Sistem Terdistribusi

Distributed File System

Background

- Computers are everywhere
- Network as communication media
- Resources can be in different computer or different formats (context: file system)
- Service that gives access transparency to users
  - Allows remote file access via networks, but works just like local file access
  - Client doesn't need to know what file system is used

- Client Server
  - Server provide files/directories
  - Client access files/directories
  - Operation: Add/Remove, Read/Write
- Distributed file system
  - FS shared to many distributed clients
  - Communication through shared files

Background

- Distributed file system (DFS) adalah sebuah sistem di mana banyak pengguna dapat berbagi berkas dan sumber daya penyimpanan.
- A DFS manages set of dispersed storage devices
- Overall storage space managed by a DFS is composed of different, remotely located, smaller storage spaces
- There is usually a correspondence between constituent storage spaces and sets of files

DFS Structure

- Service – software entity running on one or more machines and providing a particular type of function to a priori unknown clients
- Server – service software running on a single machine
- Client – process that can invoke a service using a set of operations that forms its client interface
- A client interface for a file service is formed by a set of primitive file operations (create, delete, read, write)
- Client interface of a DFS should be transparent, i.e., not distinguish between local and remote files

Tantangan

- Transparency
  - Acces: client can acess remote files as local files
  - Location: client can't tell the file location
  - Migration: file can move to other server
  - Replication: multiple copies of file exists
  - Concurrency: multiple client access
  - Failure: client & program have to run even when server fail
- Flexibility
  - support multiple FS types
  - server can be added/removed
- Consistency
- Security
  - User access

Tantangan (2)

- Fault tolerance
- Performance
  - Load balancing
- Scalability
  - File and users exponential growth
- Heterogeneity
  - OS, Programming language, software
- Efficiency
  - Working on network introduce latency and other overhead

Layanan file terdistribusi

- Layanan Dasar
  - tempat penyimpanan tetap untuk data dan program
  - operasi terhadap file (create, open, read,…)
  - multiple remote clients (dalam intranet)
  - file sharing
  - menggunakan semantic one-copy update umum, melalui RPC
- Perkembangan baru
  - persistent object stores (storage of objects)
  - Persistent Java, Corba, …
  - multimedia terdistribusi (contoh: file server Tiger video)

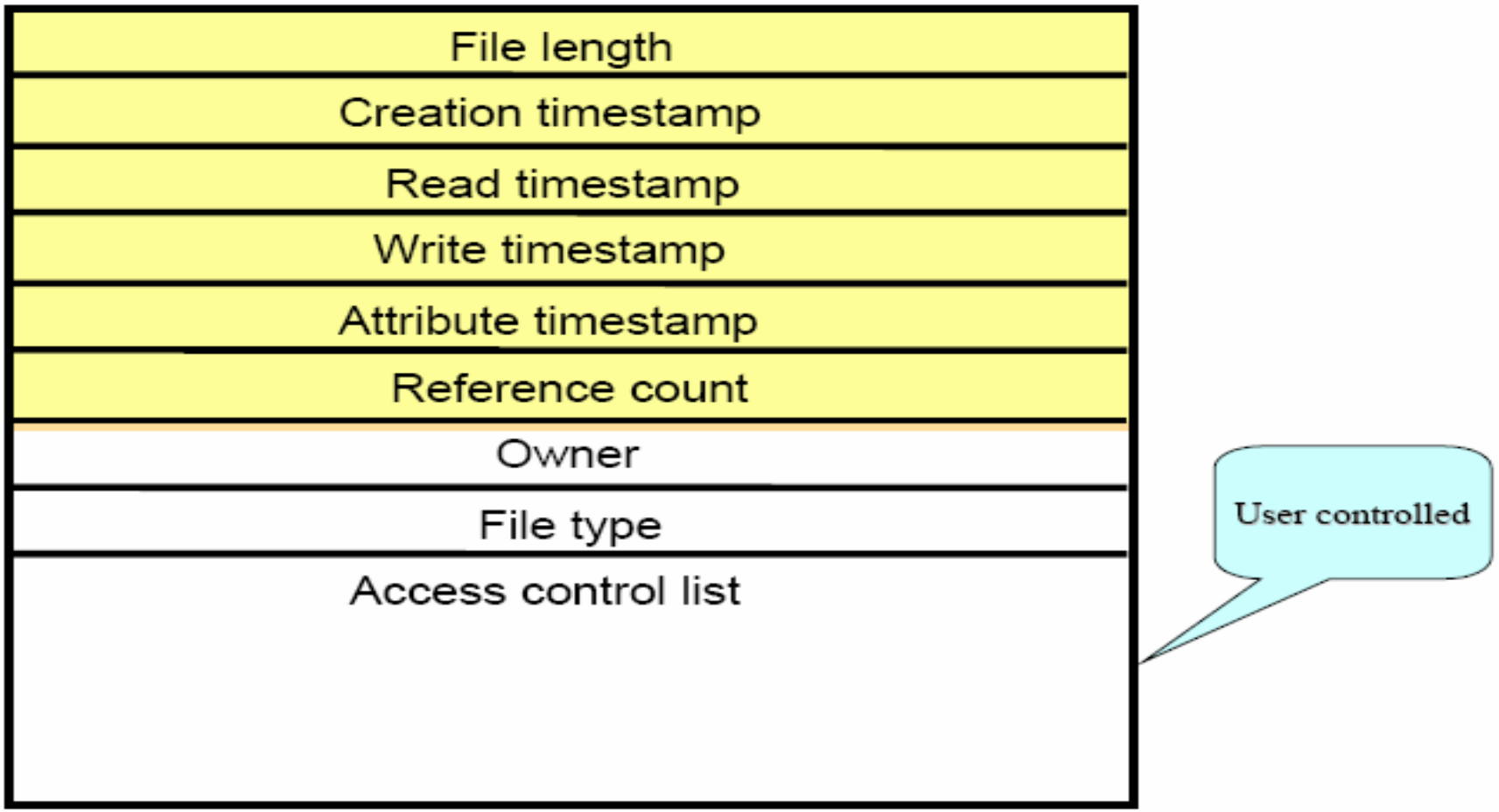
Operasi File

- Operasi terhadap files (=data + attributes)
  - create/delete
  - attribute query/modify
  - open/close
  - read/write
  - access control

Organisasi Storage

- directory structure (hierarchical, pathnames)
- metadata (file management information)
- file attributes
- informasi struktur directory

Atribut file



Layanan file

- Interface
  - Files: sequence of bytes
  - Attributes: owner, size, date, etc
  - Protection: access control list
  - Immutable file: no modification to the file
- Permission on Linux
  - Read (4)
  - Write (2)
  - Execute (1)

Opsi layanan file

- Stateful
  - server menyimpan informasi tentang file yang open, posisi sekarang (current position) dan file locks
  - open (dibuka) sebelum access dan kemudian ditutup
  - performa yang lebih baik – pesan yang lebih pendek, dimungkinkan untuk read-ahead
  - server failure - kehilangan state
  - client failure - tables fill up
  - menyediakan file locks

Opsi Layanan File

- Stateless
  - server tidak menyimpan state informasi
  - file operations idempotent, harus mengandung semua yang diperlukan (longer message)
  - perancangan file server yang lebih simpel
  - dapat dengan mudah di-recovery apabila client ataupun server crash
  - locking membutuhkan extra lock server untuk mempertahankan state

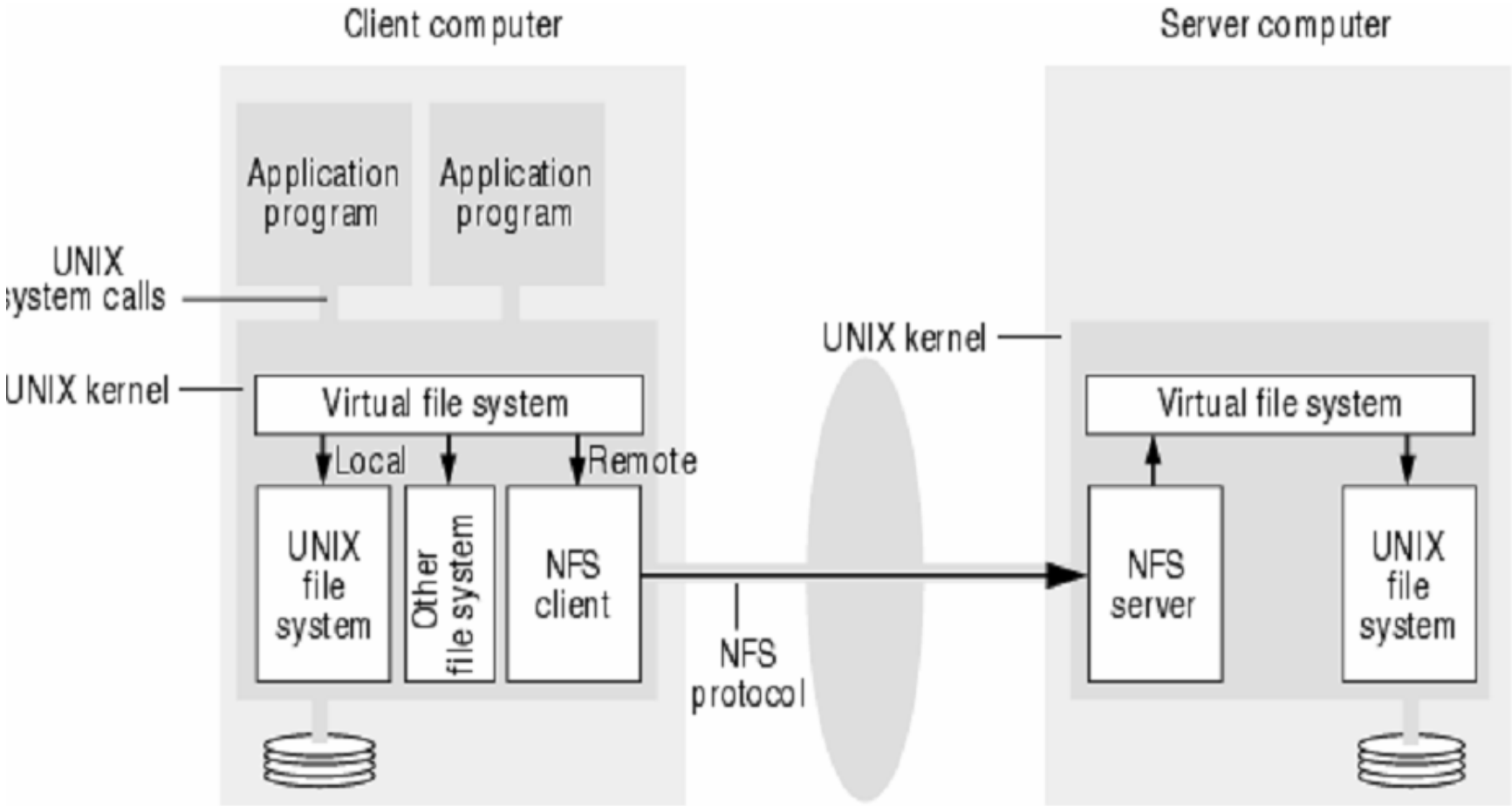
File sharing

- Multiple clients berbagi (share) file yang sama untuk akses read/write
- One-copy update semantics
  - setiap read melihat dampak semua writes sebelumnya
- suatu proses write akan segera visible ke client yang sudah siap membuka file untuk reading
- Problems!
  - caching: memelihara konsistensi antara beberapa copies yang sulit di achieve
  - akses serialisasi dengan menggunakan file locks (memperngaruhi performance)
  - trade-off antara consistency dan performance

NFS (Network file system)

- Dikembangkan oleh Sun Microsystem (1984)
- Dipakai secara luas pada industri dan pendidikan pada tahun 1985
- Sekarang menjadi public source
  - RFC 1094, http://tools.ietf.org/html/rfc1094
  - RFC 1813, http://tools.ietf.org/html/rfc1813
  - RFC 3530, http://tools.ietf.org/html/rfc3530
- Sudah diimplementasikan pada hampir semua Sistem operasi modern (client/server)

Arsitektur NFS



NFS

- Server side
  - NFS protocol independent from file system
  - NFS Server run as a daemon
  - /etc/exports (Linux) specify what directory are exported to whom under which policy
  - Transparent caching (read ahead)
- Client side
  - Mounting (explicit/auto)
  - Support diskless workstations (thin clients)
  - /etc/fstab (Linux)

Operasi-operasi NFS

Operation	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory in a given directory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory from a directory
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a file name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name stored in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more attribute values for a file
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

NFS di Linux

- Linux support NFS Client/Server
- Require portmap to handle RPC connection
  - server that converts RPC program numbers into DARPA protocol port numbers
- Configuration file on server: /etc/exports
- Client only mount with option nfs on /etc/fstab
- http://nfs.sourceforge.net/nfs-howto

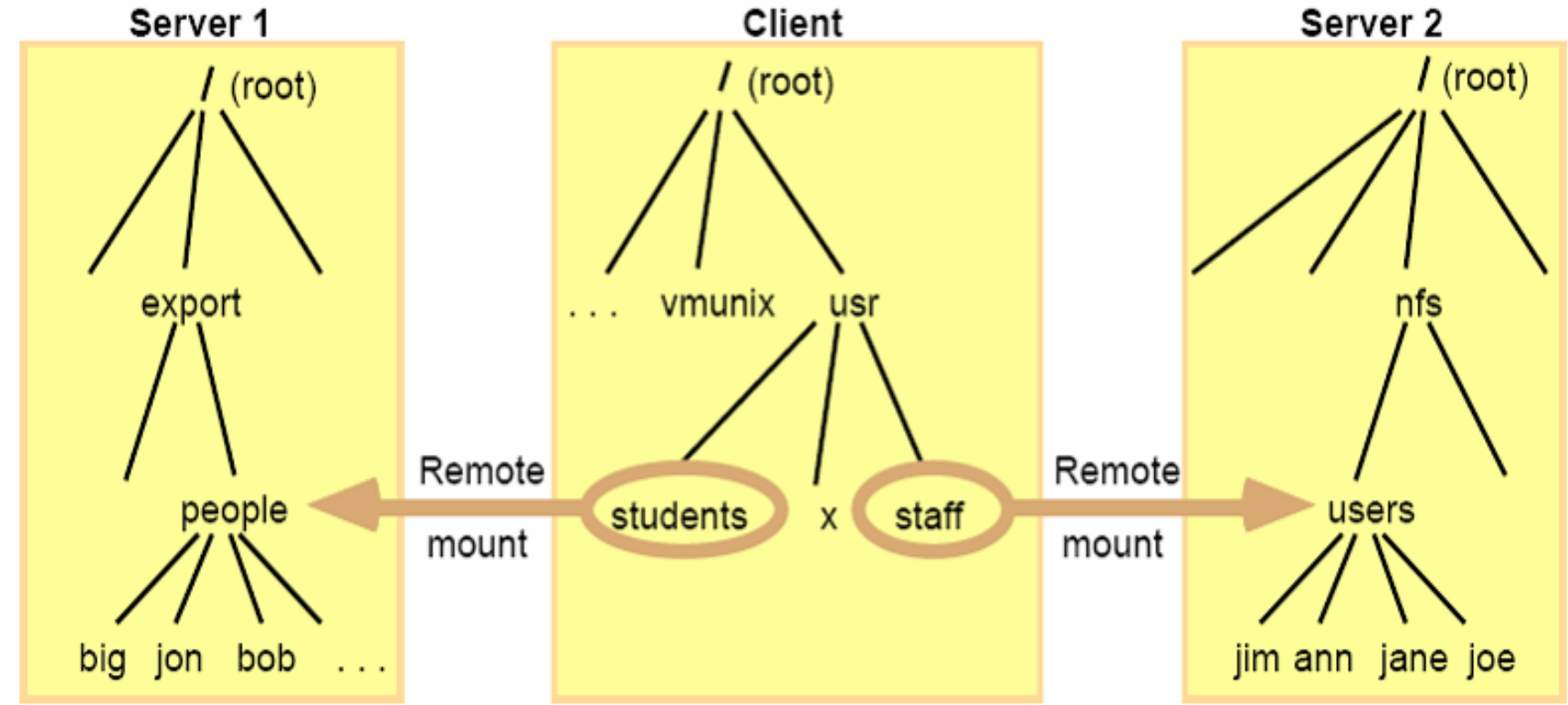
File handle pada NFS

- Setiap file memiliki identifier
- File identifier dalam NFS disebut
  File Handle
- File handle menyimpan informasi tentang
  lokasi penyimpanan file (file system)
- NFS Client Module mensimulasikan opera si file biasa (lokal), meskipun sebenarnya dia bekerja secara remote (access transpa rency)

Sun NFS

- Implementasi NFS dengan menggunakan Sun
- RPC (Sun Remote Procedure Call)
- NFS Client 'call' fungsi pada NFS Server
- Spesifikasi NFS mendefinisikan remote interface yang bisa dipakai oleh client
- Pengiriman data bisa menggunakan TCP/ UDP

Remote mount



Note: The file system mounted at */usr/students* in the client is actually the sub-tree located at */export/people* in Server 1; the file system mounted at */usr/staff* in the client is actually the sub-tree located at */nfs/users* in Server 2.

Caching

- Akses data yang terus menerus bisa membebani server dan berpengaruh pada performa
- Solusi: Caching (menyimpan sebagian data pada RAM)
- Caching sebaiknya dilakukan pada server/client
- Server Caching
  - Read-ahead
  - Delayed write

Cache Location – Disk vs. Main Memory

- Advantages of disk caches
  - More reliable
  - Cached data kept on disk are still there during recovery and don’t need to be fetched again
- Advantages of main-memory caches:
  - Permit workstations to be diskless
  - Data can be accessed more quickly
  - Performance speedup in bigger memories
  - Server caches (used to speed up disk I/O) are in main memory regardless of where user caches are located; using main-memory caches on the user machine permits a single caching mechanism for servers and users

Caching

- Read-Ahead
  - Asumsi file dibaca secara sequential
  - Kernel membaca blok berikutnya dari file sebelum aplikasi memintanya
- Bisa menghasilkan I/O yg useless dan memakan resource memory jika ternyata tidak sesuai yang diharapkan
- Delayed Write
  - Perubahan akan disimpan pada disk jika buffer hendak dipakai oleh request lain
  - Pada UNIX, operasi sync akan dilakukan per 30s

Caching

- Write-through
  - Data disimpan pada memory cache server
    dan ditulis pada disk mengirimkan reply ke client
  - Client yakin bahwa data sudah tersimpan permanen k etika menerima jawaban
- Write-commit
  - Penyimpanan pada disk dilakukan jika
    diterima perintah commit
  - Banyak dipakai oleh NFS Client standar

Caching

- Client : Melakukan cache terhadap operasi read, write, getattr, lookup, dan readdir
- Bisa menimbulkan masalah konsistensi karena seringkali data sudah diupdate oleh proses lain
- Solusi: client poll server secara interval apakah data masih up-to-date

File Replication

- Replicas of the same file reside on failure-independent machines
- Improves availability and can shorten service time
- Naming scheme maps a replicated file name to a particular replica
  - Existence of replicas should be invisible to higher levels
  - Replicas must be distinguished from one another by different lower-level names
- Updates – replicas of a file denote the same logical entity, and thus an update to any replica must be reflected on all other replicas
- Demand replication – reading a nonlocal replica causes it to be cached locally, thereby generating a new nonprimary replica

Remote file sharing

- Dikembangkan oleh AT&T, dikembangkan untuk Unix system (tidak heterogenous)
- Didesain secara statefull dan connection-oriented.
- Server dapat mendeteksi client crash, sehingga konsistensi dari cache terjamin.

Andrew File System

- Dikembangkan oleh Carnegie Mellon Univesity.
- Mendukung sharing informasi untuk skala besar (100 – 10000+ pengguna).
- Asumsi penggunaan file pada AFS :
  - Kebanyakan dari file-file merupakan file yang kecil.
  - Lebih sering membaca daripada menulis file.
  - Kebanyakan file dibaca/tulis oleh satu pengguna.
- AFS menggunakan file serving keseluruhan berada di server dan keseluruhan file caching berada di client.

Google file system

- Made by Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung (2003)
  - http://labs.google.com/papers/gfs.html
- Successor of BigFiles, created by Google Founders (Larry Page and Sergey Brin)
- Idea: “Store data reliably even in the presence of unreliable machines"
- Optimized to run on Google clusters which consists of thousands of nodes per cluster
- Ensure availability by replicating files at least on three different computers in a given server cluster

Google file system

- Assumption :
  - Failure occurs often
  - Huge files
  - Large streaming reads
  - Large appends
  - Concurrent appends
  - Bandwidth more important than latency

Data Center

NEXT