

CSci555: Advanced Operating Systems

Lecture 7 - October 15, 1999

File Systems

Dr. Katia Obraczka
University of Southern California
Information Sciences Institute

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



151

Outline

- ❖ Time in distributed systems.
- ❖ File systems.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



152

File Systems

- ❖ Provide set of primitives that abstract users from details of storage access and management.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



153

Distributed File Systems

- ❖ Promote sharing across machine boundaries.
- ❖ Transparent access to files.
- ❖ Make diskless machines viable.
- ❖ Increase disk space availability by avoiding duplication.
- ❖ Balance load among multiple servers.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



154

Sun Network File System 1

- ❖ De facto standard:
 - Mid 80's.
 - Widely adopted in academia and industry.
- ❖ Provides transparent access to remote files.
- ❖ Uses Sun RPC and XDR.
 - NFS protocol defined as set of procedures and corresponding arguments.
 - Synchronous RPC:
 - ♦ Client blocks until it gets results from server.

Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



155

Sun NFS 2

- ❖ Stateless server:
 - Remote procedure calls are self-contained.
 - Servers don't need to keep state about previous requests.
 - ♦ Flush all modified data to disk before returning from RPC call.
 - Robustness.
 - ♦ No state to recover.
 - ♦ Clients retry.

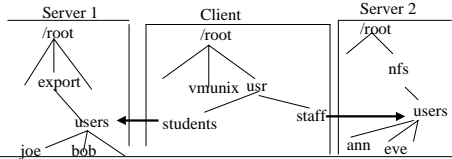
Copyright © 1995-1998 Clifford Neuman, Katia Obraczka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



156

Location Transparency

- ❖ Client's file name space includes remote files.
 - Shared remote files are *exported* by server.
 - They need to be *remote-mounted* by client.



Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Achieving Transparency 1

- ❖ Mount service.
 - Mount remote file systems in the client's local file name space.
 - Mount service process runs on each node to provide RPC interface for mounting and unmounting file systems at client.
 - Runs at system boot time or user login time.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Achieving Transparency 2

- ❖ Automounter.
 - Dynamically mounts file systems.
 - Runs as user-level process on clients (daemon).
 - Resolves references to unmounted pathnames within files managed by the automounter by mounting them on demand.
 - Maintains a table of mount points and the corresponding server(s); sends probes to server(s).
 - Primitive form of replication.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Transparency?

- ❖ Early binding.
 - Mount system call attaches remote file system to local mount point.
 - Client deals with host name once.
 - But, mount needs to happen before remote files become accessible.

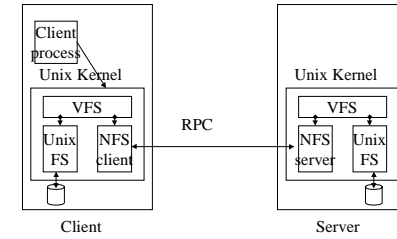
Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Other Functions

- ❖ NFS file and directory operations:
 - read, write, create, delete, getattr, etc.
- ❖ Access control:
 - File and directory access permissions.
- ❖ Path name translation:
 - Lookup for each path component.
 - Caching.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Implementation



Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE

Virtual File System 1

- ❖ VFS added to UNIX kernel.
 - Location-transparent file access.
 - Distinguishes between local and remote access.
- ❖ @ client:
 - Processes file system system calls to determine whether access is local (passes it to UNIX FS) or remote (passes it to NFS client).
- ❖ @ server:
 - NFS server receives request and passes it to local FS through VFS.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



133

VFS 2

- ❖ If local, translates file handle to internal file id's (in UNIX i-nodes).
- ❖ V-node:
 - ♦ If file local, reference to file's i-node.
 - ♦ If file remote, reference to file handle.
- ❖ File handle: uniquely distinguishes file.

File system id	I-node #	I-node generation #
----------------	----------	---------------------

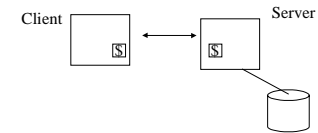
Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



134

NFS Caching

- ❖ File contents and attributes.
- ❖ Client versus server caching.



Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



135

Server Caching

- ❖ Read:
 - Same as UNIX FS.
 - Caching of file pages and attributes.
 - Cache replacement uses LRU.
- ❖ Write:
 - Write through (as opposed to delayed writes of conventional UNIX FS). Why?
 - [Delayed writes: modified written to disk when buffer space needed, sync operation (every 30 sec), file close].

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



136

Client Caching 1

- ❖ Timestamp-based cache invalidation.
- ❖ Read:
 - Cached entries have TS with last-modified time.
 - Blocks assumed to be valid for TTL.
 - ♦ TTL specified at mount time.
 - ♦ Typically 3 sec for files.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



137

Client Caching 2

- ❖ Write:
 - Modified pages marked and flushed to server at file close or sync (every 30 sec).
- ❖ Consistency?
 - Not always guaranteed!
 - E.g., client modifies file; delay for modification to reach servers + 3-sec window for cache validation from clients sharing file.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



138

Cache Validation

- ❖ Validation check performed when:
 - First reference to file after TTL expires.
 - File open or new block fetched from server.
- ❖ Done for all files (even if not being shared).
- ❖ Expensive!
 - Potentially, every 3 sec get file attributes.
 - If needed invalidate all blocks.
 - Fetch fresh copy when file is next accessed.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



152

The Sprite File System 1

- ❖ Main memory caching on both client and server.
- ❖ Write-sharing consistency guarantees.
- ❖ Variable size caches.
 - VM and FS negotiate amount of memory needed.
 - According to caching needs, cache size changes.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



203

Sprite 2

- ❖ Sprite supports concurrent writes by disabling caching of write-shared files.
 - If file shared, server notifies client that has file open for writing to write modified blocks back to server.
 - Server notifies all client that have file open for read that file is no longer cacheable; clients discard all cached blocks, so access goes thru server.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



205

Sprite 3

- ❖ Sprite servers are stateful.
 - Need to keep state about current accesses.
 - Centralized points for cache consistency.
 - ◆ Bottleneck?
 - ◆ Single point of failure?
- ❖ Tradeoff: consistency versus performance.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



202

Andrew

- ❖ Distributed computing environment developed at CMU.
- ❖ Campus wide computing system.
 - Between 5 and 10K workstations.
 - 1991: ~ 800 ws's, 40 servers.

Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



203

Andrew FS

- ❖ Goals:
 - Information sharing.
 - Scalability.
 - ◆ Key strategy: caching of **whole** files at client.
 - ◆ Whole file serving
 - Entire file transferred to client.
 - ◆ Whole file caching
 - Local copy of file cached on client's local disk.
 - Survive client's reboots and server unavailability.

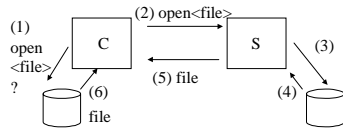
Copyright © 1995-1998 Clifford Neuman, Kato Ohnohara - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



204

Whole File Caching

- Local cache contains several most recently used files.

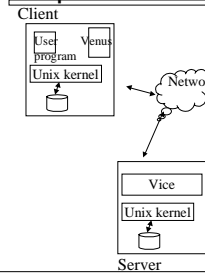


- Subsequent operations on file applied to local copy.
- On close, if file modified, sent back to server.

Implementation 1

- Network of ws's running Unix BSD 4.3 and Mach.
- Implemented as 2 user-level processes:
 - Vice: runs at each Andrew server.
 - Venus: runs at each Andrew client.

Implementation 2

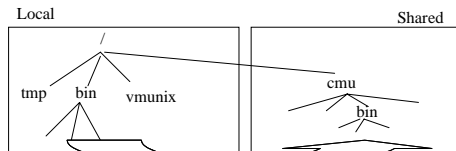


- Modified BSD 4.3 Unix kernel.
 - At client, intercept file system calls (open, close, etc.) and pass them to Venus when referring to shared files.
- File partition on local disk used as cache.
- Venus manages cache.
 - LRU replacement policy.
 - Cache large enough to hold 100's of average-sized files.

File Sharing

- Files are *shared* or *local*.
 - Shared files
 - ♦ Utilities (/bin, /lib): infrequently updated or files accessed by single user (user's home directory).
 - ♦ Stored on servers and cached on clients.
 - ♦ Local copies remain valid for long time.
 - Local files
 - ♦ Temporary files (/tmp) and files used for start-up.
 - ♦ Stored on local machine's disk.

File Name Space



- Regular UNIX directory hierarchy.
- "cmu" subtree contains shared files.
- Local files stored on local machine.
- Shared files: symbolic links to shared files.

AFS Caching 1

- AFS-1 uses timestamp-based cache invalidation.
- AFS-2 and + use *callbacks*.
 - When serving file, Vice server promises to notify Venus client when file is modified.
 - Stateless servers?
 - Callback stored with cached file.
 - ♦ Valid.
 - ♦ Canceled: when client is notified by server that file has been modified.

AFS Caching 2

- ❖ Callbacks implemented using RPC.
- ❖ When accessing file, Venus checks if file exists and if callback valid; if canceled, fetches fresh copy from server.
- ❖ Failure recovery:
 - When restarting after failure, Venus checks each cached file by sending validation request to server.
 - Also periodic checks in case of communication failures.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



211

AFS Caching 3

- ❖ @ file close time, Venus on client modifying file sends update to Vice server.
- ❖ Server updates its own copy and sends callback cancellation to all clients caching file.
- ❖ Consistency?
- ❖ Concurrent updates?

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



212

AFS Replication

- ❖ Read-only replication.
 - Only read-only files allowed to be replicated at several servers.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



213

Coda

- ❖ Evolved from AFS.
- ❖ Goal: constant data availability.
 - Improved replication.
 - ♦ Replication of read-write volumes.
 - Disconnected operation: mobility.
 - ♦ Extension of AFS's whole file caching mechanism.
- ❖ Access to shared file repository (servers) versus relying on local resources when server not available.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



214

Replication in Coda

- ❖ Replication unit: file volume (set of files).
- ❖ |Set of replicas of file volume: volume storage group (VSG).
- ❖ Subset of replicas available to client: AVSG.
 - Different clients, different AVSGs.
 - AVSG membership changes as server availability changes.
 - On write: when file is closed, copies of modified file broadcast to AVSG.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



215

Optimistic Replication

- ❖ Goal is availability!
- ❖ Replicated files are allowed to be modified even in the presence of partitions or during disconnected operation.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



216

Disconnected Operation

- ❖ AVSG = { }.
- ❖ Network/server failures or host on the move.
- ❖ Rely on local cache to serve all needed files.
- ❖ Loading the cache:
 - User intervention: list of files to be cached.
 - Learning usage patterns over time.
- ❖ Upon reconnection, cached copies validated against server's files.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



237

Normal and Disconnected Operation

- ❖ During normal operation:
 - Coda behaves like AFS.
 - Cache miss transparent to user; only performance penalty.
 - Load balancing across replicas.
 - Cost: replica consistency + cache consistency.
- ❖ Disconnected operation:
 - No replicas are accessible; cache miss prevents further progress; need to load cache before disconnection.

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



238

Replication and Caching

- ❖ Coda integrates server replication and client caching.
 - On cache hit and valid data: Venus does not need to contact server.
 - On cache miss: Venus gets data from an AVSG server, i.e., the preferred server (PS).
 - ♦ PS chosen at random or based on proximity, load.
 - Venus also contacts other AVSG servers and collect their versions; if conflict, abort operation; **if replicas stale, update them off-line.**

Copyright © 1995-1998 Clifford Neuman, Kati Obracka - UNIVERSITY OF SOUTHERN CALIFORNIA - INFORMATION SCIENCES INSTITUTE



239