CS323 Operating Systems
Multi-Computers/Distributed Systems

Yuanyuan Zhou
Lecture 32
4/16/2003

Content of this lecture

- Distributed/Network File Systems
  - Background
  - Naming and Transparency
  - Remote File Access
  - Stateful versus Stateless Service
  - File Replication
  - Example Systems
- Multi-processor (read textbook)

Midterm 2 Review

- Still grading
- Will be available early next week
- Not very well at short questions
- How to improve
  - Questions after each chapter
  - Lecture discussion questions
  - Thoroughly understand the concepts and algorithms.
- Statistics
  - Names

Background

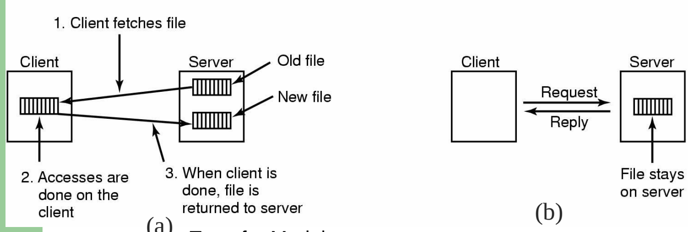
- Distributed file system (DFS)
  - allow remote file systems to be accessed as if they were local.
- Server
  - machine with remote file system
- Client
  - machine with application accessing remote file system
- Client Interface
- How does it scale?
- Is it transparent ?
- Are the names organized the same way from any machine?

Windows Example

Unix Example

```
yyzhou|csil-linux5|~/cs323/public_html/lectures|[47]% df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/sda1              4127076   2282724   1634708  59% /
none                    256816         0    256816   0% /dev/shm
/dev/sda3               505636     86794    392737  19% /var
/dev/sda5             11820060     71052  11148584   1% /scratch/scratch0
csil-linux:/mounts/csil-linux/disks/0/software
                      63322504  11696340  48409552  20% /usr/dcs/software
csil-server1:/home/student1
                      69344972  15260700  53390824  23% /home/student
csil-server1:/home/group2/class
                      27850792  23365684   4206604  85% /home/class
csil-server1:/usr/dcs/csil-projects
                      17332444   6498416  10660704  38% /usr/dcs/csil-projects
crladmin-csil:/usr/dcs/sysadm
                       8705504   3978012   4640436  47% /usr/dcs/sysadm
csil-server1:/usr/dcs/csil
                       3099096   2740444    296668  91% /usr/dcs/csil
csil-server1:/home/group1/faculty
                      15487084   6758120   8574092  45% /home/faculty
yyzhou|csil-linux5|~/cs323/public_html/lectures|[48]%
```


DFS



- Transfer Models

  (a) upload/download model

  (b) remote access model

Naming and Transparency

- Naming -- mapping between logical and physical objects
- Multilevel mapping -- hiding where and how the disk is stored
- Transparent DFS hides location of files-- is this good or bad. E.g. fault-tolerance may require copies of same file to be kept on DIFFERENT systems.
- File replication -- accessed like a single file but allows redundancy
- Ownership -- keeping the ownership of a file
- Storage unit -- limited size of disk partitions

Naming Structures

- Location Transparency
  - unique reference to set of physical blocks
  - Allow sharing conveniently (same name)
  - Expose correspondence between files and machines.
- Location Independence
  - File can be moved without changing name
  - Allows better distribution of files
  - Separates file naming hierarchy from storage hierarchy

Naming Schemes

- Files named by hostname and local name: guarantees unique system wide name
- Attach remote directories to local directories, giving appearance of a coherent directory tree. Only mounted remote directories can be accessed.
- Total integration of the component file systems.
  - Single global name space
  - Name space fragments when machines not available-- arbitrary.

Remote File Access

- RPC of file operations
- Use client-side inode-like representation of remote file to record file descriptors and file status.
- Cache disk blocks, buffer caches or whole files on local machine to improve performance and reduce network traffic.
- Cache consistency problem
- Network transfer unit (1.5k on Ethernet) is not same as block sizes. Need disassembly and reassembly.

Discussion

- Client caching using disks vs. client caching using memory tradeoff?

Location

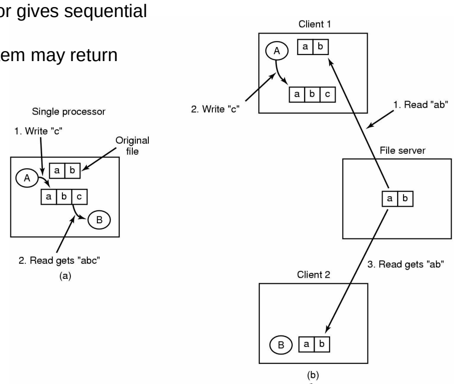
- Disk cache
  - more reliable in event of crash
  - Flushing disk cache to remote file system requires three extra reads and writes from memory.
- Memory cache:
  - permits diskless workstations
  - allows fast access to data
  - performance improves as memory available increases

Cache Update Policy

- Write through --
  - write data through to disk as soon as data is output to cache.
  - reliable but poor performance.
  - temporary files written to disk unnecessarily.
- Delayed-write -- modifications written to memory. Only written to server when close or needed by open
  - Poor reliability -- crash destroys data
  - Flush cache to remote disk periodically (30 secs.)
  - Accumulate clusters and periodically flush clusters
  - By waiting, avoid temp files since they are often removed within seconds of creation.

Consistency Problem

- Semantics of File sharing
  - (a) single processor gives sequential consistency
  - (b) distributed system may return obsolete value



Consistency

- Client Initiated Approach
  - Client initiates a validity check
  - Server checks whether local data is consistent with master copy
- Server Initiated Approach
  - Server records, for each client, the file blocks it caches.
  - When server detects inconsistency, it block access or issues a invalidate request to client.

Comparison of Caching and Remote Service

- Many remote accesses as fast as local ones. Read ahead.
- Writes infrequent -- good for caches
- Temporary files need not be written to disk.
- Servers contacted infrequently instead of on each access-- better for scalability and network traffic
- But relative overhead handling big chunks of data less than for small chunks.

Stateless versus Stateless Service

- Stateful
  - Server records which client is accessing file
  - Allows easy read/write synchronization
  - Permits easy caching of data -- knows about read ahead
  - Server doesnt know if client crashes -- clean up is problem
  - Server crash leaves clients needing to be cleaned up
  - Protocol must be reliable, exactly once -- need to know write occurred
  - UNIX stateful
- Stateless
  - Each request independent from previous requests (contains state info)
  - File operations idempotent -- can repeat writes
  - No need to open/close connections
  - Client can crash without causing any server difficulties
  - Server can crash without causing client difficulties
  - longer request messages
  - NFS stateless

File Replication

- Replicas of same file reside on failure-independent machines
- Improves availability
- Replicas should be invisible, yet distinguished at lower levels
- Updates to replicas must be duplicated -- need exactly once semantics.
- Demand replication -- build a cache of whole file

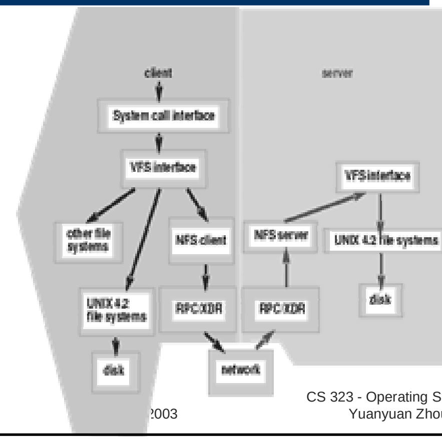
Example: SUN Network File System

- Uses UDP/IP protocol and stateless server
- Each system is regarded as independent
- A remote file system is mounted over a local file system directory
- Local file system directory is no longer visible.
- The mount command uses name of remote machine
- Access rights, users need to have same ids, group ids.
- No concurrency control mechanisms, modified data must be committed to server disk before request returned to client to avoid problems
- Works on heterogeneous machines by using a machine independent RPC (network order).

Major Layers of NFS Architecture

- vnode -- network wide unique (like an inode but for a network)
- RPC and NFS Service layer -- NFS Protocol
- Path name look up (past mount point) requires RPC per name.
- client cache of remote vnodes for remote directory names
- client cannot access another server through a server... remote file systems are always mounted directly

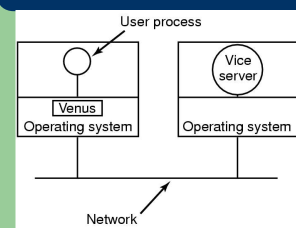
NFS Architecture



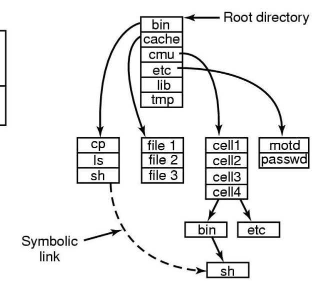
NFS Caching

- File blocks and file-attribute caches
- Attributes used only if up to date. Discarded after 60 seconds.
- Read-ahead and delayed write techniques used.
- Delayed write used even for concurrent access (not UNIX semantics.)
- New files may not be visible for 30 seconds.
- Updated files may not be visible to systems with file open for reading for a while.

Andrew File Systems



- AFS – Andrew File System
  - workstations grouped into cells
  - note position of venus and vice



Client's view

Example: Andrew

- Aimed at scalability
- Clients are not servers
- Local name space and shared name space
- Local name space is root file system
- Whole file caching
- Clients may access files from any workstation using same name space
- Security imposed at server interfaces -- no client programs run on servers.
- Access lists for files
- Client workstation interacts with servers only during opening and closing of files
- Reading and writing bytes performed by kernel