Appleshare: Every node is both a server with a disk and a client

Examples:
- LAN
- Edlib: One server node with all the disks, and a bunch of diskless workstations on a
- Share disks between nodes as if all the disks were attached to every node.
- Given a set of disks attached to different nodes

Basic Idea:
One of the most common uses of distributed systems

Today: Distributed File Systems

OS manages the communication

To use the server, the client does a procedure call

Servers export procedures for some set of clients to call

Last Class: Distributed Systems and RPCs
Most naming schemes used in practice do not have location transparency, but many have location transparency.

Location transparency: The name of the file does not reveal the physical location.

Issues

Naming and Transparency

Naming and Transparency

Replication

Remote file access

Caching

Server with state or without state

Distributed File Systems: Issues
Disadvantages:

Advantages:

Care of the mapping
- Users then refer to the <local path name> as if it were local, and the NFS takes
- At boot time, the local name is bound to the remote name.<n>remote path name @ machine name</n> and a <local path name>
- Each host has a mount table (/etc/mtab) that specifies
- Each host has a set of local names for remote locations.

Mount Points (NFS - Sun's Network File System)

Naming Strategies: Mount Points

Naming Strategies: Absolute Names
Partial contents of /etc/ftpd for Edlab machines:

```
rcf/commonرف/ exp/rcf/common
rcf/megadapt/ exp/rcf/share
config/320/322/users/training
users/users2/users/george
users/users2/users/larry
users/users2/users/david
usr/local/wx3.cs.nwu.edu/var/spool/mail
```
What happens if multiple clients cache the same file?
When are modifications propagated back to the remote file?
Where and when are the blocks cached?

Caching Issues:

1. Remotely on the server machine and then return the results using RPC.
2. Can transfer the file (or part of the file) to the requesting host, and perform local access (called caching).
3. Once the user specifies a remote file, the OS can do the access either

Remote File Access and Caching

Disadvantages:

Advantages:

Single name space: CMU's Andrew and Berkeley's Sprite

Naming Strategies: Global Name Space
Write-back: yields the quickest response time since the write need only wait cache before the process continues.

- Write-back when file is closed, a block is evicted from cache, or every 30sec.
- If a user cache time is lost: the unwritten data is lost.
- The same disk block, since only one of the writes will go across the network.
- If reduces network traffic and the number of writes to the disk for repeated writes to hit cache before the process continues.

Write-back:

- Exploits caching only for reads.
- Caching with write-through is equivalent to using remote service for all writes, and

Write-through:

- Yields the most reliable results since every write hits the remote disk before the process continues, but it has the poorest performance.

When to write local changes to the server has a central role in determining

### Cache Update Policies

<table>
<thead>
<tr>
<th>Disadvantages:</th>
<th>Advantages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Local memory</td>
<td>1. Local disk</td>
</tr>
</tbody>
</table>

### Remote File Access and Caching
name must be in /etc/exports on the server.
  2. /etc/exports lists the global names that the local nodes import. A corresponding local
     file on the server is willing to export.

     • Uses a mount protocol to make a global name local
     • Servers have no state.
     • Nodes are both servers and clients.
     • NFS is designed to run on LANs
     • NFS is the standard for distributed UNIX file access.

Case Study: Sun’s Network File System

• Invalidates caches

  Server-initiated consistency: Server detects potential conflicts and
  * Which clients are readers and which are writers.
  * Which clients have cached which parts of which files.
  - Server needs to know:

  • copies are consistent with the server’s copies.

Cache Consistency

• Can check only upon opening a file.
  - Can check at a given interval.
  - Can check every access.
### NFS Implementation

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

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

1. Directories search, directory entries
2. Manipulating links and directories
3. Accessing file attributes
4. Reading/Writing files

NFS defines a set of RPC operations for remote file access:
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