# CS 550:
# Advanced Operating Systems

## Networked File Systems

**Ioan Raicu**
Computer Science Department
Illinois Institute of Technology

CS 550
Advanced Operating Systems
March 29th, 2011

# File System Basics

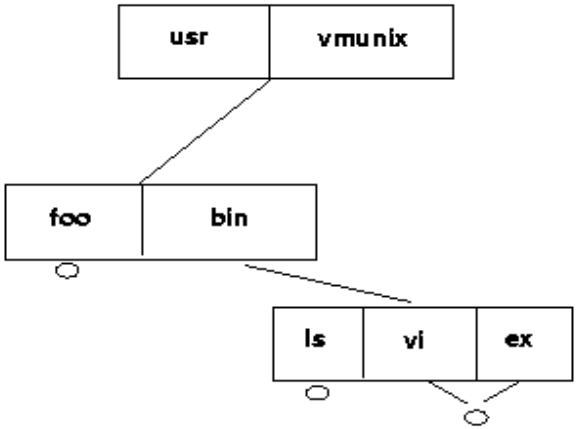
- *File*:
  - Named collection of logically related data
  - Unix file: an uninterpreted sequence of bytes
- *File system:*
  - Provides a logical view of data and storage functions
  - User-friendly interface
  - Provides facility to create, modify, organize, and delete files
  - Provides sharing among users in a controlled manner
  - Provides protection

# File Types and Attributes

- *File types:*
  - Regular files
  - Directories
  - Character special files: used for serial I/O
  - Block special files: used to model disks [buffered I/O]
- *File attributes*: varies from OS to OS
  - Name, type, location, size, protection info, password, owner, creator, time and date of creation, last modification, access
- *File operations*:
  - Create, delete, open, close, read, write, append, get/set attributes
- *File access:*
  - Sequential, random

# Directories

- Tree structure organization most common




- Access  to a file specified by *absolute file name*
- User can assign a directory as the *current working directory*
  - Access to files can be specified by *relative name* relative to the current directory
- Possible organizations: linear list of files, hash table

# File System

- Components: directory, authorization, file service and system service
  - Authorization service: between file and directory services
  - Directory service:  used to keep track of the location of all resources in the system
  - File service provides a transparent way of accessing any file in the system in the same way
  - System service: file system's interface to hardware

# Distributed File Systems

- Characteristics of a DFS
  - Access transparency
  - Location transparency
  - Concurrency transparency
  - Failure transparency
  - Performance transparency
  - Replication transparency
  - Migration transparency
  - Scalability

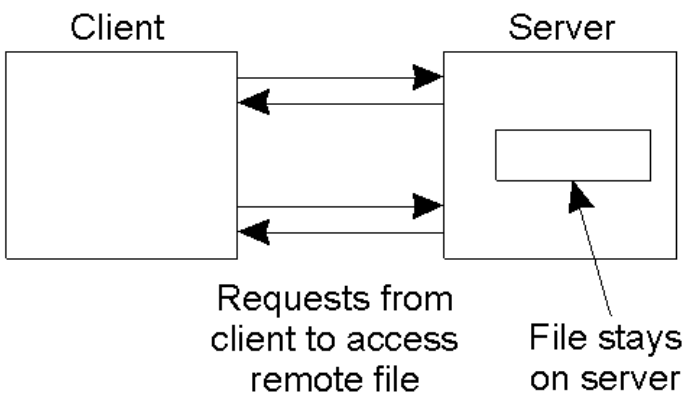
# Distributed File Systems

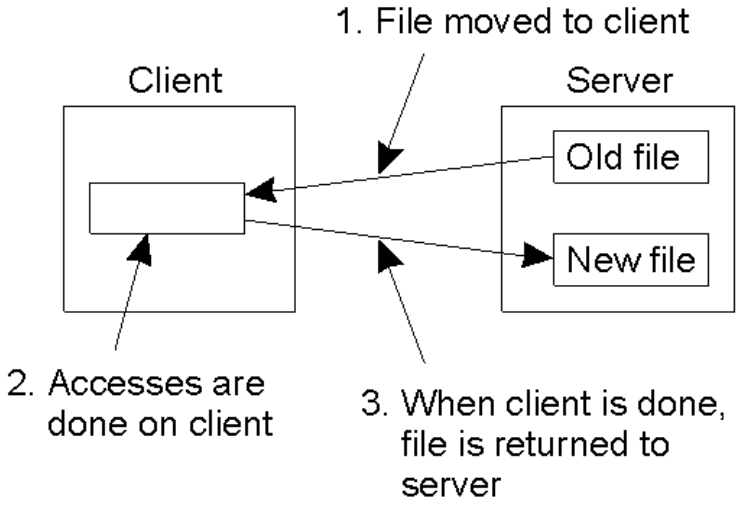
- *File service:*
  - Specification of what the file system offers to client
    - Actions
    - Client primitives
    - Parameters, application programming interface (API)
  - Does not include how service is implemented
- *File server:*
  - Process that runs on a machine and implements file service
  - Can have several servers on one machine (UNIX, DOS,…)
  - ideally, clients do not know the distributed nature

# Architectures

- How are DFS generally organized?
  - Client-server architectures
    - Example: Sun Microsystem's NFS
    - File servers with a standardized view of its local file system; clients can access these files
  - Cluster-based distributed file systems
    - Example: file striping, partitioning the whole file system, GFS
  - Symmetric architectures
    - Fully symmetric organization based on p2p
    - Example: Ivy

# Client-Server Architectures
# Shared File Systems

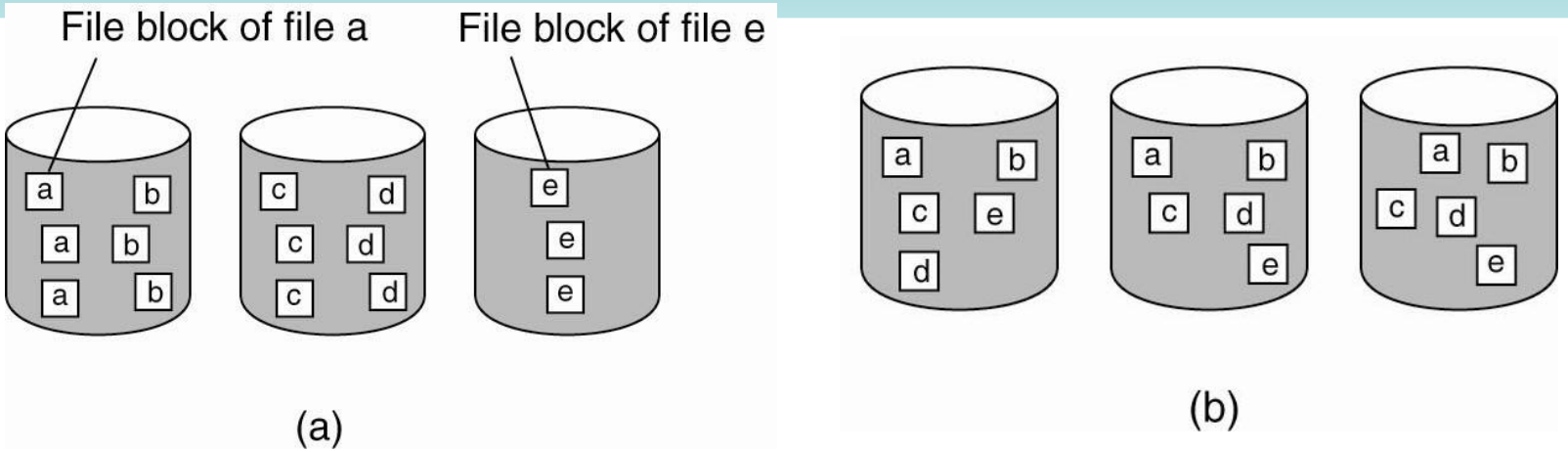



- **Remote access model**
  - Work done at the server
- Stateful server (e.g., databases)
- Pros & cons?

- **Upload/download model**
  - Work done at the client
- Stateless server
- Pros & cons?

# Cluster-based DFSs
# Distributed/Parallel File Systems




- Figure (b): When server clusters are used for parallel applications
  - File-striping techniques, a single file is distributed across multiple servers
- Figure (a): For general-purpose applications, when file striping may not be effective
  - Partition the file system as a whole and simply store different files on different servers
- External material: Google File System

# Questions