

CS 5523 Lecture 23: Distributed File Systems

- *Laboratory 2 Discussion*
- *Questions on Laboratories 3 and 4*
- *Requirements*
- *Storage system support infrastructure*
- *How does Unix do it?*
- *Flat file services*
- *Directory structure*
- *Introduction to NFS (Network File System)*

Distributed file system requirements:

- *Transparency*
 - *access transparency*
 - *location transparency*
 - *mobility transparency*
 - *performance transparency*
 - *scaling transparency*
- *Concurrency control possibility with support for transactions*
- *Replication*
- *Fault tolerance*
- *Heterogeneity*
- *Security*
- *Consistency*

Figure 8.1
Storage systems and their properties

	<i>Sharing</i>	<i>Persis- tence</i>	<i>Distributed cache/replicas</i>	<i>Consistency maintenance</i>	<i>Example</i>
Main memory	×	×	×	1	RAM
File system	×	✓	×	1	UNIX file system
Distributed file system	✓	✓	✓	✓	Sun NFS
Web	✓	✓	✓	×	Web server
Distributed shared memory	✓	×	✓	✓	Ivy (Ch. 16)
Remote objects (RMI/ORB)	✓	×	×	1	CORBA
Persistent object store	✓	✓	×	1	CORBA Persistent Object Service
Persistent distributed object store	✓	✓	✓	✓	PerDiS, Khazana

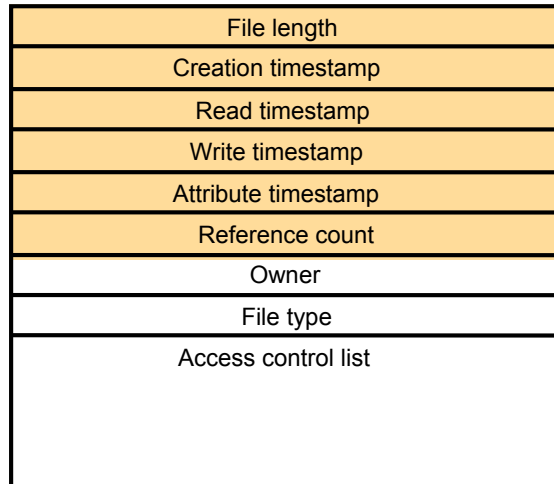
Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.2
File system modules

Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	disk I/O and buffering

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.3
File attribute record structure



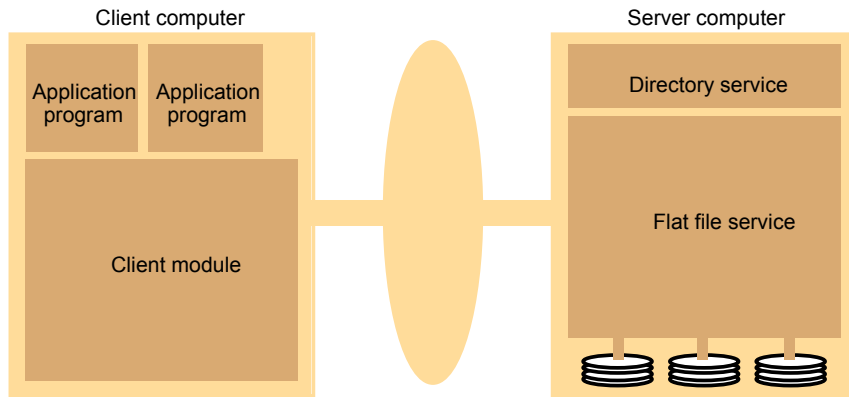
Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.4
UNIX file system operations

<i>filedes</i> = <i>open</i> (<i>name</i> , <i>mode</i>)	Opens an existing file with the given <i>name</i> .
<i>filedes</i> = <i>creat</i> (<i>name</i> , <i>mode</i>)	Creates a new file with the given <i>name</i> .
	Both operations deliver a file descriptor referencing the open file. The <i>mode</i> is <i>read</i> , <i>write</i> or both.
<i>status</i> = <i>close</i> (<i>filedes</i>)	Closes the open file <i>filedes</i> .
<i>count</i> = <i>read</i> (<i>filedes</i> , <i>buffer</i> , <i>n</i>)	Transfers <i>n</i> bytes from the file referenced by <i>filedes</i> to <i>buffer</i> .
<i>count</i> = <i>write</i> (<i>filedes</i> , <i>buffer</i> , <i>n</i>)	Transfers <i>n</i> bytes to the file referenced by <i>filedes</i> from <i>buffer</i> .
	Both operations deliver the number of bytes actually transferred and advance the read-write pointer.
<i>pos</i> = <i>lseek</i> (<i>filedes</i> , <i>offset</i> , <i>whence</i>)	Moves the read-write pointer to offset (relative or absolute, depending on <i>whence</i>).
<i>status</i> = <i>unlink</i> (<i>name</i>)	Removes the file <i>name</i> from the directory structure. If the file has no other names, it is deleted.
<i>status</i> = <i>link</i> (<i>name1</i> , <i>name2</i>)	Adds a new name (<i>name2</i>) for a file (<i>name1</i>).
<i>status</i> = <i>stat</i> (<i>name</i> , <i>buffer</i>)	Gets the file attributes for file <i>name</i> into <i>buffer</i> .

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.5
File service architecture



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

File service architecture:

■ Flat file service:

- provides operations on file content
- uses a UFID (unique file ID)

■ Directory service

- maps text file names to UFIDs
- may be hierarchical (text names can be directories)

■ Client module

- runs on each client
- emulates local operations
- may use caching and other strategies for improving performance

Figure 8.6
Flat file service operations

<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 (only those attributes that are not shaded in).

Proposed flat file service versus Unix:

- *No open or close*
- *Most operations (except create) are idempotent (UNIX read and write are not)*
- *Access control is different:*
 - *In UNIX access rights are checked at the open. Process uses a file handle.*
 - *In the proposed service access rights are checked by the server:*
 - *Either when a name is converted to UFID access rights are encoded in a capability. The capability is then presented with each subsequent operations*
 - *Or, the user credentials are submitted with each request*

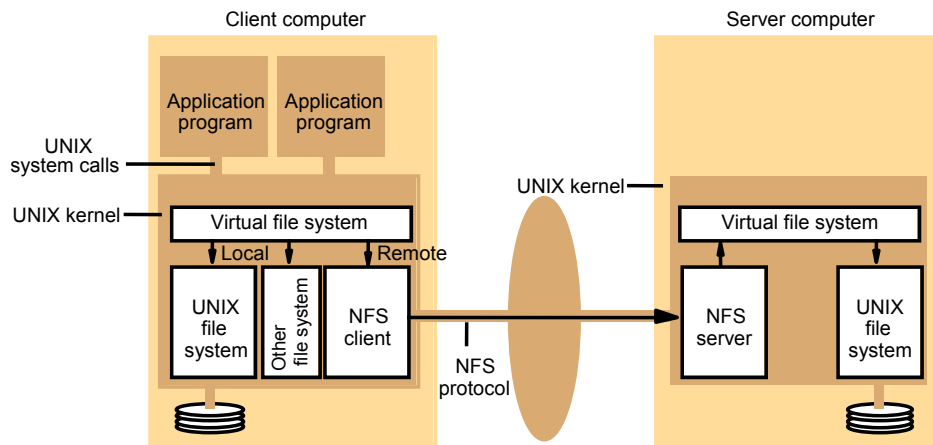
Figure 8.7 Directory service operations

<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> .

Proposed directory system versus Unix:

- *Text names are translated to UIDs*
- *UNIX file names are mapped to inodes in a specific device partition*
- *Hierarchical organization can be supported in both (need a type designation)*
- *File groups used to allocate blocks of files in a distributed file system to particular servers (analogous to a filesystem in UNIX)*

Figure 8.8
NFS architecture



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.9
NFS server operations (simplified) – 1

<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 <i>name</i> 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 <i>name</i> 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>to dirfh</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> .

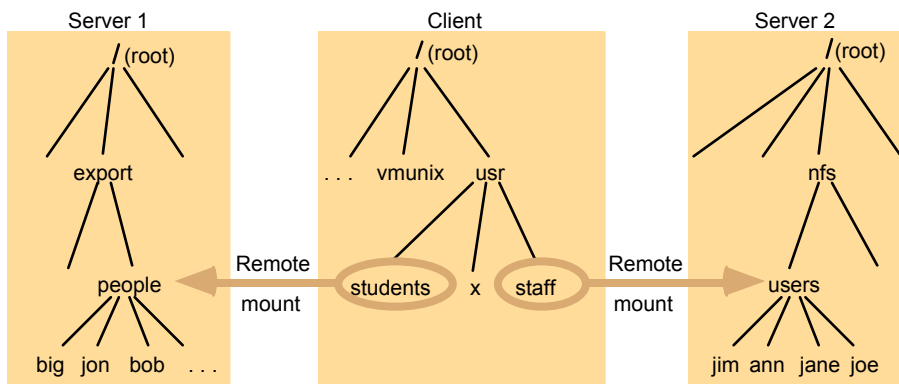
Continues on next slide ...

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3
© Addison-Wesley Publishers 2000

Figure 8.9
NFS server operations (simplified) – 2

<i>symlink(newdirfh, newname, string)</i> -> status	Creates an entry <i>newname</i> in the directory <i>newdirfh</i> of type symbolic link with the value <i>string</i> . The server does not interpret the <i>string</i> but makes a symbolic link file to hold it.
<i>readlink(fh)</i> -> string	Returns the string that is associated with the symbolic link file identified by <i>fh</i> .
<i>mkdir(dirfh, name, attr)</i> -> <i>newfh, attr</i>	Creates a new directory <i>name</i> with attributes <i>attr</i> and returns the new file handle and attributes.
<i>rmdir(dirfh, name)</i> -> status	Removes the empty directory <i>name</i> from the parent directory <i>dirfh</i> . Fails if the directory is not empty.
<i>readdir(dirfh, cookie, count)</i> -> <i>entries</i>	Returns up to <i>count</i> bytes of directory entries from the directory <i>dirfh</i> . Each entry contains a file name, a file handle, and an opaque pointer to the next directory entry, called a <i>cookie</i> . The <i>cookie</i> is used in subsequent <i>readdir</i> calls to start reading from the following entry. If the value of <i>cookie</i> is 0, reads from the first entry in the directory.
<i>statfs(fh)</i> -> <i>fsstats</i>	Returns file system information (such as block size, number of free blocks and so on) for the file system containing a file <i>fh</i> .

Figure 8.10
Local and remote file systems accessible on an NFS client



Note: The file system mounted at */usr/students* in the client is actually the sub-tree located at */export/people* in Server 1; the file system mounted at */usr/staff* in the client is actually the sub-tree located at */nfs/users* in Server 2.

For next time:

■ *Read CDK 8.3-8.4*