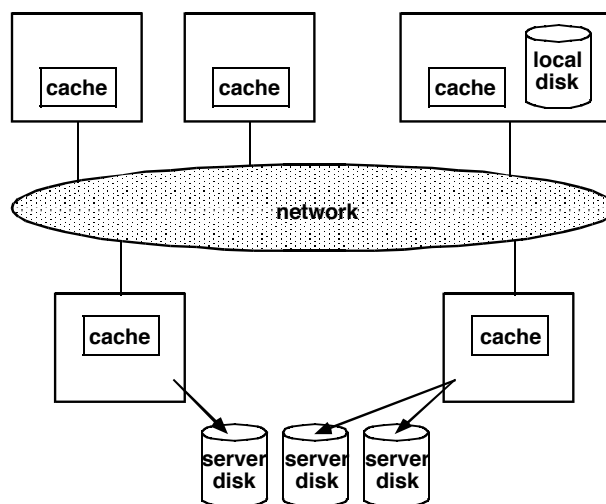


## Distributed File Systems

- **Distributed file system** — a distributed implementation of a file system
  - **File service** — specification of the file system interface as seen by the clients
  - **File server** — a process running on some machine which helps implement the file service by supplying files
- **Goals of a distributed file system**
  - **Network transparency**
    - Provide same operations for accessing remote and local files
    - Ideally, clients should not have to know the location of files to access them
  - **Availability / robustness** — file service should be maintained even in the presence of partial system failures
  - **Performance** — should overcome bottlenecks of a centralized file system

## Distributed File Systems (cont.)



- In principle, files in a distributed file system can be stored at any machine
  - However, a typical distributed environment has a few dedicated machines called *file servers* that store all the files

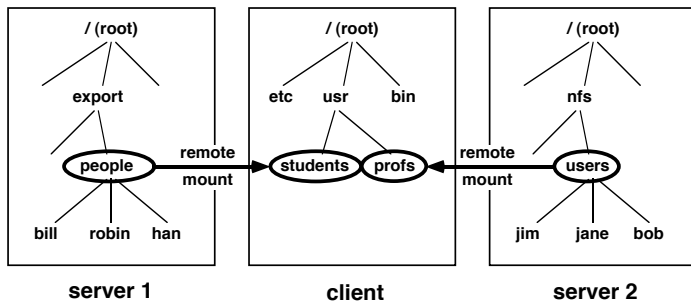
## Distributed Naming Structures

- **Two related concepts in naming:**
  - **Location transparency** — the name of the file does not reveal the physical storage location (disk)
    - May expose name of machine
    - True for many naming schemes
  - **Location independence** — the name of the file need not change if the file's storage location changes
    - False for most naming schemes
- **Absolute names**
  - Names of form: *machine : pathname*
  - Used by:
    - Old UNIX distributed file systems
    - Current web browsers (e.g., Netscape)
  - ✓ User can use same tools and file operations for local and remote access
  - ✗ Not location transparent or independent

## Distributed Naming Structures (cont.)

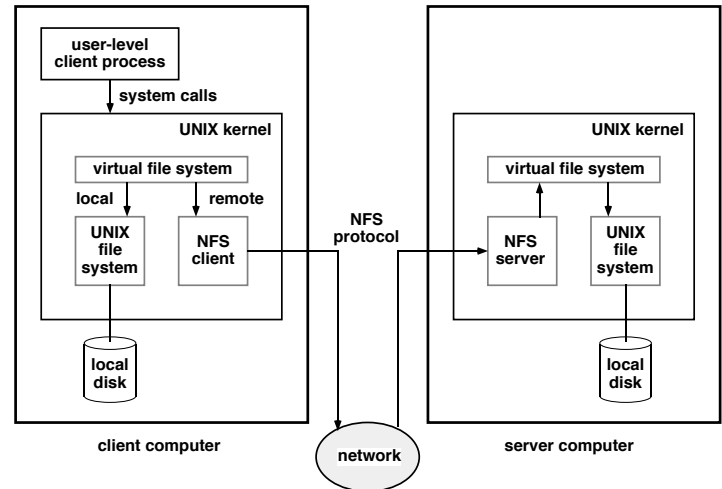
- **Mount remote directories onto local directories (possibly on demand)**
  - **Client-maintained mount information:**
    - Used by UNIX and NFS — Sun's Network File System
    - Client maintains:
      - A set of local names for remote locations
      - A *mount table* (*/etc/fstab*) that specifies a:
        - » < remote machine name : pathname >
        - » and < local pathname >
    - At boot time, the local name is bound to the remote name
      - Afterwards, users refer to local pathname as if it were local, and the distributed OS takes care of the mapping
      - Location transparent and independent after the mount operation, but not before
  - **Server-maintained mount information:**
    - If files are moved to a different server, mount information need only be updated at servers

## Mounting Remote File Systems



- NFS supports mounting of remote file systems by client machines
  - Name space seen by each client may be different
  - Same file on server may have different path names on different clients
  - NFS does not enforce a single network-wide name space, but a uniform name space (and location transparency) can be established if desired

## NFS Software Architecture



- Virtual file system:
  - Separates generic file-system operations from their implementation (can have different types of local file systems)
  - Based on a file descriptor called a vnode that is unique networkwide (UNIX inodes are only unique on a single file system)

## NFS Protocol

- NFS protocol provides a set of RPCs for remote file operations
  - Looking up a file within a directory
  - Manipulating links and directories
  - Creating, renaming, and removing files
  - Getting and setting file attributes
  - Reading and writing files
- NFS is stateless
  - Servers do not maintain information about their clients from one access to the next
    - There are no open-file tables on the server
  - There are no open and close operations
    - Each request must provide a unique file identifier, and an offset within the file
  - Easy to recover from a crash, but file operations must be idempotent

## NFS Protocol (cont.)

- Because NFS is stateless, all modified data must be written to the server's disk before results are returned to the client
  - Server crash and recovery should be invisible to client —data should be intact
  - ✗ Lose benefits of caching
    - Solution — RAM disks with battery backup (un-interruptable power supply), written to disk periodically
- A single NFS write is guaranteed to be atomic, and not intermixed with other writes to the same file
  - However, NFS does not provide concurrency control
    - A write system call may be decomposed into several NFS writes, which may be interleaved
    - Since NFS is stateless, this is not considered to be an NFS problem