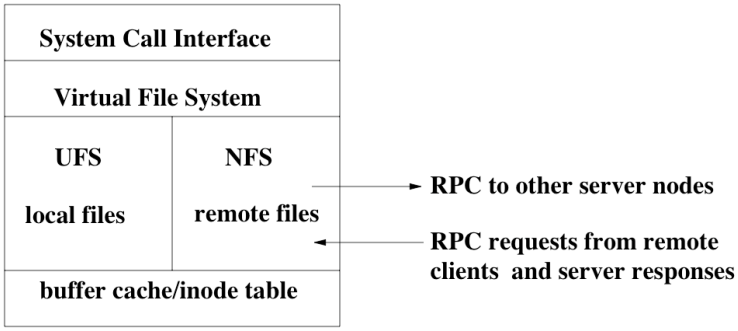
Case Study: Sun's Network File System

- NFS is the standard for distributed UNIX file access.
- NFS is designed to run on LANs.
- Nodes are both servers and clients.
- Servers have no state.
- Uses a mount protocol to make a global name local
  1. /etc/exports lists the local names the server is willing to export.
  2. /etc/fstab lists the global names that the local nodes import. A corresponding global name must be in /etc/exports on the server.

NFS Implementation

- NFS defines a set of RPC operations for remote file access:
  1. directory search, reading directory entries
  2. manipulating links and directories
  3. accessing file attributes
  4. reading/writing files
- Does not rely on node homogeneity - heterogeneous nodes must simply support the NFS mount and remote access protocols using RPC.
- Users may need to know different names depending upon the node to which they logon.

NFS Implementation



NFS Implementation

- NFS defines new layers in the Unix file system
- The virtual file system provides a standard interface, using vnodes as file handles. A vnode describes either a local file or a remote file.
- The ``buffer cache'' caches remote file blocks and attributes.
- On an *open*, the client asks the server whether its cached blocks are up to date.
- Once a file is open, different clients can write to it and get inconsistent data.
- Modified data is flushed back to the server every 30s.
- What file contents do new clients see?
  - Effects of last flush. Writers might have made changes but not updated remote file yet.
- What file contents do existing clients see?
  - For cached blocks, they see out of date info. For new blocks, same as new client

Today: Review

Final Exam covers all material, but emphasizes post midterm material.

- 70% of the exam is on memory management, I/O systems and distributed systems
- 30% of the exam is on the rest of the course

Distributed Systems

1. What is the difference between a distributed system and a parallel system?
2. What advantages do distributed systems have over isolated systems?
3. What advantages do isolated systems have over distributed systems?

Networks

1. What is a LAN?
2. What is a WAN?
3. What are common network topologies? Which are most suitable to WANs? Which to LANs?
4. How do node failures affect the different network topologies?
5. What are the expected communication costs for the different network topologies?
6. What are packets?
7. What is a network protocol stack? What is TCP/IP?

Distributed sharing

1. What is data migration? When would you use it?
2. What is computation migration? When would you use it?
3. What is job migration? When would you use it?

Remote Procedure Call

1. What is RPC?
2. How does RPC differ from normal procedure call?
3. What extra computation is required to do RPC instead of a normal procedure call?
4. Would you ever use RPC to communicate between two processes on the same machine?

Distributed file systems

1. What are location transparent names?
2. What are location independent names?
3. What does it mean to say that a distributed file system has a single (global) namespace?
4. What is a cache?
5. What are the advantages of using a cache in a distributed file system? What are the disadvantages?
6. What are the advantages and disadvantages of write-back and write-through caches?

Highlights of Process Management

1. What is a context switch? What happens during a context switch? What causes a context switch to occur?
2. What is the difference between a process and a thread?
3. What are FCFS, Round Robin, SJF, and Multilevel Feedback Queue algorithms?
4. What is an I/O bound process? What is a CPU bound process? Is there any reason to treat them differently for scheduling purposes?
5. What is a semaphore? What are the three things a semaphore can be used for?
6. What is a monitor? What is a condition variable?
7. What is busy waiting?
8. What are the four necessary conditions for deadlock to occur?
9. What is the difference between deadlock detection and deadlock prevention?
10. After detecting deadlock, what options are conceivable for recovering from deadlock?

Highlights of Memory and I/O Management

1. What is virtual memory and why do we use it?
2. What is paging, a page?
3. What does the OS store in the page table?
4. What is a TLB? How is one used?
5. What is a page fault, how does the OS know it needs to take one, and what does the OS do when a page fault occurs?
6. Page replacement algorithms: FIFO, MIN, LRU, Second chance. For each understand how they work, advantages and disadvantages.
7. How does the OS communicate with I/O devices?
8. What are I/O buffers used for?
9. What are I/O caches used for? How do they affect reading and writing to I/O devices?
10. What is seek time?
11. What is rotational latency?
12. What is transfer time?
13. Disk scheduling algorithms: FIFO, SSTF, SCAN, C-SCAN. How do they work, advantages and disadvantages.

General Skills

- You should have a good sense of how the pieces fit together and how changes in one part of the OS might impact another.
- You will **not** be asked to read or write Java code.
- You will **not** be asked detailed questions about any specific operating system such as Unix, Windows NT.

Wrapping up

- Four key topics: Processes, Memory, I/O & Files and Distributed Systems
- Practical skills: experience with C++
  - Pthreads, syncronization
  - Malloc / heap management
  - Simple File System
- Follow on courses
  - 577: Linux kernel programing
  - 677: Distributed operating systems (grad)
  - Experimental class in iphone programming

Sermons in Computer Science

- Simplicity
- Performance
- Programming as Craft
- Information is Property
- Stay Broad