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

- LAN
- Example:

  - 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
independence, but many have location transparency.

 Most naming schemes used in practice do not have location

file's storage location changes.

location independence: The name of the file need not change if the

storage location.

location transparency: the name of the file does not reveal the physical

Issues

Naming and Transparency

Distributed File Systems

Issues
have different names

Disadvantages: Single unified strategy hard to maintain, same file can
reboots

Advantages: Location transparent, remote name can change across care of the mapping

- Users think relative 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.
- Each host has a mount table ( /etc/exports) 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

- Not fault tolerant
- Make sharing harder
- File is location dependent, and thus cannot move.
- User must know the complete name and is aware of which files are local and which

Disadvantages:

- Scales easily
- No global state
- It is easy to add and delete new names
- Finding a fully specified file name is simple.

Advantages:

Examples: AppleShare, Win NT

Absolute names: machine name: path name

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

```
root
```

```
```
Disadvantages:

- Global names are bound late, moving them is easier.
- Since names are bound late, moving them is easier.
- The global name space is consistent regardless of where you login.
- Naming is consistent and easy to keep consistent.

Advantages:

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

Naming Strategies: Global Name Space

Global Name Space
Remote File Access and Caching

What happens if multiple clients cache the same file?

What are modifications propagated back to the remote file?

Where and when are file blocks cached?

Caching Issues:

1. Perform local access (called caching).
2. Can transfer the file (or part of the file) to the remote host, and then return the results using RPC.

Once the user specifies a remote file, the O/S can do the access either

Remote File Access and Caching
Write-back: When the file is closed, a block is evicted from cache, or every 30sec.

- If a user machine crashes, 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: Yields the quickest response time since the write need only exploit caching only for reads.

- Caching with write throughput 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 distributed file system performance.

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

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 global
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

---

Whichever clients are readers and which are writers.
* Which clients have cached which parts of which files.

- Server needs to know:
  - Invalidates caches
  - Server-initialized consistency: Server detects potential conflicts and
  - Can check only upon opening a file.
  - Can check at a given interval.
  - Can check every access.

- Copy is consistent with the server's copy.

Client-initialized consistency: Client contacts the server and asks if its

Cache Consistency
Clients and server responses
RPC requests from remote
RPC to other server nodes

buffer cache/inode table
remote files
local files
NFS
UFS
Virtual File System
System Call Interface

NFS Implementation

• Which they logon, users may need to know different names depending on the node to support the NFS mount and remote access protocols using RPC.

• Does not rely on node homogeneity - heterogeneous nodes must simply
  1. directory search, read/write 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:

NFS Implementation
Need to write changes back to disk

- Speed up remote file access with caching

- Location dependent names are most prevalent
- Desirable independence, but it is difficult to attain

**Summary**