

Agenda

- **Last time (Tues Mar 13)**
 - Dis File systems (chpt 8)
 - Recover from Spring Break
 - Study for OS midterm
- **This time**
 - Dis File systems: NFS, CIFS/SAMBA, and AFS (chpt 8)
 - Name Services (chpt 9)
- **Thursday (5pm, D222)/Friday(3pm D223) (Make-up)**
 - Finish Name services (chpt 9)
- **Tues Mar 20: Midterm (Closed books, closed notes)**
- **Thurs Mar 22:**
 - Web Services
 - Assignment #3 out
- **I am not in town on Mon and Tues (no office hours on Mon)**

CS451: Distributed Systems (Spring 2007)

Schedule

Sun	Tues	Thurs	Fri
	DFS (8) 13	DFS (8) / Naming (9) Naming (9) 15	Naming (9) Uva 12:15pm 16
	Midterm – closed books, closed notes 20	Web services (19) PA#3 out 22	
	P2P (10) 27	P2P, #3 due, #4 out Time/global(11) 29	Time/global (11) 30
	Time/global PA#5 Out 3	Coordination/agreement 5	Coordination/agreement (12)
	Coordination/agreement PA#4 due 10	Transactions (13) PA#5 proposal due 12	
	Transactions (13) Dis transactions(14)	Replication (15) 19	
	Student presentations PA#5 due 24	Student presentations 26	

CS451: Distributed Systems (Spring 2007)

Before we start

- **Prof. Weaver has an NSF REU program**
 - Computer Applications for Medicine
 - June 11--August 8.
 - We provide housing, meals, and a \$3200 stipend
 - The target audience is (current) 3rd years, although (current) 2nd years and 4th years will also be considered.
 - <http://www.cs.virginia.edu/~acw/REU>
- **I'm looking for good people too**

CS451: Distributed Systems (Spring 2007)

Before we start: midterm

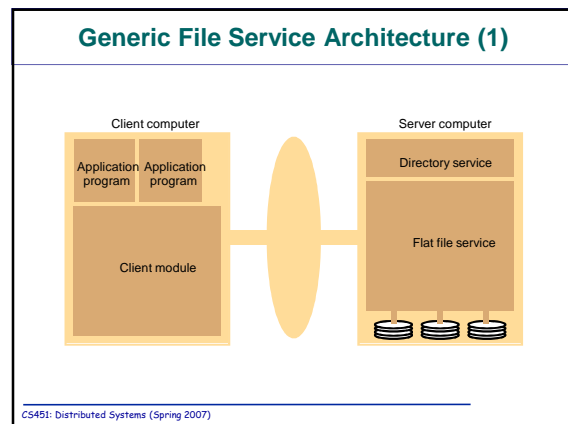
• <http://www.cs.virginia.edu/~humphrey/cs451/Assignments/MidtermPrep.html>

CS451: Distributed Systems (Spring 2007)

Recap

- **A distributed file service:**
 - Enables programs to store and access remote files exactly as they do local ones.
 - Reduces the need for local disk storage and eases management and archiving.
- **Files, directories, metadata**
- **Main transparencies: Access and Location**
- **Issues**
 1. What model of read/write? (@ client or @ server?)
 2. Must/should all clients have the same view?
 3. How to lookup /a/b/c?
 4. Caching: when, when?

CS451: Distributed Systems (Spring 2007)



Generic File Service Architecture (2)

- **Stateless, by definition!**
- **Flat file service (FFS):**
 - Operations on the contents of files.
 - Uses UFIDs – unique file identifiers.
- **Directory service:**
 - Mapping between text names and UFIDs.
 - Generation of directories, addition of file names to directories, ...
- **Client module:**
 - Provides API for flat file services and directory service to user-level programs in client computers.
 - (optionally) Caching of recently used file blocks.

CS451: Distributed Systems (Spring 2007)

Generic flat file service operations (RPC interface to client modules)

<i>Read(FileId, i, n) -> Data</i> — throws <i>BadPosition</i>	If $1 \leq i \leq \text{Length}(\text{File})$: Reads a sequence of up to n items from a file starting at item i and returns it in <i>Data</i> .
<i>Write(FileId, i, Data)</i> — throws <i>BadPosition</i>	If $1 \leq i \leq \text{Length}(\text{File})+1$: Writes a sequence of <i>Data</i> to a file, starting at item i , extending the file if necessary.
<i>Create() -> FileId</i>	Creates a new file of length 0 and delivers a UFID for it.
<i>Delete(FileId)</i>	Removes the file from the file store.
<i>GetAttributes(FileId) -> Attr</i>	Returns the file attributes for the file.
<i>SetAttributes(FileId, Attr)</i>	Sets the file attributes.

CS451: Distributed Systems (Spring 2007)

Sidebar: Generic flat file service (FFS) vs. UNIX

- **UNIX has open and close operations and a read/write pointer.**
 - FFS has no open or close operations and no notion of a read/write pointer.
- **FFS characteristics:**
 - Repeatable operations, *idempotent* operations, allowing at-least-once RPC semantics.
 - Stateless servers, possible to resume operation after failure without restoration of state.

CS451: Distributed Systems (Spring 2007)

Access Control

- **UNIX: access control performed at invocation of *open***
- **Generic flat file system:**
 - Cannot check at client, because (malicious) client could always say it lied
 - **Can't we just have a list of trusted clients?**
- **Generic flat file system: stateless, so two options on the RPC invocation**
 - *Filename*-to-UFID translation: check access and encode results a *la* capability
 - Submit credentials on every client request (more common)
- **What about sniffing?**

CS451: Distributed Systems (Spring 2007)

Generic directory service operations

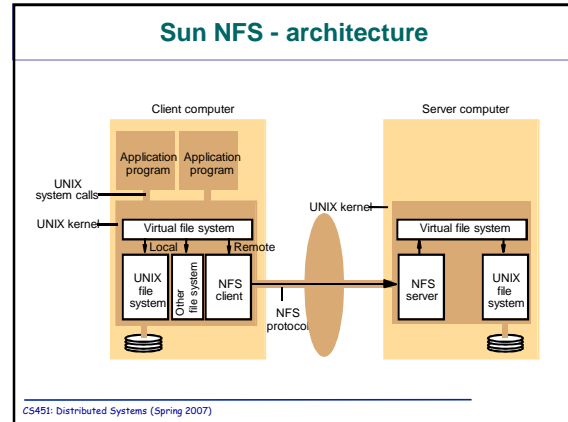
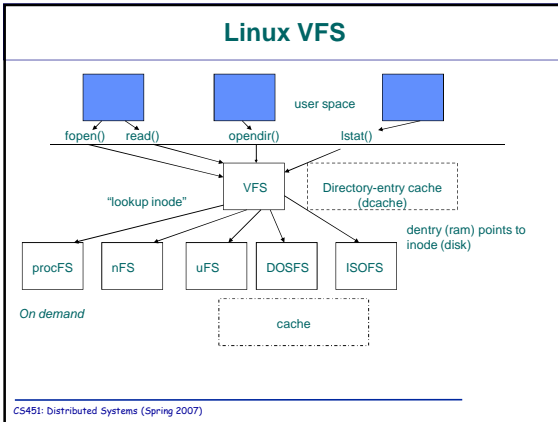
<i>Lookup(Dir, Name) -> FileId</i> — throws <i>NotFound</i>	Locates the text name in the directory and returns the relevant UFID. If <i>Name</i> is not in the directory, throws an exception.
<i>AddName(Dir, Name, File)</i> — throws <i>NameDuplicate</i>	If <i>Name</i> is not in the directory, adds (<i>Name, File</i>) to the directory and updates the file's attribute record. If <i>Name</i> is already in the directory: throws an exception.
<i>UnName(Dir, Name)</i> — throws <i>NotFound</i>	If <i>Name</i> is in the directory: the entry containing <i>Name</i> is removed from the directory. If <i>Name</i> is not in the directory: throws an exception.
<i>GetNames(Dir, Pattern) -> NameSeq</i>	Returns all the text names in the directory that match the regular expression <i>Pattern</i> .

CS451: Distributed Systems (Spring 2007)

Sun NFS - basics

- **NFS: Network File System (1985)**
- **Placed in public domain**
 - (1989) RFC 1094: NFS v2
 - (1995) RFC 1813: NFS v3
 - (2000) RFC 3010: NFS v4
- **NFS Versions 2, 3, and 4 are supported on 2.6 and later kernels.**
- **Nodes are both clients and servers in NFS**
- **The NFS server module resides in the kernel on each computer that acts as an NFS server.**
- **Requests referring to files in a remote file system are translated by the client module to NFS protocol operations.**
- **Uses SUN RPC (either UDP or TCP)**

CS451: Distributed Systems (Spring 2007)



Sun NFS – NFS client module

- Supplies an interface for application programs.
- Emulates the semantics of the standard UNIX file system primitives.
- Is integrated with the UNIX kernel
 - No need to recompile user programs (because access is through the system call interface)
 - Single client module serves all user-level processes, with shared cache (same cache as local I/O system)
- Transfers blocks of files to and from the server.
- Access Control: conventional UNIX 16-bit user ID and group ID
 - Problems?

CS451: Distributed Systems (Spring 2007)

Some NFS server module operations

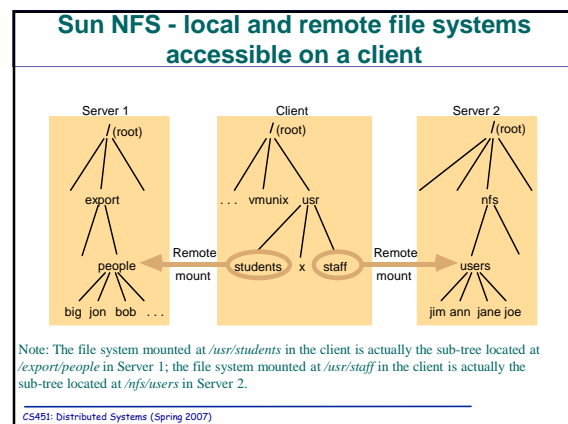
<code>lookup(dirfh, name) -> fh, attr</code>	Returns file handle and attributes for the file <i>name</i> in the directory <i>dirfh</i> .
<code>create(dirfh, name, attr) -> newfh, attr</code>	Creates a new file name in directory <i>dirfh</i> with attributes <i>attr</i> and returns the new file handle and attributes.
<code>remove(dirfh, name) status</code>	Removes file name from directory <i>dirfh</i> .
<code>getattr(fh) -> attr</code>	Returns file attributes of file <i>fh</i> . (Similar to the UNIX <i>stat</i> system call.)
<code>setattr(fh, attr) -> attr</code>	Sets the attributes (mode, user id, group id, size, access time and modify time of a file). Setting the size to 0 truncates the file.
<code>read(fh, offset, count) -> attr, data</code>	Returns up to <i>count</i> bytes of data from a file starting at <i>offset</i> . Also returns the latest attributes of the file.
<code>write(fh, offset, count, data) -> attr</code>	Writes <i>count</i> bytes of data to a file starting at <i>offset</i> . Returns the attributes of the file after the write has taken place.
<code>rename(dirfh, name, todirfh, toname) -> status</code>	Changes the name of file <i>name</i> in directory <i>dirfh</i> to <i>toname</i> in directory <i>todirfh</i> .
<code>link(newdirfh, newname, dirfh, name) -> status</code>	Creates an entry <i>newname</i> in the directory <i>newdirfh</i> which refers to file <i>name</i> in the directory <i>dirfh</i> .

CS451: Distributed Systems (Spring 2007)

Sun NFS - mount service

- A mount service process runs on each NFS server computer.
- A file (*/etc/exports*) contains the names of local filesystems that are available for remote mounting (by which machine names)
- Clients request mounting of a remote filesystem by specifying:
 - Remote host name.
 - Pathname of a directory in the remote filesystem.
 - The local name with which it is to be mounted.
- Server returns file handle of specified directory.
- “hard-mount”: block until operation completes

CS451: Distributed Systems (Spring 2007)



Sun NFS Server operations

- **Pathname translation**
 - Clients perform parsing of multi-part names
 - Iteratively interacts with file servers to determine the final location
 - Client-side caching helps the efficiency
- **Automounter**
 - *Autofs*

CS451: Distributed Systems (Spring 2007)

Sun NFS – server caching

- **File pages are held in a main memory buffer cache until the buffer space is required for other pages.**
- **Read operations raise no consistency problem.**
- **Write operations offer two options:**
 - **Write-through:** Data in write operations received from clients is stored in the memory cache at the server and written to disk before a reply is sent to the client.
 - **Commit required:** Data in write operations is stored only in the memory cache. It will be written to disk when a commit operation is received for the relevant file. (standard NFS uses this)

CS451: Distributed Systems (Spring 2007)

Sun NFS – client caching

- **Possibly different versions of files or portions of files in different client nodes.**
- **NFS client module caches results of *read*, *write*, *getattr*, *lookup*, and *readdir* operations.**
- **Timestamp-based method used to validate cached blocks before they are used.**
 - Typical freshness: 3-30 for files, 30-60 directories
- **Validation may involve *getattr* calls to server.**
- **When a cached page is modified, it is marked as dirty and scheduled to be flushed to the server asynchronously.**

CS451: Distributed Systems (Spring 2007)

Finishing up NFS

- **Looking up */a/b/c* ... hmmm... a couple of different ways to do it, right?**
 - Clients performs parsing of multi-part names
 - Iteratively interacts with file servers to determine the final location
 - Client-side caching helps the efficiency
- **Benefits of kernel-space client and server implementations**
 - Broader re-use of cached material ..
 - No need to recompile user programs
- **Security issues**
- **Write operations offer two options:**
 - **Write-through:** Data in write operations received from clients is stored in the memory cache at the server and written to disk before a reply is sent to the client.
 - **Commit required:** Data in write operations is stored only in the memory cache. It will be written to disk when a commit operation is received for the relevant file. (standard NFS uses this)

CS451: Distributed Systems (Spring 2007)

NFS performance

- `[mah2h@grad13 pa3]$ ls -al xalan-j-current-bin-2jars.tar.gz`
-rw-rw-r-- 1 mah2h mah2h 10556153 Feb 29 2004 xalan-j-current-bin-2jars.tar.gz
- **NFS**
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m2.125s
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m1.288s
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m1.202s
- **Non-NFS**
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m0.470s
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m0.095s
 - `[mah2h@grad13 pa3]$ time cp xalan-j-current-bin-2jars.tar.gz bar`
• real 0m0.094s
- **Hmmm... what does this show? (1.202 / 0.094 = 12.78)**

CS451: Distributed Systems (Spring 2007)

NFS performance (take two)

- `[mah2h@grad13 cs414.cs-t11]$ ls -al root_fs.rh-7.2-full.pristine.20020312`
-rwxrwx--- 1 mah2h mah2h 711983104 Sep 18 2004 root_fs.rh-7.2-full.pristine.20020312
- **NFS**
 - `[mah2h@grad13 cs414.cs-t11]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 1m57.531s
 - `[mah2h@grad13 cs414.cs-t11]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 1m56.955s
 - `[mah2h@grad13 cs414.cs-t11]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 1m56.084s
- **Non-NFS**
 - `[mah2h@grad13 pa3]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 0m58.398s
 - `[mah2h@grad13 pa3]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 1m3.189s
 - `[mah2h@grad13 pa3]$ time cp root_fs.rh-7.2-full.pristine.20020312 foo`
• real 1m4.200s
- **Hmmm... what does this show? (116.084 / 64.200 = 1.808)**

CS451: Distributed Systems (Spring 2007)

NFS performance (take three)

```
•[mah2h@grad13 ~]$ ls -al test_file
-rw-rw-r-- 1 mah2h mah2h 57 Mar 24 07:10 test_file
•NFS
•[mah2h@grad13 ~]$ time cp test_file foo
•real 0m0.047s
•[mah2h@grad13 ~]$ time cp test_file foo
•real 0m0.009s
•[mah2h@grad13 ~]$ time cp test_file foo
•real 0m0.009s
•Non-NFS
•[mah2h@grad13 pa3]$ time cp test_file foo
•real 0m0.636s
•[mah2h@grad13 pa3]$ time cp test_file foo
•real 0m0.003s
•[mah2h@grad13 pa3]$ time cp test_file foo
•real 0m0.003s
•Hmmm... what does this show? (0.009 / 0.003 = 3)
```

CS451: Distributed Systems (Spring 2007)

NFS performance summary

	NFS-first	NFS-others	Non-NFS-first	Non-NFS-others	NFS-others / non-NFS others
57	0m0.047s	0m0.009s	0m0.636s	0m0.003s	3
10M	0m2.125s	0m1.202s	0m0.470s	0m0.094s	12.78
0.7G	1m57.531s	1m56.084s	0m58.398s	1m4.200s	1.808

CS451: Distributed Systems (Spring 2007)

NFS: Evaluation

- Access transparency: API is identical; no mods to existing progs
- Location transparency: uniform name space *can* be established
- Mobility transparency: Filesystems can be moved, and clients must be updated (so mobility transparency is not fully achieved)
- Scalability: "hot spots" can be moved easily
- Fault tolerance: NFS is stateless and idempotent (but note that there is generally only one source of info)
- NFSv4: attempts to scale to the Internet-scale and deal with Security issues (although not ready for prime-time)

CS451: Distributed Systems (Spring 2007)