Last Class: Distributed Systems and RPCs

- Servers export procedures for some set of clients to call
- To use the server, the client does a procedure call
- OS manages the communication

Today: Distributed File Systems

- One of the most common uses of distributed systems
- **Basic idea:**
  - Given a set of disks attached to different nodes
  - share disks between nodes as if all the disks were attached to every node.
- **Examples:**
  - **Edlab:** One server node with all the disks, and a bunch of diskless workstations on a LAN.
  - **AppleShare:** Every node is both a server with a disk and a client

Distributed File Systems: Issues

- Naming and Transparency
- Remote file access
- Caching
- Server with state or without
- Replication

Naming and Transparency

- **Issues**
  - How are files named?
  - Do file names reveal their location?
  - Do file names change if the file moves?
  - Do file names change if the *user* moves?
- **Location transparency:** the name of the file does not reveal the physical storage location.
- **Location independence:** The name of the file need not change if the file's storage location changes.
- Most naming schemes used in practice do not have location independence, but many have location transparency

Naming Strategies: Absolute Names

- **Absolute names:** <machine name: path name>
- Examples: AppleShare, Win NT
- **Advantages:**
- **Disadvantages:**

Naming Strategies: Mount Points

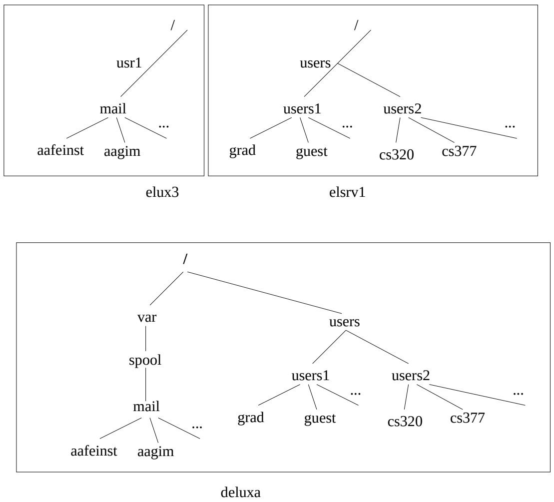
- Mount Points (NFS - Sun's Network File System)
  - Each host has a set of local names for remote locations.
  - Each host has a mount table (/etc/fstab) that specifies <remote path name @ machine name> and a <local path name>.
  - At boot time, the local name is bound to the remote name.
  - Users then refer to the <local path name> as if it were local, and the NFS takes care of the mapping
- **Advantages:**
- **Disadvantages:**

NFS: Example

Partial contents of /etc/fstab for Edlab machines:

```
/usr1/mail@elux3.cs.umass.edu:/var/spool/mail
/users/users1@elsrv1:/users/users1
/users/users2@elsrv1:/users/users2
/users/users3@elsrv2:/users/users3
/users/users4@elsrv2:/users/users4
/courses/cs300@elsrv3:/courses/cs300
/rcf/mipsel/4.2/share@elsrv1:/exp/rcf/share
/rcf/common@elsrv1:/exp/rcf/common
```


NFS: Example



Naming Strategies: Global Name Space

- Single name space: CMU's Andrew and Berkeley's Sprite
  - No matter which node you are on, the file names are the same.
  - Set of workstation clients, and a set of dedicated file server machines.
  - When a client starts up, it gets its file name structure from a server.
  - As users access files, the server sends copies to the workstation and the workstation caches the files
- **Advantages:**
- **Disadvantages:**

Remote File Access and Caching

Once the user specifies a remote file, the OS can do the access either

1. remotely, on the server machine and then return the results using RPC (called *remote service*), or
2. can transfer the file (or part of the file) to the requesting host, and perform local accesses (called *caching*)

Caching Issues:

- Where and when are file blocks cached?
- When are modifications propagated back to the remote file?
- What happens if multiple clients cache the same file?

Remote File Access and Caching

Location

1. Local disk
   - **Advantages:**
   - **Disadvantages:**
2. Local memory
   - **Advantages:**
   - **Disadvantages:**

Cache Update Policies

When to write local changes to the server has a central role in determining distributed file system performance.

- **Write through:** yields the most reliable results since every write hits the remote disk before the process continues, but it has the poorest performance.
  - Caching with write through is equivalent to using remote service for all writes, and exploits caching only for reads.
- **Write back:** yields the quickest response time since the write need only hit cache before the process continues.
  - It reduces network traffic and the number of writes to the disk for repeated writes to the same disk block, since only one of the writes will go across the network.
  - If a user machine crashes, the unwritten data is lost.
  - Write-back when file is closed, a block is evicted from cache, or every 30sec.

Cache Consistency

- **Client-initiated consistency:** Client contacts the server and asks if its copy is consistent with the server's copy.
  - Can check every access.
  - Can check at a given interval.
  - Can check only upon opening a file.
- **Server-initiated consistency:** Server detects potential conflicts and invalidates caches
  - Server needs to know:
    * which clients have cached which parts of which files.
    * Which clients are readers and which are writers.

Case Study: Sun's Network File System

- NFS is the standard for distributed UNIX file access.
- NFS is designed to run on LANs
- Nodes are both servers and clients.
- Servers have no state.
- Uses a mount protocol to make a global name local
  1. /etc/exports lists the local names the server is willing to export.
  2. /etc/fstab lists the global names that the local nodes import. A corresponding global name must be in /etc/exports on the server.

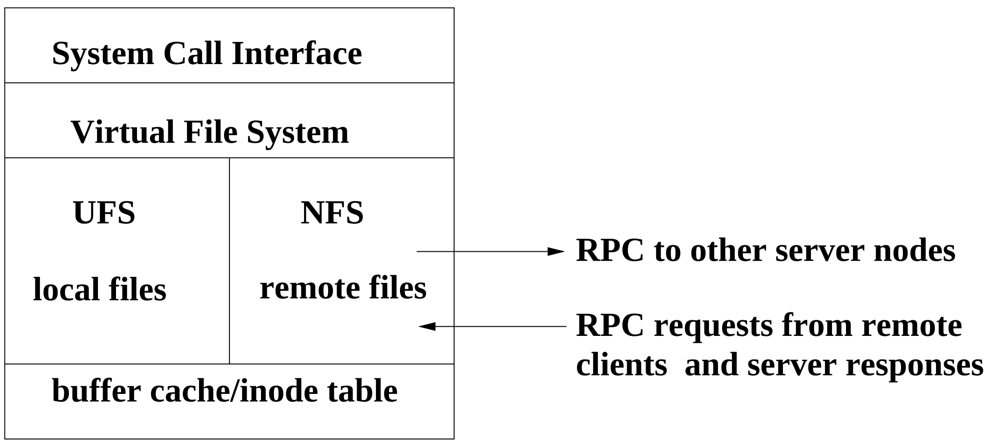
NFS Implementation

- NFS defines a set of RPC operations for remote file access:
  1. directory search, reading directory entries
  2. manipulating links and directories
  3. accessing file attributes
  4. reading/writing files

- Does not rely on node homogeneity - heterogeneous nodes must simply support the NFS mount and remote access protocols using RPC.

- Users may need to know different names depending upon the node to which they logon.

NFS Implementation



Summary

- Naming
  - Desire name independence, but it is difficult to attain
  - Location dependent names are most prevalent
- Speed up remote file access with caching
- Need to write changes back to disk