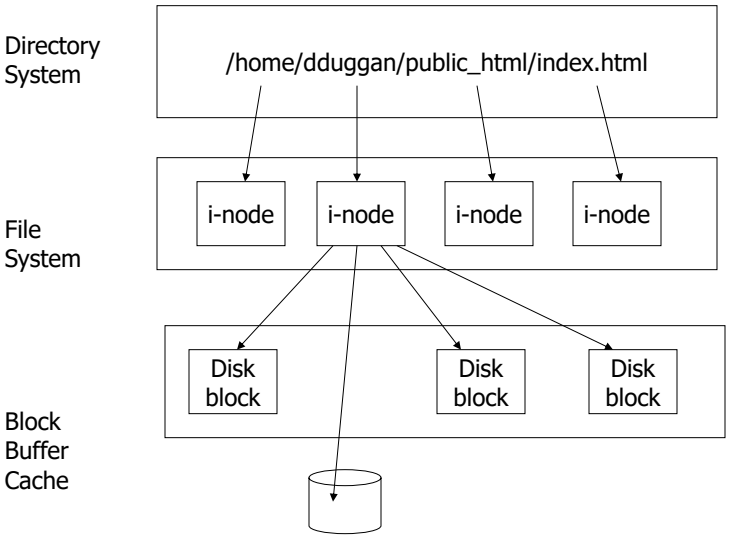
FILE SYSTEMS

2

2

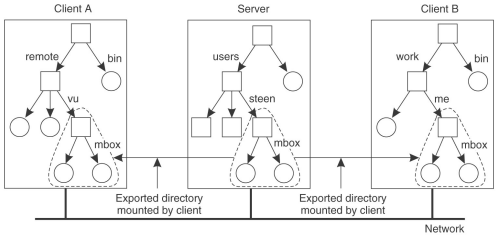
Layers in File Systems



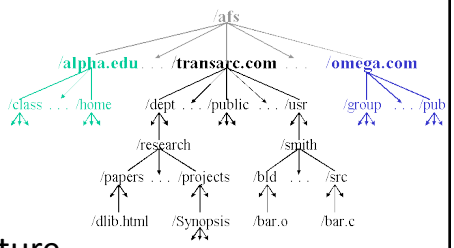
3

3

Naming Scheme



- Mount remote directories to local
  - Coherent **local** directory tree
  - Ex: Unix/Linux with NFS; Windows mapped drives

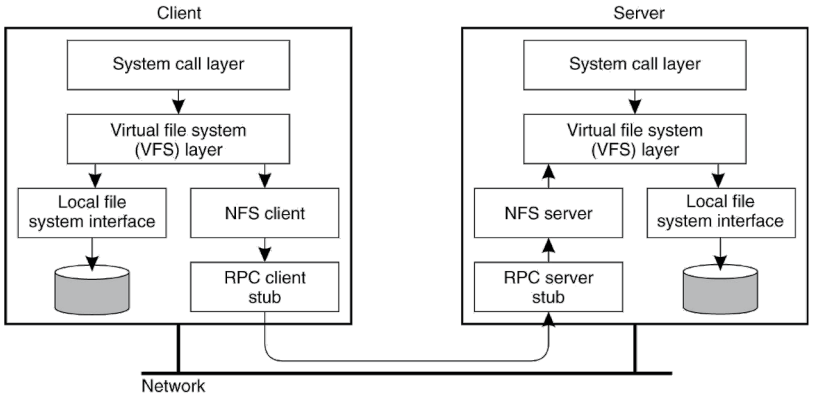


- Total integration of component file systems
  - Single **global** name structure
  - Ex: AFS

4

4

Location Transparency

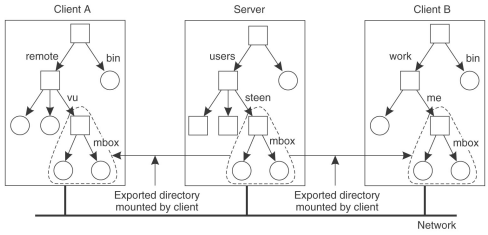


5

5

Location Independence (Migration Transparency)

- NFS: Not location independent
  - Server: `export /root/fs1/`
  - Client: `mount server:/root/fs1 /fs1`

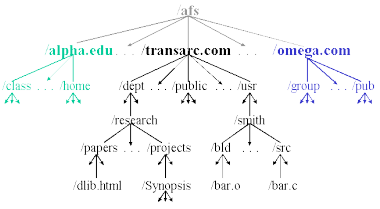


6

6

Location Independence (Migration Transparency)

- AFS: global volumes
  - Global directory /afs;
  - /afs/cs.stevens.edu/vol1/…; /afs/cs.njit.edu/vol1/
  - File id = <vol_id, vnode #, uniquifier>
  - "Volume location database"
    - vol_id→ server_ip mappings
    - Shared by servers



7

7

File Server Semantics

- Stateless
  - + Crash recovery
  - - File locking
  - Ex: NSF v3
- Stateful
  - - Crash recovery
  - + File locking
  - Ex: AFS, NFS v4

8

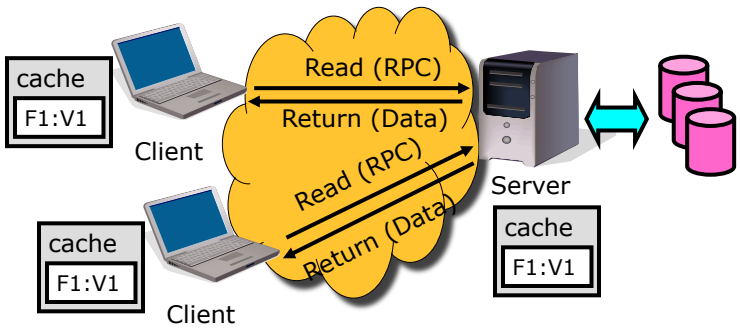
8

CACHING POLICIES

9

9

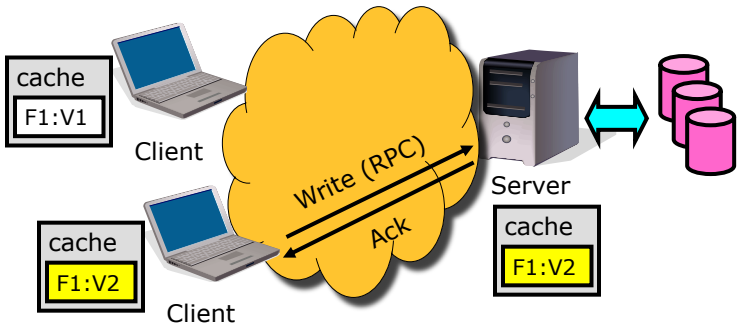
File Caching



10

10

Cache Consistency



11

11

Cache Update Policies

- When does the client update the master file?
- Write-through: write data to server ASAP
- Delayed-write: cache, write to server later
  - Better local performance
  - Network I/O
  - Poor reliability
- Write-on-close

12

12

File Sharing Semantics

- Sequential Semantics
  - No cache
  - Performance problems
  - Write-through cache
    - Must notify clients holding copies
- Session Semantics
- Immutable Files
- Atomic Transactions

13

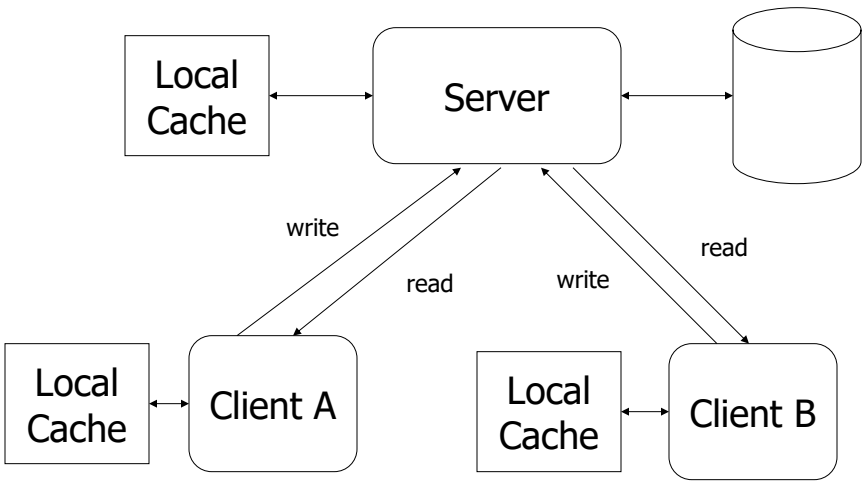
13

NFS: NETWORK FILE SYSTEM

14

14

Caching in NFS



15

15

Client Caching

- Client checks validity of cached files
  - File open
  - Periodic polling
- Client responsible for writing out cache
  - Periodic scan, flush of dirty blocks

NFS Semantics

- Locking
  - Originally separate (stateless)
  - Stateful for locking since NFS v4
- Unix file semantics not guaranteed
  - E.g., read after write
- Session semantics not even guaranteed
  - Intermediate writes
  - Client may implement close-to-open

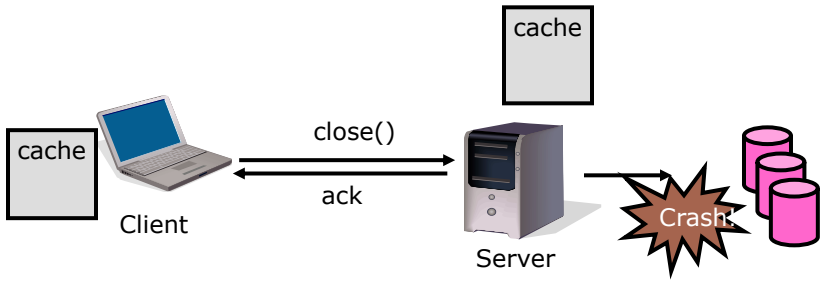
NFS Implementation

- Remote procedure calls for all operations
  - Originally over UDP
  - Using TCP since NFSv4
- Lost requests are simply re-transmitted
  - At-least-once semantics

NFS Failure Recovery

- Server crashes transparent to client
  - Each client request self-contained
  - Client retransmits request if crash
    - *"Server not responding, still trying"*
- Client crashes transparent to server

Caching and Failures

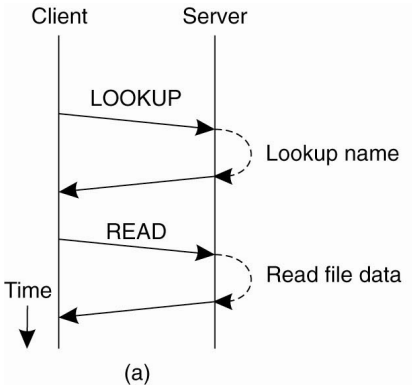


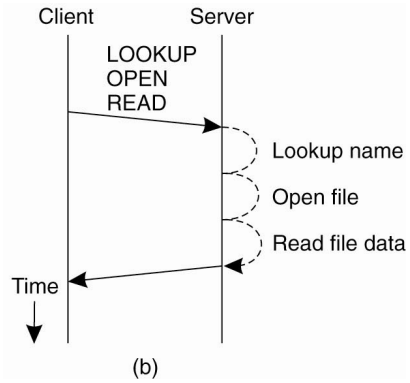
- Suppose client implements close-to-open
- Server acks updates
- Server crashes before flushing updates to disk
- Fixes:
  - Server flushes updates before ack
  - NVRAM for server
  - NFS v3: client buffers updates until COMMIT acknowledged

20

20

Compound RPC





21

21

Open Delegation

- Server may delegate open/close/locking to client
- Operations done locally at client
- Periodic cache checks unnecessary
- Lease and revocation via callback
  - RPC from server to client

22