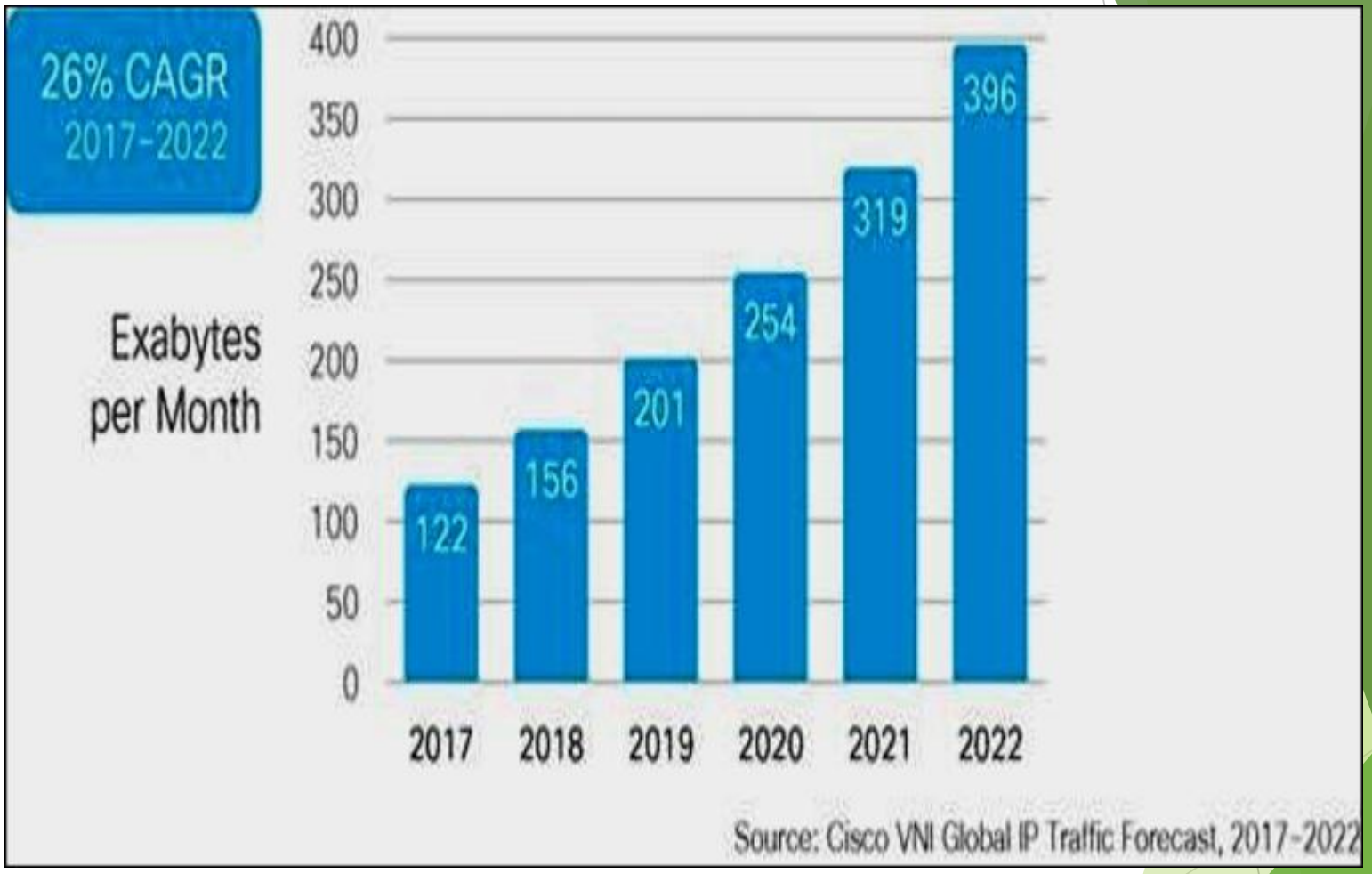
Sistem Penyimpanan

Storage System

Penyimpanan Data di Cloud

- Penyimpanan dan pemrosesan di cloud terkait erat satu sama lain.
  - Sebagian besar aplikasi cloud memproses data dalam jumlah yang sangat besar. Replikasi data dan strategi manajemen penyimpanan yang efektif sangat penting untuk komputasi yang dilakukan di cloud.
  - Strategi untuk mengurangi waktu akses dan untuk mendukung akses multimedia waktu nyata diperlukan untuk memenuhi persyaratan pengiriman konten.
- Sensor memberikan aliran data yang berkelanjutan ke aplikasi cloud.
- Semakin banyak layanan berbasis cloud yang mengumpulkan data terperinci tentang layanan mereka dan informasi tentang pengguna layanan ini. Penyedia layanan menggunakan awan untuk menganalisis data.

Data Traffic Internet



Keterangan 1 exabyte (EB) = 10^{18}bytes

Big Data

- New Consep → mencerminkan fakta bahwa banyak aplikasi menggunakan kumpulan data yang tidak dapat disimpan dan diproses menggunakan sumber daya lokal.
- Aplikasi dalam genomik, biologi struktural, fisika energi tinggi, astronomi, meteorologi, dan studi lingkungan melakukan analisis kompleks kumpulan data sering kali dalam urutan TB (terabyte). Contoh:
  - Pada tahun 2010, empat detektor utama di Large Hadron Collider (LHC) menghasilkan 13 PB data.
  - Sloan Digital Sky Survey (SDSS) mengumpulkan sekitar 200 GB data per malam.
- Fenomena tiga dimensi.
  - Peningkatan volume data.
  - Memerlukan peningkatan kecepatan pemrosesan untuk memproses lebih banyak data dan menghasilkan lebih banyak hasil.
  - Melibatkan keragaman sumber data dan tipe data.

Model penyimpanan dan data

- Model penyimpanan → menggambarkan tata letak struktur data dalam penyimpanan fisik - disk lokal, media yang dapat dipindahkan, atau penyimpanan yang dapat diakses melalui jaringan.
- Model data → menangkap aspek logis terpenting dari struktur data dalam database.
- Dua model penyimpanan abstrak digunakan.
- Penyimpanan sel (Cell Storage) mengasumsikan bahwa penyimpanan terdiri dari sel-sel dengan ukuran yang sama dan bahwa setiap objek tepat berada dalam satu sel. Model ini mencerminkan organisasi fisik dari beberapa media penyimpanan; memori utama komputer diatur sebagai array sel memori dan perangkat penyimpanan sekunder, misalnya, disk, diatur dalam sektor atau blok yang dibaca dan ditulis sebagai satu unit.
- Penyimpanan jurnal (journal storage) → sistem yang melacak perubahan yang akan dibuat dalam jurnal (biasanya log melingkar di area khusus sistem file) sebelum memasukkannya ke sistem file utama. Jika terjadi sistem crash atau mati listrik, sistem file tersebut lebih cepat untuk dibawa kembali online dan kecil kemungkinannya menjadi rusak.

Data Base Management System (DBMS)

- Basis data kumpulan catatan yang berhubungan secara logis.
- Sistem Manajemen Basis Data (DBMS) → perangkat lunak yang mengontrol akses ke database.
- Bahasa query → bahasa pemrograman khusus yang digunakan untuk mengembangkan aplikasi database.
- Sebagian besar aplikasi cloud tidak berinteraksi langsung dengan sistem file, tetapi melalui DBMS.
- Model database mencerminkan keterbatasan perangkat keras yang tersedia pada saat itu dan persyaratan aplikasi yang paling popular setiap periode.
  - model navigasi tahun 1960-an.
  - model relasional tahun 1970-an.
  - model berorientasi objek tahun 1980-an.
  - Model NoSQL dekade pertama tahun 2000-an

Persyaratan aplikasi cloud

- Sebagian besar aplikasi cloud bersifat intensif data dan menguji keterbatasan infrastruktur yang ada. Persyaratan:
  - Pengembangan aplikasi yang cepat dan waktu yang singkat ke pasar.
  - Latensi rendah.
  - Skalabilitas.
  - Ketersediaan tinggi.
  - Tampilan data yang konsisten.
- Persyaratan ini tidak dapat dipenuhi secara bersamaan oleh model database yang ada; misalnya, database relasional mudah digunakan untuk pengembangan aplikasi tetapi tidak berskala dengan baik.
- Model NoSQL berguna ketika struktur data tidak memerlukan model relasional dan jumlah datanya sangat besar.
- Tidak mendukung SQL sebagai bahasa query.
- Mungkin tidak menjamin properti ACID (Atomicity, Consistency, Isolation, Durability) dari database tradisional; biasanya menjamin konsistensi akhir untuk transaksi terbatas pada satu item data.

Organisasi logis dan fisik file

- File → array linier sel yang disimpan pada perangkat penyimpanan persisten. Dilihat oleh aplikasi sebagai kumpulan catatan logis; file disimpan pada perangkat fisik sebagai satu set catatan fisik, atau blok, dengan ukuran yang ditentukan oleh media fisik.
- Penunjuk file → mengidentifikasi sel yang digunakan sebagai titik awal untuk membaca atau operasi tulis.
- Organisasi logis dari file → mencerminkan model data, tampilan data dari perspektif aplikasi.
- Organisasi fisik file → mencerminkan model penyimpanan dan menjelaskan cara file disimpan pada media penyimpanan tertentu.

Sistem File

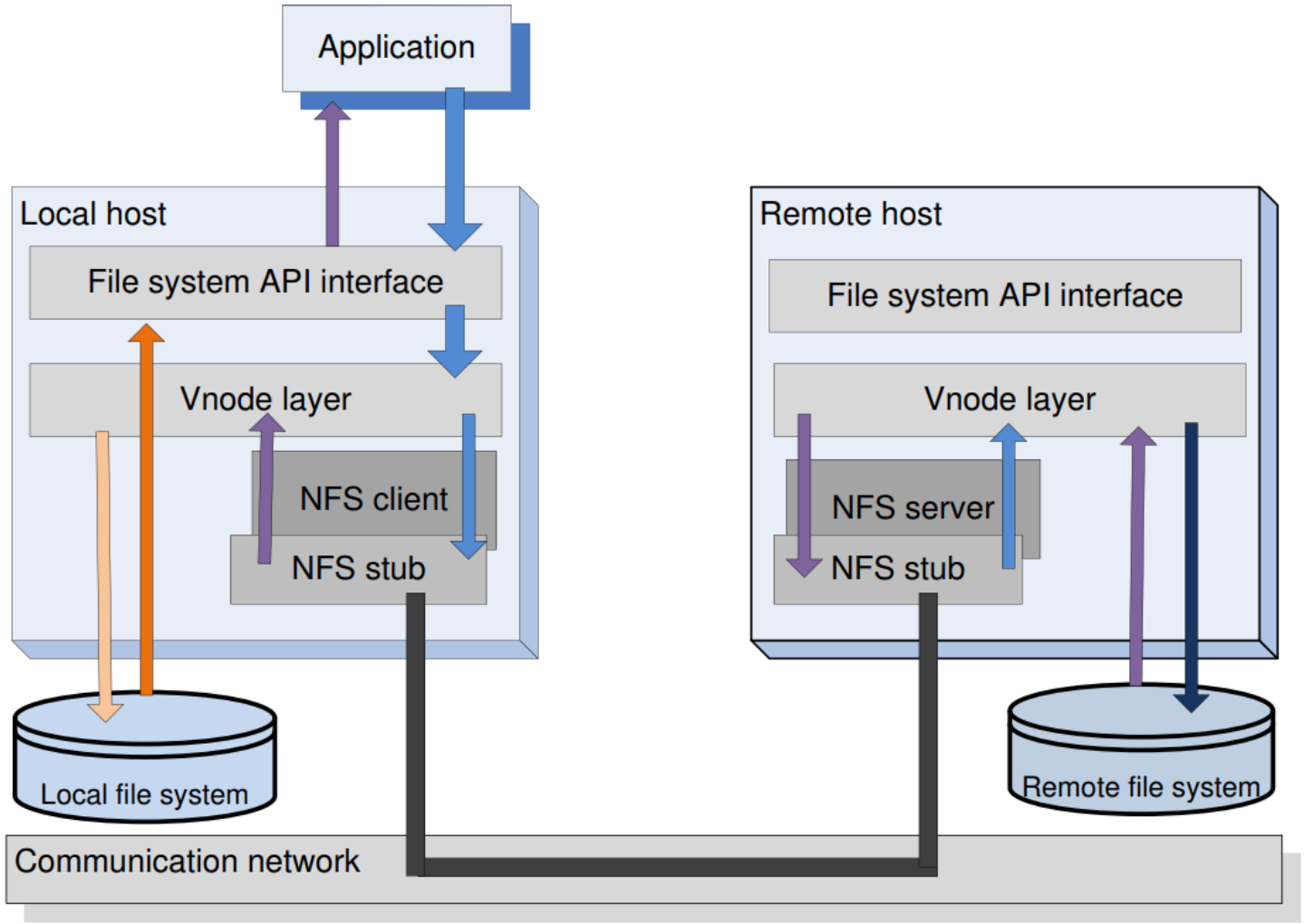
- Sistem file → kumpulan direktori; setiap direktori menyediakan informasi tentang sekumpulan file.
  - Tradisional – Sistem File Unix.
  - Sistem file terdistribusi→ Network File System (NFS) - sangat populer, telah digunakan selama beberapa waktu, tetapi tidak berskala dengan baik dan memiliki masalah keandalan; server NFS bisa menjadi satu titik kegagalan. Storage Area Networks (SAN) - memungkinkan server cloud menangani perubahan yang tidak mengganggu dalam konfigurasi penyimpanan. Penyimpanan dalam SAN dapat dikumpulkan dan kemudian dialokasikan berdasarkan kebutuhan server. Implementasi sistem file berbasis SAN bisa mahal, karena setiap node harus memiliki adaptor Fibre Channel untuk terhubung ke jaringan.
  - Sistem File Paralel (PFS) - scalable, mampu mendistribusikan file di sejumlah besar node, dengan ruang penamaan global. Beberapa node I/O melayani data ke semua node komputasi; itu termasuk juga server metadata yang berisi informasi tentang data yang disimpan di node I/O. Jaringan interkoneksi PFS dapat berupa SAN.

Unix File System (UFS)

- Desain berlapis memberikan fleksibilitas.
  - Desain berlapis memungkinkan UFS untuk memisahkan perhatian untuk struktur file fisik dari yang logis.
  - Lapisan vnode memungkinkan UFS untuk memperlakukan akses file lokal dan jarak jauh secara seragam.
- Desain hierarki mendukung skalabilitas yang dicerminkan oleh konvensi penamaan file. Ini memungkinkan pengelompokan direktori file, mendukung berbagai tingkat direktori, dan kumpulan direktori dan file, yang disebut sistem file.
- Metadata mendukung filosofi desain sistematis dari sistem file dan independensi perangkat.
  - Metadata meliputi: pemilik file, hak akses, waktu pembuatan, waktu modifikasi terakhir, ukuran file, struktur file, dan sel perangkat penyimpanan persisten tempat data disimpan.
  - Inode berisi informasi tentang file dan direktori individual. Inode disimpan pada media persisten bersama dengan data.

Network File System (NFS)

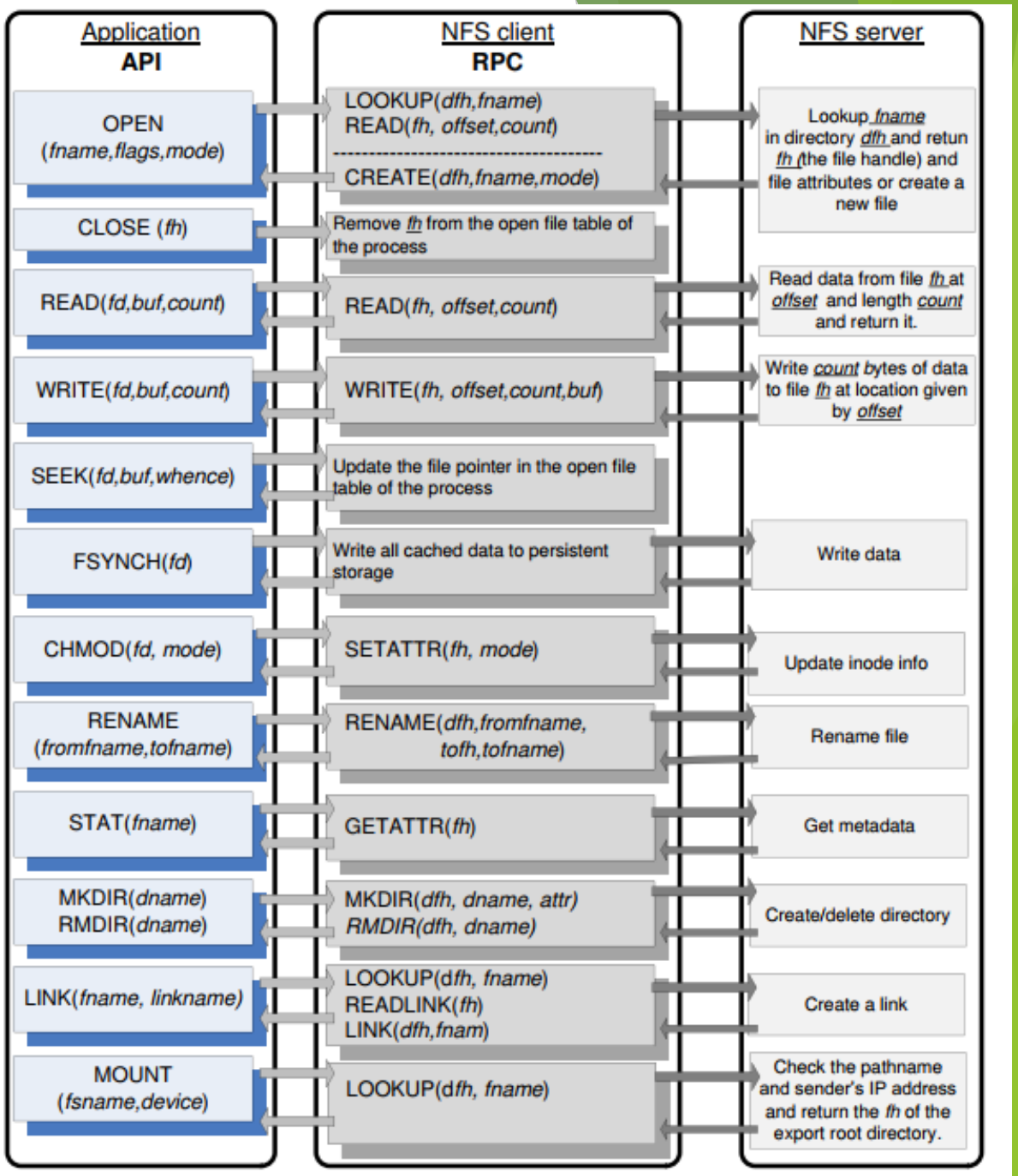
- Tujuan desain:
  - Menyediakan semantik yang sama dengan Unix File System (UFS) lokal untuk memastikan kompatibilitas dengan aplikasi yang ada.
  - Memfasilitasi integrasi yang mudah ke dalam UFS yang ada.
  - Pastikan bahwa sistem akan digunakan secara luas; dengan demikian, dukung klien yang berjalan di sistem operasi yang berbeda.
  - Menerima penurunan kinerja sederhana karena akses jarak jauh melalui jaringan dengan bandwidth beberapa Mbps.
- NFS didasarkan pada paradigma client-server. Klien berjalan di host lokal saat server berada di lokasi sistem file jarak jauh; mereka berinteraksi melalui Remote Procedure Calls (RPC).
- File jarak jauh diidentifikasi secara unik oleh file handle (fh) daripada deskriptor file. file handle adalah nama internal 32-byte - kombinasi dari identifikasi sistem file, nomor inode, dan anomor generasi.



Interaksi klien-server NFS. Lapisan vnode mengimplementasikan operasi file dengan cara yang seragam, terlepas dari apakah file tersebut lokal atau jauh. Operasi yang menargetkan file lokal diarahkan ke sistem file lokal, sedangkan operasi untuk file jarak jauh melibatkan NFS; klien NSF mengemas informasi yang relevan tentang target dan server NFS meneruskannya ke lapisan vnode pada host jarak jauh yang, pada gilirannya, mengarahkannya ke sistem file jarak jauh.

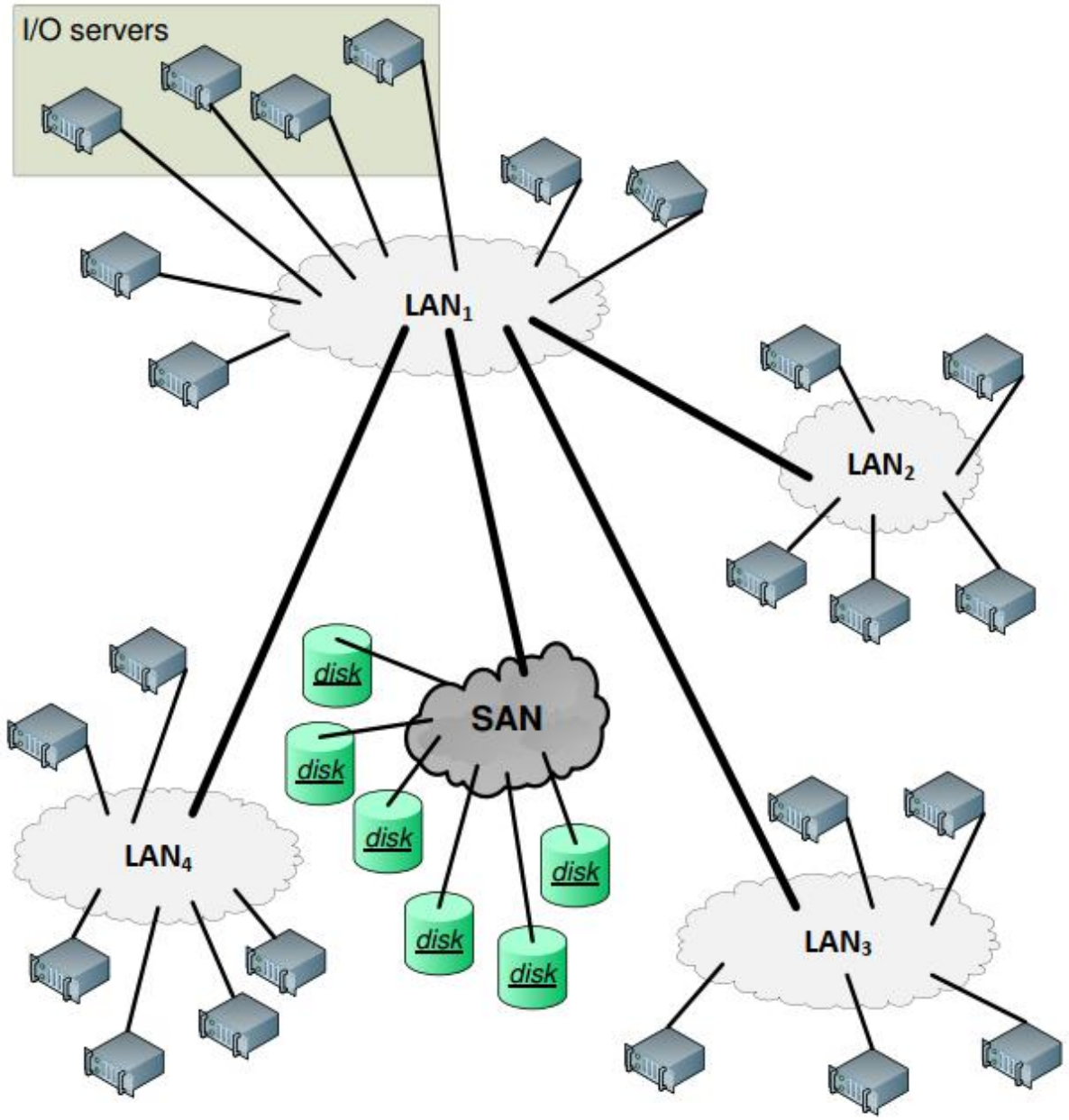
API sistem file UNIX dan RPC terkait yang dikeluarkan oleh klien NFS ke server NFS.

- fd →deskriptor file.
- fh → untuk menangani file.
- fname → nama file,
- dname → nama direktori.
- dfh → direktori tempat file handle dapat ditemukan.
- Count → jumlah byte yang akan ditransfer.
- buf → buffer untuk mentransfer data ke/dari.
- Device → perangkat di mana sistem file berada.



General Parallel File System (GPFS)

- I/O paralel menyiratkan eksekusi bersamaan dari beberapa operasi input/output. Dukungan untuk I/O paralel sangat penting untuk kinerja banyak aplikasi.
- Kontrol konkurensi adalah masalah penting untuk sistem file paralel. Beberapa semantik untuk menangani akses bersama dimungkinkan. Misalnya, ketika klien berbagi penunjuk file, pembacaan berturut-turut yang dikeluarkan oleh beberapa klien memajukan penunjuk file; semantik lain adalah mengizinkan setiap klien memiliki penunjuk file sendiri.
- GPFS.
  - Dikembangkan di IBM pada awal 2000-an sebagai penerus sistem file multimedia TigerShark.
  - Dirancang untuk kinerja optimal cluster besar; itu dapat mendukung sistem file hingga 4 PB yang terdiri dari hingga 4.096 disk masing-masing 1 TB.
  - Ukuran file maksimum adalah (263 -1) byte.Sebuah file terdiri dari blok dengan ukuran yang sama, mulai dari 16 KB hingga 1 MB, dilucuti di beberapa disk.


I/O servers
LAN1
LAN2
LAN3
LAN4
SAN
disk
disk
disk
disk
disk
disk

Keandalan GPFS

- Untuk memulihkan dari kegagalan sistem, GPFS mencatat semua pembaruan metadata dalam file log write-ahead.
- Write-ahead pembaruan ditulis ke penyimpanan persisten hanya setelah catatan log ditulis.
- File log dipelihara oleh setiap node I/O untuk setiap sistem file yang di-mount; setiap node I/O dapat memulai pemulihan atas nama node yang gagal.
- Data Striping memungkinkan akses bersamaan dan meningkatkan kinerja, tetapi dapat memiliki efek samping yang tidak menyenangkan. Ketika satu disk gagal, sejumlah besar file terpengaruh.
- Sistem menggunakan perangkat RAID dengan garis-garis yang sama dengan ukuran blok dan pengontrol RAID yang terpasang ganda.
- Untuk lebih meningkatkan toleransi kesalahan sistem, file data GPFS serta metadata direplikasi pada dua disk fisik yang berbeda.

Google File System (GFS)

- GFS → dikembangkan pada akhir 1990-an; menggunakan ribuan sistem penyimpanan yang dibangun dari komponen murah untuk menyediakan penyimpanan petabyte ke pengguna dengan beragam kebutuhan.
- Pertimbangan desain.
  - Skalabilitas dan keandalan adalah fitur penting dari sistem; mereka harus dipertimbangkan dari awal, bukan pada beberapa tahap desain.
  - Sebagian besar ukuran file berkisar dari beberapa GB hingga ratusan TB.
  - Operasi yang paling umum adalah menambahkan file yang sudah ada; operasi tulis andom ke file sangat jarang.
  - Operasi membacanya berurutan.
  - Pengguna memproses data secara massal dan kurang peduli dengan waktu respons.
  - Model konsistensi harus dilonggarkan untuk menyederhanakan implementasi sistem tetapi tanpa membebani pengembang aplikasi.

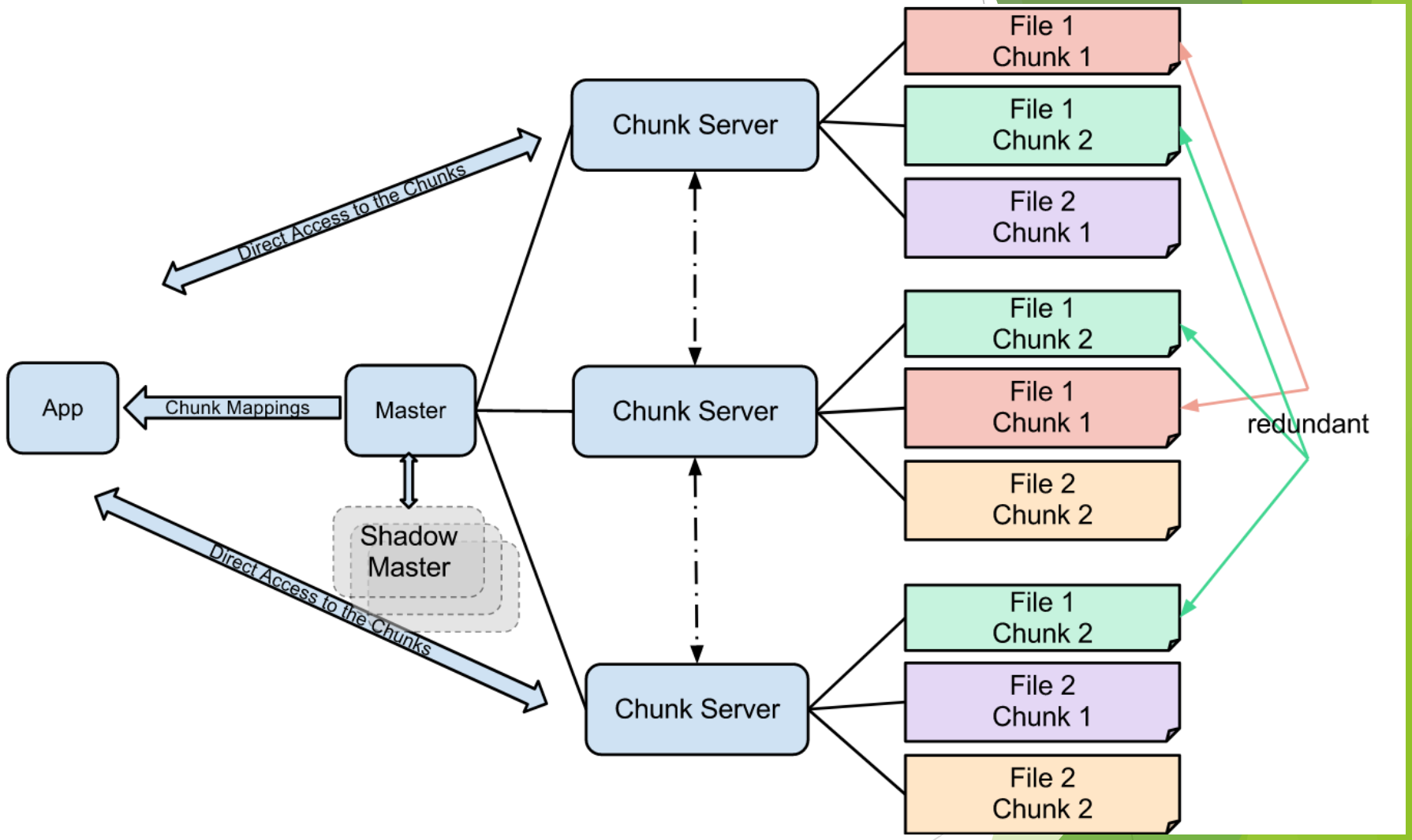
keputusan desain GFS

- Segmentasikan file dalam potongan besar.
- Menerapkan operasi penambahan file yang memungkinkan beberapa aplikasi yang beroperasi secara bersamaan untuk menambahkan ke file yang sama.
- Bangun klaster di sekitar jaringan interkoneksi bandwidth tinggi daripada latensi rendah. Pisahkan aliran kontrol dari aliran data. Transfer data pipa melalui koneksi TCP. Eksploitasi topologi jaringan dengan mengirimkan data ke node terdekat dalam jaringan.
- Hilangkan caching di situs klien. Caching meningkatkan overhead untuk menjaga konsistensi.
- Pastikan konsistensi dengan menyalurkan operasi file penting melalui master, komponen cluster yang mengontrol seluruh sistem.
- Minimalkan keterlibatan master dalam operasi akses file untuk memastikan skalabilitas.
- Mendukung chekpoint yang efisien dan mekanisme pemulihan yang cepat.
- Mendukung mekanisme pengumpulan file yang tidak berguna yang efisien.

GFS chunks

- File GFS adalah kumpulan segmen berukuran tetap yang disebut chunks.
- Ukuran potongan adalah 64 MB; pilihan ini dimotivasi oleh keinginan untuk mengoptimalkan kinerja untuk file besar dan untuk mengurangi jumlah metadata yang dikelola oleh sistem.
- Ukuran chunk yang besar meningkatkan kemungkinan bahwa beberapa operasi akan diarahkan ke chunk yang sama, sehingga mengurangi jumlah permintaan untuk mencari lokasi chunk dan, pada saat yang sama, memungkinkan aplikasi untuk mempertahankan koneksi jaringan yang persisten dengan server di mana potongan itu berada.
- Sebuah chunk terdiri dari 64 KB blok dan setiap blok memiliki checksum 32 bit.
- Potongan disimpan di sistem file Linux dan direplikasi di banyak situs; pengguna dapat mengubah jumlah replika, dari nilai standar tiga, ke nilai yang diinginkan.
- Pada saat pembuatan file, setiap potongan diberi pegangan potongan yang unik.

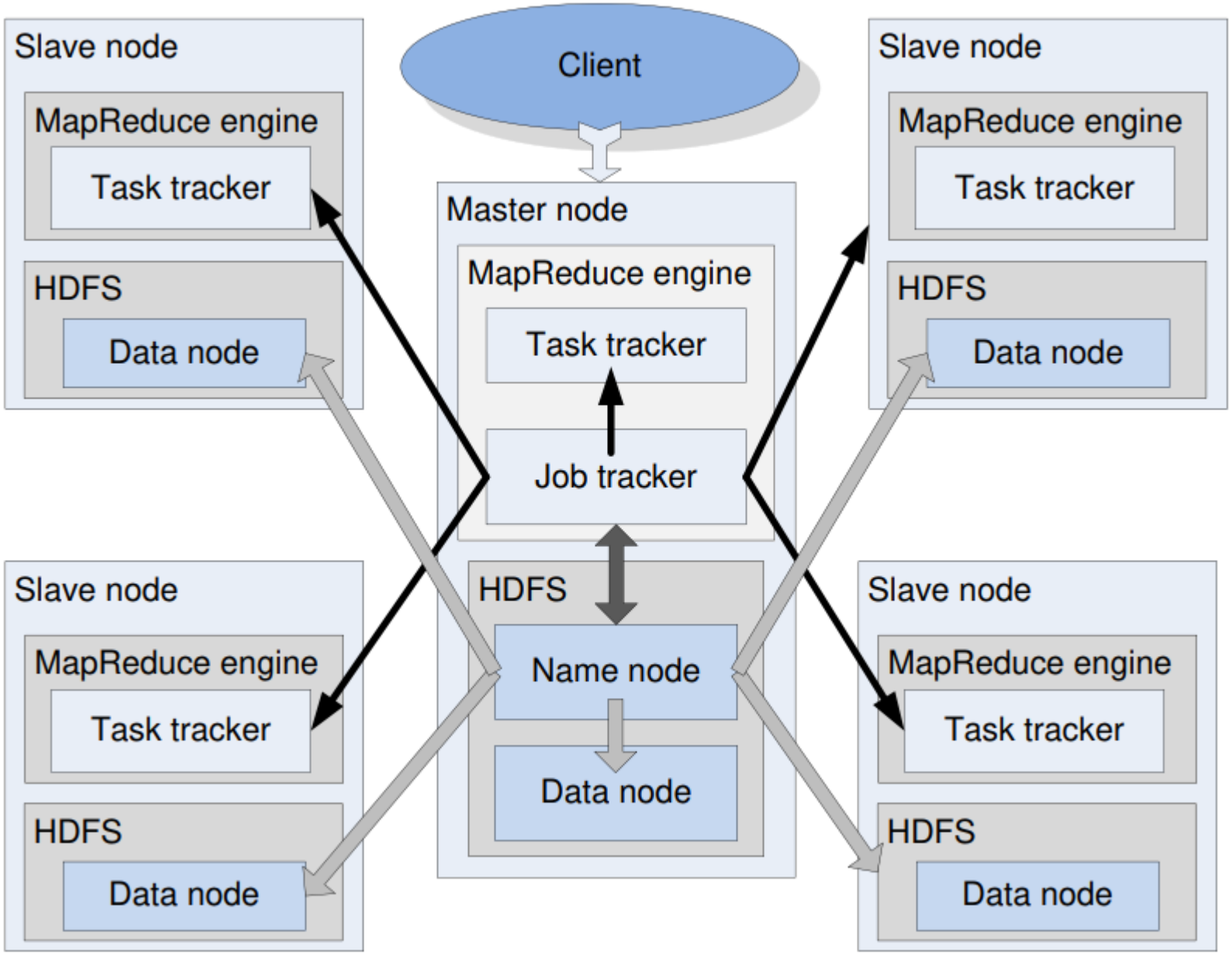
Arsitektur cluster GFS; master memelihara informasi status tentang semua komponen sistem; itu mengontrol sejumlah server chunk. Sebuah server chunk berjalan di Linux; menggunakan metadata yang disediakan oleh master untuk berkomunikasi langsung dengan aplikasi. Data dan jalur kontrol ditampilkan secara terpisah. Panah menunjukkan aliran kontrol antara aplikasi, master dan server chunk.



Apache Hadoop

- Apache Hadoop → open source, perangkat lunak berbasis Java, mendukung aplikasi terdistribusi yang menangani volume data yang sangat besar.
- Hadoop digunakan oleh banyak organisasi dari industri, pemerintah,dan penelitian; perusahaan IT besar misalnya, Apple, IBM, HP, Microsoft, Yahoo, dan Amazon, perusahaan media misalnya, New York Times dan Fox, jaringan sosial termasuk, Twitter, Facebook, dan Linkedln, dan lembaga pemerintah seperti Federal Reserve.
- Sistem Hadoop memiliki dua komponen, mesin MapReduce dan sebuah database. Basis data dapat berupa Hadoop File System (HDFS), S3 Amazon, atau CloudStore, implementasi GFS.
- HDFS adalah sistem file terdistribusi yang ditulis dalam Java; itu portabel, tetapi tidak dapat langsung dipasang pada sistem operasi yang ada.

Sebuah cluster Hadoop menggunakan HDFS; cluster termasuk master dan empat node slave. Setiap node menjalankan mesin MapReduce dan mesin database. Pelacak pekerjaan mesin master berkomunikasi dengan pelacak tugas di semua node dan dengan nama node HDFS. Node nama HDFS berbagi informasi tentang penempatan data dengan pelacak pekerjaan untuk meminimalkan komunikasi antara node tempat data berada dan node yang membutuhkannya.



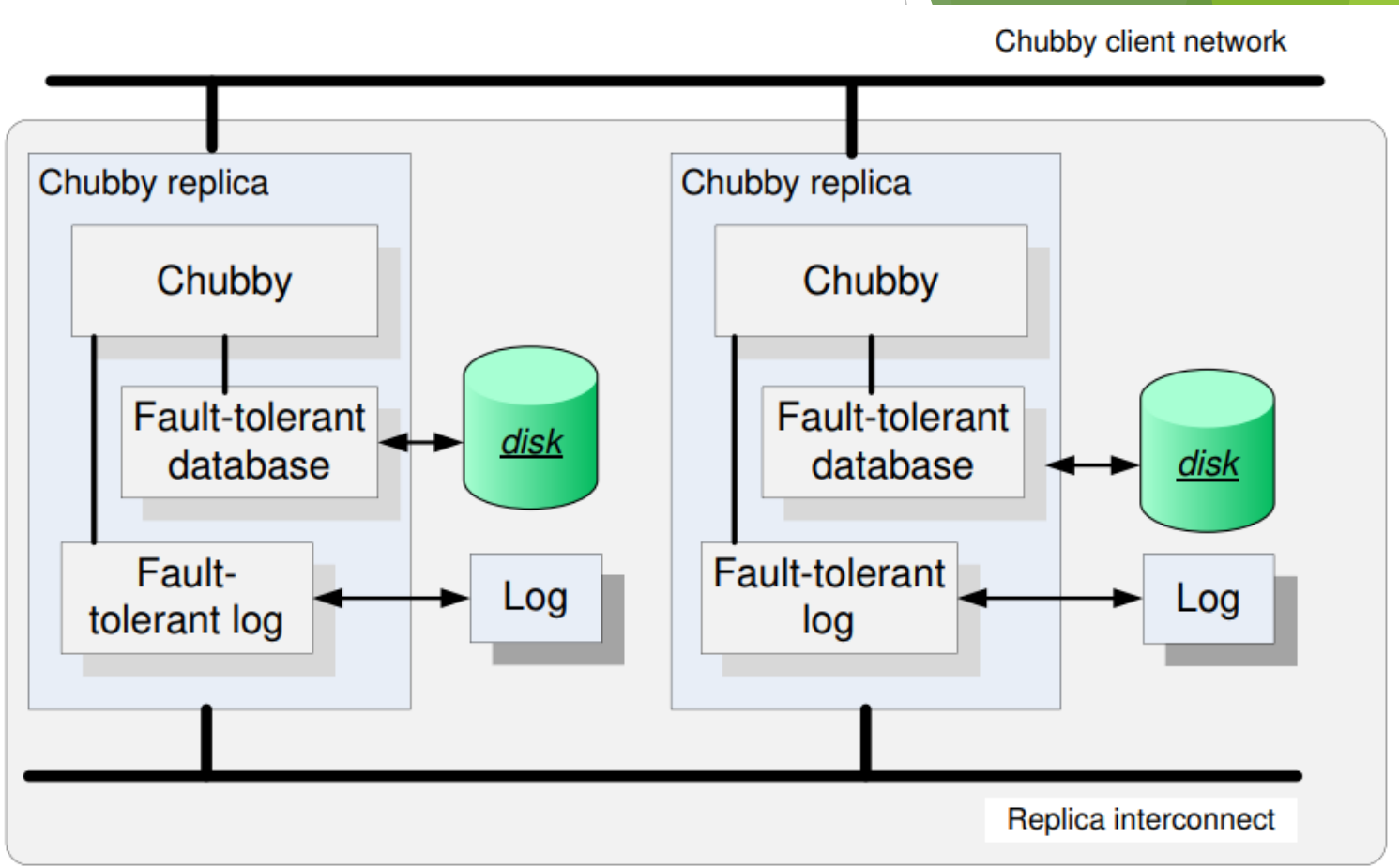
Chubby - layanan penguncian

- Kunci mendukung implementasi penyimpanan yang andal untuk sistem terdistribusi; memungkinkan akses terkontrol ke penyimpanan bersama dan memastikan atomitas operasi baca dan tulis.
- Permasalahan system terdistribusi, seperti pemilihan master dari sekelompok server data; misalnya, master GFS memelihara informasi status tentang semua komponen sistem.
- Dua pendekatan yang mungkin:
  - mendelegasikan kepada klien penerapan algoritma konsensus dan menyediakan perpustakaan fungsi yang diperlukan untuk tugas ini.
  - membuat layanan penguncian yang mengimplementasikan versi algoritma Paxos asinkron dan menyediakan perpustakaan untuk dihubungkan dengan klien aplikasi.
- Chubby -Berdasarkan algoritma Paxos yang menjamin keamanan tanpa asumsi waktu, kondisi yang diperlukan dalam sistem skala besar ketika penundaan komunikasi tidak dapat diprediksi; algoritme harus menggunakan jam untuk memastikan keaktifan dan mengatasi ketidakmungkinan mencapai konsensus dengan satu proses yang salah.

Algoritma Paxos

- Digunakan untuk mencapai kesepakatan pada kumpulan nilai, misalnya, urutan entri dalam log yang direplikasi.
- Fase-fase algoritma.
  - Pilih replika untuk menjadi master/koordinator. Ketika master gagal, beberapa replika mungkin memutuskan untuk mengambil peran master; untuk memastikan bahwa hasil pemilihan adalah unik, setiap replika menghasilkan nomor urut yang lebih besar dari nomor urut yang pernah dilihatnya, dalam rentang (1,r) di mana r adalah jumlah replika, dan menyiarkannya dalam pesan yang diusulkan. Replika yang belum melihat nomor urut yang lebih tinggi menyiarkan jawaban janji dan menyatakan bahwa mereka akan menolak proposal dari calon master lainnya; jika jumlah responden mewakili mayoritas replika, orang yang mengirim pesan usul terpilih sebagai master.
  - Master menyiarkan ke semua replika pesan terima termasuk nilai yang telah dipilihnya dan menunggu balasan, baik mengakui atau menolak.
  - Kesepakatan tercapai ketika sebagian besar replika mengirim pesan pengakuan; kemudian master menyiarkan pesan komit.

Arsitektur replika yang chubby; komponen Chubby mengimplementasikan protokol komunikasi dengan klien. Sistem menyertakan komponen untuk mentransfer file ke database yang toleran terhadap kesalahan dan komponen log yang toleran terhadap kesalahan untuk menulis entri log. Log toleransi kesalahan menggunakan protokol Paxos untuk mencapai konsensus. Setiap replika memiliki sistem file lokalnya sendiri; replika berkomunikasi satu sama lain menggunakan interkoneksi khusus dan berkomunikasi dengan klien melalui jaringan klien.



Pemrosesan transaksi

- Pemrosesan Transaksi Online (Online Transaction Processing (OLTP)) banyak digunakan oleh banyak aplikasi cloud.
- Persyaratan utama:
  - Waktu respons yang singkat.
  - Skalabilitas.
    - Penskalaan vertikal data dan beban kerja didistribusikan ke sistem yang berbagi sumber daya, misalnya, inti/prosesor, disk, dan mungkin RAM
    - Penskalaan horizontal sistem tidak berbagi baik primer maupun penyimpanan sekunder.
- Pencarian model alternatif untuk menyimpan data di cloud dimotivasi oleh kebutuhan aplikasi OLTP:
  - kurangi latensi dengan menyimpan data yang sering digunakan di memori.
  - memungkinkan beberapa transaksi terjadi pada saat yang sama dan mengurangi waktu respons dengan mendistribusikan data pada sejumlah besar server.

Permasalahan pada OLTP

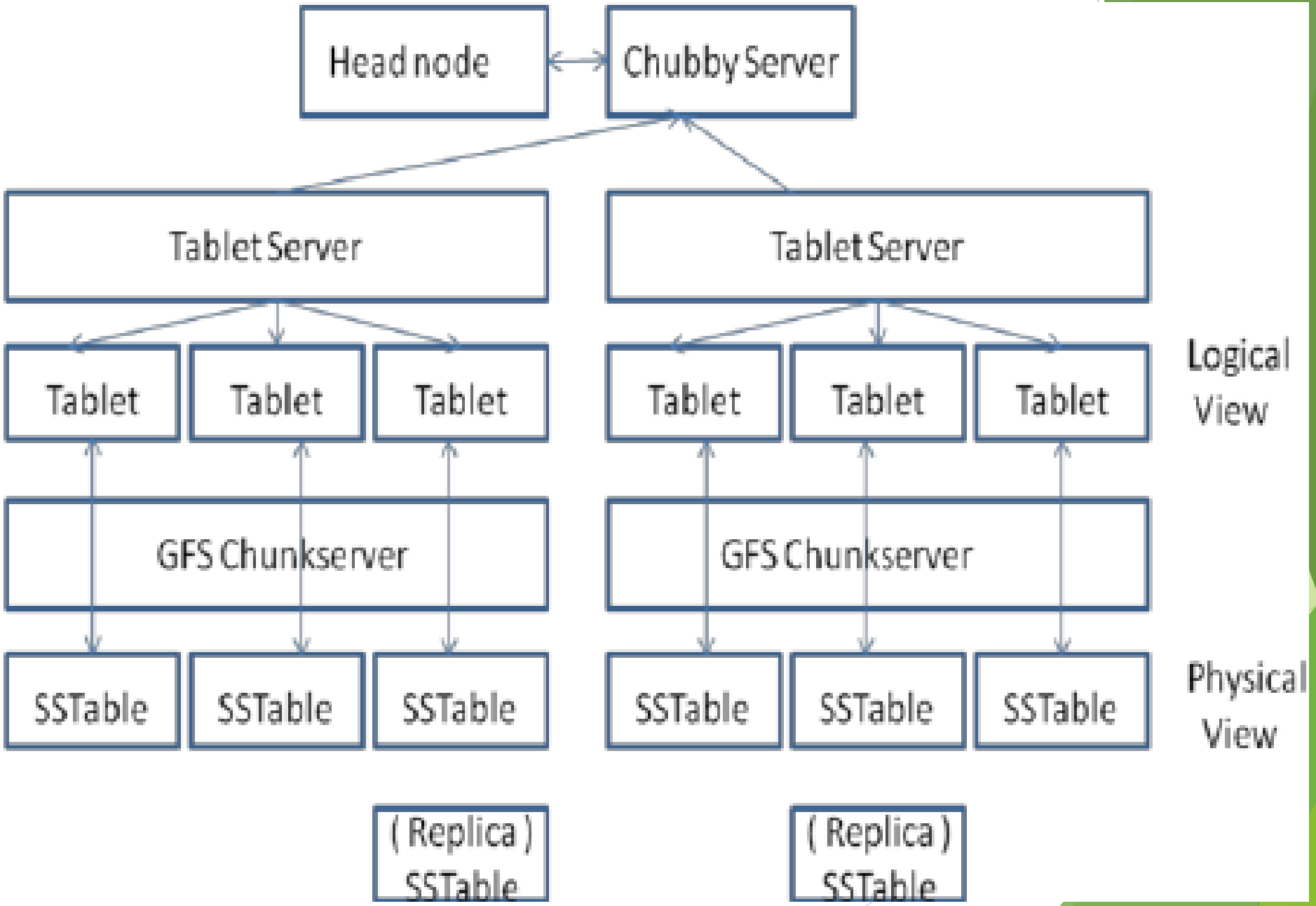
- Empat hal menjadi masalah pada OLTP:
  - Logging - mahal karena database tradisional memerlukan ketahanan transaksi sehingga, setiap penulisan ke database hanya dapat diselesaikan setelah log diperbarui.
  - Locking - untuk menjamin atomisitas, transaksi mengunci setiap catatan dan ini membutuhkan akses ke tabel kunci.
  - Latching – banyak operasi memerlukan multi-threading dan akses ke struktur data bersama, seperti tabel kunci, menuntut latches jangka pendek untuk koordinasi. Latch adalah penghitung yang memicu suatu peristiwa ketika mencapai nol; misalnya threads master memulai penghitung dengan jumlah worker threads dan menunggu untuk diberi tahu ketika semuanya selesai.
  - Buffer Management.
- Rincian jumlah instruksi untuk operasi ini diDBMS yang ada adalah: 34,6% untuk manajemen buffer, 14,2% untuk latching, 16,2% untuk locking, 11,9% untuk logging, dan 16,2% untuk optimasi manual.

Database NoSQL

- Penamaan NoSQL menyesatkan. Stonebreaker mencatat bahwa “kinerja yang membutakan bergantung pada pemindahan overhead. Overhead semacam itu tidak ada hubungannya dengan SQL, ini berkisar pada implementasi tradisional dari transaksi ACID, multi-threading, dan manajemen disk.”
- Pendekatan soft-state memungkinkan data menjadi tidak konsisten dan mentransfer tugas untuk mengimplementasikan hanya subset dari properti ACID yang diperlukan oleh aplikasi tertentu ke pengembang aplikasi.
- Sistem NoSQL memastikan bahwa data pada akhirnya akan konsisten di beberapa titik waktu mendatang, alih-alih menegakkan konsistensi pada saat transaksi dilakukan.
- Atribut:
  - Skala dengan baik.
  - Jangan tunjukkan satu pun titik kegagalan.
  - Memiliki dukungan bawaan untuk keputusan berbasis kesepakatan.
  - Mendukung partisi dan replikasi sebagai dasar.

Bigtable

- Sistem penyimpanan terdistribusi yang dikembangkan oleh Google
  - untuk menyimpan sejumlah besar data.
  - skala hingga ribuan server penyimpanan.
- Sistem menggunakan
- Sistem File Google → untuk menyimpan data pengguna dan informasi sistem.
- Chubby distributed lock service → untuk menjamin operasi baca dan tulis atom; direktori dan file di namespace Chubby digunakan sebagai kunci.
- Data sederhana dan fleksibel memodelkan array sel multidimensi.
  - A row key → arbitrer hingga 64 KB dan rentang baris dipartisi menjadi tablet yang berfungsi sebagai unit untuk penyeimbangan beban. timestamps waktu yang digunakan untuk mengindeks berbagai versi data dalam sel adalah bilangan bulat 64-bit; interpretasinya dapat ditentukan oleh aplikasi, sedangkan defaultnya adalah waktu suatu peristiwa dalam mikrodetik.
  - A column key → terdiri dari string, satu set karakter yang dapat dicetak, dan string arbitrer sebagai kualifikasi.


Head node
Chubby Server
Tablet Server
Tablet Server
Tablet
Tablet
Tablet
Tablet
Tablet
Tablet
Logical View
GFS Chunkserver
GFS Chunkserver
SSTable
SSTable
SSTable
SSTable
SSTable
SSTable
Physical View
(Replica) SSTable
(Replica) SSTable

Megastore

- Penyimpanan yang dapat diskalakan untuk layanan online. Banyak digunakan secara internal di Google, menangani sekitar 23 miliar transaksi setiap hari, 3 miliar transaksi tulis dan 20 miliar baca.
- Sistem yang tersebar di beberapa pusat data memiliki kapasitas yang sangat besar.
- Setiap partisi direplikasi di pusat data di wilayah geografis yang berbeda. Sistem mendukung semantik ACID penuh dalam setiap partisi dan memberikan jaminan konsistensi terbatas di seluruh partisi.
- Algoritme konsensus Paxos digunakan untuk mereplikasi data pengguna utama, metadata, dan informasi konfigurasi sistem di seluruh pusat data dan untuk mengunci. Versi algoritma Paxos tidak memerlukan master tunggal, sebagai gantinya setiap node dapat memulai operasi baca dan tulis ke log write-ahead yang direplikasi ke sekelompok rekan simetris.
- Sistem memanfaatkan Bigtable secara ekstensif.

Link Video Penjelasan

https://drive.google.com/file/d/1AXfFlnn8TDBbYrsZsIV9XxbvC0oPb4PU/view?usp=sharing

Terimakasih