan dengan VMware vSphere, adalah solusi perangkat lunak yang memungkinkan perusahaan untuk membangun private *cloud* multitenant yang aman dengan menggabungkan sumber daya infrastruktur ke dalam pusat data virtual dan memaparkannya kepada pengguna melalui portal berbasis web dan antarmuka terprogram secara penuh. otomatis, layanan berbasis katalog. VMware vClou*d* Director mengabstraksi lingkungan komputasi virtual dari sumber daya yang mendasarinya dan menyediakan arsitektur multitenant yang menampilkan sumber daya virtual yang terisolasi, autentikasi LDAP independen, kontrol kebijakan khusus, dan katalog unik. Teknologi VMware vShield digunakan untuk memberikan keamanan pada lingkungan ini dengan menggunakan layanan seperti perlindungan perimeter, firewall tingkat port, layanan NAT dan DHCP, VPN situs-ke-situs, isolasi jaringan, dan penyeimbangan beban web. Direktur VMware v*Cloud* memungkinkan pengguna untuk membuat katalog infrastruktur dan layanan aplikasi dari konfigurasi yang diinginkan dan menerapkan serta menggunakannya sesuai kebutuhan. Interaksi dengan pusat data virtual atau katalog dilakukan melalui portal web yang mudah digunakan atau v*Cloud* API. v*Cloud* API adalah API berbasis REST terbuka yang menyediakan akses skrip, sesuai dengan format virtualisasi terbuka (OVF). API dapat digunakan bersama dengan VMware vCenter Orchestrator untuk mengotomatisasi dan mengatur proses operasional seperti tugas rutin, aktivitas, dan alur kerja.

Solusi *cloud* publik dan hybrid disediakan oleh VMware dengan bermitra dengan perusahaan lain, yang disertifikasi sebagai penyedia layanan. Layanan VMware vCloud Datacenter dan VMware vCloud Express menawarkan solusi efisien untuk memanfaatkan IaaS baik sebagai *cloud* publik maupun *cloud* hybrid. Layanan vCloud Datacenter menyediakan lingkungan yang dapat diskalakan, di mana sumber daya internal ditambah dengan sumber daya eksternal. Layanan vCloud Datacenter dibangun di atas teknologi dan fondasi yang sama dengan VMware vCloud Director dan VMware vSphere untuk memungkinkan interoperabilitas antar lingkungan *cloud*. Dengan demikian, pengguna bebas untuk meledakkan *cloud* pribadinya ke *cloud* publik dari penyedia layanan pilihannya.

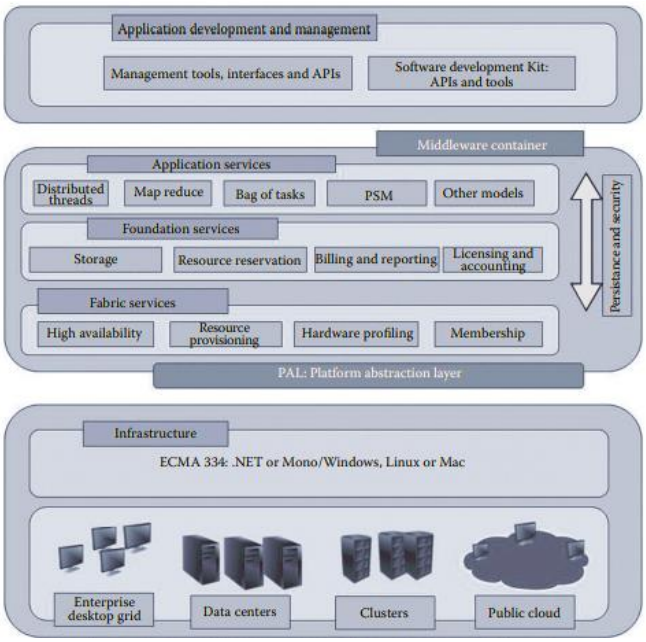
vCloud Express adalah penawaran IaaS yang disampaikan oleh mitra penyedia layanan VMware terkemuka. Ini adalah layanan merek bersama yang menyediakan infrastruktur yang andal, sesuai permintaan, dan bayar sesuai penggunaan. Penyedia VMware vCloud Express adalah Virtacore vCloud Express, Hosting.com, Melbourne IT, dan vCloud Express dari Terremark. Jenis instans, penyeimbangan muatan, opsi penyimpanan, dan harga berbeda-beda di antara penyedia layanan.

## 11.11 MANJRASOFT

Manjrasoft adalah salah satu penyedia layanan *cloud* nonutama. Namun telah hadir sebuah platform bernama Aneka yang menyediakan serangkaian layanan yang membantu pengembangan aplikasi dengan cara yang lebih mudah. Manjrasoft mengembangkan platform komputasi awan berorientasi pasar yang memungkinkan seseorang membangun, mempercepat, dan mengelola aplikasi yang pada akhirnya menghemat waktu dan uang, yang mengarah pada peningkatan produktivitas dan keuntungan bisnis.

**Platform Aneka**

Aneka menyediakan serangkaian layanan yang membuat konstruksi *cloud* perusahaan dan pengembangan aplikasi semudah mungkin tanpa mengorbankan fleksibilitas, skalabilitas, keandalan, dan ekstensibilitas.




**Gambar 11.5** Tinjauan platform Aneka.

Gambar 11.5 memberikan gambaran tentang platform Aneka. Fitur utama yang didukung oleh Aneka adalah sebagai berikut:

1. Platform (kontainer) eksekusi yang dapat dikonfigurasi dan fleksibel yang memungkinkan layanan plug-gable dan implementasi keamanan. Berbagai mekanisme autentikasi/otorisasi seperti keamanan berbasis peran dan autentikasi berbasis domain Windows dipertimbangkan untuk tujuan ini.
2. Beberapa opsi persistensi termasuk *Relational Database Management System* (RDBMS), *Structured Query Language* (SQL) *Express*, MySQL, dan file datar.
3. Kit pengembangan perangkat lunak (SDK) yang mendukung beberapa model pemrograman termasuk model thread berorientasi objek, model tugas untuk aplikasi lawas, dan model MapReduce untuk aplikasi intensif data.
4. Alat khusus seperti Design Explorer untuk studi sapuan parameter.
5. Alat manajemen yang mudah digunakan untuk negosiasi SLA dan Kualitas Layanan (QoS) dan alokasi sumber daya dinamis.
6. Mendukung penerapan aplikasi di *cloud* pribadi atau publik selain integrasinya yang mulus.

Aneka memungkinkan server dan PC desktop dihubungkan bersama untuk membentuk infrastruktur komputasi yang sangat kuat. Hal ini memungkinkan perusahaan menjadi hemat energi dan menghemat uang tanpa berinvestasi di sejumlah komputer untuk menjalankan aplikasi kompleks mereka.

Setiap node Aneka terdiri dari wadah yang dapat dikonfigurasi yang mencakup layanan informasi dan pengindeksan, penjadwalan, eksekusi, dan penyimpanan. Aneka mendukung berbagai model pemrograman, keamanan, persistensi, dan protokol komunikasi.

## 11.12 RINGKASAN

Pada bab ini, kita telah membahas tentang berbagai perusahaan yang mendukung *cloud computing* dengan menyediakan alat dan teknologi untuk beradaptasi dengan lingkungan *cloud*. Setiap bagian menjelaskan secara singkat fitur *cloud* yang didukung di perusahaan ini. Beberapa layanan seperti Google Docs dan Google *Cloud* Print gratis, sedangkan AWS, Microsoft, dll., adalah hak milik. Berdasarkan persyaratan khusus, pengguna harus melakukan trade-off antara alat/layanan open source dan closed source. Upaya telah dilakukan untuk membuat daftar alat/layanan yang ditawarkan oleh masing-masing perusahaan pada Tabel 11.1.

Meskipun ada sejumlah perusahaan, kami telah memilih beberapa perusahaan yang telah berkembang pesat di bidang ini. Tabel 11.2 memberikan informasi tentang beberapa penyedia dan harga per jam beserta model layanan dan model penyebaran yang ditawarkan oleh sistem tersebut. Rincian lebih lanjut mengenai perusahaan dapat dilihat di bagian referensi.

**Tabel 11.1** Alat dan Layanan yang Ditawarkan oleh Perusahaan

| **Nama perusahaan** | **Alat/Layanan** |
|---|---|
| EMC | Perangkat Captiva Cloud |

| | |
|---|---|
| Google | Google App Engine, Google Docs, Google Cloud Connect Google Cloud Print |
| Amazon | Amazon EC2, Amazon S3, Amazon SQS |
| Microsoft | Perangkat Penilaian dan Perencanaan Microsoft, Windows Azure Sharepoint |
| IBM | IBM Smart Cloud |
| Tenaga penjualan | Cloud Sales, Cloud Service |
| SAP LABS | SAP HANA Cloud |
| VMware | vCloud |
| Manjrasoft | Platform Aneka |
| topi merah | OpenShift Enterprise, Asal OpenShift |
| Gigaspace | Cloudify |

**Tabel 11.2** Detail Penyedia Layanan Cloud

| Nama Penyedia | Model Layanan | Model Penerapan | Sistem Operasi Server |
|---|---|---|---|
| Layanan Web Amazon | IaaS | Publik | Windows, Linux |
| Mesin Aplikasi Google | PaaS | Publik | Windows |
| Windows Azure | IaaS | Publik | Windows, Linux |
| Awan IBM | IaaS | Pribadi, hibrida | Windows, Linux |
| Platform Tenaga Penjualan | PaaS | Publik | Windows, Linux |
| Ruang rak | IaaS | Publik, pribadi, hibrida | Windows, Linux |
| SAP HANA Cloud | PaaS | Publik | Linux |

**Tinjau Poin**

- MDW: Ini adalah titik masuk di toolkit Captiva *Cloud*. Modul ini membantu mengimpor dokumen dari folder/repositori tertentu.
- REST: Ini adalah gaya arsitektur dalam sistem terdistribusi. Google *Cloud* Storage adalah layanan web penyimpanan online RESTfull.
- AMI: Ini adalah jenis alat virtual khusus yang digunakan untuk membuat mesin virtual di Amazon EC2.
- RRS: Ini adalah solusi penyimpanan baru dari Amazon S3 yang memungkinkan penyimpanan data nonkritis yang hemat biaya dengan mengurangi redundansi. Fitur ini memberikan solusi yang lebih baik dalam hal penyimpanan untuk analitik data di Amazon SQS.
- MAT: Ini adalah alat yang disediakan oleh Microsoft Azure yang membahas pertimbangan migrasi termasuk server aplikasi, database, integrasi, keamanan, dan instrumentasi untuk platform yang berbeda.

**Latihan Soal**

1. Apa yang dimaksud dengan penyedia layanan cloud? Manakah penyedia layanan cloud utama?
2. Sebutkan alat/layanan yang disediakan oleh Microsoft dan jelaskan secara singkat.
3. Apa itu Google Cloud Print? Apa kelebihannya?
4. Jelaskan SAP HANA Cloud secara singkat.
5. Apa saja layanan yang ditawarkan EMC IT? Menjelaskan.
6. Jelaskan layanan yang disediakan oleh IBM SmartCloud.
7. Apa saja layanan dukungan yang ditawarkan oleh Amazon Web Services? Menjelaskan.
8. Apa yang Anda maksud dengan Pengetahuan sebagai Layanan? Perusahaan mana yang menyediakan layanan ini? Menjelaskan.
9. Jelaskan fitur Aneka.
10. Apa itu vCloud? Jelaskan secara singkat.

# BAB 12
# OPEN SOURCE UNTUK *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan utama dari bab ini adalah untuk memberikan gambaran tentang dukungan open source untuk komputasi awan. Setelah membaca bab ini, Anda akan

- Memahami perbedaan antara alat sumber terbuka dan sumber tertutup
- Ketahui kelebihan alat sumber terbuka dibandingkan sumber tertutup
- Memahami berbagai alat sumber terbuka untuk Infrastruktur sebagai Layanan (IaaS), Platform sebagai Layanan (PaaS), Perangkat Lunak sebagai Layanan (SaaS), dll.
- Mengetahui arsitektur alat yang tersedia
- Memahami fitur yang didukung oleh alat

**Pengantar**

Bab ini memberikan ikhtisar berbagai alat sumber terbuka yang mendukung komputasi awan. Kami mulai dengan pengenalan alat sumber terbuka dan keunggulannya dibandingkan alat sumber tertutup. Bagian selanjutnya memberikan gambaran singkat tentang alat yang disediakan untuk IaaS, PaaS, dan SaaS, simulator untuk penelitian, dan komputasi terdistribusi untuk mengelola sistem terdistribusi. Subbagian di masing-masing bagian ini menjelaskan arsitektur dan fitur alat. Kami fokus untuk memberikan gambaran singkat tentang berbagai alat yang tersedia sebagai sumber terbuka. Setelah akhir bab ini, pembaca dapat memperoleh pengetahuan tentang berbagai alat dan membuat keputusan yang tepat untuk memilihnya.

## 12.1 PENDAHULUAN

Komputasi awan adalah salah satu teknologi paling populer saat ini. Menurut National Institute of Standards and Technology (NIST), komputasi awan adalah model penyediaan layanan cepat sesuai permintaan melalui Internet dalam bentuk komputasi, jaringan, penyimpanan, dan aplikasi dengan upaya manajemen minimal. Karakteristik *cloud computing* adalah on-demand self-service, akses jaringan yang luas, *resource pooling*, elastisitas yang cepat, dan layanan yang terukur. Untuk mendukung komputasi awan, berbagai alat tersedia secara gratis dan ada perangkat lunak berpemilik. Dukungan sumber terbuka untuk komputasi awan merupakan langkah perkembangan utama untuk pertumbuhan pesat teknologi awan.

***Open Source* di *Cloud computing*: Gambaran Umum**

*Open source* mengacu pada program atau perangkat lunak di mana kode sumber (bentuk program ketika programmer menulis program dalam bahasa pemrograman tertentu) tersedia untuk masyarakat umum untuk digunakan dan/atau dimodifikasi dari desain aslinya. gratis. Kode sumber terbuka biasanya dibuat sebagai upaya kolaboratif di mana pemrogram meningkatkan kode dan berbagi perubahan dalam komunitas. Sumber terbuka gratis sedangkan perangkat lunak berpemilik dimiliki dan dikendalikan secara pribadi. Dalam industri komputer, kepemilikan dianggap kebalikan dari keterbukaan. Desain atau teknik berpemilik adalah salah satu yang dimiliki oleh perusahaan. Ini juga menyiratkan bahwa perusahaan

belum membocorkan spesifikasi yang memungkinkan perusahaan lain untuk menduplikasi produk tersebut.

Dalam lingkungan komputasi awan, dukungan open source telah memunculkan banyak inovasi dengan menyediakan berbagai hal sebagai layanan, yaitu X sebagai Layanan, di mana X dapat berupa Perangkat Lunak, Platform, Infrastruktur, dll.

**Perbedaan antara Sumber Terbuka dan Sumber Tertutup**

Beberapa perangkat lunak memiliki kode sumber yang tidak dapat dimodifikasi oleh siapa pun kecuali orang, tim, atau organisasi yang membuatnya dan mempertahankan kontrol eksklusif atasnya. Perangkat lunak semacam ini sering disebut perangkat lunak berpemilik atau perangkat lunak sumber tertutup, karena kode sumbernya adalah milik pembuat aslinya, yang merupakan satu-satunya yang diizinkan secara hukum untuk menyalin atau memodifikasinya. Microsoft Word dan Adobe Photoshop adalah contoh perangkat lunak berpemilik. Untuk menggunakan perangkat lunak berpemilik, pengguna komputer harus setuju (biasanya dengan menandatangani lisensi yang ditampilkan saat pertama kali menjalankan perangkat lunak ini) bahwa mereka tidak akan melakukan apa pun dengan perangkat lunak yang tidak diizinkan oleh pembuat perangkat lunak secara eksplisit.

Perangkat lunak sumber terbuka berbeda. Pembuatnya membuat kode sumbernya tersedia bagi orang lain yang ingin melihat kode itu, menyalinnya, mempelajarinya, mengubahnya, atau membagikannya. LibreOffice dan Program Manipulasi Gambar GNU adalah contoh perangkat lunak sumber terbuka. Seperti yang mereka lakukan dengan perangkat lunak berpemilik, pengguna harus menerima persyaratan lisensi saat mereka menggunakan perangkat lunak sumber terbuka, tetapi persyaratan hukum lisensi sumber terbuka berbeda secara dramatis dari lisensi berpemilik. Lisensi perangkat lunak open source mempromosikan kolaborasi dan berbagi karena memungkinkan orang lain membuat modifikasi pada kode sumber dan memasukkan kode itu ke dalam proyek mereka sendiri. Beberapa lisensi sumber terbuka memastikan bahwa siapa pun yang mengubah dan kemudian berbagi program dengan orang lain juga harus membagikan kode sumber program tersebut tanpa membebankan biaya lisensi untuk itu. Dengan kata lain, pemrogram komputer dapat mengakses, melihat, dan memodifikasi perangkat lunak open source kapan pun mereka suka selama mereka membiarkan orang lain melakukan hal yang sama saat mereka berbagi pekerjaan. Faktanya, mereka dapat melanggar ketentuan beberapa lisensi open source jika mereka tidak melakukannya.

**Keuntungan Memiliki Sumber Terbuka**

1. Dukungan pengembang yang lebih besar: Sumber terbuka sangat membantu untuk mengembangkan platform. Ini memberikan dukungan yang jauh lebih besar untuk pengembang dan memberi mereka rasa memiliki karena mereka dapat mengubah apa pun yang mereka suka.
2. Dapat disesuaikan: Dalam skenario sumber tertutup, pengembang hanya diberi opsi untuk mengubah apa yang dipilih pengembang asli, sedangkan sumber terbuka memungkinkan pengembang memiliki kontrol penuh dan menyesuaikan nuansa tampilan.

3. Lebih aman: Sumber terbuka jauh lebih transparan daripada sumber tertutup. Siapa pun dapat melihat kodenya. Memiliki ribuan orang membaca kode seseorang, bug dan kerentanan ditemukan lebih cepat dan dikirim untuk diperbaiki. Ini juga memberi tahu pengguna jika bug telah diperbaiki karena seseorang dapat memeriksa kode setelah setiap rilis.
4. Dukungan komunitas yang lebih luas: Seiring bertambahnya usia produk, pengembang asli mungkin terus maju dan berhenti berkembang, membiarkan produk menua tanpa perbaikan atau fitur baru, tetapi jika terbuka, biasanya komunitas akan mengambil alih dan terus mengerjakannya memungkinkan masa pakai produk diperpanjang jauh melampaui apa yang dimaksudkan oleh pengembang asli.

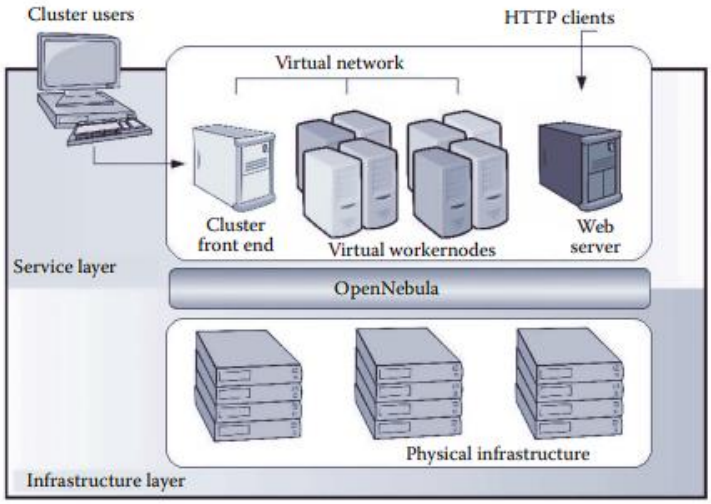
## 12.2 LAYANAN OPEN SOURCE UNTUK IAAS

IaaS adalah model penyediaan di mana organisasi mengalihdayakan peralatan yang digunakan untuk mendukung operasi, termasuk penyimpanan, perangkat keras, server, dan komponen jaringan. Penyedia layanan memiliki peralatan dan bertanggung jawab untuk menampung, menjalankan, dan memeliharanya. Klien biasanya membayar per penggunaan. Subbagian berikut menjelaskan beberapa alat sumber terbuka yang tersedia untuk menyediakan IaaS dalam komputasi awan.

**OPEN Nebula**

OpenNebula adalah alat fleksibel yang mengatur teknologi penyimpanan, jaringan, dan virtualisasi untuk memungkinkan penempatan dinamis layanan pada infrastruktur terdistribusi. OpenNebula menyediakan arsitektur modular yang dimaksudkan agar fleksibel. Salah satu modul ini adalah penjadwal, sebuah algoritme menarik yang menempatkan mesin virtual (VM) tergantung pada kebutuhannya. Komunikasi klien juga dikelola oleh modul yang menawarkan antarmuka berdasarkan layanan web. Ini mungkin satu-satunya platform manajemen terbuka yang berinvestasi ke dalam algoritme penempatan VM yang dapat disesuaikan. Dengan demikian, ini dapat memberikan lingkungan yang baik bagi para peneliti yang ingin membandingkan dan mengembangkan strategi alokasi sumber daya yang berbeda. Keterbatasan yang ditemukan dengan OpenNebula adalah, seperti XCP, infrastruktur mereka mengasumsikan arsitektur seperti cluster klasik dengan front-end dan tanpa layanan berlebihan. Gambar 12.1 menunjukkan virtualisasi infrastruktur OpenNebula.

OpenNebula adalah alat manajemen sumber terbuka yang membantu pusat data tervirtualisasi mengawasi *cloud* pribadi, *cloud* publik, dan *cloud* hybrid. Ini menggabungkan teknologi virtualisasi yang ada dengan fitur canggih untuk multitenancy, penyediaan otomatis, dan elastisitas. Manajer jaringan virtual bawaan memetakan jaringan virtual ke jaringan fisik. OpenNebula adalah vendor netral, serta platform agnostik dan antarmuka pemrograman aplikasi (API) agnostik. Itu dapat menggunakan mesin virtual berbasis kernel (KVM), Xen, atau VMware hypervisor.



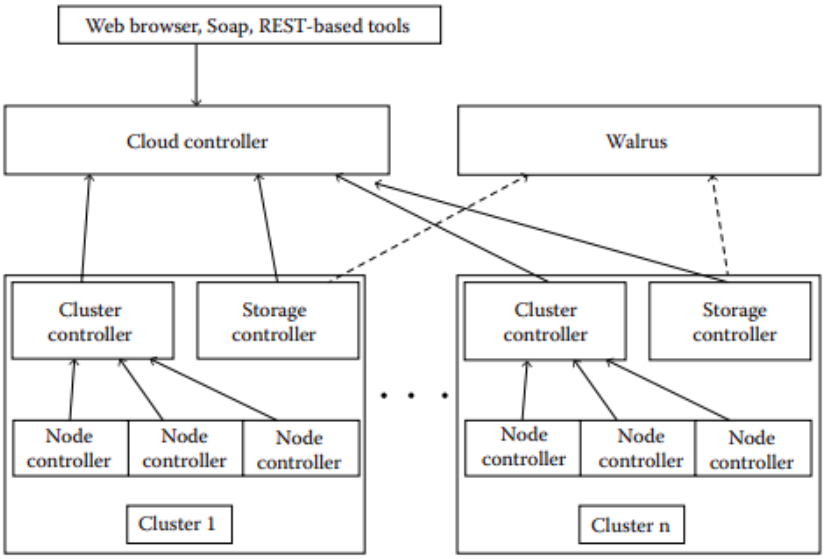

**Gambar 12.1** Virtualisasi Infrastruktur Di Opennebula

Beberapa alasan mengapa OpenNebula terutama digunakan adalah sebagai berikut:

1. Ini adalah fungsionalitas kelas perusahaan yang paling canggih dan inovatif untuk pengelolaan pusat data tervirtualisasi untuk membangun *cloud* pribadi, publik, dan hybrid.
2. OpenNebula sepenuhnya platform independen dengan dukungan luas untuk komoditas dan hypervisor tingkat perusahaan, penyimpanan, dan sumber daya jaringan, memungkinkan untuk memanfaatkan infrastruktur TI yang ada, melindungi investasi, dan menghindari vendor lock-in.
3. Ini menyediakan arsitektur, antarmuka, dan komponen yang terbuka, dapat disesuaikan, dan dapat diperluas untuk membangun layanan atau produk *cloud* yang disesuaikan.
4. Ini mendukung interoperabilitas dan portabilitas *cloud* yang memberi konsumen *cloud* pilihan lintas standar dan antarmuka *cloud* paling populer.
5. OpenNebula bukan edisi terbatas fitur atau kinerja versi perusahaan, ini benar-benar kode sumber terbuka yang didistribusikan di bawah lisensi Apache.

**Eucalyptus**

Eucalyptus mengimplementasikan *cloud* pribadi dan hybrid bergaya IaaS. Platform ini menyediakan antarmuka tunggal yang memungkinkan pengguna mengakses sumber daya infrastruktur komputasi (mesin, jaringan, dan penyimpanan) yang tersedia di *cloud* pribadi—diimplementasikan oleh Eucalyptus di dalam pusat data organisasi yang ada—dan sumber daya yang tersedia secara eksternal di layanan *cloud* publik. Perangkat lunak ini dirancang dengan arsitektur berbasis layanan web modular dan dapat diperluas yang memungkinkan Eucalyptus untuk mengekspor berbagai API ke pengguna melalui alat klien.




**Gambar 12.2** Arsitektur Eucalyptus *Cloud*.

Saat ini, Eucalyptus mengimplementasikan API Amazon Web Services (AWS) standar industri, yang memungkinkan interoperabilitas Eucalyptus dengan layanan dan alat AWS yang ada. Eucalyptus menyediakan rangkaian alat baris perintahnya sendiri yang disebut Euca2ools, yang dapat digunakan secara internal untuk berinteraksi dengan instalasi *cloud* pribadi Eucalyptus atau secara eksternal untuk berinteraksi dengan penawaran *cloud* publik, termasuk Amazon EC2. Gambar 12.2 mewakili arsitektur Eucalyptus *Cloud*. Komponen utama dalam hal ini adalah cluster controller, *cloud* controller, node controller, walrus storage controller, dan storage controller:

1. Pengontrol kluster (CC): Ia mengelola satu atau lebih pengontrol node dan bertanggung jawab untuk menerapkan dan mengelola instans pada pengontrol tersebut. Ini berkomunikasi dengan pengontrol node dan pengontrol *cloud* secara bersamaan. CC juga mengelola jaringan untuk menjalankan instans di bawah jenis mode jaringan tertentu yang tersedia di Eucalyptus.
2. Cloud controller (CLC): Ini adalah ujung depan untuk seluruh ekosistem. CLC menyediakan antarmuka layanan web yang sesuai dengan Amazon EC2/S3 ke alat klien di satu sisi dan berinteraksi dengan komponen infrastruktur Eucalyptus lainnya di sisi lain.
3. Node controller (NC): Merupakan komponen dasar untuk node. Itu mempertahankan siklus hidup dari instance yang berjalan di setiap node. Ini berinteraksi dengan OS, hypervisor, dan CC secara bersamaan.
4. Pengontrol penyimpanan Walrus (WS3): Ini adalah sistem penyimpanan file sederhana. WS3 menyimpan gambar dan snapshot mesin. Itu juga menyimpan dan melayani file menggunakan API S3.
5. Pengontrol penyimpanan (SC): Ini memungkinkan pembuatan snapshot volume. Ini menyediakan penyimpanan blok persisten melalui AoE atau iSCSI ke instans.

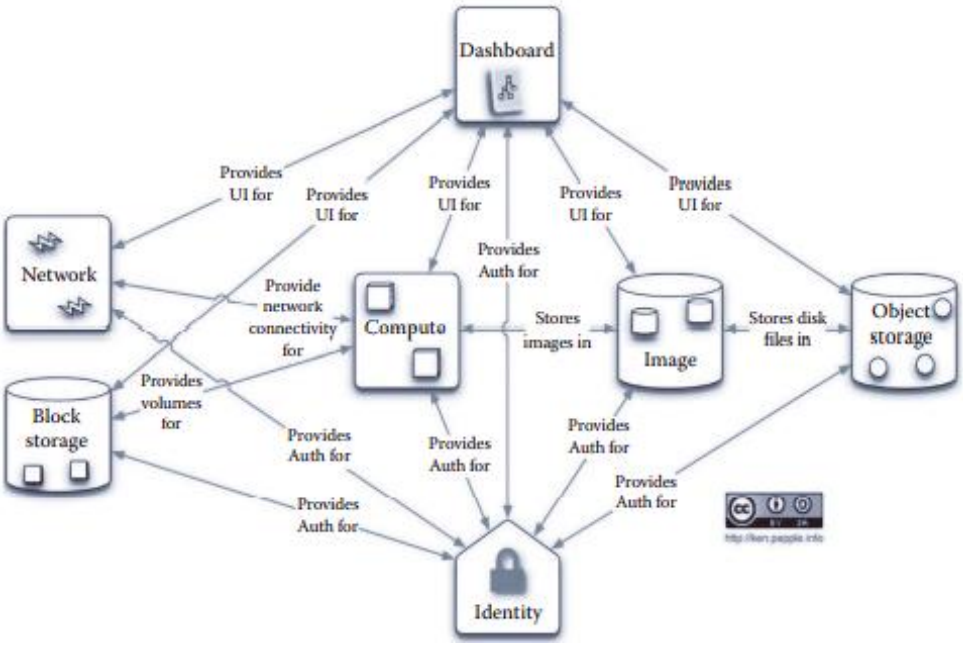
Saat mengerjakan hypervisor Xen dan KVM, Eucalyptus menyediakan solusi open source untuk mengelola infrastruktur virtual *cloud*. Penggunaan antarmuka berdasarkan layanan web

adalah salah satu karakteristik utamanya, yang memungkinkan integrasi asli dengan layanan Amazon. Selain itu, arsitektur hierarki dirancang untuk mengurangi campur tangan manusia.

**OpenStack**

OpenStack, sebuah proyek *cloud-computing*, bertujuan untuk menyediakan IaaS. Ini adalah kolaborasi global pengembang dan teknologi komputasi awan yang menghasilkan platform komputasi awan sumber terbuka di mana-mana untuk membangun awan publik dan pribadi. Ini memberikan solusi untuk semua jenis *cloud* dengan sederhana untuk diimplementasikan, dapat diskalakan secara besar-besaran, dan kaya fitur. Teknologi tersebut terdiri dari serangkaian proyek yang saling terkait yang menghadirkan berbagai komponen untuk solusi infrastruktur *cloud*.

Tujuan dari inisiatif OpenStack adalah untuk mendukung interoperabilitas antara layanan *cloud* dan memungkinkan bisnis untuk membangun layanan *cloud* seperti Amazon di pusat data mereka sendiri. OpenStack, yang tersedia secara gratis di bawah lisensi Apache 2.0, sering disebut di media sebagai Linux dari *Cloud* dan dibandingkan dengan proyek Eucalyptus dan Apache *Cloud*Stack.




**Gambar 12.3** Arsitektur konseptual OpenStack.

Gambar 12.3 menunjukkan arsitektur dan hubungan proyek OpenStack. Ada tujuh komponen inti OpenStack: komputasi, penyimpanan objek, identitas, dasbor, penyimpanan blok, jaringan, dan layanan gambar.

1. Penyimpanan objek memungkinkan pengguna untuk menyimpan atau mengambil file. Nama kode yang diberikan untuk ini cepat. Arsitektur swift sangat terdistribusi untuk mencegah satu titik kegagalan serta untuk menskalakan secara horizontal.
2. Gambar menyediakan katalog dan penyimpanan untuk gambar disk virtual. Gambar disk ini sebagian besar umum digunakan di OpenStack Compute. Nama kode yang diberikan untuk ini adalah pandangan sekilas. Sekilas melayani peran sentral untuk keseluruhan gambar IaaS. Itu menerima permintaan API untuk gambar (atau metadata

gambar) dari pengguna akhir atau komponen nova dan dapat menyimpan file disknya di layanan penyimpanan objek, cepat.

3. Compute menyediakan server virtual berdasarkan permintaan. Nama kode yang diberikan untuk ini adalah nova. Nova adalah komponen OpenStack yang paling rumit dan terdistribusi. Sejumlah besar proses bekerja sama untuk mengubah permintaan API pengguna akhir menjadi VM yang berjalan.
4. Dasbor menyediakan antarmuka pengguna (UI) berbasis web modular untuk semua layanan OpenStack. Nama kode yang diberikan untuk ini adalah horizon.
5. Identitas memberikan otentikasi dan otorisasi untuk semua layanan OpenStack. Ini juga menyediakan katalog layanan layanan dalam *cloud* OpenStack tertentu. Nama kode yang diberikan untuk ini adalah keystone. Keystone menyediakan satu titik integrasi untuk kebijakan, katalog, token, dan otentikasi OpenStack.
6. Jaringan menyediakan konektivitas jaringan sebagai layanan antara perangkat antarmuka yang dikelola oleh layanan OpenStack lainnya (kemungkinan besar nova). Layanan ini bekerja dengan mengizinkan pengguna untuk membuat jaringan mereka sendiri dan kemudian memasang antarmuka ke jaringan tersebut. Nama kode yang diberikan untuk ini adalah neutron. Neutron berinteraksi terutama dengan nova, di mana ia menyediakan jaringan dan konektivitas untuk instansnya.
7. Penyimpanan blok menyediakan penyimpanan blok persisten ke VM tamu. Nama kode yang diberikan untuk ini adalah cinder. Cinder memisahkan fungsionalitas penyimpanan blok persisten yang sebelumnya merupakan bagian dari komputasi OpenStack (dalam bentuk volume nova) ke dalam layanannya sendiri.

**Apache CloudStack**

Apache CloudStack adalah perangkat lunak sumber terbuka yang dirancang untuk menerapkan dan mengelola jaringan VM yang besar, sebagai platform komputasi awan IaaS yang sangat tersedia dan dapat diskalakan. CloudStack digunakan oleh sejumlah penyedia layanan untuk menawarkan layanan *cloud* publik dan oleh banyak perusahaan untuk menyediakan penawaran *cloud* di tempat (pribadi), atau sebagai bagian dari solusi *cloud* hybrid. CloudStack adalah solusi yang lebih baik yang mencakup hampir semua fitur yang diharapkan sebagian besar organisasi dari *cloud* IaaS. Itu dapat dicantumkan sebagai berikut:

- Hitung orkestrasi
- Jaringan sebagai Layanan
- Pengguna dan manajemen akun
- API asli penuh dan terbuka
- Akuntansi sumber daya
- UI

**Nimbus**

Nimbus adalah open source yang sangat bagus untuk pekerjaan IaaS dalam administrasi jaringan virtual. Ini didukung oleh Secure Shell (SSH) ke semua node komputasi. Performa *cloud computing* bergantung pada parameter yang berbeda seperti kecepatan CPU, jumlah memori, jaringan, dan kecepatan hard drive. Dalam lingkungan virtual, perangkat keras dibagi di antara VM. Fitur Nimbus adalah sebagai berikut:

- Operator pusat data dapat dengan mudah membangun layanan *cloud* dalam infrastruktur mereka yang ada untuk menawarkan layanan *cloud* elastis sesuai permintaan.
- Ini adalah platform perangkat lunak IaaS open source yang memungkinkan pengguna membangun, mengelola, dan menerapkan lingkungan *cloud* komputasi.

**GoGrid Cloud**

GoGrid Cloud adalah layanan hosting awan yang memungkinkan penyediaan otomatis infrastruktur virtual dan perangkat keras melalui Internet. Ini adalah penyedia IaaS yang menawarkan server virtual dan fisik, penyimpanan, jaringan, penyeimbangan beban, dan keamanan dalam waktu nyata dan di beberapa pusat data. Layanan ini diakses dan dioperasikan menggunakan protokol jaringan standar dan alamat IP melalui Internet. GoGrid Cloud juga menyediakan hosting hibrid, tempat server virtual dan fisik dapat disediakan di jaringan yang sama. Server virtual dapat dibuat dari berbagai citra server dan ukuran server, berdasarkan alokasi RAM. Gambar ini tersedia dalam berbagai ukuran—mulai dari 0,5 hingga 16 Gb RAM. GoGrid Cloud menyediakan sistem operasi Windows dan Linux standar sebagai image, dengan akses root/administrator. Gambar khusus juga dapat dibuat menggunakan GoGrid MyGSI, proses yang mudah dan semiotomatis untuk membuat, mengedit, menyimpan, dan menerapkan gambar server GoGrid (GSI). GoGrid juga bermitra dengan berbagai perusahaan, yang telah membuat rangkaian lengkap gambar server yang dibuat dan dikelola oleh komunitas mitra kami. Gambar-gambar ini dibundel dengan perangkat lunak dan aplikasi populer, memberikan solusi seperti pemulihan bencana, solusi pencadangan, dan keamanan server *cloud*. Serupa dengan Amazon EC2, GoGrid juga menyediakan kemampuan untuk menempatkan server virtual di beberapa lokasi geografis atau beberapa pusat data, memungkinkan failover atau pemulihan bencana.

Interaksi dengan infrastruktur GoGrid dilakukan melalui portal GoGrid atau API GoGrid. Portal GoGrid adalah antarmuka pengguna grafis (GUI) berbasis web yang memungkinkan pengguna untuk menyediakan atau memesan server *cloud* (virtual) dan khusus (fisik), penyeimbangan beban perangkat keras F5, penyimpanan *cloud*, jaringan pengiriman konten, dan gambar server khusus di waktu sebenarnya. API GoGrid adalah antarmuka kueri seperti representational state transfer (REST). Komunikasi didasarkan pada metode permintaan HTTP seperti GET atau POST. API sumber terbuka mendukung Java, PHP, Python, Ruby, C#, dan bahkan bahasa skrip shell seperti bash. Manajemen *cloud* dapat dilakukan dengan menggunakan mitra GSI untuk memastikan kontrol penuh atas *cloud*. Skalabilitas, keamanan, portabilitas, manajemen, dan kontrol adalah layanan inti yang disediakan oleh mitra GSI.

Penyeimbangan muatan untuk menangani lonjakan lalu lintas aplikasi diaktifkan oleh penyeimbang muatan f5 yang terintegrasi penuh dan redundan, yang dapat disediakan dari portal web atau API. Waktu henti aplikasi dicegah dengan menyebarkan lalu lintas di antara beberapa server. Keandalan failover disediakan oleh penyeimbang beban f5 dengan mengalihkan lalu lintas yang ada ke server online jika server tertentu tidak tersedia. Penyeimbang muatan f5 ini memungkinkan penskalaan cepat untuk menangani peningkatan lalu lintas yang drastis dan mempertahankan lalu lintas aplikasi.

Penyimpanan disediakan melalui GoGrid Cloud Storage, layanan pencadangan tingkat file yang dapat diskalakan dan andal secara instan untuk server cloud Windows dan Linux yang berjalan di cloud. Server ini memasang volume penyimpanan melalui jaringan pribadi yang aman dan menggunakan protokol umum untuk mentransfer data. GoGrid Cloud Storage mendukung SCP, FTP, SAMBA/CIFS, dan RSYNC. GoGrid Cloud menyediakan solusi keamanan dalam bentuk firewall perangkat keras. Terowongan jaringan pribadi virtual (VPN) digunakan untuk menyediakan akses administratif yang aman ke server cloud, sehingga melindunginya dari ancaman eksternal.

## 12.3 LAYANAN *OPEN SOURCE* UNTUK PAAS

PaaS adalah kategori layanan komputasi awan yang menyediakan platform komputasi untuk pengembangan aplikasi. Penawaran PaaS memfasilitasi penerapan aplikasi tanpa biaya dan kompleksitas pembelian dan pengelolaan perangkat keras dan perangkat lunak yang mendasarinya serta menyediakan kemampuan hosting. Ini juga dapat mencakup fasilitas untuk desain aplikasi, pengembangan, pengujian, dan penerapan. Ini juga menawarkan layanan seperti integrasi layanan web, keamanan, integrasi basis data, dan penyimpanan. Subbagian berikut memberikan ikhtisar tentang beberapa alat sumber terbuka yang menyediakan PaaS.

**Pembuat Paas**

Paasmaker adalah PaaS sumber terbuka. Ini memberikan visibilitas penuh kepada pengguna. Tindakan dipecah menjadi pohon yang dapat dipantau sepanjang waktu. Ini mendukung portabilitas dengan menawarkan antarmuka untuk beberapa bahasa umum seperti PHP, Ruby, Python, dan Node.js. Pengembangan lokal dibuat karena seseorang dapat menjalankan aplikasi dari direktori lokal. Arsitektur berbasis plug-in sangat cocok untuk memperluas plug-in. Paasmaker memungkinkan penyesuaian melalui plug-in untuk memperluas sistem dengan cara tertentu dengan mudah untuk menawarkan layanan atau runtime tambahan. Ini dirancang untuk cluster. Paasmaker mendistribusikan pekerjaan ke beberapa mesin dan memantaunya. Jika penerapan gagal, penerapan dialihkan secara otomatis. Jika node pengontrol gagal, sisa cluster tetap aktif sampai kembali.

**Asal Red Hat OpenShift**

OpenShift Origin adalah open source upstream dari OpenShift, platform hosting aplikasi generasi berikutnya yang dikembangkan oleh Red Hat. Ini juga dikenal sebagai PaaS Red Hat, OpenShift menangani infrastruktur, middleware, dan manajemen. OpenShift Origin menyertakan dukungan untuk berbagai runtime bahasa dan lapisan data termasuk Java EE6, Ruby, PHP, Python, Perl, MongoDB, MySQL, dan PostgreSQL.

Platform OpenShift Origin memiliki dua unit fungsi dasar: broker dan server node. Komunikasi antar unit ini melalui layanan antrian pesan. Broker adalah titik kontak tunggal untuk semua aktivitas manajemen aplikasi. Ini bertanggung jawab untuk mengelola login pengguna, shutdown dinamis (DNS), status aplikasi, dan orkestrasi umum aplikasi. Pelanggan tidak menghubungi broker secara langsung; sebaliknya, mereka menggunakan konsol web, alat antarmuka baris perintah (CLI), atau lingkungan pengembangan terintegrasi (IDE) JBoss Tools untuk berinteraksi dengan broker melalui API berbasis REST. Server node menghosting

kartrid bawaan yang akan tersedia bagi pengguna dan roda gigi tempat aplikasi pengguna benar-benar akan disimpan dan disajikan. Klien MCollective yang berjalan di setiap node bertanggung jawab untuk menerima dan melakukan tindakan yang diminta oleh broker. OpenShift Origin mendukung beberapa kartrid bawaan berdasarkan bahasa dan basis data pengembangan aplikasi paling populer. Agar ini berfungsi, teknologi yang mendasarinya harus diinstal pada setiap server node dalam sistem Origin. Roda gigi mewakili bagian dari CPU, RAM, dan penyimpanan dasar node yang tersedia untuk setiap aplikasi. OpenShift Origin mendukung beberapa konfigurasi roda gigi, memungkinkan pengguna untuk memilih dari berbagai ukuran roda gigi pada waktu penyetelan aplikasi. Saat aplikasi dibuat, broker menginstruksikan server node untuk membuat perlengkapan baru untuk menampungnya. Setiap kali roda gigi baru dibuat di server node, bagian CPU dan RAM dialokasikan untuknya dan struktur direktori dibuat.

**Platform Awan Xen**

Xen Cloud Platform (XCP) mengelola penyimpanan, VM, dan jaringan di *cloud*. XCP tidak menyediakan arsitektur *cloud* secara keseluruhan melainkan berfokus pada konfigurasi dan pemeliharaan *cloud*. Ini juga memungkinkan alat eksternal, termasuk Eucalyptus dan OpenNebula, untuk memanfaatkan hypervisor Xen dengan lebih baik. XCP adalah alat pengelola infrastruktur sumber terbuka untuk *cloud* yang tidak menyediakan keseluruhan arsitektur untuk komputasi *cloud* karena tidak menyediakan antarmuka bagi pengguna akhir untuk berinteraksi dengan *cloud*. Namun, XCP menyediakan lingkungan yang berguna untuk administrator dan API untuk pengembang sistem manajemen *cloud*.

**Cloudify**

Cloudify adalah PaaS pribadi open source dari GigaSpaces Technology, Inc. yang memungkinkan pengguna untuk menerapkan, mengelola, dan menskalakan aplikasi. Ini adalah sistem manajemen siklus hidup runtime perangkat lunak untuk layanan yang dihosting di *cloud*. Ini mengelola secara otomatis:

- Penyediaan infrastruktur *cloud*
- Instalasi dan konfigurasi layanan pada infrastruktur tersebut
- Memantau layanan tersebut untuk kesehatan dan parameter. Menanggapi pelanggaran perjanjian tingkat layanan (SLA).
- Penghapusan instalasi perangkat lunak layanan dan deprovisioning infrastruktur *cloud*
- Menyediakan cara yang disesuaikan untuk berinteraksi dengan sistem yang sedang berjalan yang menyembunyikan penerapannya yang tidak jelas

Cloudify dirancang untuk menghadirkan aplikasi apa pun ke perusahaan yang mendukung *cloud*, vendor perangkat lunak independen (ISV), dan penyedia layanan terkelola untuk mendapatkan manfaat cepat dari otomatisasi *cloud* dan elastisitas yang dibutuhkan organisasi saat ini. Cloudify membantu pengguna memaksimalkan onboarding dan otomatisasi aplikasi dengan mengatur deployment dan runtime aplikasi secara eksternal. Pendekatan DevOps Cloudify memperlakukan infrastruktur sebagai kode, memungkinkan pengguna untuk menjelaskan langkah penerapan dan pasca penerapan untuk aplikasi apa pun melalui cetak biru eksternal (alias, resep, yang kemudian dapat diambil dari *cloud* ke *cloud*, tidak berubah),

yaitu, Cloudify mendukung portabilitas. Cloudify menyediakan UI web, CLI, dan REST API untuk menerapkan, mengelola, dan mengonfigurasi aplikasi dengan lebih cepat.

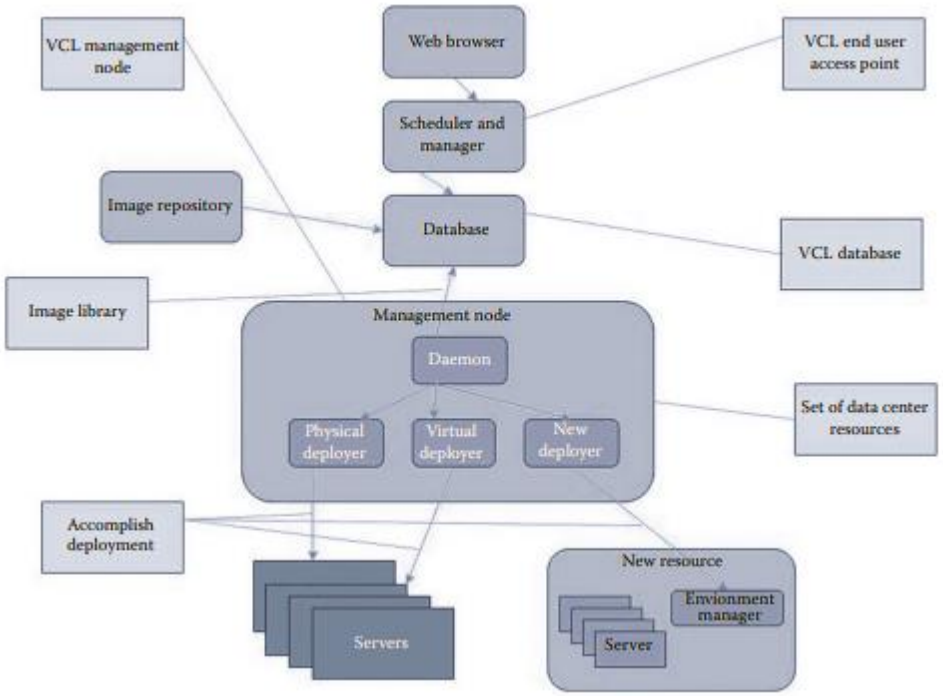
## 12.4 LAYANAN *OPEN SOURCE* UNTUK SAAS

SaaS adalah model distribusi perangkat lunak di mana aplikasi dihosting oleh vendor atau penyedia layanan dan tersedia untuk pelanggan melalui jaringan, biasanya Internet. SaaS menjadi model pengiriman yang semakin umum sebagai teknologi dasar yang mendukung layanan web dan arsitektur berorientasi layanan (SOA) yang matang dan pendekatan pengembangan baru, seperti Ajax, menjadi populer. Subbagian berikut menjelaskan secara singkat beberapa alat sumber terbuka untuk SaaS.

**Apache VCL**

VCL adalah singkatan dari lab komputasi virtual. Apache VCL adalah solusi sumber terbuka untuk akses jarak jauh melalui Internet untuk menyediakan dan mencadangkan sumber daya komputasi secara dinamis untuk beragam aplikasi, bertindak sebagai solusi SaaS. VCL memiliki arsitektur sederhana dengan empat komponen utama: web portal, database, management node, dan compute node. Gambar 12.4 menunjukkan arsitektur VCL.

Server web mewakili portal VCL dan menggunakan solusi Linux/Apache/PHP. Portal ini menyediakan UI yang memungkinkan permintaan dan pengelolaan sumber daya VCL.
Server basis data menyimpan informasi tentang reservasi VCL, kontrol akses, dan inventaris mesin dan lingkungan. Ini menggunakan solusi Linux/SQL.

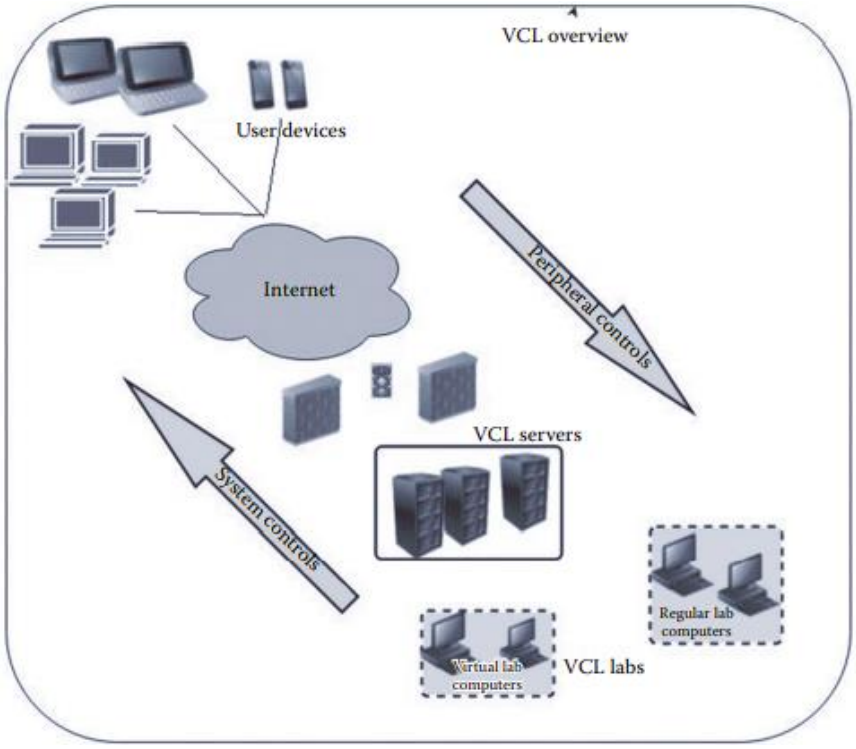



**Gambar 12.4** Arsitektur VCL

Node manajemen adalah mesin pemrosesan. Node manajemen mengontrol subset sumber daya VCL, yang mungkin berupa server blade fisik, rak tradisional, atau VM. Ini menggunakan solusi pustaka Linux/VCLD (perl)/gambar. VCLD adalah middleware yang

bertanggung jawab untuk memproses reservasi atau pekerjaan yang diberikan oleh portal web VCL. Sesuai dengan jenis lingkungan yang diminta, VCLD harus memastikan bahwa layanan (lingkungan komputasi) akan tersedia untuk pengguna.

Node komputasi mencakup server fisik, VM, mesin lab komputasi, serta sumber daya komputasi awan. Gambaran konseptual Apache VCL diberikan pada Gambar 12.5. Pengguna jarak jauh terhubung ke Aplikasi Penjadwalan VCL (portal web VCL) dan meminta akses ke lingkungan aplikasi yang diinginkan. Lingkungan aplikasi terdiri dari sistem operasi dan rangkaian aplikasi. Jenis komputer adalah server blade ruang mesin, VMware VMs, dan mesin yang berdiri sendiri.




**Gambar 12.5** Tinjauan konseptual Apache VCL.

**Google Drive**

Google Drive adalah layanan penyimpanan dan sinkronisasi file yang disediakan oleh Google yang memungkinkan penyimpanan *cloud* pengguna, berbagi file, dan pengeditan kolaboratif. File yang dibagikan secara publik di Google Drive dapat dicari dengan mesin pencari web.

Google Drive memungkinkan pengguna menyimpan dan mengakses file di mana saja — di web, di hard drive, atau saat dalam perjalanan. Ia bekerja sebagai berikut:

- Buka Google Drive di web di drive.google.com.
- Instal Google Drive di komputer atau perangkat seluler.
- Simpan file di Google Drive. Ini tersedia di perangkat dari mana ia diakses.

Setelah melakukannya, pengguna akan dapat mengakses file dari mana saja dia mau. Jika file diubah di web, dengan menggunakan komputer, atau perangkat seluler, file tersebut diperbarui di setiap perangkat yang memasang Google Drive.

**Google Dokumen**

Penawaran SaaS lain dari Google adalah Google Docs. Ini adalah salah satu dari banyak layanan berbagi dokumen komputasi awan. Sebagian besar layanan berbagi dokumen memerlukan biaya pengguna, sedangkan Google Docs gratis. Popularitasnya di kalangan bisnis semakin meningkat karena fitur berbagi dan aksesibilitasnya yang ditingkatkan.

Selain itu, Google Documents telah menikmati popularitas yang meningkat pesat di kalangan pelajar dan institusi pendidikan. Google Cloud Connect adalah plugin untuk Microsoft Office 2003, 2007, dan 2010 di Windows yang dapat secara otomatis menyimpan dan menyinkronkan dokumen Microsoft Word, presentasi PowerPoint, atau spreadsheet Excel ke Google Docs dalam format Google Docs atau Microsoft Office. Salinan Google Dokumen diperbarui secara otomatis setiap kali dokumen Microsoft Office disimpan. Dokumen Microsoft Office dapat diedit secara offline dan disinkronkan nanti saat online. Google *Cloud* Sync mempertahankan versi dokumen Microsoft Office sebelumnya dan memungkinkan banyak pengguna berkolaborasi dengan mengerjakan dokumen yang sama pada waktu yang sama.

Google Spreadsheets dan Google Sites juga menggabungkan Google Apps Script untuk menulis kode di dalam dokumen dengan cara yang mirip dengan Visual Basic for Applications (VBA) di Microsoft Office. Skrip dapat diaktifkan baik oleh tindakan pengguna atau oleh pemicu sebagai respons terhadap suatu peristiwa.

Google Forms dan Google Drawings telah ditambahkan ke suite Google Docs. Google Formulir adalah alat yang memungkinkan pengguna mengumpulkan informasi melalui survei atau kuis yang dipersonalisasi. Informasi tersebut kemudian dikumpulkan dan secara otomatis terhubung ke spreadsheet dengan nama yang sama. Spreadsheet diisi dengan survei dan respons kuis.

Google Drawings memungkinkan pengguna untuk berkolaborasi membuat, berbagi, dan mengedit gambar atau gambar. Google Drawings berisi subset fitur di Google Presentation (Google Slides) tetapi dengan template yang berbeda.

**Dropbox**

Dropbox adalah layanan hosting file yang dioperasikan oleh Dropbox, Inc. yang menawarkan penyimpanan *cloud*, sinkronisasi file, dan perangkat lunak klien. Dropbox memungkinkan pengguna untuk membuat folder khusus di setiap komputer mereka, yang kemudian disinkronkan oleh Dropbox sehingga tampak folder yang sama (dengan konten yang sama) terlepas dari komputer mana yang digunakan untuk melihatnya. File yang ditempatkan di folder ini juga dapat diakses melalui situs web dan aplikasi ponsel.

Dropbox menyediakan perangkat lunak klien untuk Microsoft Windows, Mac OS X, Linux, Android, iOS, BlackBerry OS, dan browser web, serta port tidak resmi untuk Symbian, Windows Phone, dan MeeGo.

Perangkat lunak server Dropbox dan klien desktop pada dasarnya ditulis dengan Python. Klien desktop menggunakan perangkat GUI seperti wxWidgets dan Kakao. Pustaka Python terkenal lainnya termasuk Twisted, ctypes, dan pywin32. Dropbox mengirimkan dan bergantung pada perpustakaan librsync binary-delta (yang ditulis dalam C). Klien Dropbox memungkinkan pengguna untuk meletakkan file apa pun ke dalam folder yang ditunjuk yang

kemudian disinkronkan dengan layanan Internet Dropbox dan ke komputer dan perangkat pengguna lainnya dengan klien Dropbox. Pengguna juga dapat mengunggah file secara manual melalui browser web.

Klien Dropbox mendukung sinkronisasi dan berbagi bersama dengan penyimpanan pribadi. Ini mendukung riwayat revisi, sehingga file yang dihapus dari folder Dropbox dapat dipulihkan dari komputer mana pun yang disinkronkan. Dropbox mendukung kontrol versi multipengguna, memungkinkan beberapa pengguna untuk mengedit dan mengirim ulang file tanpa menimpa versi. Riwayat versi secara default disimpan selama 30 hari, dengan versi tak terbatas yang disebut Pack-Rat tersedia untuk dibeli.

Dropbox juga menyediakan teknologi yang disebut sinkronisasi LAN, yang memungkinkan komputer di jaringan area lokal untuk mengunduh file secara aman dari satu sama lain secara lokal alih-alih selalu masuk ke server pusat.

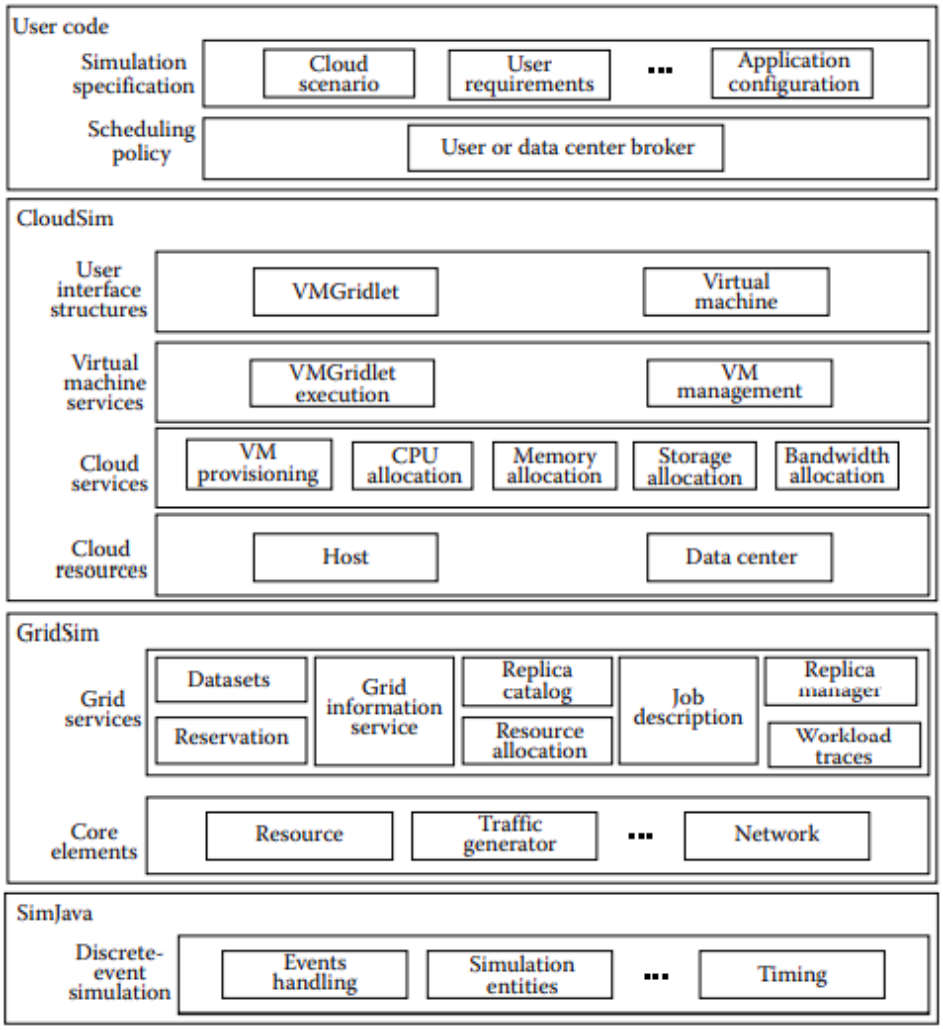
## 12.5 LAYANAN *OPEN SOURCE* UNTUK RISET

Upaya terbaru untuk merancang dan mengembangkan teknologi *cloud* berfokus pada penentuan metode, kebijakan, dan mekanisme baru untuk mengelola infrastruktur *cloud* secara efisien. Untuk menguji metode dan kebijakan yang baru dikembangkan ini, peneliti membutuhkan alat, terutama simulator yang memungkinkan mereka mengevaluasi hipotesis sebelum penerapan nyata di lingkungan tempat seseorang dapat mereproduksi pengujian. Pendekatan berbasis simulasi dalam mengevaluasi sistem komputasi awan dan perilaku aplikasi menawarkan manfaat yang signifikan, karena memungkinkan pengembang awan (i) untuk menguji kinerja kebijakan penyediaan dan pengiriman layanan mereka dalam lingkungan yang dapat diulang dan dikontrol tanpa biaya, dan (ii) untuk menyesuaikan kemacetan kinerja sebelum penerapan dunia nyata di *cloud* komersial. Beberapa alat yang dapat digunakan untuk pekerjaan penelitian dijelaskan pada subbab berikut.

**CloudSim**

CloudSim adalah toolkit yang dirancang untuk menciptakan lingkungan simulasi untuk bekerja di *cloud*. Sebagai alat yang sepenuhnya dapat disesuaikan, ini memungkinkan perluasan dan definisi kebijakan di semua komponen tumpukan perangkat lunak, yang menjadikannya cocok sebagai alat penelitian yang dapat menangani kerumitan yang timbul dari lingkungan simulasi.

Gambar 12.6 menunjukkan arsitektur berlapis dari *Cloud*Sim. Ini menjelaskan berbagai lapisan yang melibatkan UI, layanan *cloud*, dan sumber daya. Pemetaan kode pengguna ke lingkungan simulasi ditampilkan dengan jelas dalam arsitektur berlapis.

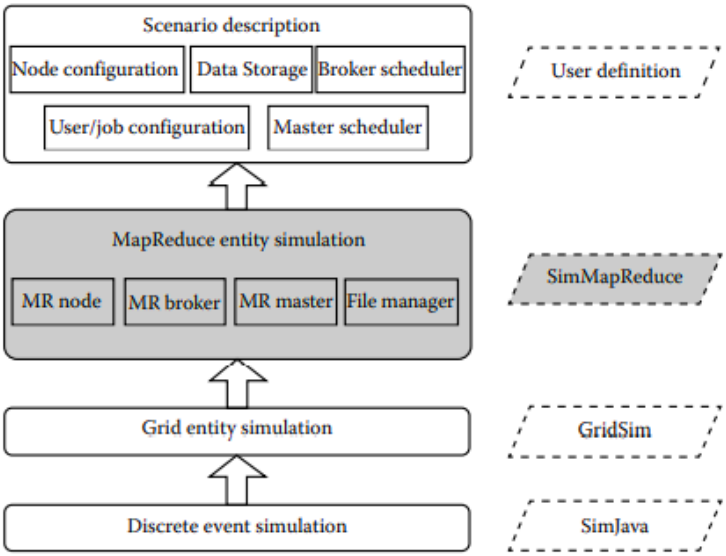



**Gambar 12.6** Arsitektur CloudSim berlapis.

**SimMapReduce**

SimMapReduce adalah simulator yang dirancang untuk menjadi toolkit yang fleksibel dan nyaman untuk diwariskan atau diwariskan oleh paket lain. SimMapReduce berusaha untuk memodelkan lingkungan MapReduce yang hidup, dengan mempertimbangkan beberapa fitur khusus seperti lokalitas data dan ketergantungan antara peta dan pengurangan, dan menyediakan layanan entitas penting yang dapat ditentukan sebelumnya dalam format XML. Selanjutnya, dengan menggunakan simulator ini, pemodel dapat mewujudkan algoritma penjadwalan multilayer pada level pengguna, level pekerjaan, atau level tugas dengan memperluas kelas yang diawetkan dengan mudah.

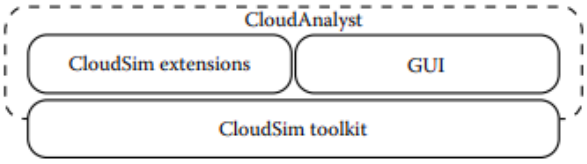
SimMapReduce sangat berguna untuk menganalisis program MapReduce di lingkungan simulasi. Gambar 12.7 memberikan diagram arsitektur SimMapReduce. Ini menjelaskan pemetaan kode pengguna dengan lingkungan simulasi.




**Gambar 12.7** Arsitektur SimMapReduce empat lapis

**Analis Awan**

Cloud Analyst adalah alat berbasis CloudSim yang dikembangkan di University of Melbourne yang bertujuan untuk mendukung evaluasi alat jejaring sosial sesuai dengan distribusi geografis pengguna dan pusat data. Dalam alat ini, komunitas pengguna dan pusat data yang mendukung jejaring sosial dicirikan berdasarkan lokasinya; parameter seperti pengalaman pengguna saat menggunakan aplikasi jejaring sosial dan beban di pusat data diperoleh/dicatat. Gambar 12.8 menunjukkan arsitektur alat *Cloud* Analyst.

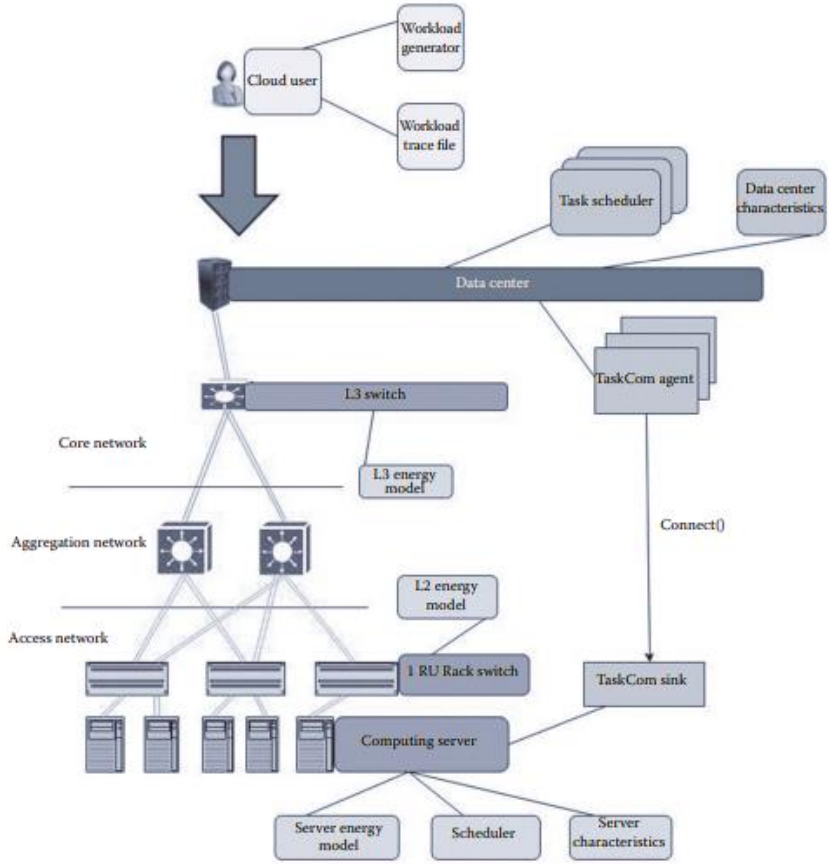



**Gambar 12.8** arsitektur CloudAnalyst.

**Awan Hijau**

GreenCloud adalah simulator tingkat paket canggih untuk pusat data komputasi awan sadar energi dengan fokus pada komunikasi awan. Ini menawarkan pemodelan mendetail dari energi yang dikonsumsi oleh peralatan TI pusat data, seperti server komputasi, sakelar jaringan, dan tautan komunikasi. Green*Cloud* dapat digunakan untuk mengembangkan solusi baru dalam pemantauan, alokasi sumber daya, penjadwalan beban kerja, serta optimalisasi protokol komunikasi dan infrastruktur jaringan. Ini dapat mensimulasikan pusat data yang ada, memandu keputusan perluasan kapasitas, serta membantu merancang fasilitas pusat data di masa depan.

GreenCloud, dirilis di bawah Perjanjian Lisensi Publik Umum, merupakan perpanjangan dari simulator jaringan NS2 yang terkenal. Sekitar 80% dari kode GreenCloud diimplementasikan dalam C++, sedangkan 20% sisanya dalam bentuk skrip tool command language (TCL). Gambar 12.9 menunjukkan arsitektur simulator GreenCloud. Ini menyajikan struktur ekstensi GreenCloud yang dipetakan ke arsitektur pusat data tiga tingkat.




**Gambar 12.9** Arsitektur simulator GreenCloud.

## 12.6 ALAT KOMPUTASI TERDISTRIBUSI UNTUK PENGELOLAAN SISTEM TERDISTRIBUSI

Sistem komputer terdistribusi terdiri dari banyak komponen perangkat lunak yang ada di banyak komputer, tetapi dijalankan sebagai satu sistem. Komputer yang berada dalam sistem terdistribusi dapat secara fisik berdekatan dan terhubung dengan jaringan lokal, atau secara geografis jauh dan terhubung dengan jaringan area luas. Sistem terdistribusi dapat terdiri dari sejumlah kemungkinan konfigurasi, seperti mainframe, komputer pribadi, workstation, dan komputer mini. Tujuan komputasi terdistribusi adalah membuat jaringan seperti itu bekerja sebagai satu komputer. Subbagian berikut secara singkat menjelaskan beberapa alat open source yang tersedia untuk komputasi terdistribusi.

**Cassandra**

Apache Cassandra adalah sistem manajemen basis data terdistribusi open source yang dirancang untuk menangani data dalam jumlah besar di banyak server komoditas, menyediakan ketersediaan tinggi tanpa titik kegagalan tunggal. Cassandra menawarkan dukungan kuat untuk klaster yang menjangkau beberapa pusat data, dengan replikasi tanpa master asinkron yang memungkinkan operasi latensi rendah untuk semua klien.

Cassandra juga menjunjung tinggi kinerja. Model data Cassandra adalah penyimpanan baris yang dipartisi dengan konsistensi yang dapat disesuaikan. Baris diatur ke dalam tabel; komponen pertama dari kunci utama tabel adalah kunci partisi; dalam sebuah partisi, baris

dikelompokkan oleh kolom kunci yang tersisa. Kolom lain dapat diindeks secara terpisah dari kunci utama.

Database Apache Cassandra adalah pilihan yang tepat ketika seseorang membutuhkan skalabilitas dan ketersediaan tinggi tanpa mengorbankan kinerja. Skalabilitas linier dan toleransi kesalahan yang telah terbukti pada perangkat keras komoditas atau infrastruktur *cloud* menjadikannya platform yang sempurna untuk data yang sangat penting. Dukungan Cassandra untuk replikasi di beberapa pusat data adalah yang terbaik di kelasnya, memberikan latensi yang lebih rendah untuk pengguna, dan tabel dapat dibuat, dihapus, dan diubah saat runtime tanpa memblokir pembaruan dan kueri. Cassandra tidak mendukung penggabungan atau subkueri, kecuali untuk analisis batch melalui Hadoop. Sebaliknya, Cassandra menekankan denormalisasi melalui fitur seperti koleksi.

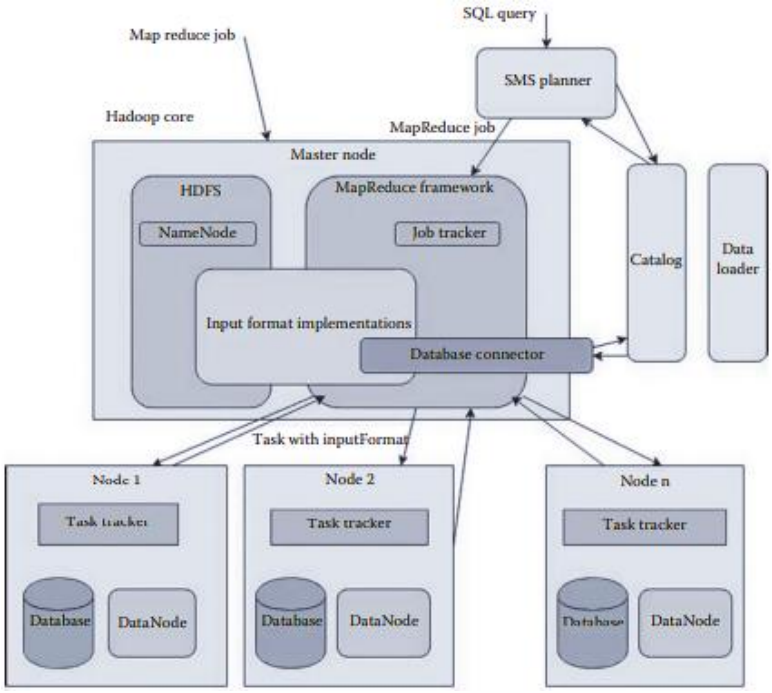
**Hadoop**

Pustaka perangkat lunak Apache Hadoop adalah kerangka kerja yang memungkinkan pemrosesan terdistribusi kumpulan data besar di seluruh kluster komputer menggunakan model pemrograman sederhana. Ini dirancang untuk meningkatkan dari satu server ke ribuan mesin, masing-masing menawarkan komputasi dan penyimpanan lokal. Daripada mengandalkan perangkat keras untuk memberikan ketersediaan yang tinggi, perpustakaan itu sendiri dirancang untuk mendeteksi dan menangani kegagalan pada lapisan aplikasi, sehingga memberikan layanan yang sangat tersedia di atas sekelompok komputer, yang masing-masing mungkin rentan terhadap kegagalan.

Kerangka kerja Apache Hadoop terdiri dari modul-modul berikut:

- Hadoop umum: Ini berisi perpustakaan dan utilitas yang dibutuhkan oleh modul Hadoop lainnya.
- Sistem file terdistribusi Hadoop (HDFS): Ini adalah sistem file terdistribusi yang menyimpan data pada mesin komoditas, menyediakan bandwidth agregat sangat tinggi di seluruh cluster.
- Hadoop YARN: Ini adalah platform manajemen sumber daya yang bertanggung jawab untuk mengelola sumber daya komputasi dalam kluster dan menggunakannya untuk penjadwalan aplikasi pengguna.
- Hadoop MapReduce: Ini adalah model pemrograman untuk pemrosesan data berskala besar.

Semua modul di Hadoop dirancang dengan asumsi mendasar bahwa kegagalan perangkat keras (dari mesin individu atau rak mesin) adalah umum dan karenanya harus secara otomatis ditangani dalam perangkat lunak oleh kerangka kerja. Komponen MapReduce dan HDFS Apache Hadoop awalnya diturunkan, masing-masing, dari kertas MapReduce dan Google File System (GFS) Google. Gambar 12.10 menunjukkan arsitektur Hadoop. Ini menjelaskan cara kerja node master dan node slave dalam kerangka kerja MapReduce. Node master memiliki pelacak pekerjaan dan node nama, sedangkan node budak memiliki pelacak tugas dan node data. Node master memetakan pekerjaan ke budak dan kemudian mengumpulkan hasilnya dan menguranginya menjadi keluaran yang diperlukan. Node budak melakukan perhitungan yang diperlukan dan memberikan hasilnya kepada master.




**Gambar 12.10** arsitektur Hadoop.

**MongoDB**

MongoDB adalah database dokumen lain yang memberikan kinerja tinggi, ketersediaan tinggi, dan skalabilitas mudah. Dokumen (objek) dipetakan dengan baik ke tipe data bahasa pemrograman. Dokumen dan larik yang disematkan mengurangi kebutuhan untuk bergabung. Skema dinamis membuat polimorfisme lebih mudah. Penyematan juga mempercepat proses baca dan tulis. Indeks dapat menyertakan kunci dari dokumen tersemat dan larik yang mencapai kinerja tinggi. Server yang direplikasi memberikan ketersediaan tinggi jika terjadi kegagalan master otomatis. Pecahan otomatis mendistribusikan data pengumpulan ke seluruh mesin memastikan skalabilitas yang mudah. Akhirnya, pembacaan yang konsisten dapat didistribusikan melalui server yang direplikasi.

MongoDB adalah proses server yang berjalan di Linux, Windows, dan OS X. MongoDB dapat dijalankan sebagai aplikasi 32-bit dan 64-bit.

**NGrid**

NGrid adalah kerangka kerja komputasi grid open source yang ditulis dalam C #. NGrid bertujuan untuk menjadi platform independen melalui proyek Mono. Ini menyediakan

- Model pemrograman multithread transparan untuk pemrograman grid
- Kerangka grid fisik dan beberapa implementasi grid
- Utilitas umum untuk pemrograman grid dan implementasi grid

NGrid, atau tepatnya NGrid.Core, adalah lapisan abstraksi kisi dua sisi untuk .Net. Sisi pertama adalah model pemrograman grid, dan sisi kedua adalah abstraksi layanan grid. NGrid.Core tidak mengencangkan ke jaringan tertentu. Dengan mengimplementasikan layanan grid, NGrid.Core dapat dihubungkan ke jaringan fisik manapun.

NGrid.Core menyediakan model pengumpulan sampah grid (GC). Dengan kata lain, objek dibuat dan hidup di grid. Saat tidak direferensikan lagi, objek dikumpulkan. Utas kisi

berjalan di atas objek kisi tersebut. Sinkronisasi dan komunikasi antara utas jaringan bekerja dengan cara yang sama seperti utas lokal.

Tujuan dari setiap implementasi grid fisik adalah untuk mengelola model grid ini seefisien mungkin. Agar masalah pengoptimalan dapat ditelusuri, NGrid.Core menyertakan beberapa metadata kode, yang dikenal sebagai atribut di C#, yang dapat digunakan untuk menghias kode klien. NGrid. Atribut penyetelan inti dapat digunakan oleh jaringan fisik untuk mengadaptasi perilaku jaringan untuk kinerja yang lebih baik. Atribut ini memiliki efek yang terbatas pada efisiensi komputasi grid. Secara khusus, mereka tidak memiliki dampak semantik: dengan atau tanpa atribut penyetelan, hasil yang dikembalikan pada akhir perhitungan grid adalah sama. Satu-satunya parameter yang mungkin berbeda adalah waktu eksekusi.

**Ganglia**

Ganglia adalah sistem pemantauan terdistribusi yang dapat diskalakan untuk sistem komputasi berperforma tinggi seperti cluster dan grid. Ini didasarkan pada desain hierarki yang ditargetkan pada federasi cluster. Ini memanfaatkan teknologi yang digunakan secara luas seperti XML untuk representasi data, XDR untuk transportasi data yang ringkas dan portabel, dan alat RRD untuk penyimpanan dan visualisasi data. Ini menggunakan struktur data dan algoritma yang direkayasa dengan hati-hati untuk mencapai overhead per-node yang sangat rendah dan konkurensi tinggi. Implementasinya kuat, telah diporting ke rangkaian sistem operasi dan arsitektur prosesor yang ekstensif, dan saat ini digunakan di ribuan klaster di seluruh dunia. Ini telah digunakan untuk menghubungkan klaster di seluruh kampus universitas dan di seluruh dunia dan dapat disesuaikan untuk menangani klaster dengan 2000 node.

## 12.7 RINGKASAN

Pada bab ini, kita telah membahas tentang berbagai alat untuk lingkungan awan yang disediakan sebagai sumber terbuka. Kami telah mengkategorikan alat ke dalam kelompok yang berbeda seperti dukungan sumber terbuka untuk IaaS, PaaS, dan SaaS, peneliti, dan komputasi terdistribusi. Di bawah setiap bagian, kami telah mencantumkan dan menjelaskan secara singkat arsitektur dan fitur dari beberapa alat untuk memberikan informasi yang berguna bagi pengguna. Tabel 12.1 memberikan informasi terkait model layanan dan model penyebaran yang ditawarkan oleh beberapa penyedia. Ini juga memberikan rincian sistem operasi server. Untuk bacaan lebih lanjut, makalah dan tautan yang diberikan di bagian referensi akan berguna.

**Tabel 12.1** Penyedia Layanan Sumber Terbuka

| Nama Penyedia | Model Layanan | Model Penerapan | Sistem Operasi Server |
|---|---|---|---|
| GoGrid | IaaS | Privasi Umum | Windows, Linux |
| Asal OpenShift | PaaS | Privasi Umum | Linux |
| OpenNebula | IaaS | Publik | Windows |
| kayu putih | IaaS | Publik, pribadi, hibrida | Windows, Linux |

| | | | |
|---|---|---|---|
| *Cloud*ify | PaaS | Pribadi | Linux |
| Paasmaker | PaaS | Pribadi | Linux |

**Tinjau Poin**

- Pengontrol kluster (CC): Ini adalah salah satu komponen utama arsitektur Eucalyptus yang mengelola pengontrol node dan membantu dalam penerapan dan pengelolaan instans pada node.
- Node controller (NC): Ini adalah komponen dasar untuk semua node yang mempertahankan siklus hidup instance yang berjalan pada setiap node.
- Cloud controller (CLC): Ini adalah ujung depan yang menyediakan antarmuka layanan web untuk klien dan juga berinteraksi dengan komponen infrastruktur Eucalyptus lainnya.
- Pengontrol penyimpanan Walrus (WS3): Ini adalah sistem penyimpanan file sederhana dari infrastruktur Eucalyptus yang menyimpan gambar mesin dan snapshot.
- Storage controller (SC): Ini adalah sistem penyimpanan yang menyediakan penyimpanan blok persisten. Ini memungkinkan pembuatan snapshot volume.
- Horizon: Ini adalah nama kode untuk dasbor dalam arsitektur OpenStack.
- Nova: Ini adalah nama kode untuk komputasi dalam arsitektur OpenStack.
- Keystone: Ini adalah nama kode untuk identitas dalam arsitektur OpenStack.
- Cinder: Ini adalah nama kode untuk penyimpanan blok dalam arsitektur OpenStack.
- Swift: Ini adalah nama kode untuk penyimpanan objek dalam arsitektur OpenStack.
- Neutron: Ini adalah nama kode untuk jaringan dalam arsitektur OpenStack.
- Sekilas: Ini adalah nama kode untuk gambar dalam arsitektur OpenStack.
- Kartrid: Kartrid mewakili komponen yang dapat dipasang yang dapat digabungkan dalam satu aplikasi.
- Roda gigi: Roda gigi mewakili potongan CPU, RAM, dan penyimpanan dasar node yang tersedia untuk setiap aplikasi.
- Resep: Resep adalah rencana eksekusi atau peristiwa siklus hidup (menginstal, memulai, mengkonfigurasi, memantau, meningkatkan dan, dll.) dari aplikasi.
- Simulator awan: Ini adalah alat yang berguna bagi peneliti untuk menguji metode dan kebijakan yang baru dikembangkan dan dengan demikian mengevaluasi hipotesis.

**Latihan Soal**

1. Sebutkan perbedaan antara alat sumber terbuka dan alat sumber tertutup.
2. Jelaskan komponen utama Eucalyptus.
3. Apa itu OpenShift Origin? Menjelaskan.
4. Jelaskan arsitektur OpenStack.
5. Jelaskan pentingnya alat open source bagi peneliti.
6. Apa perbedaan alat sumber terbuka untuk SaaS? Jelaskan salah satunya.
7. Apa itu MongoDB? Jelaskan secara rinci.
8. Jelaskan arsitektur framework Hadoop.

# BAB 13
# KEAMANAN DI *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan utama dari bab ini adalah untuk memberikan gambaran tentang masalah keamanan dalam komputasi awan. Setelah membaca bab ini, Anda akan

- Memahami aspek keamanan yang berbeda
- Memahami masalah keamanan dalam model layanan *cloud*
- Memahami manajemen identitas dan masalah kontrol akses yang berkaitan dengan keamanan
- Memahami audit dan kepatuhan dalam keamanan

**Pengantar**

Keamanan merupakan aspek penting yang harus diperhatikan dalam lingkungan *cloud computing*. Bab ini berfokus pada berbagai aspek keamanan. Kami mulai dengan pengantar keamanan *cloud*. Bagian selanjutnya berbicara tentang keamanan data; keamanan virtualisasi; masalah keamanan dalam model Perangkat Lunak sebagai Layanan (SaaS), Infrastruktur sebagai Layanan (IaaS), dan Platform sebagai Layanan (PaaS); dll. Bab ini juga mempertimbangkan tantangan privasi dan identitas serta masalah manajemen akses di *cloud*. Setelah membaca bab ini, pembaca dapat memperoleh ikhtisar tentang masalah keamanan di berbagai model layanan di *cloud*. Pembaca juga bisa mendapatkan gambaran tentang tantangan dalam masalah keamanan ini.

## 13.1 PENDAHULUAN

Komputasi awan telah memasuki kehidupan setiap orang saat ini terlepas dari teknologi atau aspek lainnya. Setiap majalah teknologi atau setiap situs web organisasi teknologi informasi (TI) berbicara tentang komputasi awan. Apa sebenarnya komputasi awan ini? Apa fungsinya untuk membuat hidup kita lebih mudah? Untuk menjawab semua pertanyaan ini, mari kita lihat detail komputasi awan sehubungan dengan teknologi dengan lebih tepat.

### *Cloud* dalam Teknologi Informasi

Komputasi awan telah merevolusi industri TI selama dekade terakhir dan masih mengembangkan cara-cara kreatif untuk memecahkan masalah saat ini. Perusahaan dan lembaga penelitian secara perlahan beralih ke *cloud* untuk memenuhi kebutuhan komputasi mereka.

Jadi, mengapa *cloud* menarik bagi sektor publik dan swasta? Untuk menjawab pertanyaan ini, mari kita lihat apa saja yang dimasukkan ke dalam *cloud* oleh sebuah organisasi. Untuk membuat *cloud* lebih ramah pengguna untuk komputasi, industri telah banyak berinvestasi dalam aspek-aspek berikut:

1. Waktu dan keuangan: *Cloud* adalah sistem terpusat dan memperbarui informasi waktu nyata. Bisnis dengan data sensitif waktu dengan cepat memanfaatkan peluang ini dan

memanfaatkan efisiensi *cloud*. Sebagai contoh, penelitian medis yang membutuhkan waktu berbulan-bulan untuk menghitung angka internal dipindahkan ke sistem terdistribusi, secara signifikan mengurangi waktu dan biaya komputasi.

2. Orang dan asosiasi: Dengan hadirnya *cloud*, kolaborasi online antara tim terdistribusi menjadi mudah. Sekarang lebih mudah untuk berkomunikasi dan bekerja dengan orang-orang yang berada di area yang berbeda, terkadang di negara yang berbeda, selama jam kerja. Tim sekarang terdiri dari anggota yang tersebar di wilayah geografis yang luas. Seperti yang telah disebutkan, kemampuan *cloud* untuk memperbarui informasi secara *real time* memungkinkan tim untuk segera mengatasi masalah. Bekerja bersama tidak lagi berarti bertemu di ruang rapat. Telepon Protokol Internet (IP), seperti Skype dan Google Hangout, menyediakan platform yang memungkinkan anggota tim untuk mendiskusikan tugas tanpa melangkah keluar dari bilik mereka.
3. Mengganti perangkat keras: Merelokasi sistem informasi dan data ke *cloud* tidak hanya menghemat uang tetapi juga mengurangi sumber daya yang terbuang. Perusahaan tidak perlu lagi membeli perangkat keras dan sistem yang memerlukan instalasi dan pemeliharaan. Pusat data di *cloud* dapat mengalokasikan kembali sumber daya ini ke klien dengan menghemat uang perusahaan hanya dengan membayar apa yang digunakan dan menghindari pembelian mesin yang tidak akan berguna dalam jangka panjang. Penyedia layanan *cloud* (CSP) juga memiliki kemampuan untuk mengoptimalkan sistem mereka untuk mengurangi pemborosan. Mereka juga memiliki kemampuan untuk meningkatkan sistem mereka sesuai dengan permintaan layanan. Ini biasanya sangat mahal untuk dilakukan oleh bisnis dan menghasilkan sumber daya yang terbuang percuma. Mesin internal yang lebih sedikit berarti bahwa perusahaan dapat mengalihkan dana untuk meningkatkan aspek lain dari operasi bisnis.
4. Hemat energi: Sebuah penelitian melaporkan bahwa klien Salesforce menghasilkan karbon 95% lebih sedikit dibandingkan perusahaan dengan sistem di tempat mereka.
5. Studi dari Accenture, Microsoft, dan WSP Environment and Energy: Studi tahun 2010 dari Accenture, Microsoft, dan WSP Environment and Energy melaporkan dampak besar awan terhadap emisi CO2. Mereka menemukan bahwa bisnis dengan sistem dan aplikasi di *cloud* dapat mengurangi jejak karbon per pengguna sebesar 30% untuk perusahaan besar dan 90% untuk bisnis kecil.
6. Going green: Greenpeace menunjukkan dalam studi baru-baru ini bahwa sementara efisiensi meningkat, sumber energinya juga beragam. Operasi internal pusat data berwarna hijau, tetapi dangkal jika sumber daya tidak terbarukan. Dengan meningkatnya permintaan komputasi awan, konsumsi energi diperkirakan akan meningkat sebesar 12% setiap tahunnya. Analisis Greenpeace menunjukkan bahwa dari 10 perusahaan teknologi terkemuka—Akamai, Amazon, Apple, Facebook, Google, HP, IBM, Microsoft, Twitter, dan Yahoo!—Akamai dan Yahoo! adalah yang paling ramah lingkungan dan Apple paling tidak. Laporan tersebut juga menyoroti upaya Google dalam menghijaukan sumber energinya.

7. Masa depan *cloud* dan lingkungan: Dua perusahaan di Islandia, Green Earth Data dan GreenQloud, keduanya mengklaim menawarkan 100% energi terbarukan dengan memberi daya pada pusat data mereka dengan sumber daya panas bumi dan tenaga air, yang melimpah di negara ini. "Internet dengan komputasi awan menjadi kontributor besar emisi karbon karena penggunaan energi yang kotor," GreenQloud bertujuan untuk memberikan contoh kepada raksasa komputasi awan dalam menciptakan layanan awan yang ramah lingkungan. Karena industri *cloud* berharap untuk tumbuh menjadi pasar Rp. 75.000.000.000 pada akhir tahun, pengguna semakin menuntut layanan ramah lingkungan. Teknologi *cloud* dengan cepat lepas landas, dan ini adalah kesempatan bagi perusahaan dan bisnis untuk memikirkan cara kreatif memanfaatkan kekuatannya sambil menyelamatkan lingkungan.

**Tantangan Umum *Cloud***

Penggunaan *cloud* memberikan sejumlah peluang seperti mengaktifkan layanan untuk digunakan tanpa memahami infrastrukturnya. Komputasi awan bekerja menggunakan skala ekonomi. Vendor dan penyedia layanan mengklaim biaya dengan membangun aliran pendapatan yang berkelanjutan. Data dan layanan disimpan dari jarak jauh tetapi dapat diakses dari mana saja.

Meskipun *cloud* adalah teknologi paling populer saat ini, ada banyak masalah yang terkait dengannya. Empat masalah utama berikut menonjol dengan komputasi awan:

1. Kebijakan ambang batas: Untuk menguji apakah program berhasil, berkembang, atau ditingkatkan, dan diterapkan, kebijakan ambang batas adalah studi percontohan sebelum memindahkan program ke lingkungan produksi. Periksa bagaimana kebijakan mendeteksi peningkatan permintaan secara tiba-tiba dan menghasilkan pembuatan instans tambahan untuk mengisi permintaan. Juga, periksa untuk menentukan bagaimana sumber daya yang tidak terpakai akan dialokasikan kembali dan dialihkan ke pekerjaan lain.
2. Masalah interoperabilitas: Masalah pencapaian interoperabilitas aplikasi antara dua vendor *cloud computing*. Kebutuhan untuk memformat ulang data atau mengubah logika dalam aplikasi.
3. Biaya Tersembunyi: *Cloud computing* tidak memberi tahu Anda apa itu biaya tersembunyi. Dalam contoh biaya jaringan yang timbul, perusahaan yang jauh dari lokasi penyedia *cloud* dapat mengalami latensi, terutama ketika ada lalu lintas yang padat.
4. Perilaku tak terduga: Pengujian yang akan dilakukan untuk menunjukkan hasil validasi yang tak terduga atau melepaskan sumber daya yang tidak terpakai. Kebutuhan untuk memperbaiki masalah sebelum menjalankan aplikasi di *cloud*.

## 13.2 ASPEK KEAMANAN

Masalah keamanan di *cloud* tidak jauh berbeda dengan penawaran layanan non*cloud* meskipun menjengkelkan—karena dalam lingkungan non-*cloud* penyewa tunggal, Anda umumnya mengetahui di mana informasi berada dan bagaimana informasi disimpan. Ada

banyak pelanggan yang berbeda dan tidak ada mekanisme yang diikuti untuk mengisolasi data satu sama lain.

Komputasi *cloud* menempatkan data bisnis ke tangan penyedia luar dan membuat kepatuhan terhadap peraturan secara inheren lebih berisiko dan lebih kompleks daripada saat sistem dikelola sendiri. Kehilangan pengawasan langsung berarti bahwa perusahaan klien harus memverifikasi bahwa penyedia layanan bekerja untuk memastikan bahwa keamanan dan integritas data sangat ketat. Berikut ini adalah bidang penelitian terkait keamanan saat ini dalam komputasi awan:

1. Aplikasi terdistribusi yang andal berbasis Internet, seperti sistem *e-commerce*, sangat bergantung pada jalur kepercayaan di antara pihak-pihak yang terlibat.
2. Permintaan yang meroket untuk generasi baru aplikasi konsumen dan bisnis berbasis *cloud* mendorong kebutuhan akan pusat data generasi berikutnya yang harus dapat diskalakan secara besar-besaran, efisien, gesit, andal, dan aman. Untuk menskalakan layanan *cloud* secara andal bagi jutaan pengembang layanan dan miliaran pengguna akhir, komputasi *cloud* generasi mendatang dan infrastruktur pusat data harus mengikuti evolusi yang mirip dengan evolusi yang menyebabkan terciptanya jaringan telekomunikasi yang dapat diskalakan.
3. Di masa mendatang, CSP berbasis jaringan akan memanfaatkan teknologi virtualisasi untuk dapat mengalokasikan tingkat yang tepat dari komputasi virtual, jaringan, dan sumber daya penyimpanan ke aplikasi individu berdasarkan permintaan bisnis real-time sementara juga menyediakan layanan penuh- tingkat jaminan ketersediaan, kinerja, dan keamanan dengan biaya yang masuk akal.

**Keamanan Data**

Karena infrastruktur yang sangat besar, organisasi biaya secara perlahan beralih ke teknologi *cloud*. Data disimpan dalam infrastruktur CSP. Karena data tidak berada di wilayah organisasi, banyak tantangan kompleks muncul. Beberapa tantangan keamanan data yang kompleks di *cloud* adalah sebagai berikut:

- Kebutuhan untuk melindungi data rahasia bisnis, pemerintah, atau peraturan
- Model layanan *cloud* dengan beberapa penyewa berbagi infrastruktur yang sama
- Mobilitas data dan masalah hukum terkait dengan aturan pemerintah seperti Petunjuk Privasi Data Uni Eropa (UE).
- Kurangnya standar tentang cara CSP mendaur ulang ruang disk dengan aman dan menghapus data yang ada
- Masalah audit, pelaporan, dan kepatuhan
- Kehilangan visibilitas terhadap keamanan utama dan intelijen operasional yang tidak lagi tersedia untuk mendukung intelijen keamanan dan manajemen risiko TI perusahaan
- Jenis orang dalam baru yang bahkan tidak bekerja untuk perusahaan Anda, tetapi mungkin memiliki kontrol dan visibilitas ke dalam data Anda

Masalah seperti itu meningkatkan tingkat kecemasan tentang risiko keamanan di *cloud*. Perusahaan khawatir apakah mereka dapat mempercayai karyawan mereka atau perlu menerapkan kontrol internal tambahan di *cloud* pribadi dan apakah penyedia pihak ketiga

dapat memberikan perlindungan yang memadai di lingkungan multitenant yang juga dapat menyimpan data pesaing.

Ada juga kekhawatiran yang sedang berlangsung tentang keamanan pemindahan data antara perusahaan dan *cloud*, serta bagaimana memastikan bahwa tidak ada sisa data sisa saat dipindahkan ke CSP lain.

Tidak diragukan lagi, lingkungan tervirtualisasi dan *cloud* pribadi melibatkan tantangan baru dalam mengamankan data, tingkat kepercayaan yang beragam, dan potensi melemahnya pemisahan tugas dan tata kelola data. Awan publik melengkapi tantangan ini dengan data yang mudah dibawa-bawa, dapat diakses oleh siapa saja yang terhubung dengan server awan, dan direplikasi untuk ketersediaan. Dan dengan *cloud* hybrid, tantangannya adalah melindungi data saat bergerak bolak-balik dari perusahaan ke *cloud* publik.

Namun, keamanan dan privasi masih disebut oleh banyak organisasi sebagai penghambat utama adopsi layanan *cloud*, yang menyebabkan pengenalan sistem enkripsi *cloud* dalam 18 bulan terakhir.

Masalah-masalah yang harus diatasi adalah sebagai berikut:

Pemberitahuan pelanggaran dan residensi data: Tidak semua data memerlukan perlindungan yang sama, sehingga bisnis harus mengkategorikan data yang ditujukan untuk penyimpanan *cloud* dan mengidentifikasi persyaratan kepatuhan apa pun sehubungan dengan pemberitahuan pelanggaran data atau jika data tidak boleh disimpan di yurisdiksi lain.

Gartner juga merekomendasikan agar perusahaan menerapkan rencana keamanan data perusahaan yang menetapkan proses bisnis untuk mengelola permintaan akses dari otoritas penegak hukum pemerintah. Rencana tersebut harus mempertimbangkan pemangku kepentingan, seperti hukum, kontrak, dan unit bisnis, keamanan, dan TI.

Manajemen data saat istirahat: Bisnis harus mengajukan pertanyaan khusus untuk menentukan siklus hidup penyimpanan data dan kebijakan keamanan CSP.

Bisnis harus mencari tahu apakah

- Penyimpanan multitenant sedang digunakan, dan jika ya, cari tahu mekanisme pemisahan apa yang digunakan antara penyewa
- Mekanisme seperti penandaan digunakan untuk mencegah replikasi data ke negara atau wilayah tertentu

Penyimpanan yang digunakan untuk arsip dan cadangan dienkripsi dan strategi manajemen utama mencakup identitas yang kuat dan kebijakan manajemen akses untuk membatasi akses dalam yurisdiksi tertentu.

Gartner merekomendasikan agar bisnis menggunakan enkripsi untuk menerapkan strategi akhir masa pakai dengan menghapus kunci untuk menghancurkan data secara digital sambil memastikan bahwa kunci tidak disusupi atau direplikasi.

Pergerakan perlindungan data: Sebagai persyaratan minimum, Gartner merekomendasikan agar bisnis memastikan bahwa CSP akan mendukung protokol komunikasi yang aman seperti Secure Socket Layer (SSL)/Transport Layer Security (TLS) untuk akses

browser atau virtual private network (VPN) koneksi berbasis untuk akses sistem untuk akses yang dilindungi ke layanan mereka.

Catatan penelitian mengatakan bahwa bisnis selalu mengenkripsi data sensitif yang bergerak ke *cloud*, tetapi jika data tidak dienkripsi saat digunakan atau disimpan, perusahaan berkewajiban untuk mengurangi pelanggaran data.

Di IaaS, Gartner merekomendasikan agar bisnis memilih CSP yang menyediakan pemisahan jaringan di antara penyewa, sehingga satu penyewa tidak dapat melihat lalu lintas jaringan yang lain.

**Keamanan Pusat Data**

Data disimpan di luar wilayah pengguna di lokasi yang disebut pusat data, yang tidak diketahui oleh pengguna. Karena lokasi pusat data tidak diketahui oleh pengguna, itu menjadi pusat data virtual. Tulang punggung pusat data virtual ini adalah infrastruktur virtual, atau mesin virtual (VM); namun, platform virtual bergantung pada banyak komponen lain yang sering terlupakan dari pusat data fisik dan virtual.

Biasanya ada tujuh bidang perhatian yang menyertai implementasi atau migrasi platform virtual utama. Seringkali, masalah ini tidak terlihat selama staging dan pengujian dan hanya muncul saat VM mengambil beban yang sama dengan mesin fisik. Titik kritis mewakili dua landasan pusat data: jaringan dan penyimpanan.

Kurangnya kinerja dan ketersediaan: Virtualisasi memindahkan banyak tugas I/O (Input/Output) yang disesuaikan untuk perangkat keras ke perangkat lunak melalui hypervisor. Lapisan terjemahan virtualisasi bertanggung jawab untuk menerjemahkan kode yang dioptimalkan untuk chip perangkat lunak ke chip fisik atau CPU yang berjalan pada perangkat keras yang mendasarinya. Aplikasi intensif I/O, seperti aplikasi pemrosesan kriptografi untuk SSL, tidak berjalan dengan baik saat divirtualisasikan karena lapisan terjemahan ini.

Kejenuhan VM yang disebabkan oleh virtualisasi sprawl dapat menyebabkan kendala sumber daya yang tidak terduga di semua bagian pusat data. Dengan mesin fisik yang menjalankan aplikasi jaringan, aplikasi tersebut dapat memiliki akses penuh ke sumber daya kartu jaringan. Ini dapat menyebabkan masalah kinerja jaringan secara keseluruhan, pengurangan bandwidth, dan peningkatan latensi; aplikasi mungkin tidak dapat menangani semua masalah ini. Bahkan masalah yang lebih kecil seperti ketersediaan alamat IP dapat dipengaruhi oleh perluasan virtualisasi.

Kurangnya kesadaran aplikasi: Salah satu keterbatasan solusi virtualisasi berbasis hypervisor dan kernel adalah bahwa mereka hanya memvirtualisasikan sistem operasi (OS). Virtualisasi OS tidak melakukan virtualisasi dan bahkan tidak mengetahui aplikasi yang berjalan di OS. Bahkan aplikasi yang sama tidak menyadari bahwa mereka menggunakan perangkat keras virtual di atas hypervisor. Dengan menyisipkan lapisan manajemen perangkat lunak tambahan ini antara aplikasi dan perangkat keras, aplikasi mungkin mengalami masalah kinerja yang berada di luar kendali.

Platform infrastruktur virtual biasanya menyertakan perangkat lunak yang dapat memigrasikan instance VM langsung dari satu perangkat fisik ke perangkat lainnya; VMware Distributed Resource Scheduler (DRS) dan VMotion adalah contoh

solusi migrasi langsung. Seperti virtualisasi OS dasar, alat migrasi ini tidak mengetahui status aplikasi dan juga tidak memiliki wawasan tentang jaringan pengiriman aplikasi.

Tambahan, biaya tak terduga: Dua pendorong utama virtualisasi adalah pengurangan biaya dan konsolidasi pusat data; namun, mengimplementasikan solusi VM lengkap di pusat data dan memigrasikan mesin fisik ke VM tidaklah murah. Setelah perangkat keras dan perangkat lunak virtualisasi diperoleh, biaya operasional dapat tumbuh tanpa batas. Pengelolaan alat baru ini dapat menjadi biaya berulang jangka panjang, terutama jika virtualisasi dilakukan secara internal. Mungkin ada persyaratan pertumbuhan tambahan untuk jaringan aplikasi dan penyimpanan karena VM ini mulai membebani infrastruktur yang ada. Biaya tak terduga dan tidak terencana dapat menjadi masalah serius saat mengimplementasikan atau bermigrasi dari fisik ke VM, menghambat atau bahkan menghentikan penerapan sepenuhnya.

Fitur virtualisasi yang tidak digunakan: Platform virtual baru menyertakan banyak teknologi jaringan canggih, seperti pengalihan perangkat lunak dan dukungan untuk segmentasi jaringan area lokal virtual (VLAN). Seringkali, teknologi baru bekerja dengan sempurna dalam pengembangan dan pementasan, tetapi mereka tidak dapat menskalakan ke tingkat produksi setelah diterapkan. Platform baru ini mungkin memiliki masalah integrasi dengan aplikasi yang ada dan jaringan penyimpanan, yang membutuhkan desain ulang pusat data.

Masalah integrasi penyimpanan cenderung muncul segera setelah VM dipindahkan ke lingkungan produksi. Pertama dan terpenting, penyimpanan jaringan merupakan persyaratan untuk platform virtual yang mengimplementasikan migrasi mesin secara langsung; terpasang langsung dan penyimpanan lokal hanya akan berfungsi untuk menjalankan VM lokal. Meskipun banyak jaringan penyimpanan perusahaan menyertakan teknologi untuk replikasi data yang menjangkau beberapa pusat data geografis, alat migrasi VM seringkali terbatas pada grup penyimpanan lokal. Jaringan penyimpanan yang meluap: Meskipun mengonversi mesin fisik ke VM merupakan aset untuk membangun pusat data dinamis, hard drive menjadi gambar disk virtual file datar yang sangat besar. Akibatnya, penyimpanan file menjadi tidak terkelola.

Jaringan penyimpanan padat: Karena sifat portabel dari virtualisasi OS, dapat terjadi peningkatan dramatis dalam data yang melintasi jaringan penyimpanan. Untuk alasan yang sama bahwa file disk VM dapat melebihi penyimpanan fisik, setelah image ini dibuat portabel, memindahkan image VM ini melintasi jaringan dari satu host ke host lain atau dari satu larik penyimpanan ke yang lain menjadi mudah. Ini bisa menjadi tantangan untuk mencegah banjir jaringan penyimpanan saat merencanakan migrasi atau pemindahan VM besar. Dan saat virtual sprawl dan VM yang tidak terkelola mulai muncul di pusat data, migrasi VM yang tidak direncanakan dapat benar-benar membuat jaringan macet, bahkan di LAN.

Kompleksitas manajemen: Di seluruh area pusat data, mengelola VM sebagai bagian dari solusi manajemen lengkap bisa menjadi perjuangan yang terbaik. VM akan

melaporkan metrik seperti latensi dan waktu respons semua mesin fisik. Tantangan manajemen dengan VM muncul dalam dua bentuk:

1. Penambahan dua komponen baru yang perlu dikelola: Hypervisor dan sistem host. Tak satu pun dari perangkat ini yang ada di dunia server fisik bukan bagian dari solusi manajemen pusat data yang ada, tetapi mereka memiliki dampak besar pada kinerja VM. Mengelola perangkat ini dan memahami perbedaan performa ini sangatlah penting.
2. Mengelola VM, jaringan aplikasi, dan jaringan penyimpanan secara bersamaan: Banyak platform VM menyertakan alat manajemen bawaan, beberapa di antaranya sangat canggih, seperti Server Virtual VMware.

Meskipun alat ini menyediakan tugas manajemen penting, seperti migrasi langsung tamu virtual dari satu host ke host lainnya, alat ini tidak memperhitungkan informasi eksternal apa pun. Dengan server fisik, ada garis yang memisahkan kepemilikan dan tanggung jawab manajemen. Tim jaringan memiliki struktur jaringan (switch, router, VLAN, IP), dan tim server memiliki server (perangkat keras, OS, aplikasi, waktu aktif). Virtualisasi OS memadukan tanggung jawab ini ke dalam satu platform, mengaburkan garis kepemilikan dan manajemen.

**Kontrol Akses**

Karena data disimpan di pusat data, mengakses data penting ini menjadi perhatian utama. Menjadi platform berbasis web, *cloud* bertindak sesuai dengan hak akses yang disediakan bagi pengguna untuk mengakses data. Hak akses ini meskipun didefinisikan dengan baik oleh masing-masing perusahaan masih menimbulkan masalah di *cloud*. Gartner merekomendasikan bahwa bisnis memerlukan CSP untuk mendukung kebijakan pembatasan akses subnet IP sehingga perusahaan dapat membatasi akses pengguna akhir dari rentang alamat IP dan perangkat yang diketahui.

Perusahaan harus menuntut agar penyedia enkripsi menawarkan akses pengguna dan kontrol administratif yang memadai, alternatif autentikasi yang lebih kuat seperti autentikasi dua faktor, pengelolaan izin akses, dan pemisahan tugas administratif seperti keamanan, jaringan, dan pemeliharaan.

**Enkripsi dan Dekripsi**

Karena data disimpan di *cloud* di luar wilayah pengguna, disarankan agar pengguna menyimpan data dalam bentuk terenkripsi. Perusahaan harus selalu bertujuan untuk mengelola kunci enkripsi, tetapi jika dikelola oleh penyedia enkripsi *cloud*, perusahaan harus memastikan bahwa kontrol manajemen akses tersedia yang akan memenuhi persyaratan pemberitahuan pelanggaran dan residensi data.

Jika kunci dikelola oleh CSP, maka bisnis harus memerlukan sistem manajemen kunci berbasis perangkat keras dalam serangkaian proses manajemen kunci yang terdefinisi dan terkelola dengan ketat. Saat kunci dikelola atau tersedia di *cloud*, vendor harus memberikan kontrol dan pemantauan yang ketat terhadap snapshot potensial dari beban kerja langsung untuk mencegah risiko analisis konten memori untuk mendapatkan kunci.

Bisnis juga harus membutuhkan:

Mencatat semua akses pengguna dan administrator ke sumber daya *cloud* dan untuk menyediakan log ini ke perusahaan dalam format yang sesuai untuk manajemen log atau informasi keamanan dan sistem manajemen kejadian

CSP untuk membatasi akses ke alat manajemen sistem sensitif yang mungkin memotret beban kerja langsung, melakukan migrasi data, atau mencadangkan dan memulihkan data

Gambar yang diambil dengan alat migrasi atau snapshot diperlakukan dengan keamanan yang sama seperti data perusahaan sensitif lainnya.

**Keamanan Virtualisasi**

Virtualisasi adalah teknologi yang mendorong konsolidasi server dan operasi pusat data menjadi unsur utama dalam menciptakan infrastruktur sesuai permintaan yang fleksibel. Ketika mengadopsi virtualisasi untuk komputasi awan, menjadi jelas bahwa alat manajemen yang digunakan dalam penyebaran berbasis server fisik tidak akan cukup dalam virtualisasi yang sangat dinamis. Pertama-tama, dalam model penyebaran server fisik, otomatisasi penyediaan umumnya tidak banyak digunakan kecuali ada jumlah OS server yang cukup signifikan untuk menjaminnya.

Virtualisasi terutama berfokus pada tiga area berbeda: jaringan virtual (virtualisasi jaringan), virtualisasi penyimpanan, dan virtualisasi server. Dalam virtualisasi jaringan, sumber daya yang tersedia dikumpulkan ke dalam jaringan dan bandwidth jaringan dibagi menjadi beberapa saluran di mana setiap saluran terpisah satu sama lain. Virtualisasi penyimpanan menggabungkan penyimpanan fisik dari beberapa perangkat penyimpanan jaringan, dan penyimpanan yang tersedia ini dipandang sebagai beberapa perangkat penyimpanan tunggal yang berbeda. Dalam virtualisasi server, identitas masing-masing perangkat server disembunyikan dari pengguna, dan server dirancang untuk dilihat sebagai server individual di mana pembagian sumber daya dan kompleksitas pemeliharaan dikelola secara seimbang. Kombinasi dari ketiga komponen virtualisasi ini memberikan kemampuan swakelola untuk sumber daya, dan swakelola ini memainkan peran utama dalam komputasi awan.

Strategi tipikal untuk penyediaan server fisik melibatkan langkah berulang. Dalam lingkungan yang sangat tervirtualisasi seperti *cloud*, penyediaan OS akan dengan cepat bertransisi menjadi proses yang sangat otomatis. Area kritis yang menjadi perhatian selama virtualisasi adalah sebagai berikut.

Ancaman baru: Virtualisasi mengubah hubungan antara OS dan perangkat keras. Ini menantang perspektif keamanan tradisional. Ini merusak kenyamanan yang mungkin Anda rasakan saat menyediakan OS dan aplikasi di server yang dapat Anda lihat dan sentuh. Beberapa sudah percaya bahwa rasa nyaman ini salah tempat dalam kebanyakan situasi. Untuk pengguna rata-rata, postur keamanan sebenarnya dari PC desktop dengan koneksi Internet sulit dilihat secara realistis.

Virtualisasi memperumit gambaran tetapi tidak serta merta membuat keamanan menjadi lebih baik atau lebih buruk. Ada beberapa masalah keamanan penting yang perlu Anda tangani dalam mempertimbangkan penggunaan virtualisasi untuk komputasi awan.

Salah satu potensi risiko baru berkaitan dengan potensi kompromi hypervisor VM. Jika hypervisor rentan untuk dieksploitasi, itu akan menjadi target utama. Pada skala *cloud*, risiko seperti itu akan berdampak luas jika tidak dimitigasi. Ini membutuhkan tingkat isolasi jaringan tambahan dan deteksi yang ditingkatkan dengan pemantauan keamanan.

Dalam memeriksa masalah ini, pertama-tama pertimbangkan sifat hypervisor. Terlihat bahwa "Hypervisor dibuat khusus dengan serangkaian fungsi yang kecil dan spesifik. Hypervisor lebih kecil, lebih fokus daripada sistem operasi tujuan umum, dan kurang terekspos, memiliki lebih sedikit atau tidak ada port jaringan yang dapat diakses secara eksternal. Hypervisor tidak sering mengalami perubahan dan tidak menjalankan aplikasi pihak ketiga. Sistem operasi tamu, yang mungkin rentan, tidak memiliki akses langsung ke hypervisor. Faktanya, hypervisor sepenuhnya transparan terhadap lalu lintas jaringan dengan pengecualian lalu lintas ke/dari antarmuka manajemen hypervisor khusus."

Masalah penyimpanan: Masalah keamanan lain dengan virtualisasi berkaitan dengan sifat pengalokasian dan pembatalan alokasi sumber daya seperti penyimpanan lokal yang terkait dengan VM. Selama penerapan dan pengoperasian VM, data ditulis ke dalam memori fisik. Jika tidak dihapus sebelum sumber daya tersebut dialokasikan kembali ke VM berikutnya, ada potensi paparan.

Masalah-masalah ini tentu tidak unik untuk virtualisasi. Mereka telah ditangani oleh setiap OS yang umum digunakan. Tidak semua OS mengelola pembersihan data. Beberapa mungkin menghapus data setelah rilis sumber daya; yang lain mungkin melakukannya berdasarkan alokasi.

Intinya adalah menghapus data sendiri, menangani operasi dengan hati-hati terhadap data sensitif, dan memberi perhatian khusus pada kontrol akses dan hak istimewa. Praktik keamanan luar biasa lainnya adalah memverifikasi bahwa sumber daya yang dirilis telah dihapus.

Bidang perhatian lebih lanjut dengan virtualisasi berkaitan dengan potensi serangan jaringan yang tidak terdeteksi antara VM yang ditempatkan di server fisik. Kecuali jika Anda dapat memantau lalu lintas dari setiap VM, Anda tidak dapat memverifikasi bahwa lalu lintas tidak dimungkinkan di antara VM tersebut.

Intinya, virtualisasi jaringan harus menghadirkan antarmuka jaringan yang sesuai ke VM. Antarmuka itu mungkin merupakan saluran multipleks dengan semua peralihan dan perutean ditangani dalam perangkat keras interkoneksi jaringan. Sebagian besar hypervisor berfitur lengkap memiliki sakelar virtual dan firewall yang berada di antara antarmuka fisik server dan antarmuka virtual yang disediakan untuk VM.

Manajemen lalu lintas: Teknik teoretis lain yang mungkin berpotensi membatasi arus lalu lintas antar VM adalah menggunakan pemisahan untuk mengumpulkan dan mengisolasi kelas VM yang berbeda satu sama lain. VM dapat dilacak ke pemiliknya sepanjang siklus hidup mereka. Mereka hanya akan ditempatkan di server fisik dengan VM lain yang memenuhi persyaratan yang sama untuk colocation.

Salah satu praktik aktual untuk mengelola arus lalu lintas antar VM adalah menggunakan VLAN untuk mengisolasi lalu lintas antara VM satu pelanggan dan VM pelanggan lainnya. Namun, agar benar-benar efektif, teknik ini memerlukan dukungan yang

lebih luas untuk VLAN di luar infrastruktur peralihan inti dan turun ke server fisik yang menghosting VM.

Masalah selanjutnya adalah menskalakan kemampuan seperti VLAN di luar batasnya saat ini untuk mendukung *cloud* yang lebih besar. Dukungan itu juga perlu distandarisasi untuk memungkinkan solusi multivendor. Itu juga perlu dikaitkan dengan manajemen jaringan dan hypervisor.

**Keamanan Jaringan**

*Cloud* didasarkan pada jaringan banyak hal bersama seperti jaringan infrastruktur. Meskipun jaringan merupakan tulang punggung *cloud*, banyak tantangan yang dihadapi dalam jaringan ini. Beberapa tantangan dalam jaringan *cloud* yang ada dibahas berikut ini.

Performa aplikasi: Penyewa *cloud* harus dapat menentukan persyaratan bandwidth untuk aplikasi yang dihosting di *cloud*, memastikan performa yang serupa dengan penerapan di lokasi. Banyak aplikasi berjenjang membutuhkan bandwidth yang dijamin antara server instans untuk memenuhi transaksi pengguna dalam jangka waktu yang dapat diterima dan memenuhi perjanjian tingkat layanan (SLA) yang telah ditentukan sebelumnya. Bandwidth yang tidak mencukupi antara server-server ini akan menyebabkan latensi yang signifikan pada interaksi pengguna. Oleh karena itu, tanpa kontrol eksplisit, variasi dalam beban kerja *cloud* dan kelebihan langganan dapat menyebabkan penundaan dan penyimpangan waktu respons di luar batas yang dapat diterima, yang menyebabkan pelanggaran SLA untuk aplikasi yang dihosting.

Penerapan peralatan yang fleksibel: Perusahaan menerapkan berbagai peralatan keamanan di pusat data mereka, seperti inspeksi paket dalam (DPI) atau sistem deteksi intrusi (IDS), dan firewall untuk melindungi aplikasi mereka dari serangan. Ini sering digunakan bersama peralatan lain yang melakukan penyeimbangan beban, caching, dan akselerasi aplikasi. Saat diterapkan di *cloud*, aplikasi perusahaan harus terus dapat mengeksploitasi fungsionalitas peralatan ini secara fleksibel.

Kompleksitas penegakan kebijakan: Isolasi lalu lintas dan kontrol akses ke pengguna akhir adalah beberapa kebijakan penerusan yang harus ditegakkan. Kebijakan ini berdampak langsung pada konfigurasi setiap router dan switch. Persyaratan yang berubah, protokol yang berbeda (misalnya, Open Shortest Path First [OSPF], Link Aggregation Group (LAG), Protokol Redundansi Router Virtual [VRRP]), dan rasa yang berbeda dari protokol pohon spanning L2, bersama dengan protokol khusus vendor, membuatnya sangat menantang untuk membangun, mengoperasikan, dan menghubungkan jaringan *cloud* dalam skala besar.

Kompleksitas yang bergantung pada topologi: Topologi jaringan pusat data biasanya disetel agar sesuai dengan persyaratan lalu lintas yang telah ditentukan sebelumnya. Misalnya, topologi jaringan yang dioptimalkan untuk lalu lintas timur-barat (yaitu, lalu lintas antar server di pusat data) tidak sama dengan topologi untuk utara-selatan (lalu lintas ke/dari Internet). Desain topologi juga bergantung pada bagaimana L2 dan/atau L3 memanfaatkan kapasitas jaringan yang efektif. Misalnya, menambahkan tautan dan sakelar sederhana di hadapan protokol penerusan L2 berbasis spanning tree mungkin tidak menyediakan kapasitas tambahan. Selain itu, mengembangkan topologi berdasarkan perubahan pola lalu lintas juga memerlukan konfigurasi rumit dari aturan penerusan L2 dan L3.

Penulisan ulang aplikasi: Aplikasi harus kehabisan kotak sebanyak mungkin, khususnya untuk alamat IP dan mekanisme failover yang bergantung pada jaringan. Aplikasi mungkin perlu ditulis ulang atau dikonfigurasi ulang sebelum diterapkan di *cloud* untuk mengatasi beberapa batasan terkait jaringan.

Dua masalah utama adalah (1) kurangnya abstraksi domain siaran di jaringan *cloud* dan (2) alamat IP yang diberikan *cloud* untuk server virtual.

Ketergantungan lokasi: Peralatan dan server jaringan (misalnya, hypervisor) biasanya terikat ke jaringan fisik yang dikonfigurasi secara statis, yang secara implisit menciptakan batasan yang bergantung pada lokasi. Misalnya, alamat IP server biasanya ditentukan berdasarkan VLAN atau subnet tempatnya berada. VLAN dan subnet didasarkan pada konfigurasi port switch fisik. Oleh karena itu, VM tidak dapat dimigrasikan dengan mudah dan lancar di seluruh jaringan. Migrasi VM yang dibatasi menurunkan tingkat pemanfaatan sumber daya dan fleksibilitas. Selain itu, pemetaan fisik VLAN atau ruang subnet ke port fisik switch sering mengarah ke kumpulan alamat IP yang terfragmentasi.

Kompleksitas jaringan multilayer: Jaringan pusat data tiga lapis biasanya mencakup lapisan TOR (Top of Rack) yang menghubungkan server di rak, lapisan agregasi, dan lapisan inti, yang menyediakan konektivitas ke/dari tepi Internet. Arsitektur multilayer ini memaksakan kompleksitas yang signifikan dalam menentukan batas-batas domain L2, jaringan dan kebijakan penerusan L3, dan peralatan jaringan multivendor khusus-lapisan. Penyedia layanan komputasi awan saat ini mengoperasikan pusat data mereka sendiri. Konektivitas antar pusat data untuk memberikan visi satu *cloud* sepenuhnya berada dalam kendali CSP. Mungkin ada situasi di mana organisasi atau perusahaan harus dapat bekerja dengan banyak penyedia *cloud* karena lokalitas akses, migrasi dari satu layanan *cloud* ke layanan *cloud* lainnya, penggabungan perusahaan yang bekerja dengan penyedia *cloud* yang berbeda, penyedia *cloud* yang memberikan yang terbaik dari kelas services, dan kasus serupa. Interoperabilitas *cloud* dan kemampuan untuk berbagi berbagai jenis informasi antar *cloud* menjadi penting dalam skenario seperti itu. Meskipun CSP mungkin melihat kebutuhan yang kurang mendesak untuk interoperabilitas, pelanggan perusahaan akan melihat kebutuhan untuk mendorong mereka ke arah ini. Area interoperabilitas *cloud* yang luas ini terkadang dikenal sebagai *cloud* federation.

## 13.3 KEAMANAN TERKAIT PLATFORM

CSP menawarkan layanan dalam berbagai model layanan seperti SaaS, PaaS, dan IaaS di mana pengguna ditawarkan berbagai layanan berdasarkan kebutuhannya. Setiap model layanan yang ditawarkan membawa serta banyak tantangan terkait keamanan seperti jaringan aman, lokalitas sumber daya, mengakses data aman, privasi data, dan kebijakan pencadangan.

**Masalah Keamanan dalam Model Layanan *Cloud***

Komputasi awan menggunakan tiga model pengiriman seperti SaaS, PaaS, dan IaaS di mana berbagai jenis layanan komputasi disediakan untuk pengguna akhir. Ketiga model pengiriman ini menyediakan sumber daya infrastruktur, platform aplikasi, dan perangkat lunak sebagai layanan kepada pelanggan *cloud*. Model layanan ini menempatkan tingkat

persyaratan keamanan yang berbeda di lingkungan *cloud*. IaaS adalah dasar dari semua layanan *cloud*, dengan PaaS dibangun di atasnya dan SaaS pada gilirannya dibangun di atasnya. Sama seperti kemampuan yang diwariskan, begitu pula masalah dan risiko keamanan informasi. Ada *trade-off* yang signifikan untuk setiap model dalam hal fitur terintegrasi, kompleksitas versus ekstensibilitas, dan keamanan. Jika CSP hanya menangani keamanan di bagian bawah arsitektur keamanan, konsumen menjadi lebih bertanggung jawab untuk mengimplementasikan dan mengelola kemampuan keamanan.

SaaS adalah model penyebaran perangkat lunak di mana aplikasi dihosting dari jarak jauh oleh aplikasi atau penyedia layanan dan tersedia untuk pelanggan sesuai permintaan, melalui Internet. Model SaaS menawarkan manfaat yang signifikan kepada pelanggan, seperti peningkatan efisiensi operasional dan pengurangan biaya. SaaS dengan cepat muncul sebagai model pengiriman yang dominan untuk memenuhi kebutuhan layanan TI perusahaan. SaaS dengan cepat muncul sebagai model pengiriman yang dominan untuk memenuhi kebutuhan layanan TI perusahaan. PaaS adalah satu lapisan di atas IaaS pada tumpukan dan mengabstraksi semuanya hingga OS, middleware, dll. Ini menawarkan seperangkat lingkungan pengembang terintegrasi yang dapat diketuk pengembang untuk membangun aplikasi mereka tanpa memiliki petunjuk tentang apa yang terjadi di bawahnya layanan. Ini menawarkan pengembang layanan yang menyediakan manajemen siklus hidup pengembangan perangkat lunak yang lengkap, mulai dari perencanaan hingga desain hingga membangun aplikasi hingga penerapan hingga pengujian hingga pemeliharaan. Segala sesuatu yang lain diabstraksi dari pandangan pengembang.

**Masalah Keamanan Perangkat Lunak sebagai Layanan**

Dalam model penerapan aplikasi lokal tradisional, data sensitif setiap perusahaan terus berada dalam batas perusahaan dan tunduk pada kebijakan keamanan fisik, logis, dan personel serta kontrol aksesnya. Arsitektur aplikasi berbasis SaaS dirancang khusus untuk mendukung banyak pengguna secara bersamaan (multitenancy). Aplikasi SaaS diakses melalui web, sehingga keamanan browser web sangat penting. Petugas keamanan informasi perlu mempertimbangkan berbagai metode untuk mengamankan aplikasi SaaS. Keamanan layanan web (WS), enkripsi Extensible Markup Language (XML), SSL, dan opsi yang tersedia digunakan dalam menegakkan perlindungan data yang dikirimkan melalui Internet. Dalam model SaaS, data perusahaan disimpan di luar batas perusahaan, di ujung vendor SaaS. Konsekuensinya, vendor SaaS harus melakukan pemeriksaan keamanan tambahan untuk memastikan keamanan data dan untuk mencegah pelanggaran karena kerentanan keamanan dalam aplikasi atau melalui karyawan jahat. Ini melibatkan penggunaan teknik enkripsi yang kuat untuk keamanan data dan otorisasi halus untuk mengontrol akses ke data. Poin nyeri yang menjadi perhatian dalam SaaS adalah sebagai berikut.

Keamanan jaringan: Dalam model penerapan SaaS, aliran data sensitif melalui jaringan perlu diamankan untuk mencegah kebocoran informasi sensitif. Ini melibatkan penggunaan teknik enkripsi lalu lintas jaringan yang kuat seperti SSL dan TLS untuk keamanan.

Lokalitas sumber daya: Dalam model SaaS lingkungan *cloud*, pengguna akhir menggunakan layanan yang disediakan oleh penyedia *cloud* tanpa mengetahui dengan pasti di mana sumber daya untuk layanan tersebut berada. Karena undang-undang kepatuhan dan

privasi data di berbagai negara, lokalitas data sangat penting dalam banyak arsitektur perusahaan.

Arahan tersebut melarang transfer data pribadi ke negara-negara yang tidak menjamin tingkat perlindungan yang memadai. Misalnya, pengguna Dropbox baru-baru ini harus menyetujui Ketentuan Layanan yang memberikan hak kepada penyedia untuk mengungkapkan informasi pengguna sesuai dengan undang-undang dan permintaan penegak hukum.

Standar *cloud*: Untuk mencapai interoperabilitas di antara *cloud* dan untuk meningkatkan stabilitas dan keamanannya, standar *cloud* diperlukan di seluruh organisasi. Misalnya, layanan penyimpanan saat ini oleh penyedia *cloud* mungkin tidak kompatibel dengan penyedia lainnya. Untuk mempertahankan pelanggan mereka, penyedia *cloud* dapat memperkenalkan apa yang disebut layanan lengket yang menyulitkan pengguna jika mereka ingin bermigrasi dari satu penyedia ke penyedia lainnya.

Pemisahan data: Multitenancy adalah salah satu karakteristik utama komputasi awan. Dalam situasi multitenancy, data dari berbagai pengguna akan berada di lokasi yang sama. Intrusi data dari satu pengguna oleh pengguna lain menjadi mungkin di lingkungan ini. Intrusi ini dapat dilakukan baik dengan meretas melalui lubang loop di aplikasi atau dengan menyuntikkan kode klien ke dalam sistem SaaS. Oleh karena itu, model SaaS harus memastikan batas yang jelas untuk setiap data pengguna. Batasan harus dipastikan tidak hanya pada tingkat fisik tetapi juga pada tingkat aplikasi. Layanan harus cukup cerdas untuk memisahkan data dari pengguna yang berbeda.

Akses data: Masalah akses data terutama terkait dengan kebijakan keamanan yang diberikan kepada pengguna saat mengakses data. Organisasi akan memiliki kebijakan keamanan mereka sendiri berdasarkan mana setiap karyawan dapat memiliki akses ke kumpulan data tertentu. Kebijakan keamanan mungkin memberikan beberapa pertimbangan, di mana beberapa karyawan tidak diberikan akses ke sejumlah data tertentu. Kebijakan keamanan ini harus dipatuhi oleh *cloud* untuk menghindari intrusi data oleh pengguna yang tidak berwenang. Model SaaS harus cukup fleksibel untuk memasukkan kebijakan khusus yang diajukan oleh organisasi. Model tersebut juga harus dapat memberikan batasan organisasi di dalam *cloud* karena banyak organisasi akan menerapkan proses bisnis mereka dalam satu lingkungan *cloud*.

Pelanggaran data: Karena data dari berbagai pengguna dan organisasi bisnis terletak bersama di lingkungan *cloud*, pelanggaran ke dalam lingkungan *cloud* berpotensi menyerang data semua pengguna. Dengan demikian, *cloud* menjadi target bernilai tinggi.

Pencadangan: Vendor SaaS perlu memastikan bahwa semua data perusahaan yang sensitif dicadangkan secara teratur untuk memfasilitasi pemulihan cepat jika terjadi bencana. Selain itu, penggunaan skema enkripsi yang kuat untuk melindungi data cadangan direkomendasikan untuk mencegah kebocoran informasi sensitif yang tidak disengaja. Dalam kasus vendor *cloud* seperti Amazon, data diam di S3 tidak dienkripsi secara default. Pengguna perlu mengenkripsi data dan cadangan mereka secara terpisah sehingga tidak dapat diakses atau dirusak oleh pihak yang tidak berwenang.

Manajemen identitas (IdM) dan proses masuk: IdM berurusan dengan mengidentifikasi individu dalam suatu sistem dan mengendalikan akses ke sumber daya dalam sistem itu dengan menempatkan batasan pada identitas yang ditetapkan. Ketika penyedia SaaS harus tahu bagaimana mengontrol siapa yang memiliki akses ke sistem apa di dalam perusahaan, itu menjadi tugas yang lebih menantang. Dalam skenario seperti itu, penyediaan dan deprovisioning pengguna di *cloud* menjadi sangat penting.

**Masalah Keamanan Platform sebagai Layanan**

PaaS menyediakan platform siap pakai, termasuk OS yang berjalan pada infrastruktur yang disediakan vendor. Karena infrastrukturnya adalah CSP, berbagai tantangan keamanan dari arsitektur terfokus terutama disebabkan oleh penyebaran objek pengguna di host *cloud*. Secara ketat mengizinkan akses objek ke sumber daya dan mempertahankan objek dari penyedia jahat atau korup secara wajar mengurangi risiko yang mungkin terjadi. Akses jaringan dan pengukuran layanan menyatukan kekhawatiran tentang komunikasi yang aman dan kontrol akses. Praktik-praktik terkenal, penegakan otorisasi skala objek, dan metode ketertelusuran yang tidak dapat disangkal dapat mengurangi kekhawatiran tersebut.

Terlepas dari masalah yang disebutkan di atas, privasi pengguna harus dilindungi di *cloud* publik yang dibagikan. Oleh karena itu, solusi yang diusulkan harus sadar privasi. Kontinuitas layanan menjadi perhatian lain bagi banyak perusahaan yang mempertimbangkan adopsi *cloud*. Oleh karena itu, diperlukan sistem andal yang toleran terhadap kesalahan.

**Masalah Keamanan Infrastruktur sebagai Layanan**

Komputasi awan membuat banyak janji di bidang peningkatan fleksibilitas dan ketangkasan, penghematan biaya potensial, dan keunggulan kompetitif bagi pengembang sehingga mereka dapat membangun infrastruktur dengan cepat dan efisien untuk memungkinkan mereka mengembangkan perangkat lunak untuk mendorong bisnis kesuksesan. Ada banyak masalah yang diselesaikan oleh *cloud*, terutama *cloud* pribadi, tetapi tidak terlalu bagus dalam menyelesaikan masalah yang berkaitan dengan keamanan.

Namun, di lingkungan private *cloud*, beberapa masalah tradisional yang dihadapi adalah sebagai berikut:

1. Keamanan hypervisor: Di *cloud* pribadi, sebagian besar atau semua layanan akan berjalan di lingkungan virtual dan model keamanan yang digunakan oleh hypervisor tidak dapat diterima begitu saja. Kebutuhan untuk mengevaluasi model keamanan dan pengembangan hypervisors menjadi perlu.
2. Multitenancy: Meskipun semua penyewa di lingkungan multitenancy akan berasal dari perusahaan yang sama, tidak semua penyewa dapat nyaman berbagi infrastruktur dengan pengguna lain dalam perusahaan yang sama.
3. Manajemen identitas dan kontrol akses (IdAM): Di pusat data tradisional, kami merasa nyaman dengan segelintir repositori autentikasi yang harus kami kerjakan—Active Directory menjadi salah satu yang paling populer. Tetapi dengan *cloud* pribadi, menangani autentikasi dan otorisasi untuk infrastruktur *cloud*, menangani penyewa, dan menangani pendelegasian administrasi dari berbagai aspek struktur *cloud* adalah tugas utama yang harus ditangani.

4. Keamanan jaringan: Di *cloud* pribadi, kami cenderung memiliki banyak komponen layanan yang berkomunikasi satu sama lain hanya melalui saluran jaringan virtual. Menilai lalu lintas, menggunakan beberapa kontrol akses yang kuat untuk jaringan fisik, dan mengontrol kualitas layanan, yang merupakan masalah utama dalam aspek ketersediaan model keamanan kerahasiaan, integritas, dan ketersediaan (CIA), menjadi perhatian utama.

Sebagai konsolidasi, keamanan terkait platform mempertimbangkan tiga model penyampaian layanan yang disebutkan sebelumnya seperti SaaS, PaaS, dan IaaS, dan komponen terkait juga disebutkan secara terpisah.

Menggabungkan tiga jenis *cloud* (public *cloud*, private *cloud*, dan hybrid *cloud*) bersama-sama dengan tiga model penyampaian layanan, kami mendapatkan gambaran lengkap tentang lingkungan *cloud computing* yang dihubungkan oleh perangkat konektivitas ditambah dengan komponen keamanan informasi. Sumber daya fisik tervirtualisasi, infrastruktur tervirtualisasi, platform middleware tervirtualisasi, dan aplikasi terkait bisnis disediakan sebagai layanan komputasi di *cloud*. Penyedia *cloud* dan konsumen *cloud* harus mampu menjaga dan membangun keamanan komputasi di semua tingkat antarmuka dalam arsitektur *cloud computing*.

## 13.4 AUDIT DAN KEPATUHAN

Sudah menjadi fakta yang diketahui secara luas bahwa perlindungan data dan kepatuhan terhadap peraturan adalah salah satu masalah keamanan utama bagi chief information officer (CIO) di organisasi mana pun.

Menurut Pew Internet dan American Life Project, sebagian besar pengguna layanan *cloud computing* menyatakan keprihatinan serius tentang kemungkinan penyedia layanan mengungkapkan data mereka kepada orang lain. Sembilan puluh persen pengguna aplikasi *cloud* mengatakan bahwa mereka akan sangat khawatir jika perusahaan tempat penyimpanan data mereka menjualnya ke pihak lain.

Sebuah survei yang dilakukan oleh banyak perusahaan mengungkapkan pandangan bahwa keamanan merupakan tantangan terbesar bagi model komputasi awan. Oleh karena itu, pemangku kepentingan semakin merasa perlu untuk mencegah pelanggaran data. Dalam beberapa bulan terakhir, banyak artikel surat kabar mengungkapkan kebocoran data di area sensitif seperti domain keuangan dan pemerintahan serta komunitas web.

Salah satu misi otoritas perlindungan data adalah untuk mencegah apa yang disebut fenomena Big Brother, yang mengacu pada skenario di mana otoritas publik memproses data pribadi tanpa perlindungan privasi yang memadai. Dalam situasi seperti itu, pengguna akhir dapat melihat *cloud* sebagai sarana untuk beralih ke masyarakat pengawasan totaliter.

Oleh karena itu, kekhususan komputasi awan membuat insentif perlindungan data menjadi lebih besar. Misalnya, penyedia *cloud* harus menyediakan enkripsi untuk melindungi data pribadi yang disimpan dari akses, penyalinan, kebocoran, atau pemrosesan yang tidak sah.

Dalam lingkungan *cloud*, perusahaan tidak memiliki kendali atas data mereka, yang dipercayakan kepada penyedia layanan aplikasi pihak ketiga di *cloud*, sekarang dapat berada

di mana saja di dunia. Perusahaan juga tidak akan tahu di negara mana datanya berada pada titik waktu tertentu. Ini adalah masalah utama komputasi awan yang bertentangan dengan persyaratan UE di mana perusahaan harus setiap saat mengetahui ke mana data pribadi yang dimilikinya dipindahkan. Komputasi awan dengan demikian menimbulkan masalah khusus untuk perusahaan multinasional dengan pelanggan UE tertentu.

**Pemulihan Bencana**

Pencadangan data sederhana serta pemulihan bencana yang lebih komprehensif dan perencanaan kesinambungan bisnis merupakan bagian penting dari bisnis dan kehidupan pribadi. Pencadangan sebagai Layanan dan Pemulihan Bencana sebagai Layanan kini tersedia online melalui *cloud* untuk setiap tingkat pengguna, mulai dari penyimpanan dan pengambilan data pribadi, bisnis kecil hingga perusahaan besar, baik secara publik melalui Internet atau melalui metode akses khusus yang lebih aman. Akibatnya, metode tradisional menjadi usang. Beberapa keunggulannya antara lain sebagai berikut:

- Tidak ada biaya awal yang besar untuk investasi modal atau pengelolaan infrastruktur atau kotak hitam.
- Cadangan disimpan secara fisik di lokasi yang berbeda dari sumber asli data Anda.
- Pencadangan jarak jauh tidak memerlukan campur tangan pengguna atau pencadangan manual berkala.
- Retensi data tak terbatas. Anda bisa mendapatkan ruang penyimpanan data sebanyak atau sesedikit yang Anda butuhkan.
- Pencadangan bersifat otomatis dan cerdas. Mereka terjadi terus menerus dan secara efisien mencadangkan file Anda hanya saat data berubah.

Komputasi awan, berdasarkan virtualisasi, mengambil pendekatan yang sangat berbeda untuk pemulihan bencana. Dengan virtualisasi, seluruh server, termasuk OS, aplikasi, tambalan, dan data, dienkapsulasi menjadi satu bundel perangkat lunak atau server virtual. Seluruh server virtual ini dapat disalin atau dicadangkan ke pusat data di luar lokasi dan diputar di host virtual dalam hitungan menit.

Karena server virtual tidak bergantung pada perangkat keras, OS, aplikasi, tambalan, dan data dapat ditransfer dengan aman dan akurat dari satu pusat data ke pusat data kedua tanpa beban memuat ulang setiap komponen server. Hal ini dapat secara dramatis mengurangi waktu pemulihan dibandingkan dengan pendekatan pemulihan bencana konvensional (nonvirtual) di mana server perlu dimuat dengan OS dan perangkat lunak aplikasi dan ditambal ke konfigurasi terakhir yang digunakan dalam produksi sebelum data dapat dipulihkan.

**Privasi dan Integritas**

Janji untuk menghadirkan TI sebagai layanan ditujukan kepada sejumlah besar konsumen, mulai dari usaha kecil dan menengah (UKM) dan administrasi publik hingga pengguna akhir. Pengguna membuat jumlah data pribadi yang terus bertambah.

Kuantitas data pribadi yang meningkat ini akan mendorong permintaan untuk layanan *cloud*, terutama jika komputasi *cloud* memberikan janji biaya yang lebih rendah untuk pelanggan dan munculnya model bisnis baru untuk penyedia.

Di antara tantangan privasi utama untuk komputasi awan adalah sebagai berikut.

Kompleksitas penilaian risiko di lingkungan *cloud*: Kompleksitas layanan *cloud* memperkenalkan sejumlah parameter yang tidak diketahui. Penyedia layanan dan konsumen berhati-hati dalam menawarkan jaminan untuk layanan yang siap mematuhi dan mengadopsi layanan tersebut. Dengan penyedia layanan yang mempromosikan cara sederhana untuk mengalirkan data pribadi terlepas dari batas negara, tantangan nyata muncul dalam hal memeriksa siklus hidup pemrosesan data dan kepatuhannya terhadap kerangka hukum.

Untuk mengatasi masalah seperti peran dan tanggung jawab pemangku kepentingan, replikasi data, dan kepatuhan masalah hukum, Resolusi Madrid menyatakan bahwa setiap orang yang bertanggung jawab harus memiliki kebijakan yang transparan sehubungan dengan pemrosesan data pribadi. Pemangku kepentingan perlu menentukan persyaratan untuk komputasi awan yang memenuhi tingkat keamanan dan privasi yang diharapkan. Di Eropa, *European Network and Information Security Agency* (ENISA) memberikan rekomendasi untuk memfasilitasi pemahaman tentang pergeseran keseimbangan tanggung jawab dan akuntabilitas untuk fungsi utama seperti tata kelola dan kontrol atas data dan operasi TI serta kepatuhan terhadap undang-undang dan peraturan.

Munculnya model bisnis baru dan implikasinya terhadap privasi konsumen: Sebuah laporan oleh Komisi Perdagangan Federal (FTC) tentang Melindungi privasi konsumen di era perubahan yang cepat menganalisis implikasi kemajuan teknologi di bidang TI terhadap privasi konsumen. Menurut FTC, pengguna dapat mengumpulkan, menyimpan, memanipulasi, dan berbagi sejumlah besar data konsumen dengan biaya yang sangat kecil.

Kemajuan teknologi ini telah menyebabkan ledakan model bisnis baru yang bergantung pada pengambilan data konsumen pada tingkat spesifik dan individual dan dari waktu ke waktu, termasuk pembuatan profil, periklanan perilaku online (OBA), layanan media sosial, dan layanan seluler berbasis lokasi.

## 13.5 RINGKASAN

*Cloud* menjadi platform komputasi yang efisien dan berbiaya rendah untuk industri TI berkembang pesat. Dengan pertumbuhan yang luar biasa muncul tantangan untuk menangani data penting dan menawarkan kualitas layanan kepada pengguna.

Bab ini menyoroti berbagai aspek keamanan yang terkait dengan komputasi awan. Elemen keamanan dasar yang terkait dengan berbagai model penerapan *cloud* dan model penyampaian layanan dijelaskan secara singkat di sini. Aspek keamanan seperti keamanan pusat data dan keamanan sehubungan dengan model layanan juga disorot. Karena industri TI didorong ke *cloud* untuk kapasitas komputasinya, pada gilirannya perlu melihat keamanan data penting yang disimpan di penyedia pihak ketiga. Berbagai masalah terkait keamanan, meskipun ditangani oleh industri TI, masih menjadi perhatian utama karena tidak ada prosedur pengembangan standar yang ditetapkan untuk pengembangan model *cloud*. Sebagai organisasi mengikuti model mereka sendiri untuk pengembangan, keamanan menjadi lebih menonjol untuk konsentrasi.

**Tinjau Poin**

- Kebijakan ambang batas: Untuk menguji apakah program berhasil, berkembang, atau meningkat dan diterapkan, kebijakan ambang batas adalah studi percontohan sebelum memindahkan program ke lingkungan produksi.
- Keamanan data: Data disimpan dalam infrastruktur CSP. Karena data tidak berada di wilayah organisasi, banyak tantangan rumit muncul.
- Kontrol akses: Karena data disimpan di pusat data, mengakses data penting ini menjadi perhatian utama. Menjadi platform berbasis web, *cloud* bertindak sesuai dengan hak akses yang disediakan bagi pengguna untuk mengakses data.
- Ketergantungan lokasi: Peralatan dan server jaringan biasanya diikat ke jaringan fisik yang dikonfigurasi secara statis, yang secara implisit menciptakan batasan yang bergantung pada lokasi.
- Lokalitas sumber daya: Dalam model SaaS dari lingkungan *cloud*, pengguna akhir menggunakan layanan yang disediakan oleh penyedia *cloud* tanpa mengetahui dengan pasti di mana sumber daya untuk layanan tersebut berada.
- IdM: IdM berurusan dengan mengidentifikasi individu dalam suatu sistem dan mengontrol akses ke sumber daya dalam sistem itu dengan membatasi identitas yang ditetapkan.
- Pemulihan bencana: Pencadangan data sederhana serta pemulihan bencana yang lebih komprehensif dan perencanaan kesinambungan bisnis merupakan bagian penting dari bisnis dan kehidupan pribadi.

**Latihan Soal**

1. Masalah apa yang harus ditangani dalam keamanan data? Menjelaskan.
2. Apa masalah penyimpanan dalam keamanan virtualisasi?
3. Jelaskan tantangan dalam jaringan *cloud*.
4. Apa masalah keamanan di SaaS? Menjelaskan.
5. Apa saja masalah keamanan di PaaS? Menjelaskan.
6. Apa masalah keamanan di IaaS? Menjelaskan.
7. Apa keuntungan dari Disaster Recovery as a Service?
8. Apa tantangan privasi untuk komputasi awan? Menjelaskan.

# BAB 14
# KONSEP LANJUTAN DALAM *CLOUD COMPUTING*

**Tujuan pembelajaran**

Tujuan dari bab ini adalah untuk

- Berikan wawasan tentang beberapa topik lanjutan di *cloud*
- Berikan penjelasan singkat tentang setiap topik
- Mendiskusikan masalah saat ini pada topik tertentu
- Tunjukkan beberapa pekerjaan yang sedang berlangsung di area tersebut
- Diskusikan manfaat menggunakan teknologi

**Pengantar**

Komputasi awan adalah bidang penelitian yang penting dan berkembang dengan sangat cepat. Ada topik tertentu di *cloud* di mana ada beberapa lowongan penelitian. Ada beberapa aplikasi *cloud*; kedalaman dan keluasan, awan itu meluas. Dalam bab ini, beberapa topik lanjutan dibahas. Deskripsi singkat tentang inter*cloud* dan jenis, topologi, dan manfaatnya diberikan terlebih dahulu, diikuti oleh manajemen *cloud*. Manajemen *cloud* dianggap sebagai salah satu bidang penting, dan di sini dibahas topik manajemen tertentu yang terkait dengan arsitektur. Akibatnya, *cloud* seluler dan *cloud* media dibahas secara singkat. Ini diikuti dengan diskusi tentang standar dan interoperabilitas. Di sini, masalah yang berkaitan dengan interoperabilitas ditunjukkan. Perlunya standardisasi juga dibahas. Ini diikuti dengan pengenalan singkat tentang tata kelola *cloud*, berbagai komponennya, dan kepentingannya. Subbab selanjutnya membahas secara detail tentang green computing. Ini menguraikan tentang metode untuk perhitungan energi. Ini membahas beberapa algoritma populer yang mencoba mengurangi konsumsi energi pusat data. Selanjutnya, karya-karya terbaru di bidang ini disebutkan. Subbab selanjutnya membahas tentang dampak kecerdasan komputasional dalam *cloud computing*. Area di mana itu digunakan dan tempat-tempat di mana itu dapat digunakan dibahas. Terakhir, analitik data *cloud*, salah satu topik paling populer yang mengubah dunia saat ini, dibahas. Subbagian ini menguraikan analitik data, dan alasan *cloud* digunakan di sini. Bab ini diakhiri dengan beberapa poin ulasan, pertanyaan ulasan, dan bacaan lebih lanjut.

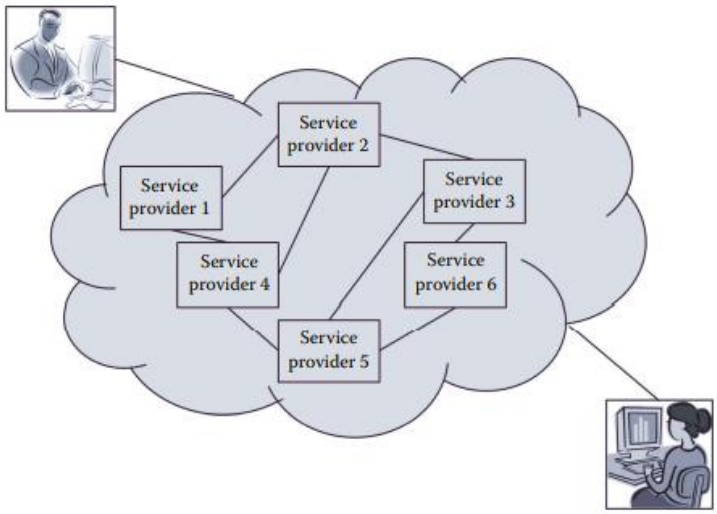
## 14.1 INTERCLOUD

Komputasi awan adalah teknologi yang telah ada selama beberapa waktu dan telah menjadi teknologi yang disambut baik. *National Institute of Standards and Technology* (NIST) mendefinisikan *cloud computing* sebagai "sebuah model untuk memungkinkan akses jaringan di mana-mana, nyaman, sesuai permintaan ke kumpulan sumber daya komputasi yang dapat dikonfigurasi bersama (misalnya, jaringan, server, penyimpanan, aplikasi, dan layanan) yang dapat disediakan dan dirilis dengan cepat dengan upaya manajemen minimal atau interaksi penyedia layanan." Saat ini, ada peningkatan permintaan untuk komputasi awan. Dengan meningkatnya jumlah pengguna *cloud*, merupakan tantangan untuk memenuhi kebutuhan semua pengguna untuk menjaga kredibilitas penyedia *cloud*. Penyedia *cloud* tidak

menawarkan sumber daya dalam jumlah tak terbatas dan karenanya mungkin jenuh pada suatu saat. Dalam beberapa kasus, situasi mungkin muncul di mana penyedia *cloud* mungkin tidak dapat memenuhi kebutuhan pelanggan. Dalam kasus seperti itu di mana penyedia *cloud* dihadapkan dengan peningkatan persyaratan atau kebutuhan yang tidak terduga, ia harus menggunakan beberapa metode untuk memastikan kepuasan pelanggan. Dalam situasi seperti itulah intercloud datang sebagai keuntungan bagi penyedia *cloud*.

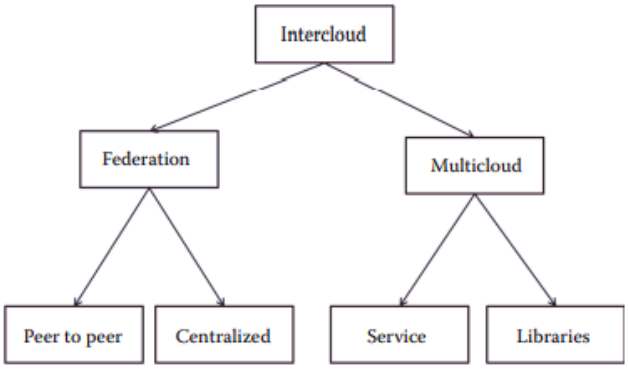
Intercloud pada dasarnya dapat dilihat sebagai awan awan. Di sini, beberapa penyedia *cloud* bergandengan tangan untuk melayani pelanggan. Intercloud dapat mengambil salah satu dari dua bentuk: federasi awan atau multicloud. Konsep intercloud diperkenalkan pada tahun 2010 di Lokakarya Internasional IEEE pertama tentang Interoperabilitas dan Layanan Komputasi Awan (InterCloud 2010). Pada bulan Oktober 2013, sebuah lingkungan yang akan berfungsi sebagai testbed untuk aplikasi terkait inter*cloud* diusulkan.

Penyedia *cloud* utama adalah Amazon, Google, Microsoft, IBM, dll., seperti yang ditunjukkan pada Gambar 14.1. Merupakan tantangan bagi penyedia *cloud* lain untuk bersaing dengan penyedia *cloud* ini. Federation of *cloud* adalah keuntungan bagi penyedia *cloud* kecil karena mereka dapat menyediakan sumber daya mereka untuk disewakan dan dengan demikian memperoleh keuntungan untuk sumber daya mereka, yang jika tidak akan dibiarkan kurang dimanfaatkan.




**Gambar 14.1** Ilustrasi Intercloud.

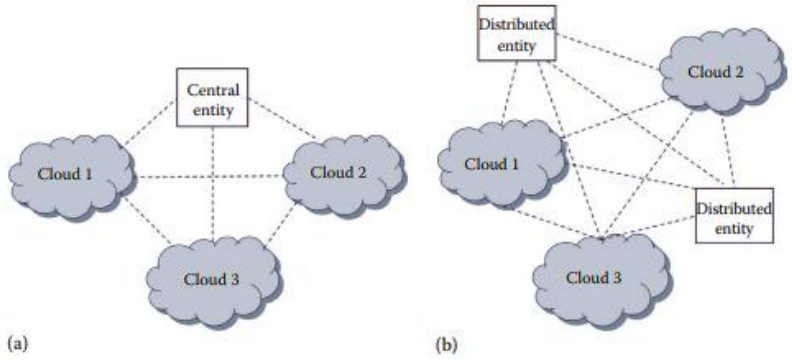
Klasifikasi dasar inter*cloud* dapat diberikan seperti pada Gambar 14.2.




**Gambar 14.2** Klasifikasi antar awan.

Dalam federasi *cloud*, penyedia *cloud* mengelola interkoneksi di antara mereka. Di sini, infrastruktur dapat dibagi sehingga memungkinkan berbagi sumber daya. Pengguna tidak perlu repot menggunakan lebih dari satu *cloud* karena penyedia *cloud* bertanggung jawab untuk menyediakan layanan transparan kepada pelanggan. Di sini, satu penyedia *cloud* dapat menyewa sumber daya dari penyedia *cloud* lain dan menawarkannya kepada pelanggan.

Di multicloud, klien atau layanan menggunakan banyak *cloud*. Dengan demikian, pengguna menyadari fakta bahwa mereka dilayani oleh lebih dari satu penyedia *cloud*. Merupakan tanggung jawab pengguna untuk menyediakan interoperabilitas antara berbagai penyedia *cloud*. Berbagai jenis multicloud dan federation *cloud* memiliki berbagai topologi, seperti yang ditunjukkan pada Gambar 14.3.




**Gambar 14.3** Topologi arsitektur intercloud yang berbeda. (a) Federasi antar awan terpusat, (b) federasi intercloud peer to peer.

Intercloud memberi pelanggan berbagai manfaat seperti berikut ini:

1. Lokasi geografis yang beragam: Penyedia layanan *cloud* mungkin memiliki pusat data mereka di berbagai lokasi geografis. Dalam kasus di mana pelanggan terikat oleh batasan legislatif mengenai lokasi penyimpanan data mereka, pusat data yang disediakan oleh penyedia layanan *cloud* di lokasi tertentu mungkin tidak cukup. Intercloud dapat digunakan untuk memungkinkan pelanggan mengontrol lokasi penyimpanan data mereka.
2. Ketahanan aplikasi yang lebih baik: Sasaran toleransi kesalahan, keandalan, dan ketersediaan juga dapat dipenuhi dengan lebih baik oleh penyedia *cloud* yang memanfaatkan intercloud. Dengan menempatkan data di lebih dari satu pusat data, aplikasi dapat dibuat lebih toleran terhadap kesalahan. Jika salah satu pusat data tidak berfungsi, selalu ada opsi untuk menggunakan salinan alternatif di pusat data lain. Ini juga meningkatkan ketersediaan. Jika terjadi peningkatan lalu lintas yang tiba-tiba, penurunan kinerja dapat terjadi. Ini dapat dihindari di intercloud. Dengan demikian, kinerja dijamin.
3. Menghindari vendor *lock-in:* Penggunaan intercloud menghindari vendor *lock-in* dimana pelanggan menjadi tergantung pada vendor tertentu. Di sini, pengguna bergantung pada banyak vendor.

4. Fleksibilitas: Dengan memanfaatkan intercloud, pengguna diberi sumber daya dengan cara yang lebih fleksibel.
5. Penghematan daya: Dalam skenario intracloud, energi dapat dihemat dengan memanfaatkan migrasi mesin virtual (VM). Mesin virtual dapat ditransfer dari host yang kelebihan beban ke host yang kekurangan beban. Penghematan energi dapat dilakukan pada tingkat yang lebih tinggi dalam skenario inter*cloud* di mana migrasi dapat dilakukan di seluruh pusat data.

Intercloud membawa serangkaian tantangan baru yang harus diselesaikan. Internet adalah jaringan dari jaringan. Datang dengan standar untuk Internet sehingga berbagai jaringan dapat bekerja sama satu sama lain membutuhkan waktu sekitar 5 tahun. Dengan cara yang sama, berbagai masalah harus diatasi sebelum intercloud dapat digunakan. Berbagai penyedia *cloud* harus memiliki beberapa mekanisme untuk membangun kepercayaan di antara mereka sendiri. Masalah lainnya termasuk persyaratan SLA dan tarif penagihan. Beberapa penyedia *cloud* mungkin memiliki mekanisme autentikasi untuk menawarkan sumber daya mereka. Sederhananya, penyedia *cloud* harus cocok satu sama lain dalam hal sumber daya, kebijakan, teknologi, dll. Berbagai teknologi yang ada, seperti *Extensible Messaging and Presence Protocol* (XMPP), dapat digunakan untuk mewujudkan intercloud di antara penyedia *cloud* yang heterogen. Masalah lainnya termasuk skalabilitas, dukungan migrasi VM lintas penyedia *cloud*, migrasi sumber daya, keamanan, manajemen kebijakan, dan pemantauan.

InterCloud adalah operator akses *cloud* yang bertindak sebagai pintu gerbang ke berbagai platform *Software-as-a-Service* (SaaS) di antara mitra intercloud, seperti Microsoft Azure dan Amazon Web Services. Ini menyediakan akses ke platform aplikasi *out-source* dengan cara yang aman.

Ada berbagai arsitektur yang diusulkan dalam literatur yang bertujuan untuk menyelesaikan berbagai masalah terkait inter*cloud* seperti pertukaran data dan berbagi sumber daya.

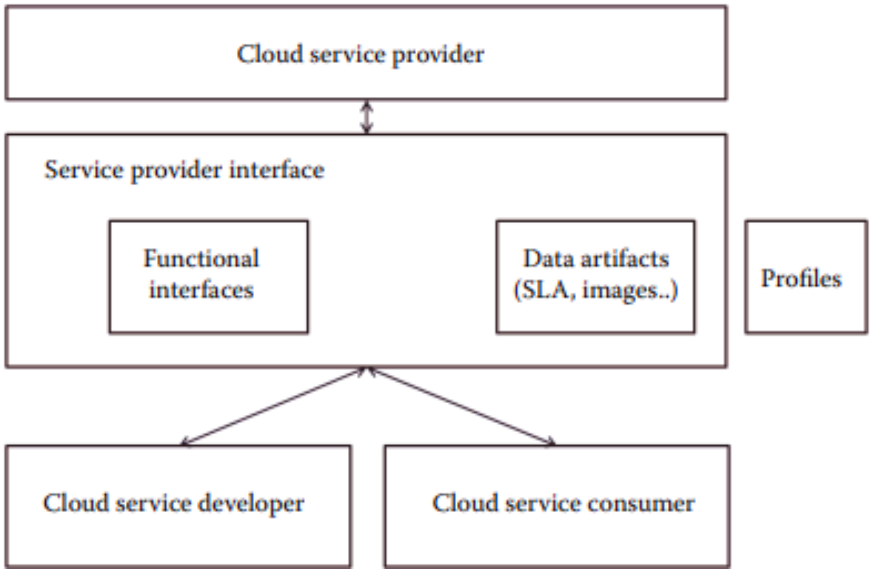
## 14.2 MANAJEMEN AWAN

Karena komputasi awan mendapatkan penerimaan yang luas di kalangan pengguna, ada kebutuhan untuk manajemen awan untuk memastikan berfungsinya layanan awan dengan baik. Manajemen *cloud* bertanggung jawab untuk mengelola infrastruktur *cloud*. Ini adalah solusi untuk masalah mewujudkan manfaat ekonomi yang diharapkan, dengan menggunakan kemampuan manajemen yang tepat.

Istilah manajemen *cloud* adalah nama yang diberikan untuk kumpulan perangkat lunak dan teknologi yang digunakan untuk mengatur dan memantau berbagai aplikasi *cloud*. Manajemen *cloud* memastikan bahwa layanan *cloud* berjalan secara optimal dan berinteraksi dengan koaplikasinya. Perangkat lunak manajemen *cloud* mungkin perlu menangani sumber daya yang heterogen. Itu harus memantau berbagai tugas seperti alokasi sumber daya.

Strategi manajemen *cloud* mencakup layanan pemantauan dan audit reguler, serta memulai dan mengelola rencana untuk pemulihan bencana.

Manajemen *cloud* biasanya menyediakan portal untuk pelanggan. Ini menyediakan otentikasi pengguna, enkripsi, dan manajemen anggaran atas nama berbagai perusahaan. Ada berbagai antarmuka manajemen *cloud* yang tersedia seperti EnStratus Enterprise *Cloud* Management Solution, yang menyediakan penyediaan, manajemen, dan otomatisasi aplikasi di *cloud* pribadi dan publik utama. Salah satu arsitektur referensi manajemen *cloud* yang diperkenalkan oleh *Distributed Management Task Force* (DMTF) ditunjukkan pada Gambar 14.4.




**Gambar 14.4** Arsitektur Referensi Layanan *Cloud*.

Komponen fungsional arsitektur adalah sebagai berikut:

1. Pengembang layanan *cloud*: Merancang, mengimplementasikan, dan memelihara template layanan.
2. Konsumen layanan *cloud*: Menyediakan akses ke layanan untuk pengguna layanan.
3. Penyedia layanan *cloud*: Memasok layanan *cloud* ke konsumen internal atau eksternal.
4. Artefak data: Elemen kontrol dan status dipertukarkan di seluruh antarmuka penyedia.
5. Antarmuka penyedia: Antarmuka yang memungkinkan konsumen untuk mengakses dan memantau layanan yang dikontrak.
6. Profil: Spesifikasi yang mendefinisikan asosiasi, metode, dan properti untuk domain manajemen.

Antarmuka manajemen *cloud* harus menjaga keamanan secara khusus. Jika penyerang mendapatkan akses ke antarmuka manajemen *cloud*, efek sampingnya bisa drastis.

## 14.3 AWAN SELULER

*Mobile Cloud computing* Forum mendefinisikan mobile *cloud computing* (MCC) sebagai berikut:

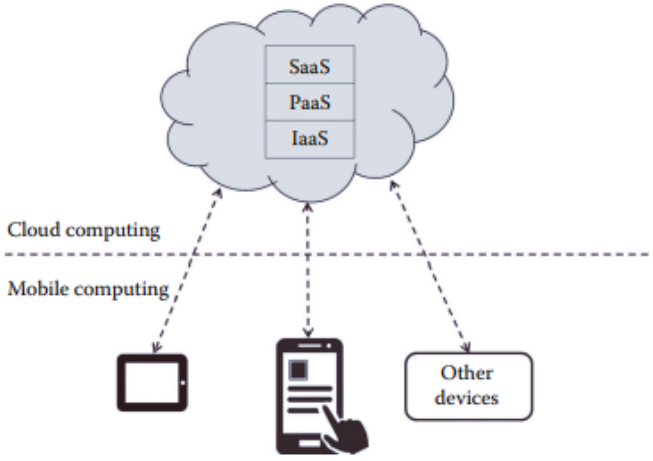
*Mobile Cloud computing* paling sederhana, mengacu pada infrastruktur tempat penyimpanan data dan pemrosesan data terjadi di luar perangkat seluler. Aplikasi *cloud* seluler memindahkan daya komputasi dan penyimpanan data dari ponsel ke *cloud*,

menghadirkan aplikasi dan komputasi seluler tidak hanya untuk pengguna smartphone, tetapi juga untuk pelanggan seluler yang lebih luas.

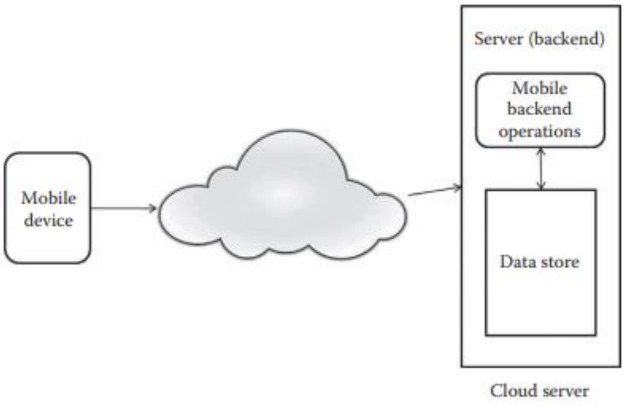
PKS pada dasarnya adalah persimpangan bidang komputasi awan dan jaringan seluler. Jaringan seluler pada dasarnya adalah jaringan yang menghubungkan pengguna seluler. Munculnya jaringan seluler ultracepat membuat domain *cloud* harus dibawa ke domain jaringan seluler. Bidang ini masih dalam tahap awal pengembangannya. MCC pada dasarnya memungkinkan pembuatan dan hosting aplikasi seluler melalui *cloud*. Ada berbagai masalah yang harus diselesaikan seperti migrasi VM langsung, keamanan, perlindungan privasi, dan toleransi kesalahan. Ada kemungkinan migrasi VM menjadi overhead di PKS.

Perangkat seluler biasanya memiliki daya dan sumber daya komputasi yang terbatas. Keterbatasan mobile computing dapat diatasi dengan MCC. Ini akan memungkinkan pengguna untuk mengakses platform dan aplikasi yang disediakan oleh *cloud* melalui perangkat seluler mereka. Di sini, pengguna tidak lagi dibatasi oleh sumber daya terbatas yang mereka miliki. Dengan demikian, aplikasi seluler yang lebih intensif komputasi dapat didukung oleh lebih banyak perangkat.

Ada banyak aplikasi *cloud* seluler, yang paling umum adalah Google Gmail dan Google Voice untuk iPhone. Diagram umum ditunjukkan pada Gambar 14.5.




**Gambar 14.5** Komputasi *cloud* seluler.




**Gambar 14.6** Server *cloud* jarak jauh yang menawarkan layanan ke perangkat seluler.

Dalam aplikasi seluler yang melibatkan pemrosesan gambar, pemrosesan bahasa alami, pencarian multimedia, dan sebagainya, kurangnya sumber daya pada perangkat seluler dapat diatasi dengan menyewa layanan yang ditawarkan oleh *cloud*. Layanan penawaran server *cloud* jarak jauh ditunjukkan pada Gambar 14.6.

Skenario lain adalah ketika perangkat seluler dapat menjadi bagian dari *cloud* dan menawarkan sumber dayanya untuk disewakan ke perangkat seluler lain. Dengan demikian, sumber daya kolektif dapat disediakan, asalkan mereka berada di sekitarnya. Skenario lain menggunakan cloudlet. Cloudlets pada dasarnya adalah kumpulan komputer multicore yang terhubung ke server *cloud* jarak jauh. Perangkat seluler biasanya bertindak sebagai perangkat klien tipis ke cloudlet.

Salah satu masalah utama di MCC adalah proses penyerahan pekerjaan ke *cloud*. Ini sangat bergantung pada jarak yang secara fisik memisahkan *cloud* dan perangkat seluler. Proses ini mungkin memerlukan biaya tambahan. Biaya ini juga harus dipertimbangkan. Ada banyak karya dalam literatur yang didedikasikan untuk analisis biaya-manfaat MCC.

Masalah besar lainnya yang tidak ada dalam komputasi awan tradisional adalah dukungan mobilitas pengguna. Harus ada mekanisme untuk mengidentifikasi lokasi klien seluler saat ini. Mobilitas pengguna seharusnya tidak mempengaruhi konektivitas ke *cloud*.

Efisiensi energi adalah aspek lain yang memerlukan penelitian intensif dalam konteks PKS. Arsitektur tertentu telah diusulkan untuk PKS. Mereka harus mempertimbangkan masalah privasi, keamanan, dan kepercayaan, di antara masalah lainnya.

Manfaat PKS adalah sebagai berikut:

1. Memperpanjang masa pakai baterai: Bagi pengguna smartphone, masa pakai baterai adalah salah satu masalah utama. Meskipun smartphone memiliki berbagai fungsi, durasi masa pakai baterai menurun secara drastis ketika lebih banyak beban kerja diserahkan ke smartphone. Dengan memindahkan eksekusi aplikasi yang membutuhkan komputasi intensif ke *cloud*, masa pakai baterai dapat ditingkatkan.
2. Meningkatkan kapasitas penyimpanan data dan daya pemrosesan: Setelah berurusan dengan masa pakai baterai, masalah umum berikutnya yang dihadapi oleh pengguna smartphone adalah kapasitas penyimpanan. *Cloud* memberikan solusi sederhana untuk masalah ini dimana data dapat diakses melalui jaringan nirkabel.
3. Meningkatkan keandalan: Saat menggunakan *cloud*, data dicadangkan di sejumlah perangkat lain. Dalam kasus kehilangan data yang tidak disengaja, data yang dicadangkan dapat diakses.
4. Meningkatkan skalabilitas: Dengan menggunakan *cloud*, skalabilitas aplikasi seluler juga dapat ditingkatkan dengan mudah.

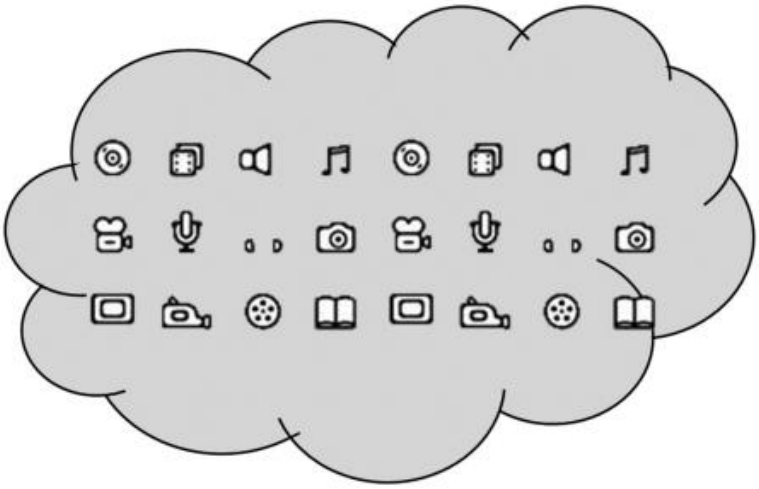
MCC menemukan aplikasi dalam perdagangan seluler, pembelajaran seluler, perawatan kesehatan seluler, dan banyak lagi.

## 14.4 MEDIA CLOUD

Data multimedia adalah bentuk utama dari data yang tersedia di Internet saat ini. Dalam hal video, ada berbagai macam format yang tersedia. Penyimpanan dan pemrosesan data multimedia memerlukan sejumlah besar fasilitas pemrosesan dan penyimpanan.

Teknologi yang muncul saat ini, komputasi awan, dapat dimanfaatkan untuk memproses dan menyimpan data tersebut secara efisien. Media *cloud* menyediakan penyimpanan untuk data media dan juga memungkinkan penyajian data menggunakan protokol pensinyalan media. Awan multimedia sederhana ditunjukkan pada Gambar 14.7.

*Cloud* media menyediakan pemrosesan data multimedia terdistribusi dan menyediakan layanan dengan *Quality of Service* (QoS) yang tinggi untuk data multimedia. Konten yang disimpan di *cloud* media dapat dengan mudah dialirkan ke berbagai klien seperti pemutar musik di mobil dan smartphone, menggunakan protokol seperti protokol kontrol transmisi (TCP), protokol diagram pengguna (UDP), dan protokol transportasi real-time (RTP).



**Gambar 14.7** Awan multimedia.

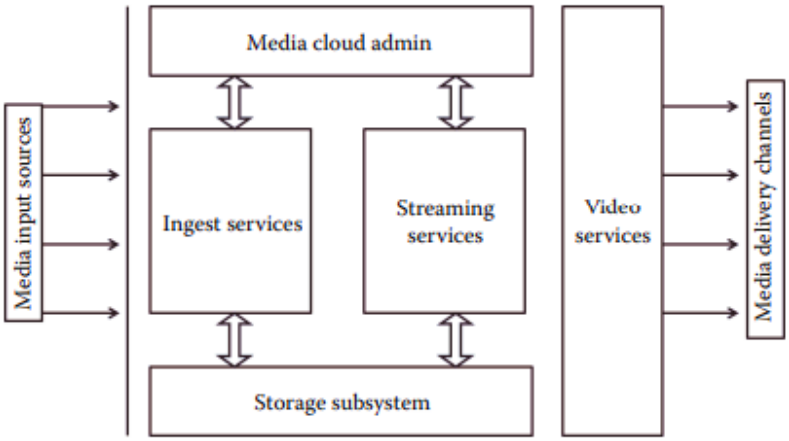
Proses streaming meliputi kegiatan buffering, rendering, recording, dan mixing data. Pengguna media *cloud* juga akan diberikan cara yang lebih mudah untuk berbagi data multimedia di antara mereka.

Ada banyak tantangan pada *cloud* media. Tantangan datang dalam bentuk format heterogen yang tersedia untuk berbagai jenis data multimedia, heterogenitas aplikasi, skalabilitas untuk beradaptasi dengan format media yang baru dikembangkan, dan pada saat yang sama membuat *cloud* multimedia menjadi *cloud* yang menguntungkan. Untuk meningkatkan keuntungan, risiko kegagalan harus dikurangi.

Referensi arsitektur media *cloud* disajikan pada Gambar 14.8. Arsitektur pada Gambar 14.8 menunjukkan layanan dasar yang harus disediakan oleh *cloud* media: manajemen penyimpanan dan infrastruktur, manajemen klaster dan jaringan, otomatisasi alur kerja, dll. Arsitektur ini berisi lima komponen utama:

1. Layanan administrasi awan
2. Menyerap layanan yang menerima masukan media dari berbagai sumber
3. Layanan streaming
4. Layanan video yang mengelola dan mengirimkan video melalui saluran media ke berbagai klien
5. Subsistem penyimpanan untuk cache konten dan pergerakan, penyimpanan, dan manajemen aset

Satu-satunya solusi untuk peningkatan pesat multimedia di tahun-tahun mendatang adalah komputasi awan. *Cloud* akan menjadi tuan rumah utama untuk semua data multimedia dan pemrosesan multimedia dalam waktu dekat. Untuk membuat sistem tidak terlalu rumit, pendekatan yang mengubah jenis data audio, video, dan multimedia lainnya yang heterogen ke dalam format standar lebih diinginkan. Area *cloud* media memiliki berbagai masalah yang belum terselesaikan seperti arsitektur *cloud* media, penyimpanan, pengiriman data multimedia, siaran seluler, efisiensi energi, dan penambangan media yang melibatkan pengambilan informasi dari data multimedia. Media *cloud* juga dapat digunakan untuk penyediaan untuk memenuhi persyaratan Video on Demand.




**Gambar 14.8** Komponen fungsional arsitektur *cloud* media

## 14.5 INTEROPERABILITAS DAN STANDAR

Interoperabilitas didefinisikan sebagai kemampuan untuk memindahkan beban kerja dari satu penyedia *cloud* ke penyedia *cloud* lainnya tanpa masalah. Dengan demikian, pengguna harus dapat menggunakan penyedia *cloud* secara bergantian atau harus dapat mengalihkan basis kerja mereka dari satu penyedia *cloud* ke penyedia *cloud* lainnya. Interoperabilitas adalah salah satu isu utama dalam komputasi awan. Salah satu masalah yang terkait dengan interoperabilitas adalah masalah vendor lock-in.

Setiap kali penyedia layanan *cloud* tidak mengizinkan pengguna untuk bermigrasi ke *cloud* lain, itu disebut vendor lock-in. Sederhananya, vendor mengunci pengguna agar tidak berpindah dari satu vendor ke vendor lainnya. Penguncian vendor telah menjadi masalah sejak lama. Di sini, setelah pengguna memilih satu penyedia layanan *cloud*, dia tidak akan dapat mengubah atau bermigrasi ke penyedia layanan lain tanpa memulai pekerjaannya lagi.

Masalah lain dengan interoperabilitas adalah penyedia layanan tidak mengizinkan pengguna untuk menggunakan produk atau komponen dari vendor lain dalam infrastruktur *cloud* mereka. Dengan demikian, pengguna harus menggunakan komponen yang diberikan oleh satu penyedia *cloud* saja.

Ada dua kemungkinan alasan untuk masalah ini. Salah satunya adalah vendor tidak ingin kehilangan pelanggannya, sehingga mereka tidak mengizinkan mereka untuk menggunakan komponen vendor lain atau bermigrasi ke vendor lain. Yang lainnya adalah

dukungan teknis yang sangat sedikit atau kemajuan yang sangat sedikit di bidang ini, jadi bahkan jika komponen *cloud* adalah open source atau tidak diatur oleh kebijakan atau aturan semacam itu, kecil kemungkinannya bahwa dua *cloud* akan dapat dioperasikan. . Interoperabilitas dan portabilitas berjalan beriringan. Ada tiga aspek portabilitas, yaitu portabilitas aplikasi, portabilitas platform, dan portabilitas infrastruktur. Jadi, jika masalah portabilitas teratasi, maka tidak akan memakan banyak waktu untuk membuat *cloud* dapat dioperasikan karena hanya ada beberapa masalah lagi yang perlu dipertimbangkan selain portabilitas. Semua masalah tersebut di atas disebabkan oleh satu alasan utama, yaitu standardisasi.

Masalah utama penyedia layanan *cloud* adalah mereka tidak memiliki standar umum untuk semua penyedia layanan *cloud*. Setiap penyedia layanan mengikuti standar mereka sendiri, tetapi standar ini bervariasi dari perusahaan ke perusahaan. Jadi, tidak ada dua perusahaan yang memiliki standar yang sama. Menurut laporan Carnegie Mellon University (CMU), terdapat perbedaan aspek standardisasi pada setiap model layanan *cloud*. Ada tiga model layanan dasar *cloud*: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), dan Software as a Service (SaaS). Standardisasi akan mempengaruhi ketiga model dengan cara yang berbeda.

Menurut CMU, model SaaS paling tidak diuntungkan oleh standar. Ini karena perangkat lunak semacam itu melibatkan persyaratan lisensi, dan standardisasi tidak akan berdampak lebih besar padanya. Demikian pula, PaaS juga mungkin tidak banyak diuntungkan dari model standardisasi. Namun, IaaS akan memiliki manfaat yang cukup besar karena beban kerja atau sumber daya dalam bentuk VM dan disk penyimpanan, dan jika standardisasi dibuat, maka pengguna dapat memigrasikan VM mereka dari satu penyedia layanan ke penyedia layanan lainnya. Ada banyak upaya yang dilakukan oleh berbagai organisasi untuk membuat standar. Ada beberapa upaya oleh beberapa komunitas untuk memiliki sistem yang dapat dioperasikan dan mempertahankan standar [6]. Proses ini akan berlanjut di masa mendatang, dan tidak lama lagi interoperabilitas akan menjadi milik setiap penyedia layanan *cloud*.

## 14.6 TATA KELOLA *CLOUD*

Tata kelola adalah istilah dalam dunia perusahaan yang umumnya melibatkan proses menciptakan nilai bagi organisasi dengan menciptakan tujuan strategis yang akan mengarah pada pertumbuhan perusahaan dan akan mempertahankan tingkat kontrol tertentu atas perusahaan. Tata kelola tidak harus bingung dengan manajemen. Kedua istilah tersebut, meski serupa, memiliki banyak perbedaan. Tata kelola menjadi gambaran di mana ada industri yang berkembang pesat yang melibatkan banyak sumber daya termasuk manusia. Tata kelola melibatkan pemeliharaan dan mengikuti kebijakan tertentu di seluruh perusahaan. Ini melibatkan pengambilan keputusan tingkat tinggi yang akan memengaruhi semua orang yang terkait dengan perusahaan.

Komputasi awan adalah salah satu teknologi yang berkembang pesat. Hampir semua perusahaan besar sudah mulai menggunakan *cloud*. Seiring berjalannya waktu, awan menjadi semakin besar dan luas. Tidak hanya sumber daya komputasi tetapi juga jumlah orang yang

bekerja di *cloud* semakin meningkat. Oleh karena itu, diperlukan mekanisme tata kelola untuk menjaga pertumbuhan *cloud*. Ada kebutuhan untuk mengatur semua sumber daya sedemikian rupa sehingga orang-orang yang terlibat atau terpengaruh oleh *cloud* secara langsung atau tidak langsung diuntungkan dan semua hal yang terjadi melalui *cloud* dipantau dengan baik. Tata kelola *cloud* sangat penting karena merupakan fakta yang diketahui bahwa organisasi yang dikelola dengan baik memiliki kemungkinan lebih tinggi untuk mempertahankan bisnis dan mempertahankan posisi mereka di industri. Jika diatur dengan baik, suatu organisasi dapat beradaptasi dengan perubahan dengan cepat.

Tata kelola *cloud* melibatkan rintangan tertentu. Ada beberapa tanggung jawab yang terlibat dalam tata kelola *cloud*, yang mencakup beberapa poin:

1. Kualitas layanan: Kualitas layanan adalah masalah utama lainnya sejauh menyangkut organisasi. Tidak ada metrik standar atau cara standar untuk memastikan kualitas layanan kepada pelanggan. Ada beberapa model atau algoritma yang diusulkan untuk memastikan kualitas layanan kepada pengguna. Perjanjian tingkat layanan (SLA) adalah parameter utama yang dipertimbangkan untuk memastikan QoS. SLA ini dianggap sangat penting. Negosiasi dilakukan antara penyedia layanan *cloud* dan pengguna *cloud* berdasarkan SLA ini. SLA memberikan fleksibilitas kepada pengguna *cloud*, dan pengguna *cloud* dapat mengklaim atau menuntut keadilan atau kompensasi jika ketentuan SLA dilanggar dengan cara apa pun oleh penyedia layanan. Oleh karena itu, merupakan tanggung jawab penyedia layanan untuk memastikan bahwa persyaratan SLA dipenuhi dengan biaya berapa pun.
2. Mematuhi hukum dan standar: Ini adalah salah satu bagian tersulit dalam hal *cloud*. Karena server back-end, disk penyimpanan, dan VM tersebar di seluruh dunia, ada banyak masalah yang muncul karena data disimpan di *cloud*. Masalahnya muncul ketika orang dari satu negara mencoba mengakses data yang disimpan di negara lain. Hukum kedua negara mungkin berbeda. Oleh karena itu, sampai pada kesimpulan tentang undang-undang mana yang harus diterapkan pada pengguna mana dan pada akhirnya apa yang harus menjadi undang-undang standar untuk perusahaan sangatlah sulit.
3. Beradaptasi dengan mekanisme layanan yang berubah: Ada berbagai mekanisme layanan yang terlibat dalam penyampaian layanan. Dari waktu ke waktu, mekanisme layanan ini berubah dan penyedia layanan harus memiliki kemampuan untuk beradaptasi dengan mekanisme layanan yang berbeda secepat mungkin.
4. Masalah privasi data: Undang-undang yang terkait dengan privasi data telah menjadi masalah sejak lama. Data pengguna yang disimpan tidak boleh dilihat oleh orang luar atau bahkan penyedia layanan. Tidak ada kepastian apakah detail data yang disimpan oleh pengguna aman. Jadi, ini adalah salah satu masalah terpenting yang perlu diselesaikan.
5. Multitenancy: Multitenancy adalah properti *cloud* yang melibatkan berbagi ruang kerja (instance) oleh banyak pengguna (klien). Ini adalah salah satu sifat penting *cloud* yang membuatnya populer. Meskipun bagus, ada beberapa masalah keamanan. Seorang pengguna dapat menembus penghalang keamanan dan dapat mengakses data orang

lain, karena semua pengguna berbagi satu sistem. Memberikan keamanan kepada pengguna masih menjadi masalah yang perlu ditangani.

6. Tata kelola di dalam organisasi (internal): Aturan di dalam organisasi untuk orang-orang yang terkait dengan proyek *cloud* harus diberikan. Karena proyek-proyek ini besar dan kompleks, aturannya harus ditentukan dengan jelas. Jumlah orang yang bekerja di *cloud* dan jumlah sumber daya yang digunakan untuk *cloud* di perusahaan sangat tinggi. Oleh karena itu, perlu ada undang-undang yang terkait dengan masalah ini, yaitu bagaimana cara menggunakan sumber daya ini juga harus ditentukan.

Berdasarkan permasalahan tersebut, peneliti telah mengusulkan beberapa model tata kelola seperti model Tata Kelola Siklus Hidup dan model Tata Kelola *Cloud* Lewam Woldu. Karena tata kelola *cloud* mirip dengan tata kelola dalam arsitektur berorientasi layanan (SOA) dan dengan demikian keduanya dianggap sebagai bagian dari tata kelola TI. Kawasan ini merupakan salah satu kawasan yang sedang berkembang.

Ada beberapa aspek tata kelola yang dapat diotomatisasi, dan beberapa perusahaan telah menghadirkan alat untuk tata kelola *cloud*. Alat-alat ini termasuk alat tata kelola *cloud* dengan nama alat Dell Platform Agility dengan jaring layanan dll. Penelitian aktif sedang berlangsung di bidang ini, dan perusahaan serta peneliti menghasilkan beberapa model dan alat untuk tata kelola.

## 14.7 KECERDASAN KOMPUTASI DI *CLOUD*

Menurut Andries P. Engelbrecht, *Computational Intelligence* (CI) adalah studi tentang mekanisme adaptif untuk mengaktifkan atau memfasilitasi perilaku cerdas dalam lingkungan yang kompleks dan berubah. Mekanisme ini termasuk paradigma kecerdasan buatan (AI) yang menunjukkan kemampuan untuk belajar atau beradaptasi dengan situasi baru, menggeneralisasi, abstrak, menemukan, dan mengasosiasikan. Aspek penting dari CI adalah adaptivitas. Bidang ini ada selama hampir empat dekade. Para ilmuwan telah menggunakan algoritma ini untuk memecahkan berbagai jenis masalah. CI pada dasarnya digunakan untuk masalah yang tidak dapat diselesaikan dengan menggunakan algoritma konvensional. Ada banyak masalah waktu nyata yang tidak dapat diselesaikan dengan menggunakan metode konvensional, dan dalam kasus ini, beberapa pendekatan alami yang praktis terbukti menjadi pilihan yang lebih baik.

CI mencakup semua algoritma heuristik yang terinspirasi oleh proses alami. Beberapa contoh yang populer adalah algoritma genetika dan pengoptimalan segerombolan. Algoritma genetika berada di bawah subdivisi yang disebut algoritma evolusioner. Algoritma evolusioner ini terinspirasi dari proses evolusi makhluk hidup. Demikian pula, pengoptimalan segerombolan juga termasuk dalam kategori ini. Teknik pengoptimalan kawanan terinspirasi oleh sekolah ikan atau kawanan burung, yaitu properti perilaku kelompok mereka digunakan. Sistem kekebalan buatan adalah salah satu metode yang didasarkan pada sistem kekebalan tubuh kita. Demikian pula, ada beberapa metode lain yang didasarkan pada proses alami. Metodologi komputasi ini banyak digunakan untuk masalah yang berhubungan dengan optimasi. Pendekatan ini memiliki penggunaan yang luas dalam komputasi awan.

Komputasi awan secara sederhana menawarkan layanan (sumber daya) kepada pelanggan. Ini melibatkan banyak sumber daya di bagian belakang. Mengelola sumber daya adalah tugas yang sulit. Manajemen sumber daya ini melibatkan beberapa tugas penting seperti penjadwalan sumber daya, penyediaan, konsolidasi, dan migrasi. Beberapa tugas yang disebutkan di atas seperti penjadwalan tidak dapat memiliki metode lengkap. Cara utama penjadwalan sumber daya adalah penjadwalan alur kerja dan penjadwalan tugas. Masalah penjadwalan seperti itu adalah masalah nondeterministic polynomial-time hard (NP-hard). Masalah NP-hard adalah masalah sulit yang tidak memiliki solusi lengkap dan hanya dapat memiliki solusi perkiraan. Jadi untuk masalah seperti ini, teknik optimisasi seperti algoritma CI dapat digunakan. Teknik optimasi biasanya dicirikan oleh sifat-sifat yang memberikan solusi perkiraan yang baik yang mendekati solusi yang benar. Di atas adalah beberapa contoh. Algoritma ini menghabiskan lebih sedikit waktu daripada algoritma pencarian lengkap. Semakin banyak masalah menjadi kompleks, semakin besar kemungkinan menggunakan algoritma CI.

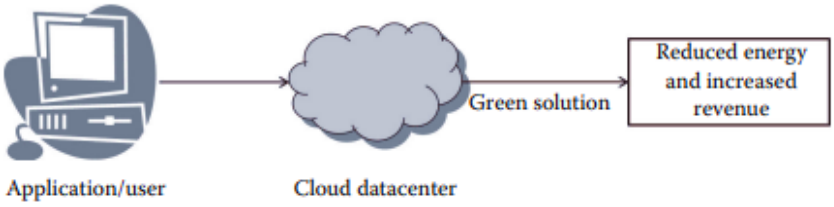
*Keuntungan*

- Cocok untuk masalah kompleks yang tidak memungkinkan pencarian menyeluruh
- Mengkonsumsi lebih sedikit waktu daripada rekan yang melelahkan

## 14.8 AWAN HIJAU

Pusat data adalah tulang punggung komputasi awan. Pusat data yang menampung awan sering menghabiskan banyak energi. Pusat data adalah fasilitas yang menampung sistem komputer dan komponen terkaitnya. Daya yang dikonsumsi oleh pusat data terutama terdiri dari daya yang dibutuhkan untuk menjalankan peralatan sebenarnya dan daya yang digunakan oleh perangkat untuk mendinginkan peralatan. Untuk mendinginkan sistem di pusat data, kami biasanya menggunakan AC presisi yang mengontrol suhu dan kelembapan sepanjang hari dan juga dapat dikelola dari jarak jauh.

Semakin banyak orang beralih ke komputasi awan, energi yang dikonsumsi olehnya menjadi signifikan. Di era yang sangat memperhatikan lingkungan yang lebih hijau, bidang komputasi awan ini juga harus dipertimbangkan. Dari kebutuhan dasar inilah komputasi awan hijau muncul. Komputasi awan hijau pada dasarnya adalah solusi komputasi untuk masalah konsumsi energi. Solusi ini juga bertujuan untuk mengurangi Operational Expenses (OPEX). Kenaikan biaya operasional mengurangi keuntungan marjinal penyedia layanan *cloud*. Konsumsi daya pusat data juga memiliki dampak yang tidak dapat diabaikan pada peningkatan emisi karbon dan pemanasan global. Dengan demikian, ada kebutuhan untuk mengembangkan solusi hemat energi untuk komputasi awan. Untuk itu, analisis mendalam tentang efisiensi daya awan perlu dilakukan (Gambar 14.9).




**Gambar 14.9** Komputasi awan hijau.

Arsitektur pusat data terdiri dari berbagai jenis seperti dua tingkat dan tiga tingkat. Dalam semua arsitektur ini, efisiensi energi harus dianalisis. Efisiensi pusat data biasanya dihitung dengan mempertimbangkan kinerja yang dihasilkan per watt.

Efektivitas penggunaan daya (PUE): Ini membandingkan energi yang digunakan untuk komputasi dengan energi yang digunakan untuk pendinginan dan overhead lainnya. Nilai idealnya adalah 1,0. Itu dapat dihitung sebagai

$$\text{PUE} = \frac{\text{Total Facility Energy}}{\text{IT Equipment Energy}}$$

PUE terendah yang dicapai dalam skenario saat ini adalah 1,13 di pusat data Google.
Efisiensi infrastruktur pusat data (DCiE): Ini kebalikan dari PUE:

$$\text{DCiE} = \frac{1}{\text{PUE}}$$

Penelitian telah menunjukkan bahwa server yang menganggur mengonsumsi sekitar 66% energi dibandingkan dengan konfigurasi yang terisi penuh. Pengelolaan modul memori dan faktor lain berkontribusi pada konsumsi energi ini. Dengan demikian, kami dapat menghemat banyak daya jika beban kerja terkonsentrasi pada jumlah minimum server komputasi, sehingga memungkinkan penutupan server yang menganggur. Pada dasarnya, berbagai teknik manajemen daya dalam arsitektur komputer adalah sebagai berikut:

1. Penskalaan voltase dinamis (DVS): Menambah atau mengurangi voltase pada suatu komponen.
2. Penskalaan frekuensi dinamis (DFS): Ini juga disebut pelambatan CPU, di mana frekuensi mikroprosesor disesuaikan untuk menghemat daya atau mengurangi jumlah panas yang dihasilkan.
3. Dynamic shutdown (DNS): Skema ini secara selektif mematikan komponen yang menganggur atau kurang dimanfaatkan.

Lapisan hijau dalam komputasi awan disediakan oleh virtualisasi. Virtualisasi adalah menciptakan entitas virtual. Menggunakan virtualisasi, banyak pengguna dapat diakomodasi pada satu host. Jika tidak ada virtualisasi, setiap pengguna harus dialokasikan mesin fisik yang terpisah. Hal ini dapat dihindari dengan membuat entitas virtual yang terisolasi untuk setiap pengguna pada mesin fisik yang sama. Dengan demikian, virtualisasi adalah alat untuk pemanfaatan sumber daya secara efisien. Itu juga bisa berfungsi sebagai alat untuk menghemat energi. Konsolidasi beban kerja dan konsolidasi server dapat digunakan untuk mengurangi konsumsi energi. Saat server kelebihan beban, VM di server dapat diangkut ke server lain yang kekurangan beban. Jika ada banyak server yang kekurangan muatan, VM dapat dikonsolidasikan ke satu atau beberapa server, memungkinkan server yang menganggur dimatikan.

Ada berbagai simulator untuk mensimulasikan lingkungan komputasi awan. Namun tidak banyak dari mereka yang mengukur efisiensi energi *cloud*. Untuk mengatasi masalah ini,

Simulator Green*Cloud* dikembangkan pada tahun 2013 oleh tim yang dipimpin oleh Dzmitry Kliazovich di University of Luxembourg. Ini memberikan simulasi halus awan hemat energi. Simulator berfokus pada komunikasi antara berbagai elemen di *cloud*. Pendekatan ini diadaptasi karena lebih dari 30% energi total dikonsumsi oleh elemen untuk komunikasi.

Ada berbagai karya dalam literatur yang termasuk dalam bidang komputasi awan hijau. Ada strategi penjadwalan hemat energi yang memanfaatkan migrasi VM. *Green routing* bertujuan untuk menyediakan layanan routing dengan cara yang hemat energi. Jaringan hijau adalah area yang berfokus pada masalah efisiensi energi dalam jaringan. Karya penting lainnya di bidang ini adalah Arsitektur *GreenCloud* yang diusulkan oleh Garg dan Buyya. Mereka mengusulkan kerangka kerja yang membatasi konsumsi energi awan. Mereka mengusulkan arsitektur dengan lapisan perantara yang disebut GreenBroker. GreenBroker ini menggunakan direktori yang berisi rincian emisi karbon dari berbagai penyedia *cloud* untuk memberikan penyedia paling ramah lingkungan kepada pelanggan sekaligus memaksimalkan keuntungan. Informasi emisi karbon dari berbagai penyedia *cloud* di tingkat IaaS diukur dengan menggunakan pengukur energi. Pada tingkat PaaS, alat profil energi digunakan.

Strategi hemat energi untuk pusat data dapat dibagi menjadi empat bagian besar:

1. Teknik hemat energi untuk server
2. Teknik hemat energi untuk jaringan
3. Teknik hemat energi untuk server dan jaringan
4. Pendinginan dan energi terbarukan

Di bawah kategori keempat, pekerjaan penting telah dilakukan oleh Baikie dan Hosman [11] di mana upaya dilakukan untuk mengurangi energi yang digunakan oleh pusat data. Ini termasuk metode seperti menggunakan energi terbarukan untuk menggerakkan pusat data dan sebagainya.

## 14.9 CLOUD ANALYTICS

*Cloud Analyst* adalah proses melakukan bisnis apa pun atau analisis intensif data (analitik data) di *cloud* publik atau pribadi. Analisis data adalah proses pemeriksaan data yang belum diproses atau mentah untuk membuat beberapa kesimpulan yang berarti dari data tersebut. Hasilnya mungkin termasuk nilai atau sekumpulan nilai atau grafik. Analisis data sangat populer. Beberapa peneliti di seluruh dunia menggunakan teknik ini secara efektif, dan berdasarkan hasilnya, keputusan penting diambil. Jika dilakukan pada data dalam jumlah besar yang bisa berada dalam kisaran terabyte dan petabyte, maka disebut sebagai big data analytics. Analitik data besar telah menjadi kata kunci baru-baru ini di industri ini. Saat ini, hampir seluruh dunia menggunakan Internet, dan jumlah data yang dihasilkan per hari sangat besar.

Ada beberapa tempat dan beberapa perusahaan yang menganalisis beberapa bagian dari data tersebut untuk mendapatkan kesimpulan yang berarti. Misalnya, situs jejaring sosial dapat dipertimbangkan. Biasanya, jumlah pengguna situs jejaring sosial banyak. Misalkan administrator situs ingin menganalisis pengguna sesuai dengan permainan yang mereka minati, dia dapat melakukannya dengan menganalisis seluruh data mereka berdasarkan satu atau lebih parameter yang menentukan atau memperkirakan hasil. Ada beberapa tempat lain

selain Internet di mana data dalam jumlah besar dihasilkan, dan data ini memerlukan analisis lengkap. Misalnya, jika perusahaan asuransi ingin mencari pelanggan terbaik dan ingin menilai pelanggan mereka berdasarkan pembayaran bulanan mereka, mereka dapat melakukannya dengan menggunakan analitik. Demikian pula, ada beberapa aplikasi analitik data.

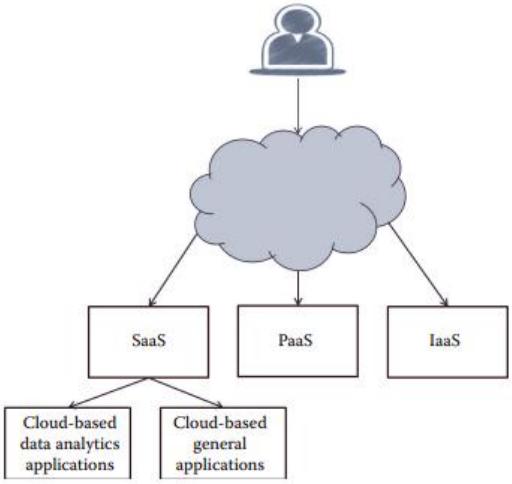
Menganalisis sejumlah kecil data itu mudah, tetapi ketika jumlah data bertambah, itu menjadi sulit. Operasi analitik data yang dilakukan di dunia saat ini adalah pada data yang berukuran terabyte dan petabyte. Untuk melakukan operasi intensif data semacam ini, diperlukan sumber daya back-end yang berat, dan yang terpenting daya pemrosesan tinggi. Ini akan sangat merugikan perusahaan atau organisasi. Perusahaan atau organisasi terkadang tidak mampu membeli sumber daya sebanyak itu, karena membeli dan memeliharanya merupakan masalah besar. Dalam hal ini, tidak ada pilihan lain bagi organisasi selain menyewa sumber daya. Ada situasi sebelumnya ketika kekuatan semacam ini dibutuhkan. Misalnya, Large Hadron Collider (LHC), yang digunakan oleh CERN, merekam data dalam petabyte per minggu dan ada kebutuhan untuk memproses data tersebut. Para ilmuwan tidak memiliki sumber daya sendiri, jadi mereka mengandalkan teknologi jaringan untuk sumber daya.

Model komputasi grid adalah salah satu teknologi komputasi yang memungkinkan pengguna untuk menggunakan sumber daya dalam jumlah besar dengan basis bayar per penggunaan. Teknologi ini terutama ditujukan bagi organisasi penelitian untuk menggunakan sumber daya pemrosesan yang luas yang tersedia. Sumber daya ini disediakan dengan biaya, dan terutama ini digunakan untuk aplikasi ilmiah kelas atas di bidang astronomi, fisika, bioinformatika, dll. Ini kadang-kadang disebut jaringan utilitas dan dianggap sebagai alasan keberhasilan banyak data tingkat tinggi. proyek intensif. Mirip dengan grid, *cloud* juga bisa digunakan untuk menyewa sumber daya. Komputasi awan adalah model komputasi di mana sumber daya, perangkat lunak, dan platform utama ditawarkan sebagai layanan. Ada perbedaan besar antara komputasi grid dan *cloud*. Tidak seperti grid, layanan *cloud* dibagi menjadi beberapa jenis. Dalam konteks ini, dua tipe utama adalah privat dan publik. *Cloud* publik menawarkan layanan kepada semua orang di seluruh dunia, sedangkan *cloud* pribadi dibatasi dan hanya dapat digunakan oleh organisasi atau individu. Ketika *cloud* pribadi atau publik digunakan untuk analisis data, itu disebut analitik *cloud*. Analitik *cloud* sangat populer karena analog dengan platform grid dan memiliki keunggulan bahwa siapa pun di dunia dapat menggunakan platform ini untuk melakukan analitik.

Orang-orang dapat menggunakan layanan *cloud* publik atau pribadi ini untuk menganalisis data dalam jumlah besar, dan mereka dapat membayar sesuai dengan apa yang mereka gunakan; tidak perlu membeli sumber daya apa pun. Ini sangat berguna bagi perusahaan secara real time, karena ini dapat digunakan kapan saja diperlukan, yaitu hanya jika diperlukan dan tidak perlu memelihara sumber daya perangkat keras apa pun. Gambar 14.10 menggambarkan klasifikasi *cloud analytics.*

Aspek lain dari analitik *cloud* adalah alih-alih menggunakan *cloud* pribadi atau *cloud* publik, orang dapat menggunakan aplikasi *cloud* yang dibuat di *cloud* dan dirancang khusus untuk analisis data. Jadi, alih-alih menggunakan sumber daya secara langsung, model pengiriman yang disebut SaaS digunakan. Di sini, perangkat lunak berbasis *cloud* dibuat oleh penyedia dan pengguna dapat menganalisis data dengan menggunakan perangkat lunak

tersebut, yang biasanya berupa aplikasi web. Aplikasi *cloud* semacam ini akan memungkinkan pengguna untuk menggunakan perangkat apa pun yang dapat mengakses aplikasi web sederhana.




**Gambar 14.10** Analitik awan.

## 14.10 RINGKASAN

Bab ini secara singkat menjelaskan topik lanjutan dalam komputasi awan. Topik yang disertakan terkait dengan kemajuan terkini di bidang ini. Semua topik yang dibahas memiliki dampak potensial. Topik-topik tersebut penting sebagai aplikasi atau memiliki kepentingan penelitian tertentu. Topik yang dibahas secara singkat adalah *mobile cloud, intercloud*, media *cloud*, *interoperability and standards, cloud governance, green cloud, cloud analytics*, dan dampak CI dalam *cloud computing* dan *cloud* management. Semua topik di atas dibahas dengan ilustrasi manfaat masing-masing teknologi dan dampaknya bagi industri atau akademisi.

**Tinjau Poin**

- Federation: Federation of *cloud* adalah penerapan dan pengelolaan beberapa layanan *cloud computing* untuk menyesuaikan kebutuhan bisnis.
- Multi*cloud*: Multi*cloud* melibatkan dua atau lebih penyedia layanan *cloud*.
- Federasi inter*cloud* peer-to-peer: Ini adalah salah satu jenis federasi *cloud* di mana tidak ada entitas pusat.
- Federasi inter*cloud* terpusat: Ini adalah jenis federasi *cloud* yang melibatkan entitas pusat.
- AC presisi: Ini adalah jenis AC yang digunakan untuk pendinginan di pusat data.
- Biaya Operasional (OPEX): Biaya Operasional adalah biaya berkelanjutan untuk menjalankan produk, bisnis, atau sistem.
- Penskalaan voltase dinamis (DVS): Ini adalah proses penyesuaian level voltase yang digunakan di pusat data secara dinamis.
- Penskalaan frekuensi dinamis (DFS): Ini melibatkan perubahan frekuensi mikroprosesor secara dinamis.
- Shutdown dinamis (DNS): Ini adalah teknik manajemen daya di mana server yang menganggur dimatikan.

- Konsolidasi server: Ini adalah proses mengurangi jumlah server yang kurang dimanfaatkan.
- Cloudlet: Cloudlet pada dasarnya dapat dilihat sebagai pusat data mini.
- Penguncian vendor: Dalam masalah penguncian vendor, penyedia layanan *cloud* tidak mengizinkan pengguna untuk bermigrasi ke *cloud* lain.
- Analitik awan: Analitik awan adalah proses melakukan bisnis apa pun atau analisis intensif data (analitik data) di awan publik atau pribadi.
- Multitenancy: Multitenancy adalah properti *cloud* yang melibatkan berbagi ruang kerja (instance) oleh banyak pengguna (klien).
- Analisis data: Analisis data adalah proses pemeriksaan data yang belum diproses atau data mentah untuk membuat beberapa kesimpulan yang berarti dari data tersebut.

**Latihan Soal**

1. Apa saja jenis topologi *intercloud?*
2. Apa perbedaan antara *multicloud* dan federasi *cloud*?
3. Sebutkan berbagai keunggulan *intercloud.*
4. Apa saja teknik yang berbeda untuk manajemen daya dalam arsitektur komputer?
5. Berikan garis besar strategi hemat energi untuk pusat data.
6. Tentukan metrik yang digunakan untuk mengukur efisiensi pusat data.
7. Diskusikan berbagai skenario di mana PKS dapat digunakan.
8. Apa isu utama dalam *cloud* multimedia?
9. Apa perlunya manajemen *cloud*?
10. Mengapa diperlukan standardisasi?
11. Apa keuntungan menggunakan CI di *cloud*?
12. Jelaskan masalah vendor lock-in.
13. Mengapa tata kelola *cloud* diperlukan?
14. Apa dua cara menggunakan analitik *cloud*?
15. Apa itu SLA?

# DAFTAR PUSTAKA


Alam, N. Survey on hypervisors. School of Informatics and Computing, Indiana University, Bloomington, IL.

Alizadeh, M., A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). SIGCOMM Computer Communications Review 40(4): 63–74, August 2010. Available

Alizadeh, M., A. Javanmard, and B. Prabhakar. Analysis of DCTCP: Stability, convergence and fairness. Proceedings of the ACM SIGMETRICS, Joint International Conference on Measurement and Modeling of Computer Systems, ser. SIGMETRICS'11. ACM, New York, 2011, pp. 73–84. Available

Badger, L. et al. Cloud computing synopsis and recommendations. NIST Special Publication 800: 146, 2012.

Baikie, B. and L. Hosman Green cloud computing in developing regions Moving data and processing closer to the end user. Telecom World (ITU WT), 2011 Technical Symposium at ITU, Geneva, Switzerland, October 24–27, 2011, pp. 24–28.

Beal, V. Cloud computing explained, 2012. http://www.webopedia.com/quick_ref/cloud_computing.asp. Accessed May 25, 2012.

Best practices for cloud computing multi-tenancy. White Paper, IBM Corporation, 2003.

Betts, D., A. Homer, A. Jezierski, M. Narumoto, and H. Zhang. Developing Multi-Tenant Applications for the Cloud on the Microsoft Windows Azure. Microsoft Press, 2010.

Bioh, M. and D. Earhart. Security issues that affect cloud computing data storage. www.slideshare.net, 2009. Accessed November 2, 2014.

Bitar, N., S. Gringeri, and T. J. Xia. Technologies and protocols for data center and cloud networking. IEEE Communications Magazine 51(9): 24–31, 2013.

Building successful enterprise SaaS apps for the cloud. White Paper. THINKstrategies, Inc., 2011.

Buyya, R. and K. Sukumar. Platforms for building and deploying applications for cloud computing. arXiv preprint arXiv:1104.4379, 2011.

Buyya, R., R. Ranjan, and R. N. Calheiros. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. International Conference on High Performance Computing and Simulation, 2009 (HPCS'09). IEEE, 2009, pp. 1–11.

Bykov, S. et al. Orleans: A framework for cloud computing. Technical Report MSR-TR-2010-159, Microsoft Research, 2010.

Carbone, M., W. Lee, and D. Zamboni. Taming virtualization. IEEE Security and Privacy 6(1): 65–67, 2008.

Chauhan, N. S. and A. Saxena. A green software development life cycle for cloud computing. IT Professional 15(1): 28–34, 2013.

Chen, W. K. Virtualization for Dummies. Hoboken, NJ: Wiley Publishing, Inc., 2007. Goth, G. Virtualization: Old technology offers huge new potential. IEEE Distributed

Chen, Y., R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding TCP incast throughput collapse in datacenter networks. Proceedings of the First ACM Workshop on Research on Enterprise Networking, ser. WREN'09. ACM, New York, 2009, pp. 73–82.

Cheng, L., C.-L. Wang, and F. C. M. Lau. PVTCP: Towards practical and effective congestion control in virtualized datacenters. 21st IEEE International Conference on Network Protocols, ser. ICNP 2013. IEEE, 2013.

Chong, R. F. Designing a database for multi-tenancy on the cloud: Considerations for SaaS vendors. Technical article, IBM Developer Works.

Cloud computing issues and impacts. White Paper, Ernst and Young. Cloud computing security and privacy issues. White Paper, CEPIS.

Cloud Incubator While Paper – DMTF, A White Paper from the Open Cloud Standards Incubator.

Cloud storage for cloud computing. OpenGrid Forum, SNIA, Advancing Storage and

Cordeiro, T., D. Damalio, N. Pereira, P. Endo, A. Palhares, G. Gonçalves, D. Sadok et al. Open source cloud computing platforms. 2010 9th International Conference on Grid and Cooperative Computing (GCC). IEEE, 2010, pp. 366–371.

Costa, P., M. Migliavacca, P. Pietzuch, and A. L. Wolf. NaaS: Network-as-a-service in the cloud. Hot-ICE, 2012.

Curran, K., S. Carlin, and M. Adams Security issues in cloud computing. Journal of Network Engineering 4069–4072, 2011.

D. Zagorodnov. The eucalyptus open-source cloud-computing system. Ninth IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009 (CCGRID'09). IEEE, 2009, pp. 124–131.

da Silva, E.A.N. and D. Lucredio. Software engineering for the cloud: A research roadmap. 26th Brazilian Symposium on Software Engineering (SBES), September 23–28, 2012, pp. 71–80.

Das, T. and K. M. Sivalingam. TCP improvements for data center networks. 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS). IEEE, 2013, pp. 1–10.

Dean, J. and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1): 107–113, 2008. Accessed December 8, 2013.

Dean, J. and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Proceedings of the Sixth Symposium on Operating System Design and Implementation, 2004.

Dean, J. and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. ACM Communications, 51: 107–113, January 2008.

Dumbre, A., S. S. Ghag, and S. P. Senthil. Practising Agile software development on the Windows Azure platform. Infosys Whitepaper, 2011.

Ekanayake, J., Pallickara, S., and G. Fox. Mapreduce for data intensive scientific analyses. IEEE Fourth International Conference on eScience, Indianapolis, IN. IEEE, Washington, DC, 2008.

Epstein, J., A. P. Black, and S. Peyton-Jones. Towards Haskell in the cloud. ACM SIGPLAN Notices 46(12): 118–129, 2011.

Fernando, N., S. W. Loke, and W. Rahayu Mobile cloud computing: A survey. Future Generation Computer Systems 29(1): 84–106, 2013.

Gagnon, S., V. Nabelsi, K. Passerini, and K. Cakici. The next web apps architecture: Challenges for SaaS vendors. IT Professional 13(5): 44–50, 2011.

Garg, S. K. and R. Buyya Green cloud computing and environmental sustain- ability. In S. Murugesan and G. R. Gangadharan (eds.), Harnessing Green IT: Principles and Practices.

Gschwind, M. Multicore Computing and the Cloud: Optimizing Systems with Virtualization. IBM Corporation, 2009.

Guha, R. and D. Al-Dabass. Impact of Web 2.0 and cloud computing platform on software engineering. 2010 International Symposium on Electronic System Design (ISED), Bhubaneswar, India, December 20–22, 2010. IEEE, New York, 2010, pp. 213–218.

Gupta, P. and S. Gupta Mobile cloud computing: The future of cloud. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE) 1(3), September 2012.

Hamlen, K., M. Kantarcioglu, L. Khan, and B. Thuraisingham. Security issues for cloud computing. International Journal of Information Security and Privacy 4(2): 39–51, 2010.

Hashizume, K., D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez. An analysis of security issues for cloud computing. Journal of Internet Services and Applications 4: 5, 2013.

He, Y. The lifecycle process model for cloud governance, 2011.

Herbst, N.R., S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, 2013.

Hurwitz, J. The role of the operating system in cloud environments. A Hurwitz White Paper, 2011.

Hwang, J., J. Yoo, and N. Choi. IA-TCP: A rate based incast-avoidance algorithm for TCP in data center networks. IEEE ICC 2012, Ottawa, Canada, 2012.

Iiard, M., M. Budiu, and Y. Yuan. Dryad: Distributed data-parallel programs from sequential building blocks. Operating Systems Review 41(3): 59–72, 2007.

Jamal, M. H. et al. Virtual machine scalability on multi-core processors based servers for cloud computing workloads. IEEE International Conference on Networking, Architecture, and Storage, 2009 (NAS 2009), Hunan, China, July 9–11, 2009. IEEE, New York, 2009, pp. 90–97.

Jayaraj, A., J. John Geevarghese, K. Rajan, U. Kartha, and V. Samuel Varghese. Programming Models for Clouds.

Jin, C. and R. Buyya. Mapreduce programming model for. NET-based cloud computing. In: H. Sips, D. Epema, and H.-X. Lin (eds.), Euro-Par 2009 Parallel Processing. Springer, Berlin, Germany, 2009, pp. 417–428.

Jin, C. and R. Buyya. Mapreduce programming model for. Net-based cloud computing. Euro-Par 2009 Parallel Processing, Delft, The Netherlands. Springer, Berlin, Germany, 2009, pp. 417–428.

Kang, S., J. Myung, J. Yeon, S. Ha, T. Cho, J. Chung, and S. Lee. A general maturity model and reference architecture for SaaS. Proceedings of Database Systems for Advanced Applications (DASFAA 2010), Part II, LNCS 5982, pp. 337–346.

Kliazovich, D., P. Bouvry, and S. U. Khan GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing 62(3): 1263–1283, 2012.

Kliazovich, D., P. Bouvry, Y. Audzevich, and S. U. Khan. GreenCloud: A packet- level simulator of energy-aware Cloud computing data centers. 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), Miami, FL, December 6–10, 2010, pp. 1–5.

Kurose, J. F. and K. W. Ross. Computer Networking: A Top Down Approach, 6th edn. Addison-Wesley, 2012.

Kuyoro, S. O., F. Ibikunle, and O. Awodele. Cloud computing security issues and challenges. International Journal of Computer Networks (IJCN) 3(5): 247–255, 2011.

Li, Y., W. Li, and C. Jiang. A survey of virtual machine system: Current technology and future trends. 2010 Third International Symposium on Electronic Commerce and Security (ISECS), IEEE Computer Society, 2010, pp. 332–336.

Liu, H. and D. Orban. Gridbatch: Cloud computing for large-scale data-intensive batch applications. 8th IEEE International Symposium on Cluster Computing and the Grid, 2008 (CCGRID'08). IEEE, 2008.

Liu, X. A programming model for the cloud platform. International Journal of Advanced Science and Technology 57: 75–81, 2013.

Ma, M. and C. Meinel. A proposal for trust model: Independent trust intermediary service (ITIS). Proceedings of IADIS International Conference WWW/Internet 2002, Lisbon, Portugal, 2002, pp. 785–790.

Mahmood, Z. and S. Saeed. Software Engineering Frameworks for Cloud Computing Paradigm. Springer-Verlag, London, U.K., 2013.

Mann, A. Virtualization 101: Technologies, benefits, and, challenges. White Paper, EMA Boulder, CO.

Mell, P. and T. Grance The NIST definition of cloud computing (draft). NIST Special Publication 800.145: 7, 2011.

Miceli, C. et al. Programming abstractions for data intensive computing on clouds and grids. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE Computer Society, 2009.

More, J. J. Cloud computing: Information technology's answer to sustainability. www. ecoseed.org. Accessed November 9, 2013.

Mosharaf Kabir Chowdhury, N. M. and Boutaba, R. A survey of network virtualiza- tion. Computer Networks 54(5): 862–876, 2010. ISSN 1389-1286.

Perez-Botero, D., J. Szefer, and R. B. Lee. Characterizing hypervisor vulnerabilities in cloud computing servers. Proceedings of the 2013 International Workshop on Security in Cloud Computing (Cloud Computing '13), ACM, New York, 2013, pp. 3–10.

Pianese, F. et al. Toward a cloud operating system. Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP. IEEE, 2010.

Pressman, R. Software Engineering: A Practitioner's Approach, 7th edn. McGraw- Hill Higher Education, New York, 2009.

Rashmi, G. Sahoo, and S. Mehfuz. Securing software as a service model of cloud com- puting: Issues and solutions. International Journal on Cloud Computing: Services and Architecture (IJCCSA) 3(4): 1–11, 2013.

Reuben, J. S. A survey on virtual machine security. Seminar of Network Security, Helsinki University of Technology, Helsinki, Finland, 2007.

Rodero-Merino, L., L. M. Vaquero, E. Caron, A. Muresan, and F. Desprez. Building safe PaaS clouds: A survey on security in multitenant software platforms. Computers and Security 31(1), 96–108, 2012. ISSN 0167-4048.

Rosen, E. and Y. Rekhter. BGP/MPLS IP virtual private networks (VPNs). RFC 4364, 2006.

S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith. Intel virtualization technol- ogy. Computer 38(5): 48–56, 2005.

Sankaralingam, K. and R. H. Arpaci-Dusseau. Get the parallelism out of my loud. Proceedings of the Second USENIX Conference on Hot Topics in Parallelism, 2010.

Scarfone, K., M. Souppaya, and P. Hoffman, Guide to security for full virtualization technologies. NIST Special Publication, 800-125, 2011.

Sempolinski, P. and D. Thain. A comparison and critique of Eucalyptus, OpenNebula and Nimbus. 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, Indianapolis, IN, 2010, pp. 417–426.

Singh, A., M. Korupolu, and D. Mahapatra. Server-storage virtualization: Integration and load balancing in data centers. International Conference for High Performance Computing, Networking, Storage and Analysis, 2008 (SC 2008), Austin, TX, November 15–21, 2008. IEEE/ACM Supercomputing (SC), 2008, pp. 1–12.

Sobolewski, M. Object-oriented service clouds for transdisciplinary computing. In Cloud Computing and Services Science. Springer, New York, 2012, pp. 3–31.

Sommerville, I. Software Engineering, 8th edn. Pearson Education, 2006.

Soylu, A., P. De Causmaecker, and P. Desmet. Context and adaptivity in perva- sive computing environments: Links with software engineering and ontological engineering. Journal of Software 4(9): 992–1013, 2009.

Strassmann, P. A. How SOA fits into cloud computing. SOA Symposium, April 22, 2010.

Stryer, P. Understanding data centers and cloud computing. Global Knowledge Instructor, CCSI, CCNA.

Teng, F., L. Yu, and F. Magoulès. SimMapReduce: A simulator for model- ing MapReduce framework. 2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE), June 28–30, 2011. IEEE, 2011, pp. 277–282.

The Service Cloud—Knowledge as a service. White paper-Salesforce.com. Varia, J. and S. Mathew. Overview of Amazon web services, 2012.

Vecchiola, C., X. Chu, and R. Buyya. Aneka: A software platform for .NET-based cloud computing. In Gentzsch, L., L. Grandinetti, and G. Joubert (eds.), High Speed and Large Scale Scientific Computing. IOS Press, Amsterdam, the Netherlands, 2009, pp. 267–295.

Wickremasinghe, B., R. N. Calheiros, and R. Buyya. Cloudanalyst:Acloudsim-based visual modeller for analysing cloud computing environments and applications. 2010 24th IEEE International Conference on Advanced Information Networking and Applications

Zhang, J., F. Ren, and C. Lin. Survey on transport control in data center net- works. IEEE Network 27(4): 22–26, 2013.

Zhang, J., F. Ren, L. Tang, and C. Lin. Taming TCP incast throughput collapse in data center networks. 21st IEEE International Conference on Network Protocols, ser. ICNP 2013. IEEE, 2013.

Zhang, L.-J. and Q. Zhou. CCOA: Cloud computing open architecture. IEEE International Conference on Web Services 2009 (ICWS 2009), 2009, pp. 607–616.

# Komputasi Awan
## (Cloud Computing)

**Dr. Joseph Teguh Santoso, M.Kom**

## BIODATA PENULIS

Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce*sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek "Raleigh" untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek "Fengjiu", yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-07-9 (PDF)
9 786238 120079

# Komputasi Awan
## (Cloud Computing)

Dr. Joseph Teguh Santoso, M.Kom

YAYASAN PRIMA AGUS TEKNIK

**PENERBIT :**
**YAYASAN PRIMA AGUS TEKNIK**
**Jl. Majapahit No. 605 Semarang**
**Telp. (024) 6723456. Fax. 024-6710144**
**Email : penerbit_ypat@stekom.ac.id**