

# CSE 451: Operating Systems Winter 2005

## Distributed filesystems

Steve Gribble

## Distributed File Systems

- Basic idea:
  - support sharing of files and sharing of devices (disks) network wide.
- To an application or a client, the filesystem looks just like a centralized, local filesystem
  - but under the covers, it has a distributed implementation
  - read blocks from remote hosts, instead of local disks

## Basic Issues

- File naming
  - how are files named?
  - are those names location transparent (is the file location visible to the user)?
  - are those names location independent?
    - do the names change if the file moves?
    - do the names change if the user moves?

3/4/05

© 2005 Steve Gribble

3

## Basic Issues

- Caching
  - caching exists for performance reasons
  - where are file blocks cached?
    - On the file server?
    - On the client machine?
- Coherency
  - what happens when a cached block/file is modified
  - how does a node know when its cached blocks are out of date?

3/4/05

© 2005 Steve Gribble

4

## Issues

- Replication
  - replication can exist for performance of availability
  - can there be multiple copies of a file in the network?
  - if multiple copies, how are updates handled?
  - what if there's a network partition and clients work on separate copies?
  - at what level is replication visible?

3/4/05

© 2005 Steve Gribble

5

## Issues

- Performance
  - what is the cost of remote operation?
  - what is the cost of file sharing?
  - how does the system scale as the number of clients grows?
  - what are the performance limitations: network, CPU, disks, protocols, data copying?

3/4/05

© 2005 Steve Gribble

6

## Example Systems: NFS

- The Sun Network File System (NFS) has become a common standard for distributed UNIX file access.
- NFS runs over LANS (even over WANs -- slowly).
- Basic idea:
  - allow a remote directory to be “mounted” (spliced) onto a local directory
  - gives access to that remote directory and all its descendants as if they were part of the local hierarchy.
- Ex:
  - I mount /usr/gribble on Node1 onto /students/foo on Node2. Users on Node2 can then access my files as /students/foo. If I had a file /usr/gribble/myfile, users on Node2 see it as /students/foo/myfile.

3/4/05

© 2005 Steve Gribble

7

## NFS

- NFS defines a set of RPC operations for remote file access:
  - searching a directory
  - reading directory entries
  - manipulating links and directories
  - reading/writing files
- Every node may be both a client and server.

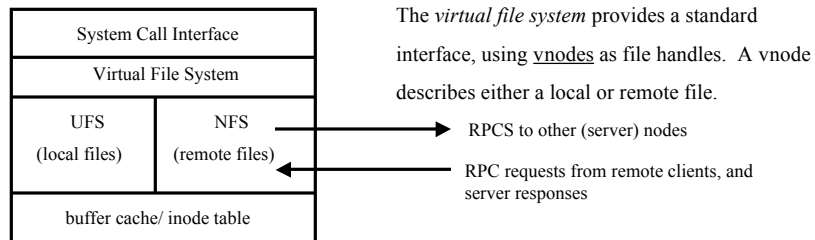
3/4/05

© 2005 Steve Gribble

8

## NFS Implementation

- NFS defines new layers in the Unix file system



- Buffer cache caches remote file blocks and attributes

3/4/05

© 2005 Steve Gribble

9

## NFS

- On an open, the client asks the server whether its cached blocks are up to date.
- Once a file is open, different clients can write it and get inconsistent data.
- Modified data is flushed back to the server every 30 seconds.

3/4/05

© 2005 Steve Gribble

10

## The Andrew File System

- Developed at CMU to support all of its student computing.
- Consists of workstation clients and dedicated file server machines.
- Workstations have local disks, used to cache files being used locally (originally whole files, now 64K file chunks).
- Andrew has a single name space -- your files have the same names everywhere in the world.
- Andrew is good for distant operation because of its local disk caching: after a slow startup, most accesses are to local disk.

3/4/05

© 2005 Steve Gribble

11

## AFS

- Need for scaling led to reduction of client-server message traffic.
- Once a file is cached, all operations are performed locally.
- On close, if the file is modified, it is replaced on the server.
- The client assumes that its cache is up to date, unless it receives a *callback* message from the server saying otherwise. On file open, if the client has received a callback on the file, it must fetch a new copy; otherwise it uses its locally-cached copy.

3/4/05

© 2005 Steve Gribble

12

## Distributed File Systems

- Performance is always an issue
  - there is a tradeoff between performance and the semantics of file operations (e.g., for shared files).
- Caching of file blocks is crucial in any file system, distributed or otherwise
  - As memories get larger, most read requests can be serviced out of file buffer cache (local memory).
  - Maintaining coherency of those caches is a crucial design issue.
- Newer systems are dealing with issues such as disconnected file operation for mobile computers.