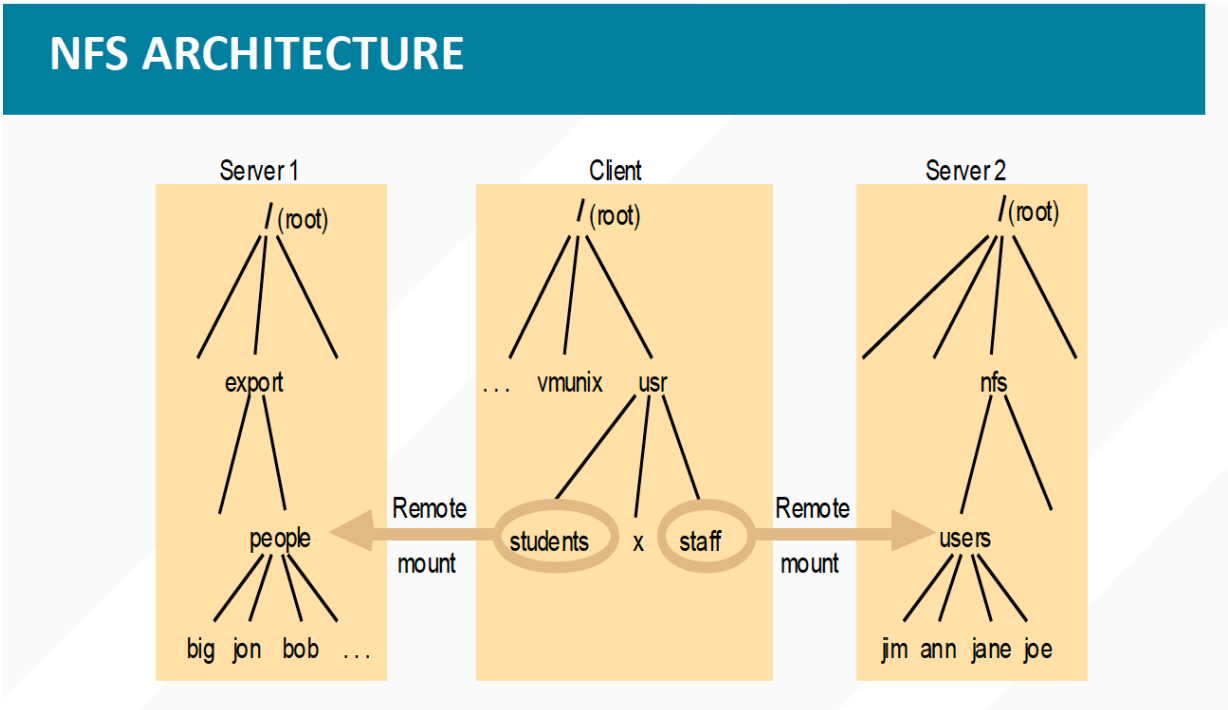
L6 NETWORKED STORAGE

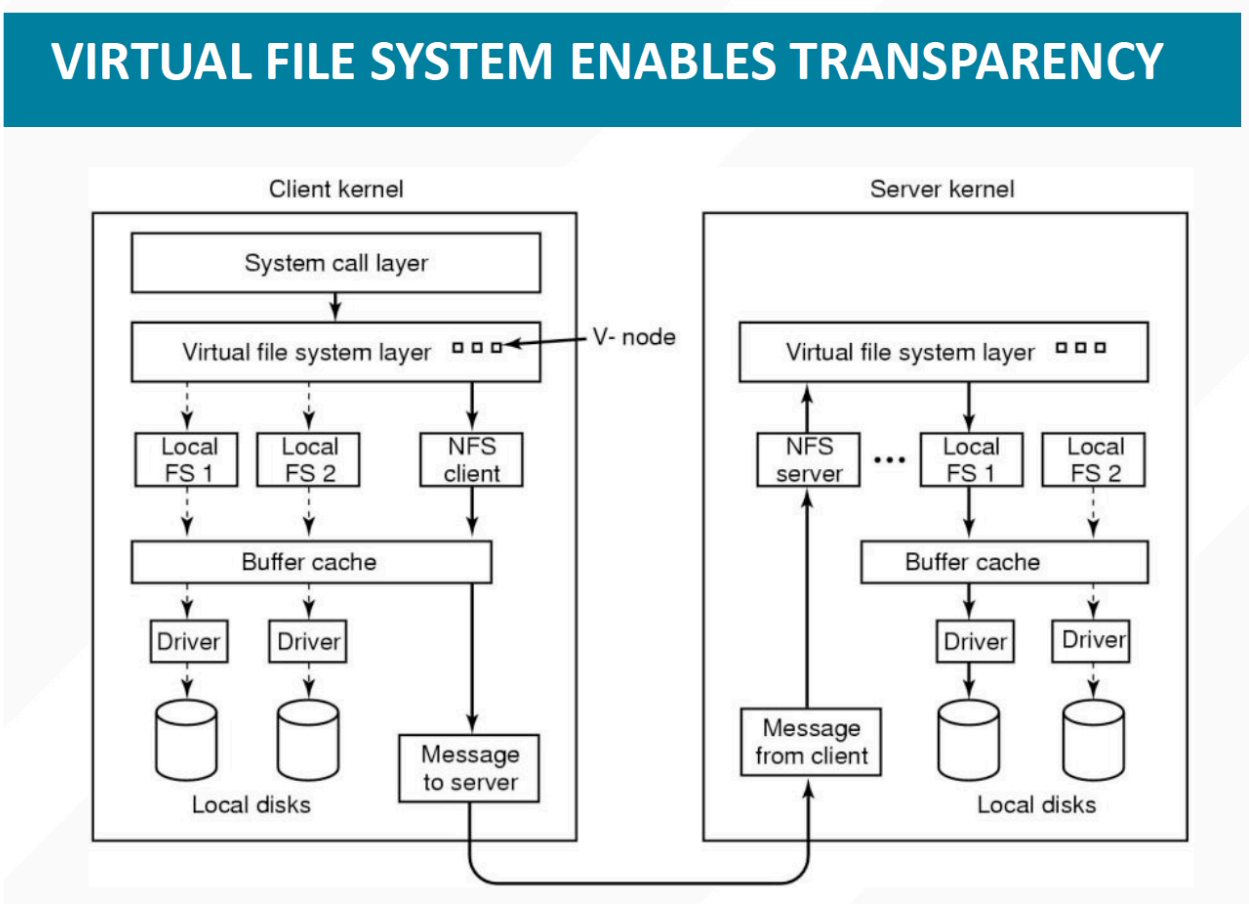
Abstraction

dicks => Directories, Files, Links => NFS(Network File System)

NFS

Mount remote file system as local directories

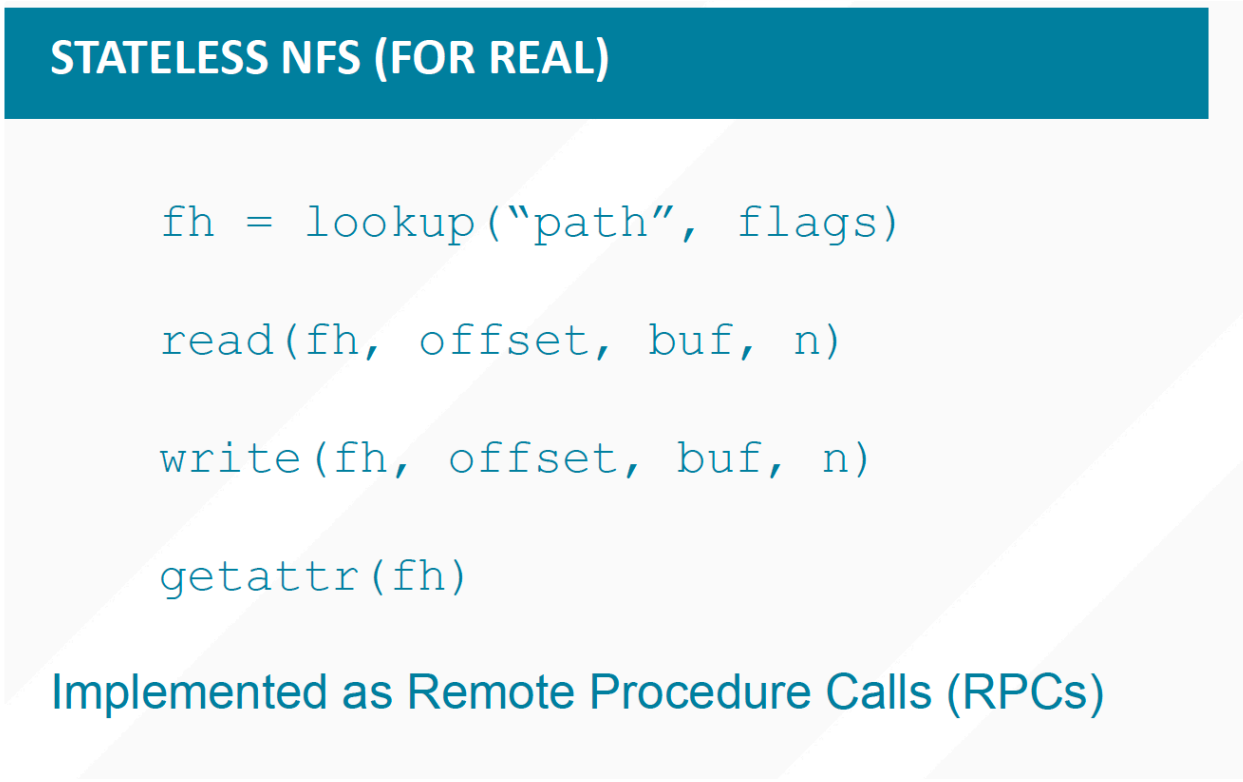




We can read file as from virtual file systems like we do on our local laptop.

Since we don't want to waste time to find the file over and over again, we use file descritptor as inode.

NFS



The server stores the server state at client.

It can be unchanged even if another user changed it.

Use generation to specify different versions.

Like a multi-version database.

Difference with local and remote FS

Local FS:

always see the data from most recent write

If we want to achieve this with NFS, it will be a huge cost due to high latency and scalability.

Caching

- Read ahead: Pre fetch blocks before needed
- Write through: All writes sent to server
- Write behind: Writes locally buffered, send as batch

Server maintain per-Client state?

Stateful

- Pros
  - Smaller requests
  - Simpler req processing
  - Better cache coherence, file locking, etc.
- Cons
  - Per-client state limits scalability
  - Fault-tolerance on state required for correctness

Stateless

- Pros
  - Easy server crash recovery
  - No open/close needed
  - Better scalability
- Cons
  - Each request must be fully self-describing
  - Consistency is harder, e.g., no simple file locking

SOFT VS. HARD STATE

- Hard state: Don't lose data
  - Durability: State not lost
    - Write to disk, or cold remote backup
    - Exact replica or recoverable (DB: checkpoint + op log)
  - Availability (liveness): Maintain online replicas
- Soft state: Performance optimization
  - Then: Lose at will
  - Now: Yes for correctness (safety), but how does recovery impact availability (liveness)?