Quiz #3

- Concisely describe how to use Copy-on-Write to implement sending and receiving large messages;
- In Mach, each region in a process' address space is backed by a memory object, and each memory object is managed by a pager.
  Describe under what situations the kernel calls the pager for service, and under what situations the pager calls the kernel for service.
- Based on your memory of results in assignment 1, describe the latency for the following operations on a platform that you tested: context switch between two processes, system call overhead, and time for a floating point multiply.

Local Disk-Based File Systems

- The access-time gap
- Disk drive modelling
- Fast File System (FFS) for UNIX

Nonremovable Disk Drives

- mechanism
  - recording components
  - positioning components
- disk controller
  - microprocessor, buffer, interface to SCSI bus
  - manages the read and write of data
  - performs mapping between logical disk address and physical disk sectors

A Picture of Disks

Recording Components

- determines storage density and transfer bandwidth
  - physical media: platters and tracks
- density
  - linear recording density (50Kb/inch and higher)
  - track density (2500 tracks/inch and higher)
- rotation speed
  - 3600 rpm and 7200 rpm, 12% increase per year

Positioning Component

- Seeking
  - components: speedup, coast, slowdown, settle
  - power applied on pivot and duration
  - recalibration
- Track following
  - positioning info: embedded or separate surface
  - head-switching: 1/3 of settle time (0.5-1.5ms)
  - track switch: same as settle time

Data Layout

- Zoning
  - longer tracks toward the outside of a platter
  - constant density => outer tracks have more sectors, and higher data transfer rate
  - 3-40 zones, 3.1 MB/s to 5.3 MB/s
- Track skewing
- Sparing
  - sector or track remapping; slip sparing

Disk Controller

- microprocessor
- bus interface: SCSI bus
  - 5MB/s, 10MB/s, 20MB/s, 40MB/s
  - 8 addresses on a bus (16 on wide SCSI)
  - speed-matching buffer
- read caching
  - on-arrival read-ahead or track buffer
  - aggressive read-ahead; segmented cache

Write Caching

- immediate-report
- nonvolatile controller buffer
- scheduling the buffered writes
- command queueing
  - with ordering constraints

Disk Read and Write

Getting a Model...

- Disk: HP97560
  - 1.3GB, 1962 cyl, 5.25", 4002RPM, SCSI-2, 10MB/s
  - 512B sector, 72 sector per track, 19 tracks per cylinder, 1 zone, track skew: 8, cyl skew: 18,
  - controller overhead: 2.2ms
  - seek time: d < 383: 3.24 + 0.4sqrt(d), d>=383: 8.00+0.008d
  - track switch: 1.6ms, disk buffer: 128KB, read fence 64KB

Calculating Disk I/O Time

- controller overhead
- in cache? write immediate reporting?
- seek time
- settle time for writes
- rotational time (track/cylinder skew, sparing)
- head and track switch cost
- media time (overlap of mechanism and bus xfer)
- read-ahead?

Bottom Line...

- You can get full media bandwidth for large sequential transfers
- Both seek latency and rotational latency contribute to I/O time
- Host disk scheduling: sort by disk addresses

UNIX FFS

- Problems with the old implementation:
  - blocks in a file not contiguous
  - 4 MB of inode followed by data; inodes in a directory not contiguous
  - freelist get scrambled quickly

=> very low utilization of disk bandwidth

Cylinder Groups

- a set of consecutive cylinders
- each cylinder group has a copy of superblock, kept on different tracks
- each cylinder group has space for I-node, a bit map of free blocks, and summary info

Blocks and Fragments

- Large block is good for disk throughput, bad for space utilization

=> 4K blocks, 1K fragments

- A file always consists of zero of more 4K blocks, and one fragmented block (copy fragments when necessary)
- roughly the same utilization as 1K blocks
- free space reserve

Layout Policies

- Goal: increase the physical locality of references; enable larger transfers
  - upper level: directory allocation
  - lower level: keep inodes in the same directory to the same cylinder group
  - upper level: keep blocks for a file in the same cylinder group; spill over every 1MB, starting from 48KB; blocks allocated sequentially
  - lower level: try to allocate requested block

File System Parameterization

- would physically contiguous blocks lead to minimal access time
- if not, how much to skip between blocks
- how to determine rotational optimal blocks

Performance

- the new system performs well, as expected
- most of operations limited by CPU speed --- no longer true today

Functional Enhancements

- long file names
- advisory locks
- symbolic links
- rename
- quotas