Announcements

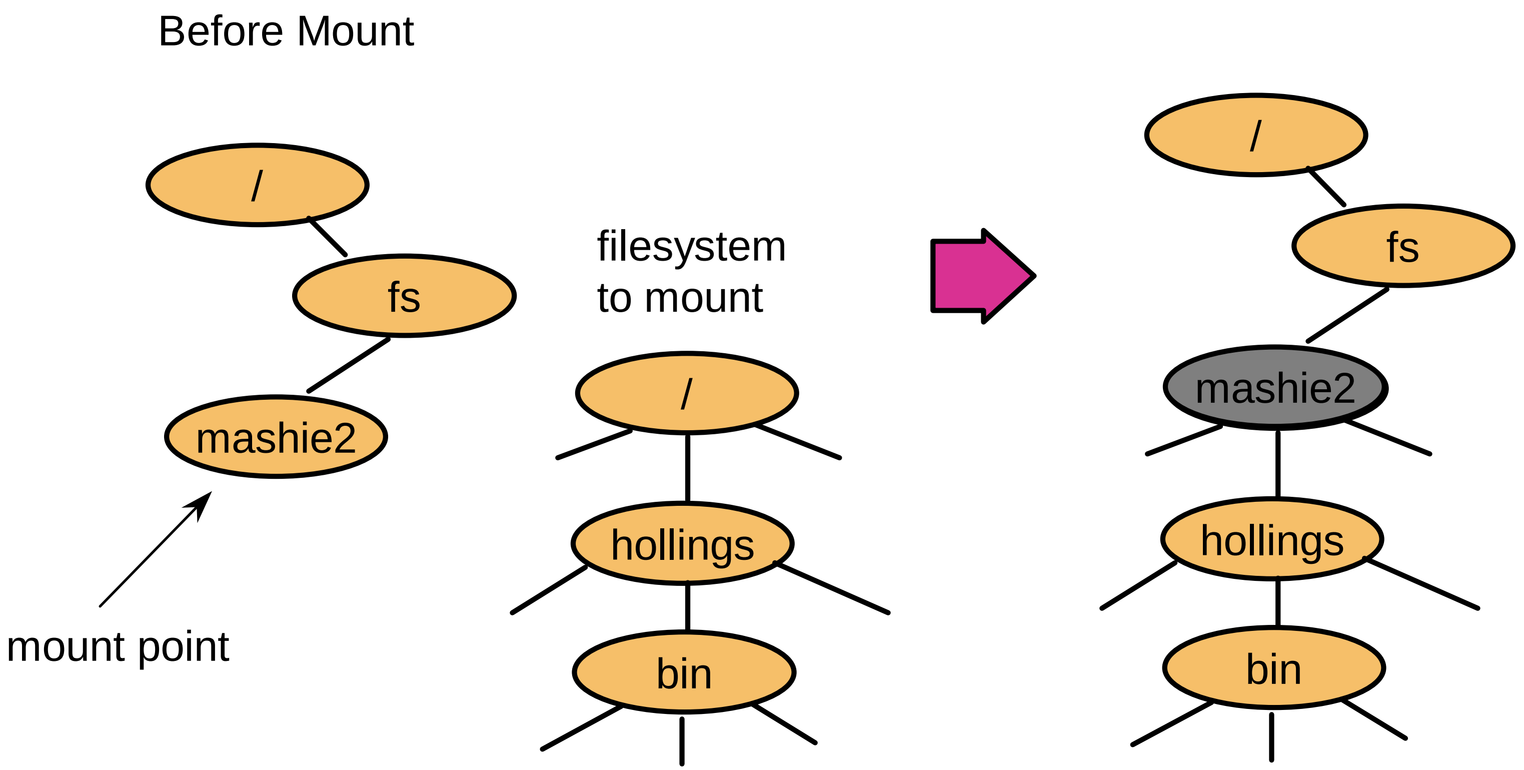
- Final is May 20, 1996 1:30-3:30 PM
- Reading: none
- Course evaluations were distributed
- Dr. Argawala is looking for students to work on realtime systems, if interested please speak to him

File Server State

- Does the fileserver maintain information between requests?
- Stateless
  - example: NFS
  - each request contains a request to read/write a specific part of a file
  - requests must be *itempotent*
    - the same request can be applied several times
  - makes recovery of failed clients/servers easier
- Stateful
  - example: AFS
  - servers maintain connections for clients
  - improves performance
  - required for server based cache management

Mounting a filesystem

- Mount attaches a filesystem to a directory
  - can be used for local or remote (NFS) filesystems



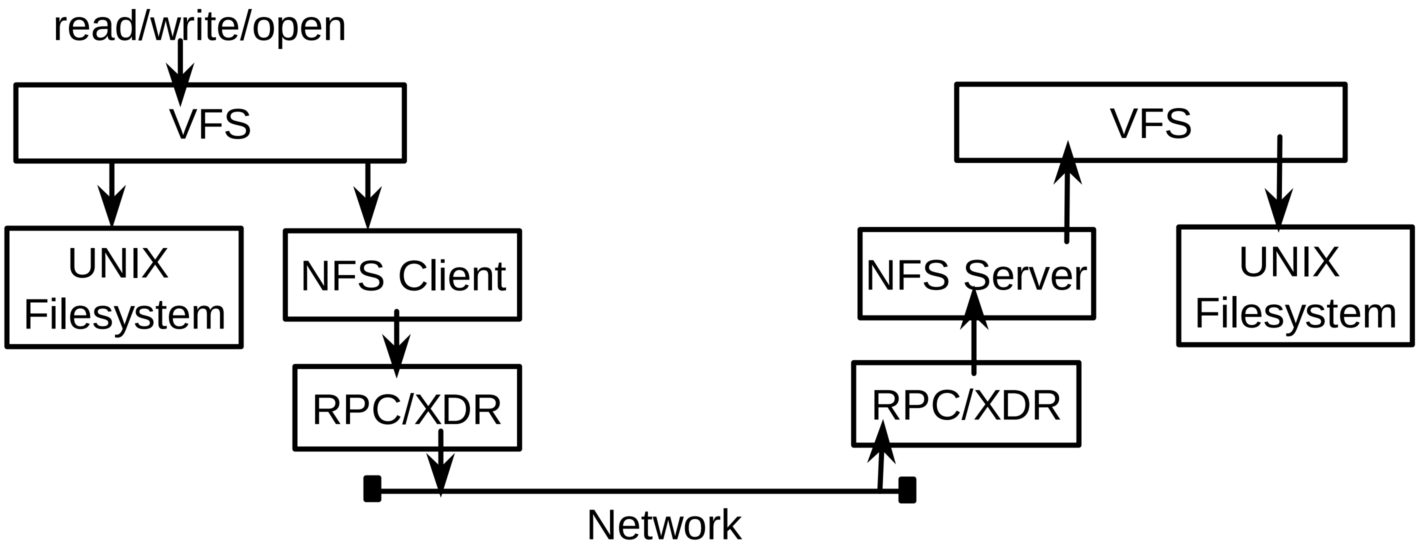
Before Mount

filesystem
to mount

mount point

NFS

- Provides a way to mount remote filesystems
  - can be done explicitly
  - can be done automatically (called an automounter)
  - clients are provided “file handle” by the server for future use
- Uses VFS: extended UNIX filesystem
  - inodes are replaced by vnodes
    - network wide unique inodes
    - can refer to local or remote files



NFS (cont.)

- Requests
  - are sent via RPC to the server
  - include read/write
  - query: lookup this directory info
    - must be done one step (directory) at a time
  - change meta data: file permissions, etc.
- Popular due to free implementations
- Provides no coherency

AFS

- Designed to scale to 5,000 or more workstations
- Location independent naming
  - within a single cell
- volumes
  - basic unit of management
  - can vary in size
  - can be migrated among servers
- names are mapped to “fids”
  - 96 bit unique id’s for a file
  - three parts: volume, vnode, and uniqidentifier
  - location information is stored in a volume to location DB
    - replicated on every server

AFS (cont.)

- File Access
  - open: file is transferred from server to client
    - very large files may only be partially transferred
  - read/write: performed on the client
  - close: file (if dirty) is written back to server
    - can fail if the disk is full
- Consistency
  - clients have callbacks
  - sever informs client when another client writes data
  - only applies to open operation
  - only requires communication when:
    - more than one client wants to write
    - one client wants to write and others to read

Process Migration

- How do you move a process from one system to another?
- Mechanism Issues:
  - need to save and restore all of the process state
    - memory
    - registers
    - pcb info
  - what if the process is talking to other processes
    - how do they find the moved process?
      - often leave a forwarding pointer
- Policy Issues:
  - when is is cost effective to move the process?
    - needs to run for a long enough time to be worth the trouble

Distributed Batch Queuing

- Problem: Many sequential compute bound jobs
- Environment: lots of semi-autonomous workstations
- Solution:
  - support submitting jobs to a pool of workstations
    - find “idle” workstations and use them
  - should look like they are running on a local workstation
- Issues:
  - what if the workstation “owner” returns?
    - need to checkpoint job and migrate it
  - how to make remote jobs look like local jobs?
- Examples:
  - Condor (aka IBM load leveler)
  - Piranha