Cloud Computing: Applications and Paradigms

Cloud Computing: Aplikasi dan Paradigma

Aplikasi Cloud computing

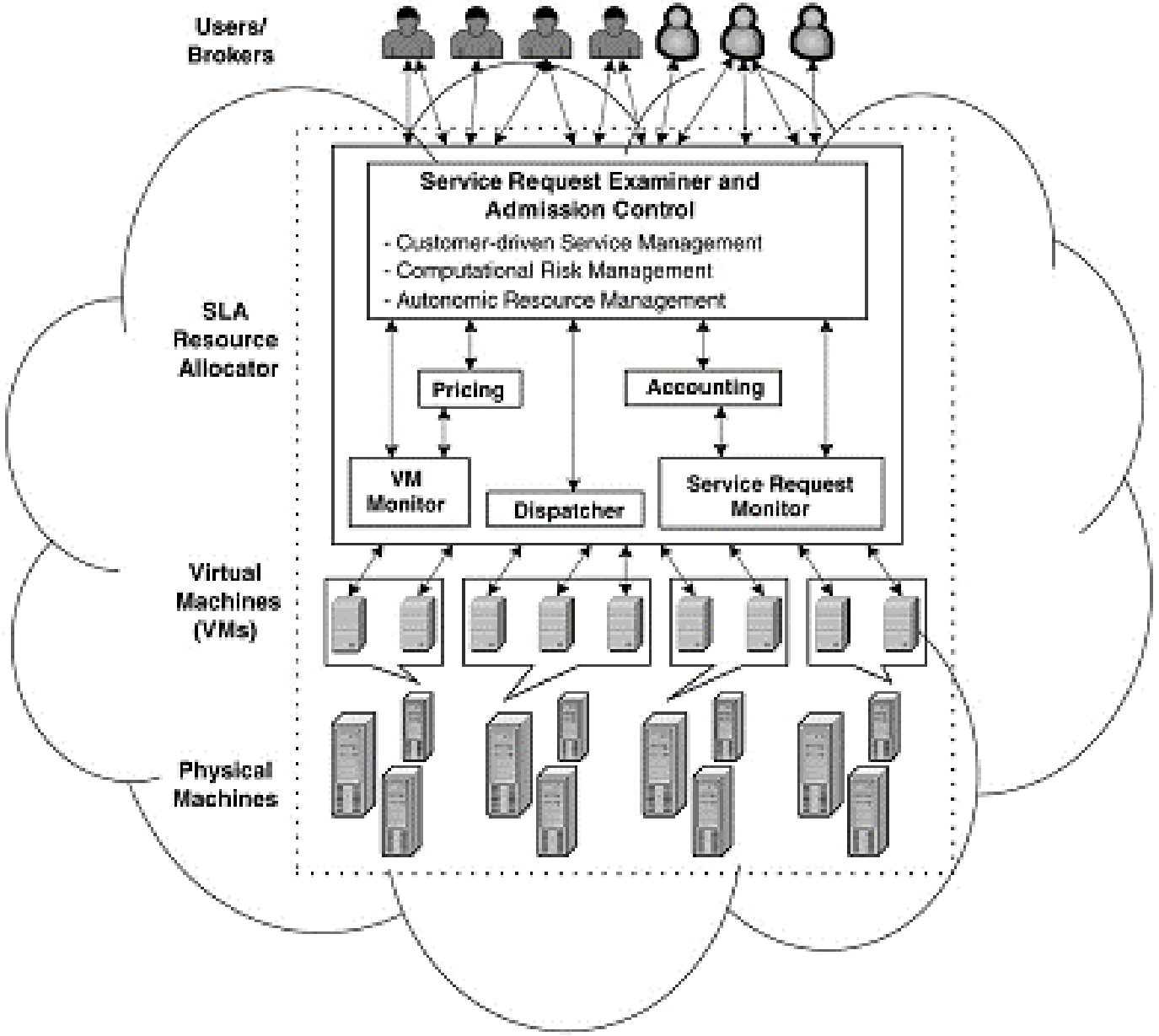
- Cloud computing sangat menarik bagi pengguna:
  - Alasan ekonomi.
    - investasi infrastruktur yang rendah.
    - biaya rendah - pelanggan hanya ditagih untuk sumber daya yang digunakan.
  - Kenyamanan dan kinerja.
    - pengembang aplikasi menikmati keuntungan dari just-in-time infrastruktur; mereka bebas mendesain aplikasi tanpa harus berkaitan dengan sistem dimana aplikasi akan dijalankan.
    - waktu eksekusi intensif komputasi dan intensif data aplikasi dapat, berpotensi, dikurangi melalui paralelisasi. Jika aplikasi dapat mempartisi beban kerja dalam n segmen dan menelurkan n instance dari dirinya sendiri, maka waktu eksekusi bisa menjadi dikurangi dengan faktor yang mendekati n.
- Cloud Computing juga bermanfaat bagi penyedia komputasi siklus - biasanya mengarah ke tingkat pemanfaatan sumber daya yang lebih tinggi.

Aplikasi Cloud computing lanjutan

- Aplikasi ideal untuk Cloud computing:
  - Layanan web Service.
  - Layanan basis data.
  - Layanan berbasis transaksi. Persyaratan sumber daya layanan berorientasi transaksi mendapat manfaat dari lingkungan elastis di mana sumber daya tersedia saat dibutuhkan dan di mana seseorang hanya membayar untuk sumber daya yang dikonsumsinya.
- Aplikasi tidak mungkin berkinerja baik di cloud:
  - Aplikasi dengan alur kerja yang kompleks dan banyak ketergantungan, seperti yang sering terjadi pada komputasi kinerja tinggi.
  - Aplikasi yang membutuhkan komunikasi intensif antar konkuren contoh untuk rendering.

Tantangan untuk pengembangan aplikasi cloud

- Isolasi kinerja (*Performance isolation*) - hampir tidak mungkin dicapai dalam sistem nyata,terutama ketika sistem heavy loaded.
- Keandalan (*Reliability*) - perhatian utama; kegagalan server diharapkan ketika besar jumlah server bekerja sama untuk melakukan komputasi.
- Infrastruktur cloud menunjukkan fluktuasi latensi dan bandwidth yang mempengaruhi kinerja aplikasi.
- Pertimbangan kinerja membatasi jumlah pencatatan data; kemampuan untuk mengidentifikasi sumber hasil dan kesalahan.


Users/
Brokers
Service Request Examiner and
Admission Control
- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management
SLA
Resource
Allocator
Pricing
Accounting
VM
Monitor
Dispatcher
Service Request
Monitor
Virtual
Machines
(VMs)
Physical
Machines

Peluang cloud computing

Tiga kategori luas dari aplikasi yang ada:

- *Processing pipelines* → seperangkat alat dan proses untuk memindahan data dari satu sistem ke sistem lainnya di mana ia dapat disimpan dan dikelola secara berbeda.
- Batch processing systems → Proses yang dilakukan secara berkala dan otomatis. Data yang akan diproses dikumpulkan terlebih dahulu oleh sistem, kemudian pada waktu-waktu tertentu yang sudah dijadwalkan, proses dijalankan secara otomatis oleh sistem.
- Aplikasi web service.

Peluang baru cloud computing

- Batch processing systems untuk sistem pendukung keputusan dan bisnis analitik.
- Aplikasi interaktif seluler yang memproses volume besar data dari berbagai jenis sensor.
- Sains dan teknik dapat sangat diuntungkan dari cloud komputasi karena banyak aplikasi di area ini adalah komputasi intensif dan data-intensif.

Processing pipelines

- Mengindeks kumpulan data besar yang dibuat oleh web crawler engines.
- Data mining - mencari koleksi besar catatan untuk menemukan item yang menarik.
- Pemrosesan gambar → Konversi gambar, misalnya memperbesar gambar atau membuat thumbnail, Kompres atau enkripsi gambar.
- Transcoding video dari satu format video ke format video lainnya, misalnya dari AVI ke MPEG.
- Pemrosesan dokumen → Konversikan banyak koleksi dokumen dari satu format kelain, misalnya, dari Word ke PDF.
- Enkripsi dokumen.
- Gunakan Pengenalan Karakter Optik untuk menghasilkan gambar digital dari dokumen.

Batch processing applications

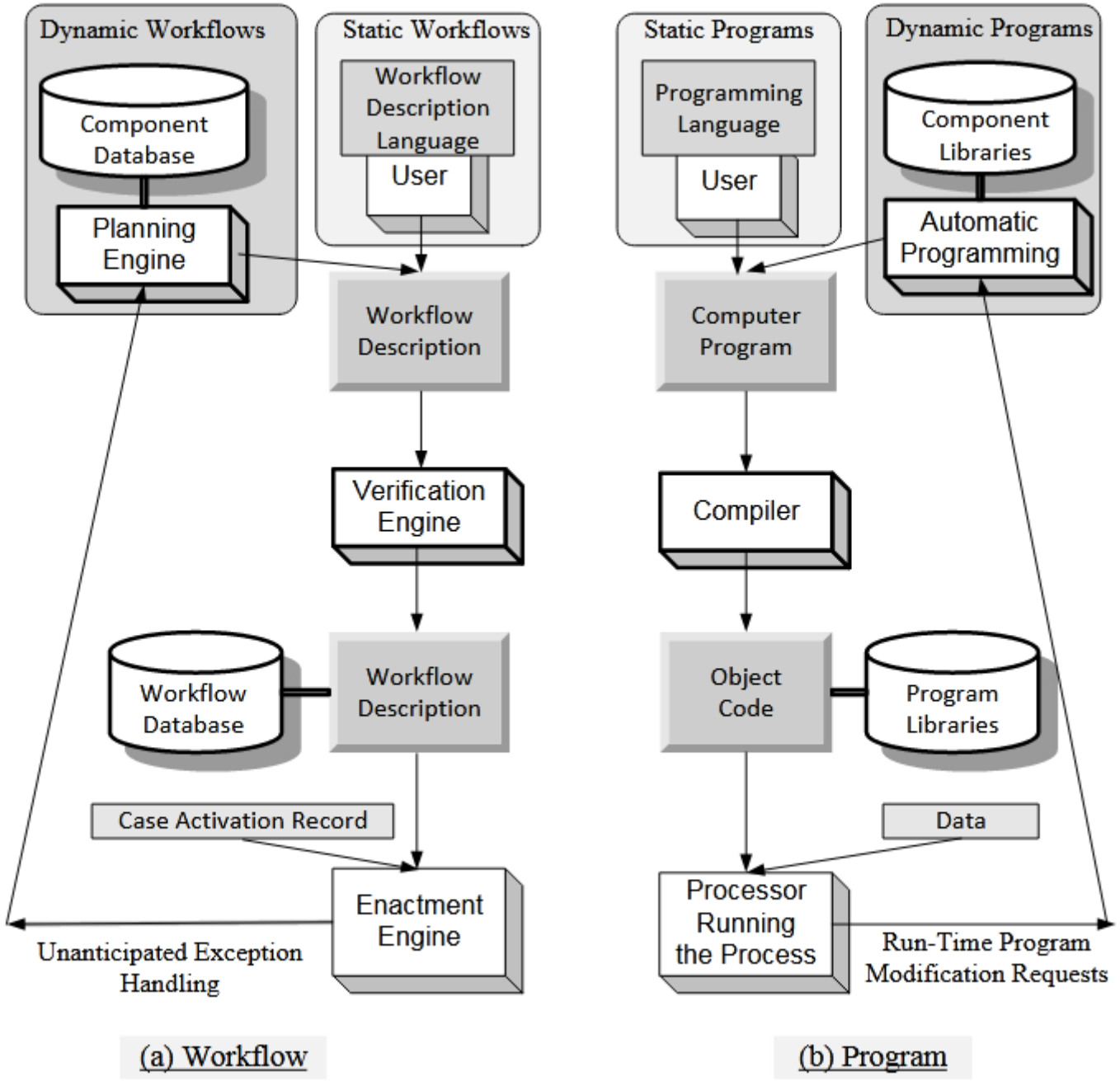
- Pembuatan laporan aktivitas harian, mingguan, bulanan, dan tahunan untuk ritel, manufaktur, sektor ekonomi lainnya.
- Pemrosesan, agregasi, dan ringkasan transaksi harian.
- Memproses catatan penagihan dan penggajian.
- Manajemen pengembangan perangkat lunak, misalnya, pembaruan repositori perangkat lunak setiap malam.
- Pengujian dan verifikasi otomatis sistem perangkat lunak dan perangkat keras.

Gaya arsitektur untuk aplikasi cloud

- Berdasarkan paradigma client-server.
- Stateless Server - lihat permintaan klien sebagai permintaan independen transaksi dan menanggapinya; klien tidak diharuskan terlebih dahulu membuat koneksi ke server.
- Seringkali klien dan server berkomunikasi menggunakan Remote Procedure Calls (RPC).
- Simple Object Access Protocol (SOAP) - protokol aplikasi untuk aplikasi web; format pesan berdasarkan XML. Menggunakan TCP atau protokol transportasi UDP.
- Representational State Transfer (REST) - arsitektur perangkat lunak untuk sistem hypermedia terdistribusi. Mendukung klien komunikasi dengan server stateless, platform independen, bahasa independen, mendukung caching data, dan dapat digunakan dengan firewall.

Alur kerja

- Deskripsi proses (*Process description*)- struktur yang menjelaskan tugas yang akan dieksekusi dan urutan pelaksanaannya. Menyerupai diagram alur.
- Case - sebuah contoh dari deskripsi proses.
- State of a case at time - didefinisikan dalam hal tugas sudah selesai pada saat itu.
- Peristiwa (event) - menyebabkan transisi antar keadaan.
- Siklus hidup alur kerja (The life cycle of a workflow) - pembuatan, definisi, verifikasi, dan pemberlakuan; mirip dengan siklus hidup program tradisional (pembuatan, kompilasi, dan eksekusi).

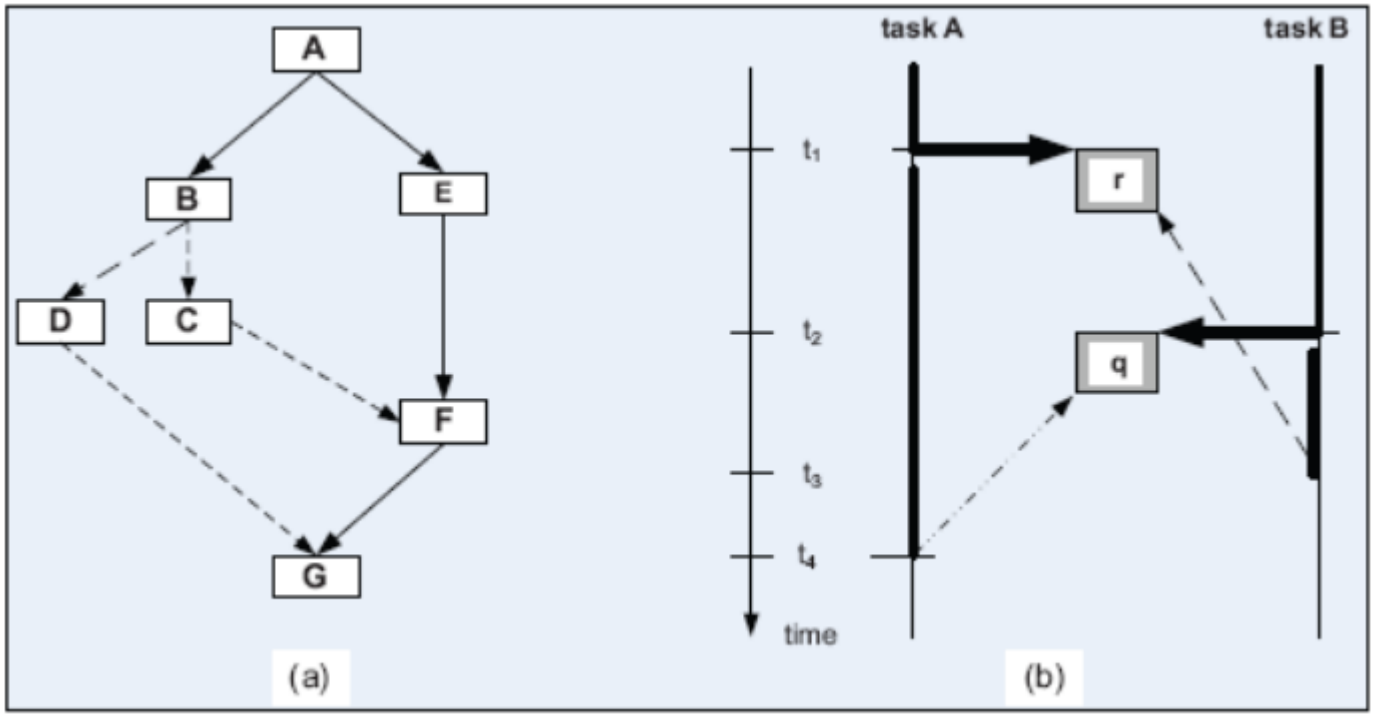

Dynamic Workflows
Component Database
Planning Engine
Static Workflows
Workflow Description Language
User
Workflow Description
Verification Engine
Workflow Database
Workflow Description
Case Activation Record
Enactment Engine
Unanticipated Exception Handling
(a) Workflow
Static Programs
Programming Language
User
Dynamic Programs
Component Libraries
Automatic Programming
Computer Program
Compiler
Object Code
Program Libraries
Data
Processor Running the Process
Run-Time Program Modification Requests
(b) Program

Safety and liveness (Keamanan dan keaktifan)

- Sifat alur kerja yang diinginkan.
- Keamanan tidak ada hal "buruk" yang pernah terjadi.
- Keaktifan sesuatu yang "baik" pada akhirnya akan terjadi

(a) Deskripsi proses yang melanggar persyaratan liveness. Jika tugas C dipilih setelah selesainya B, proses akan berhenti setelah menjalankan tugas G; jika D dipilih, maka F tidak akan pernah dipakai, karena membutuhkan penyelesaian C dan E. Proses tidak akan pernah berhenti, karena G membutuhkan penyelesaian D dan F .

(b) (b) Tugas A dan B memerlukan akses eksklusif ke dua sumber daya r dan q, dan kebuntuan dapat terjadi jika urutan kejadian berikut terjadi. Pada waktu t1 tugas A memperoleh r, pada waktu t2 tugas B memperoleh q dan terus berjalan; kemudian pada waktu t3 tugas B mencoba untuk memperoleh r dan blok karena r berada di bawah kendali A. Tugas A terus berjalan dan pada waktu t4 mencoba untuk memperoleh q dan blok karena q berada di bawah kendali B.



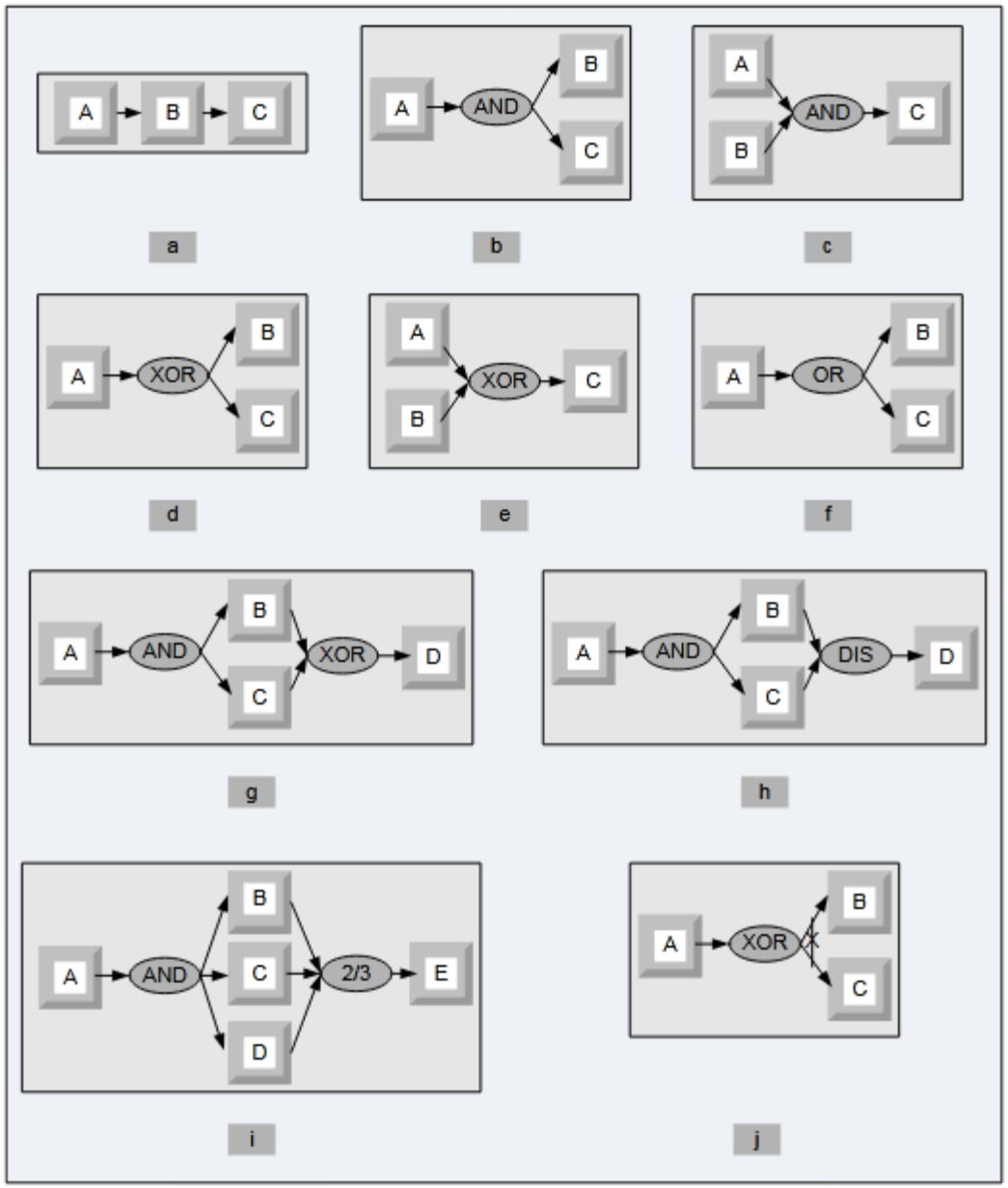
Pola alur kerja dasar

Pola alur kerja (Workflow patterns) - hubungan temporal antara tugas-tugas dari suatu proses

- Urutan - beberapa tugas harus dijadwalkan satu setelah penyelesaian yang lain.
- AND split - baik tugas B dan C diaktifkan saat tugas A berakhir.
- Sinkronisasi - tugas C hanya dapat dimulai setelah tugas A dan B berakhir.
- XOR split - setelah menyelesaikan tugas A, B atau C dapat diaktifkan.
- Penggabungan XOR - tugas C diaktifkan saat A atau B berhenti.
- OR split - setelah menyelesaikan tugas A, seseorang dapat mengaktifkan B, C, atau keduanya.

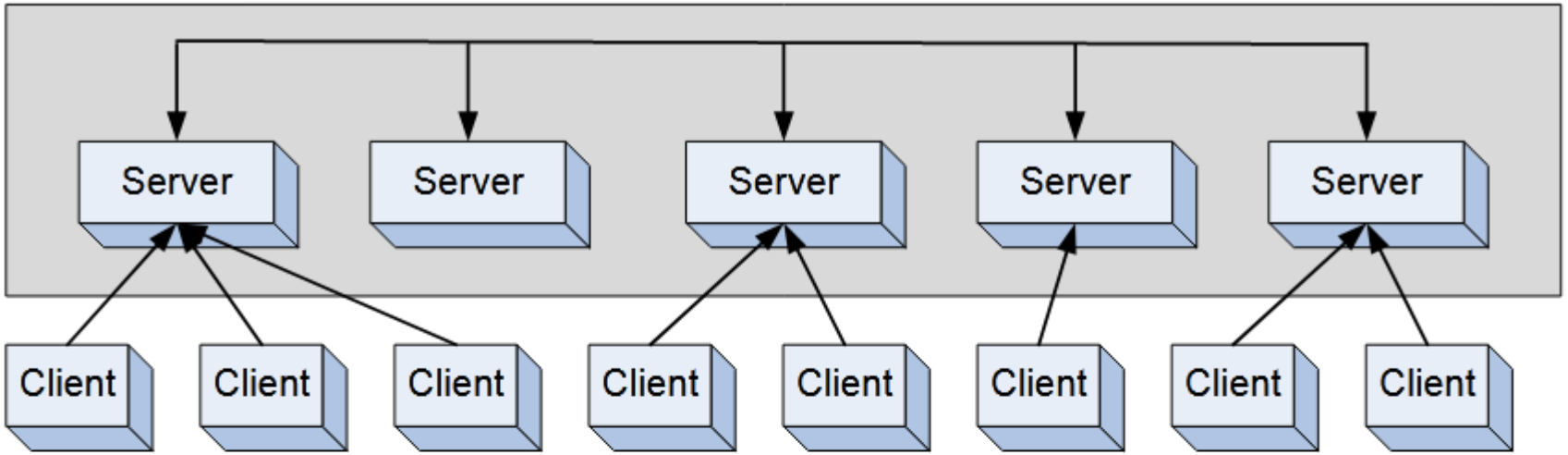
Pola alur kerja dasar Lanjutan

- Multiple Merge - setelah tugas A berakhir, B dan C dijalankan secara bersamaan; ketika yang pertama, katakanlah B, berakhir, maka D diaktifkan; kemudian, ketika C berakhir, D diaktifkan kembali.
- Diskriminator – menunggu sejumlah cabang yang masuk selesai sebelum mengaktifkan aktivitas berikutnya; kemudian tunggu cabang yang tersisa selesai tanpa mengambil tindakan apa pun sampai semuanya berakhir. Selanjutnya, me-reset sendiri.
- N dari M bergabung - sinkronisasi penghalang. Dengan asumsi bahwa M tugas berjalan secara bersamaan, N (N<M) dari mereka harus mencapai penghalang sebelum tugas berikutnya diaktifkan. Dalam contoh kita, dua dari tiga tugas A, B, dan C harus diselesaikan sebelum E diaktifkan.
- Pilihan yang Ditangguhkan - mirip dengan pemisahan XOR tetapi pilihannya tidak dibuat secara eksplisit; lingkungan run-time memutuskan cabang apa yang akan diambil.

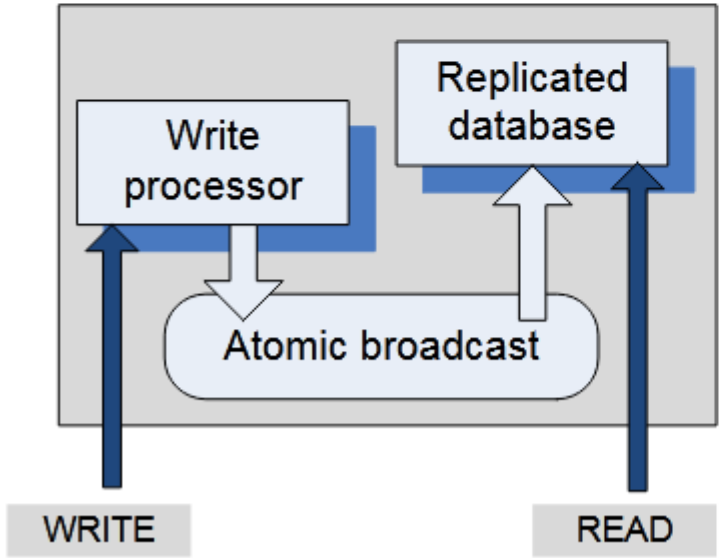

A
B
C
a
A
AND
B
C
b
A
B
AND
C
c
A
XOR
B
C
d
A
B
XOR
C
e
A
OR
B
C
f
A
AND
B
C
XOR
D
g
A
AND
B
C
DIS
D
h
A
AND
B
C
D
2/3
E
i
A
XOR
B
C
j

Koordinasi - Zookeeper

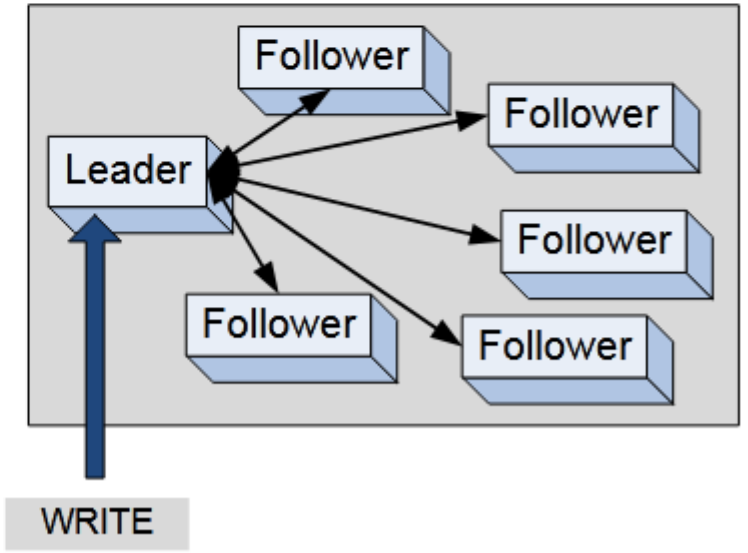
- Elastisitas Cloud → mendistribusikan komputasi dan data ke berbagai sistem; koordinasi di antara sistem ini adalah fungsi penting dalam lingkungan terdistribusi.
- Zookeeper berfungsi sebagai:
- Layanan koordinasi terdistribusi untuk sistem terdistribusi skala besar.
- Throughput tinggi dan layanan latensi rendah.
- Menerapkan versi algoritma konsensus Paxos (Paxos adalah kumpulan protokol untuk menyelesaikan konsensus dalam jaringan prosesor yang tidak dapat diandalkan atau salah. Konsensus adalah proses menyepakati satu hasil di antara sekelompok peserta. Masalah ini menjadi sulit ketika peserta atau komunikasi mereka mungkin mengalami kegagalan).
- Open source software yang ditulis dalam Java dengan binding untuk Java dan C.
- Server dalam paket berkomunikasi dan memilih server pengontrol.
- Sebuah database direplikasi pada setiap server; konsistensi replika tetap terjaga.
- Seorang klien terhubung ke satu server, menyinkronkan jamnya dengan server, dan mengirim permintaan, menerima tanggapan dan watch event melalui koneksi TCP.


Server
Server
Server
Server
Server
Client
Client
Client
Client
Client
Client
Client
Client

(a)


Write processor
Replicated database
Atomic broadcast
WRITE
READ

(b)

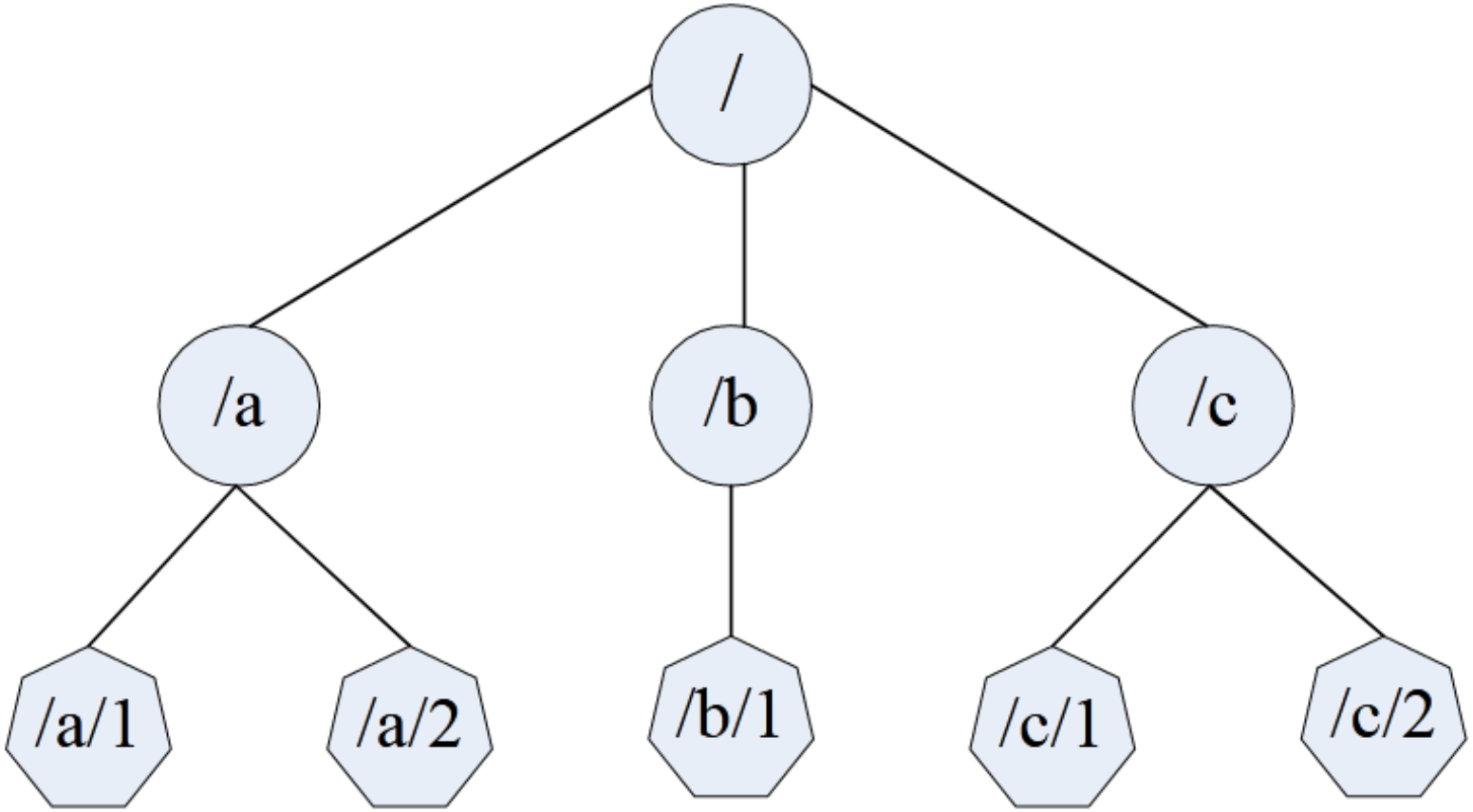

Follower
Follower
Leader
Follower
Follower
Follower
WRITE

(c)

Zookeeper communication

- Lapisan pesan (Messaging layer) → bertanggung jawab atas pemilihan controller server baru ketika server saat ini gagal.
- Protokol perpesanan menggunakan:
  - Paket - urutan byte yang dikirim melalui saluran FIFO.
  - Proposal - unit kesepakatan.
  - Pesan - urutan byte yang dibroadclast secara ke semua server.
- Sebuah pesan dimasukkan ke dalam proposal dan disetujui sebelum disampaikan.
- Proposal disepakati dengan bertukar paket dengan a kuorum server, seperti yang disyaratkan oleh algoritma Paxos.

Shared hierarchical namespace



Jaminan layanan ZooKeeper

- Atomicity - transaksi selesai atau gagal.
- Konsistensi pembaruan berurutan (Sequential consistency of updates) - pembaruan diterapkan secara ketat sesuai urutan penerimaannya.
- Gambar sistem tunggal untuk klien (Single system image for the clients) - klien menerima yang sama respons terlepas dari server yang terhubung.
- Persistensi pembaruan (Persistence of updates) - setelah diterapkan, pembaruan tetap ada hingga ditimpa oleh klien.
- Keandalan (Reliability)- sistem dijamin berfungsi dengan benar sebagaiselama sebagian besar server berfungsi dengan benar.

Zookeeper API (Application Programming Interface)

API → yang merupakan perantara perangkat lunak yang memungkinkan dua aplikasi untuk berbicara satu sama lain.

API sederhana - terdiri dari tujuh operasi:

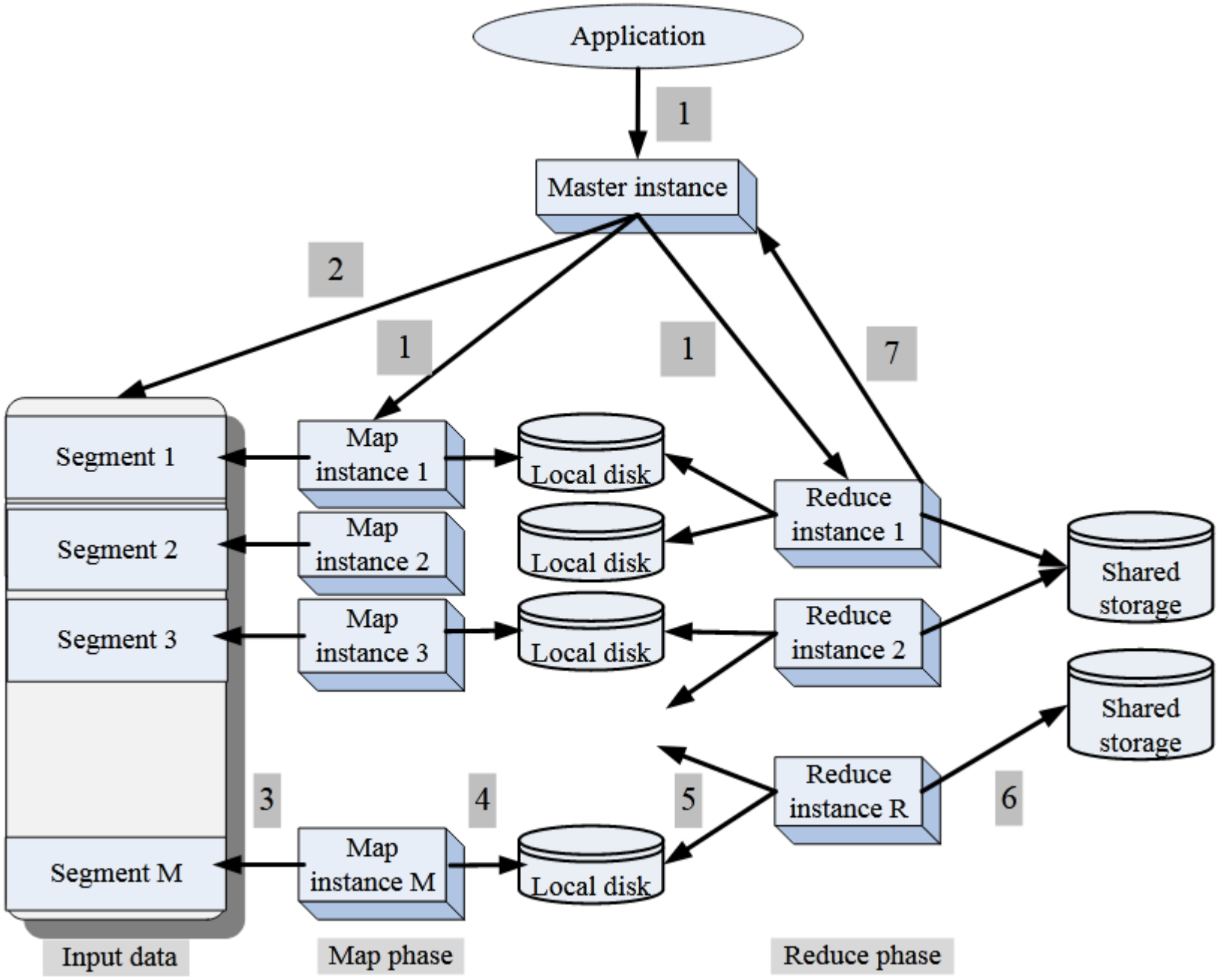
- Create - tambahkan simpul di lokasi tertentu di pohon.
- Delete - hapus sebuah simpul.
- Get data – baca data dari sebuah node.
- Set data - tulis data ke node.
- Get Children - ambil daftar anak-anak dari simpul.
- Synch - tunggu hingga data menyebar

Elastisitas dan distribusi beban (Elasticity and load distribution)

- Elastisitas → kemampuan untuk menggunakan server sebanyak yang diperlukan secara optimal menanggapi kendala biaya dan waktu aplikasi.
- Cara membagi beban
  - Sistem pemrosesan transaksi(Transaction processing systems) → front-end mendistribusikan yang masuk transaksi ke sejumlah sistem back-end. Sebagai beban kerja meningkatkan sistem back-end baru ditambahkan ke pool.
  - Untuk aplikasi batch data-intensif dua jenis beban kerja yang dapat dibagi adalah mungkin:
    - modularly divisible → pembagian beban kerja didefinisikan secara apriori.
    - arbitrarily divisible → beban kerja dapat dipartisi menjadi sejumlah besar beban kerja yang lebih kecil yang sama, atau sangat dekat ukuran.
- Banyak aplikasi dalam fisika, biologi, dan bidang lainnya ilmu dan teknik komputasi arbitrarily divisible sebagai model pembagian beban.

Filosofi MapReduce

1. Aplikasi memulai instans master, instans pekerja M untuk fase Peta dan kemudian instans pekerja R untuk fase Reduce.
2. Instance master mempartisi data input dalam segmen M.
3. Setiap instance peta membaca segmen dan proses data inputnya data.
4. Hasil pemrosesan disimpan di disk lokal server tempat instance peta dijalankan.
5. Ketika semua instance peta telah selesai memproses datanya, R kurangi instance, baca hasil fase pertama dan gabungkan hasil parsial.
6. Hasil akhir ditulis oleh pengurangan instance menjadi shared server penyimpanan.
7. Instance master memantau instance pengurangan dan ketika semuanya melaporkan penyelesaian tugas, aplikasi dihentikan


Application
1
Master instance
2
1
1
7
Segment 1
Segment 2
Segment 3
Segment M
Map instance 1
Map instance 2
Map instance 3
Map instance M
Local disk
Local disk
Local disk
Local disk
Reduce instance 1
Reduce instance 2
Reduce instance R
Shared storage
Shared storage
3
4
5
6
Input data
Map phase
Reduce phase

Cloud untuk sains dan teknik

- Masalah umum di hampir semua bidang ilmu pengetahuan adalah:
  - Pengumpulan data eksperimen.
  - Pengelolaan volume data yang sangat besar.
  - Membangun dan mengeksekusi model.
  - Integrasi data dan literatur.
  - Dokumentasi percobaan.
  - Berbagi data dengan orang lain; penyimapanan data untuk waktu yang lama.
- Semua aktivitas ini membutuhkan penyimpanan data "besar" dan sistem yang mampuuntuk memberikan siklus komputasi yang melimpah. Cloud komputasi mampu menyediakan sumber daya tersebut dan mendukung lingkungan kolaboratif.

Online data discovery

- Fase penemuan data dalam kumpulan data ilmiah besar:
  - pengenalan masalah informasi.
  - generasi permintaan pencarian menggunakan satu atau lebih mesin pencari.
  - evaluasi hasil pencarian.
  - evaluasi dokumen web.
  - membandingkan informasi dari berbagai sumber.
- Contoh Kumpulan data ilmiah besar: data biomedis dan genomik dari National Center for Informasi Bioteknologi (NCBI). data astrofisika dari NASA. data atmosfer dari National Oceanic and Atmospheric Administrasi (NOAA) dan Pusat Nasional untuk Atmosfer Penelitian (NCAR).

High performance computing on a cloud

- Tolok ukur perbandingan EC2 dan tiga superkomputer diPusat Komputasi Ilmiah Riset Energi Nasional (NERSC) diLaboratorium Nasional Lawrence Berkeley. NERSC memiliki sekitar 3.000 peneliti dan melibatkan 400 proyek berdasarkan sekitar 600 kode.
- Kesimpulan – aplikasi intensif komunikasi dipengaruhi olehpeningkatan latensi dan bandwidth cloud yang lebih rendah. Rendahlatensi dan bandwidth tinggi dari jaringan interkoneksi superkomputer tidak dapat ditandingi oleh cloud.

System	DGEMM Gflops	STREAM GB/s	Latency μs	Bndw GB/S	HPL Tflops	FFTE Gflops	PTRANS GB/s	RandAcc GUP/s
Carver	10.2	4.4	2.1	3.4	0.56	21.99	9.35	0.044
Frankl	8.4	2.3	7.8	1.6	0.47	14.24	2.63	0.061
Lawren	9.6	0.7	4.1	1.2	0.46	9.12	1.34	0.013
EC2	4.6	1.7	145	0.06	0.07	1.09	0.29	0.004

Link video penjelasan

https://drive.google.com/file/d/17GwFHDG-grKKw-IeE6XGoxTcFEtdXci9/view?usp=sharing

Terimakasih