SE350: Operating Systems

Lecture 13: I/O and Storage Devices

Outline

- I/O subsystem
- Magnetic storage
- Flash memory

What's Next?

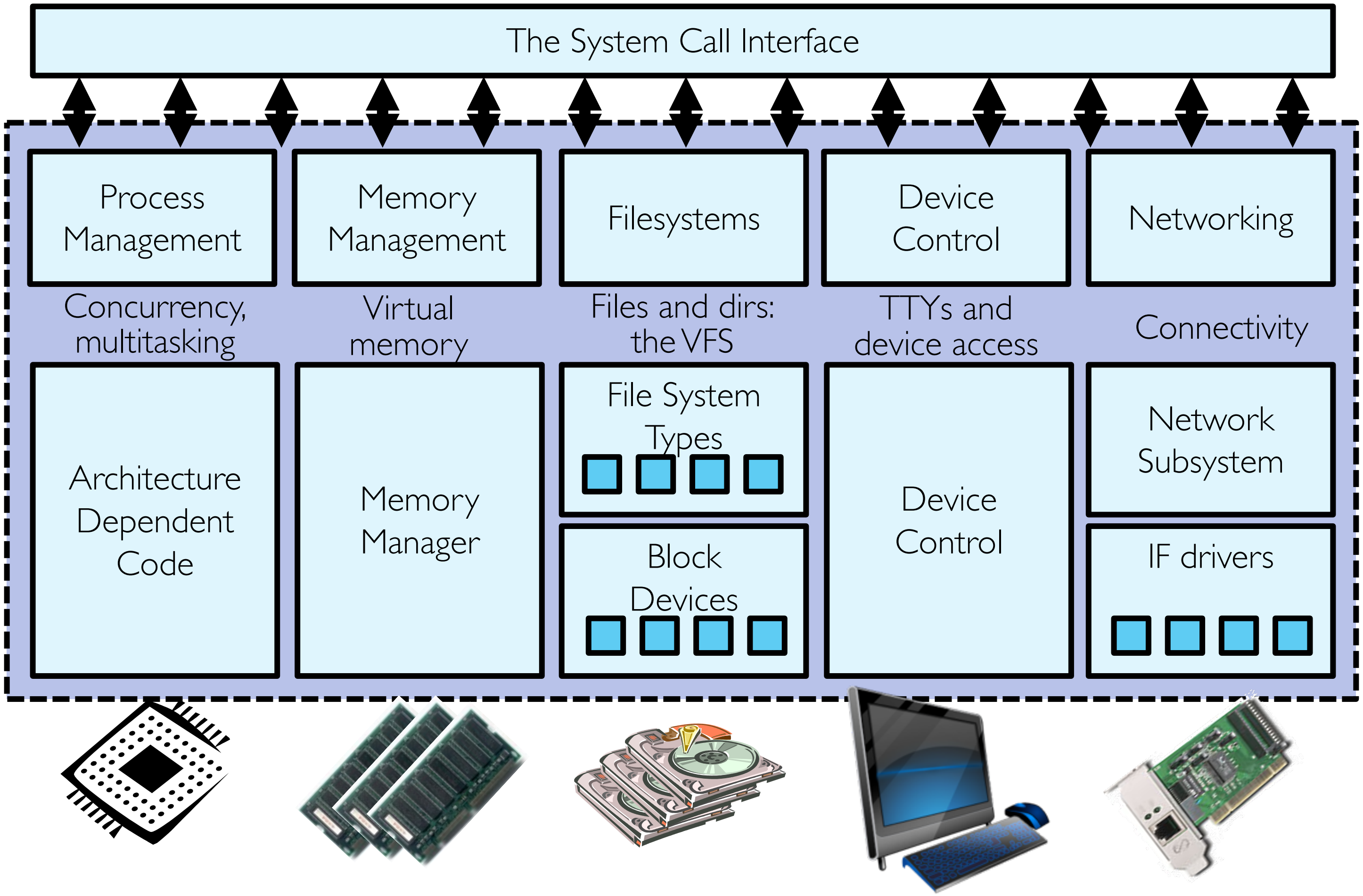
- So far in this course:
  - We have learned how to manage CPU and memory

- What about I/O?
  - Without I/O, computers are useless (disembodied brains?)
  - But … thousands of devices, each slightly different
    - How can we standardize interfaces to these devices?
  - Devices unreliable: media failures and transmission errors
    - How can we make them reliable?
  - Devices unpredictable and/or slow
    - How can we manage them if we don't know what they will do or how they will perform?

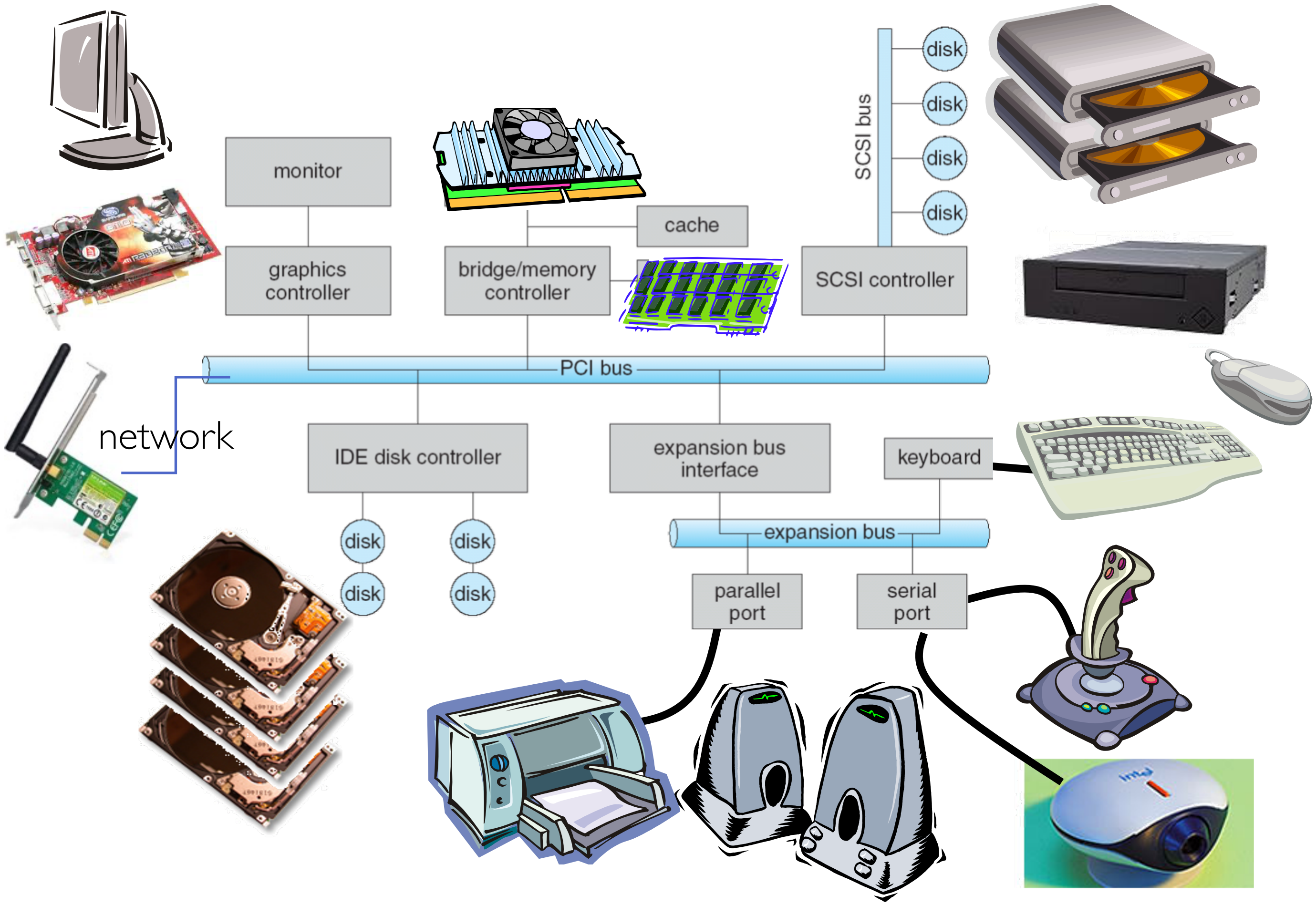
Operational Parameters for I/O

- Data granularity: Byte vs. Block
  - Some devices provide single byte at a time (e.g., keyboard)
  - Others provide whole blocks (e.g., disks, networks, etc.)
- Access pattern: Sequential vs. Random
  - Some devices must be accessed sequentially (e.g., tape)
  - Others can be accessed "randomly" (e.g., disk, cd, etc.)
    - Fixed overhead to start transfers
- Notification mechanisms: Polling vs. Interrupt
  - Some devices require continual monitoring
  - Others generate interrupts when they need service

Kernel Device Structure



Modern I/O Systems



Goal of I/O Subsystem

- Provide uniform interfaces, despite wide range of different devices
  - This code works on many different devices:

```
FILE fd = fopen("/dev/something", "rw");
for (int i = 0; i < 10; i++) {
   fprintf(fd, "Count %d\n", i);
}
close(fd);
```

  - Why? Because device drivers implements standard interface

- We will get a flavor for what is involved in controlling devices in this lecture
  - We can only scratch the surface!

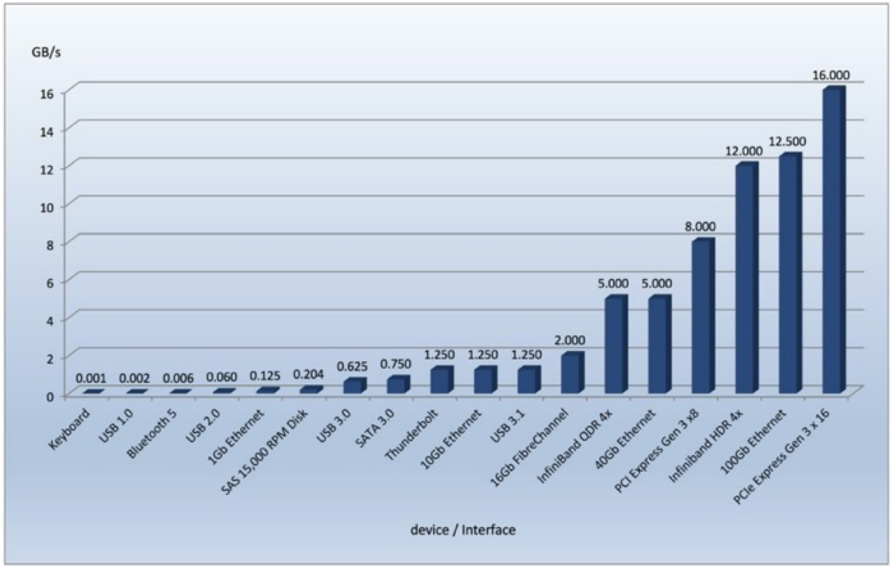
I/O Device Types

- Character devices: e.g. keyboards, mice, serial ports, some USB devices
  - Access single characters at a time
  - Commands include `get(), put()`
  - Libraries layered to allow line editing
- Block devices: e.g. disk drives, tape drives, DVD-ROM
  - Access blocks of data
  - Commands include `open(), read(), write(), seek()`
- Network devices: e.g. Ethernet, Wireless, Bluetooth
  - Different enough from block/character to have its own interface
  - Unix and Windows include socket interface
    - Separates network protocol from network operation
    - Includes `select()` functionality
  - Usage: pipes, FIFOs, streams, queues, mailboxes

I/O Standard Interfaces

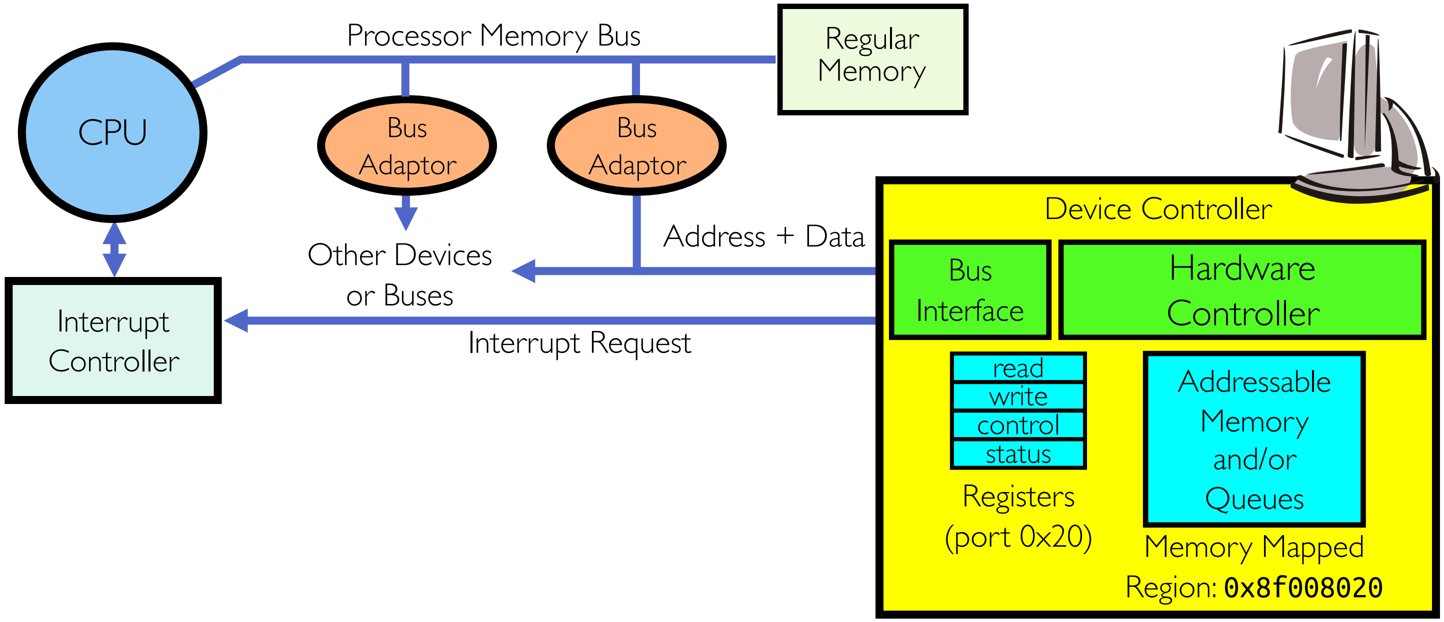
- Blocking interface: "Wait"
  - When request data (e.g. `read()` syscall), put to sleep until data is ready
  - When write data (e.g. `write()` syscall), put to sleep until device is ready
- Non-blocking interface: "Don't wait"
  - Return quickly from read or write with count of bytes successfully transferred
  - Read may return nothing, write may write nothing
- Asynchronous interface: "Tell me later"
  - When request data, take pointer to user's buffer, return immediately; later kernel fills buffer and notifies user
  - When send data, take pointer to user's buffer, return immediately; later kernel takes data and notifies user

I/O Transfer Rates



- Transfer rates vary over 7 orders of magnitude!
  - System better be able to handle this wide range
  - Better not have high overhead/byte for fast devices!
  - Better not waste time waiting for slow devices

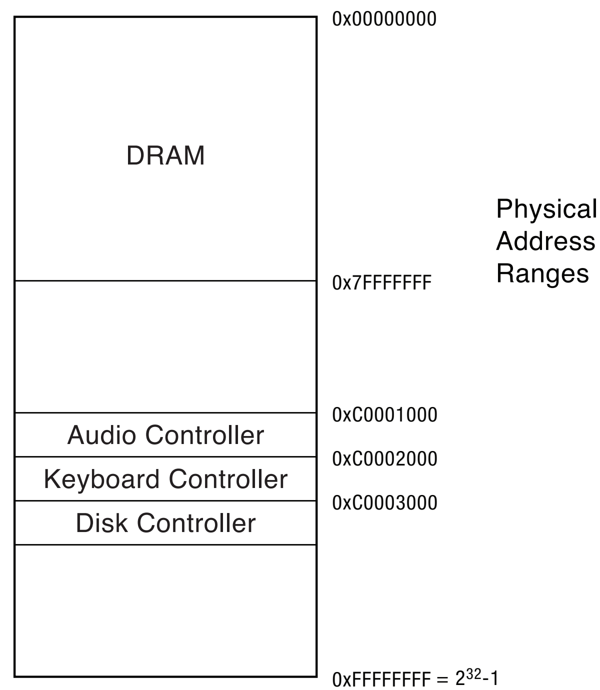
I/O Data Access



- Device controller (may) contains
  - Set of registers and memory buffers that can be read and written
- CPU accesses registers/buffers in two ways
  - Port mapped I/O: **`in/out`** instructions
    - Example from Intel architecture: **`out 0x21,AL`**
  - Memory mapped I/O: **`load/store`** instructions
    - Registers/memory appear in physical address space and accessed by **`load/store`** instructions

Memory Mapped I/O

- Physical address space is shared between DRAM and I/O
- HW maps control registers and device memory into physical address space
  - Set by HW jumpers or at boot time
- I/O-access instructions
  - `load a1, (0xC00…)` // To read
  - `store (0xC00…), b2`. // To write
  - Example: writing to display memory (also called the "frame buffer") changes image on screen



I/O Notification Mechanisms

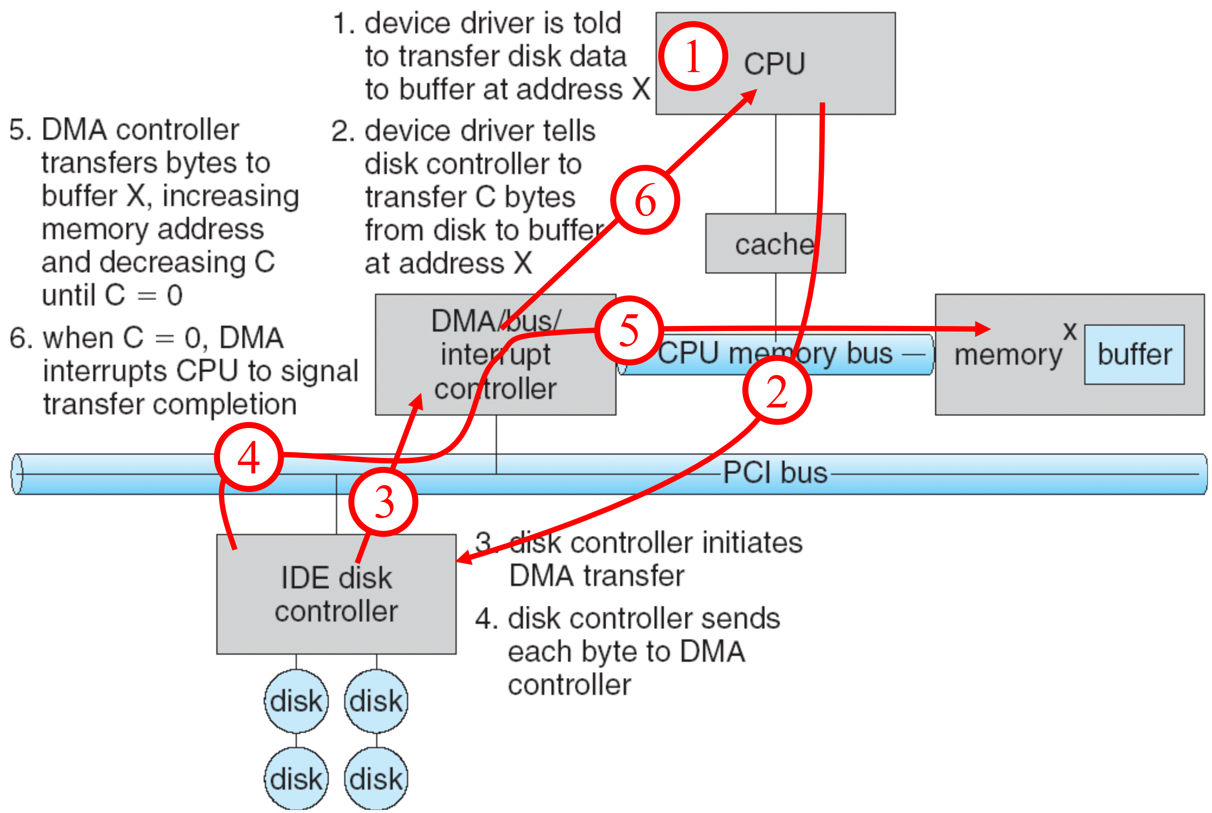
- I/O Interrupt
  - Device generates interrupt whenever it needs service
  - Pro: handles unpredictable events well
  - Con: interrupts have relatively high overhead
- Polling
  - OS periodically checks device-specific status register
    - I/O device puts completion information in status register
  - Pro: low overhead
  - Con: may waste many cycles on polling if infrequent or unpredictable I/O operations
- Actual devices combine both polling and interrupts
  - For instance – High-bandwidth network adapter
    - Interrupt for first incoming packet
    - Poll for following packets until hardware queues are empty

I/O Data Transfer

- Programmed I/O
  - Each byte transferred via processor in/out or load/store
  - Pro: Simple hardware, easy to program
  - Con: Consumes processor cycles proportional to data size

- Direct Memory Access (DMA)
  - Give controller access to memory bus
  - Ask it to transfer data blocks to/from memory directly

DMA Transfer

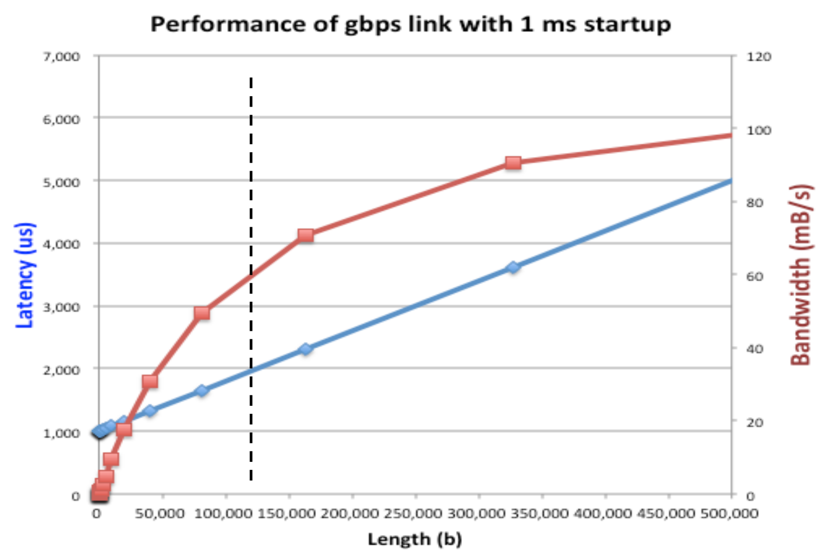


Review: Performance Concepts

- Response Time/Latency: Time to perform an operation
- Bandwidth/Throughput: Rate of executing operations (op/s)
  - Files: MB/s, Networks: Mb/s, Arithmetic: GFLOP/s
- Overhead: time to initiate an operation
- Most I/O operations are roughly linear in n bytes
  - Latency(n) = Overhead + n/Bandwidth

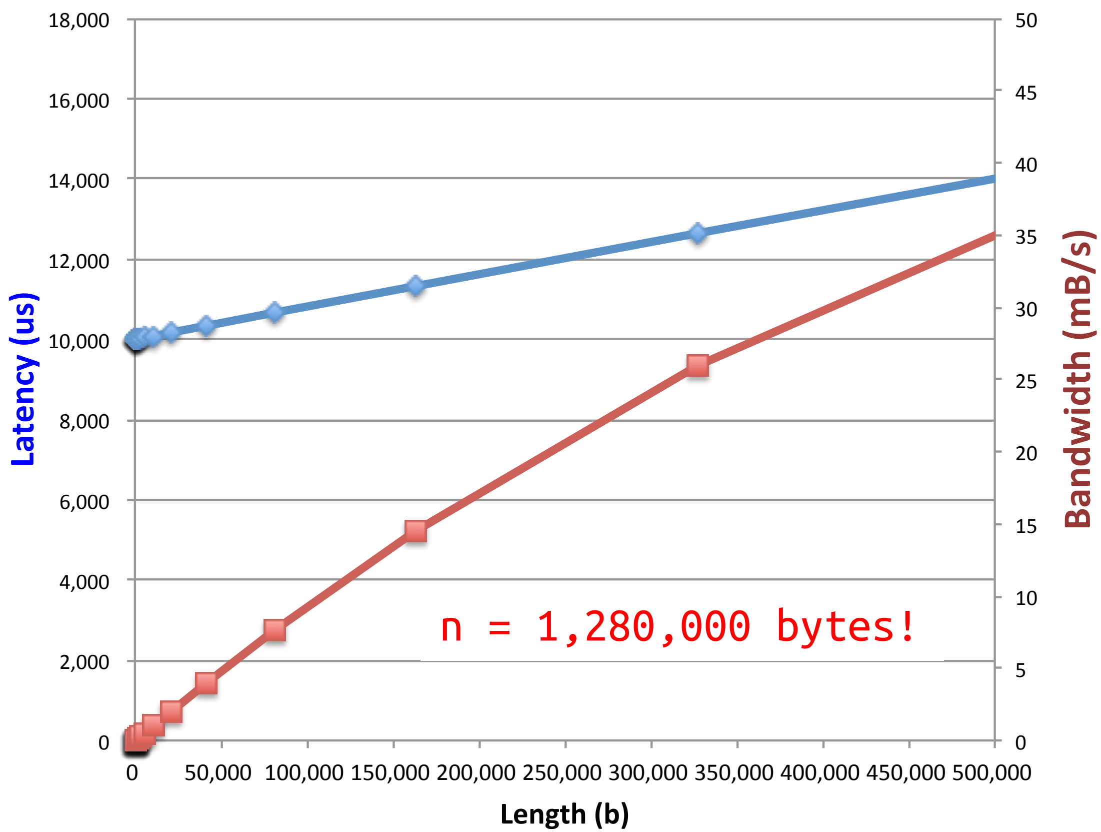
Example: Fast Network

- Consider a `1 Gb/s` link (B = `128 MB/s`)
  - With startup cost S = `1 ms`



- Latency(n) = `S + n/B`
- Bandwidth = `n/(S + n/B) = B x n/(B x S + n) = B/(B x S/n + 1)`
  - Bandwidth = `B/2` when `n = S x B`

Example: Disk (10ms Startup)



What Determines Peak BW for I/O?

- Bus speed (per lane)
  - Ultra Wide SCSI: 320 Mb/s
  - Serial Attached SCSI & Serial ATA & IEEE 1394 (firewire): 1.6 Gb/s
  - USB 3.0 – 5 Gb/s
  - PCI-X: > 8.5 Gb/s (1064 MB/s = 133 MHz x 64 bit)
  - Thunderbolt 3 – 40 Gb/s
  - PCI Express (v. 6, RS-544/514): 60 Gb/s

- Device transfer bandwidth
  - Rotational speed of disk
  - Write/Read rate of NAND flash
  - Signaling rate of network link

- Whatever is the bottleneck in the path…

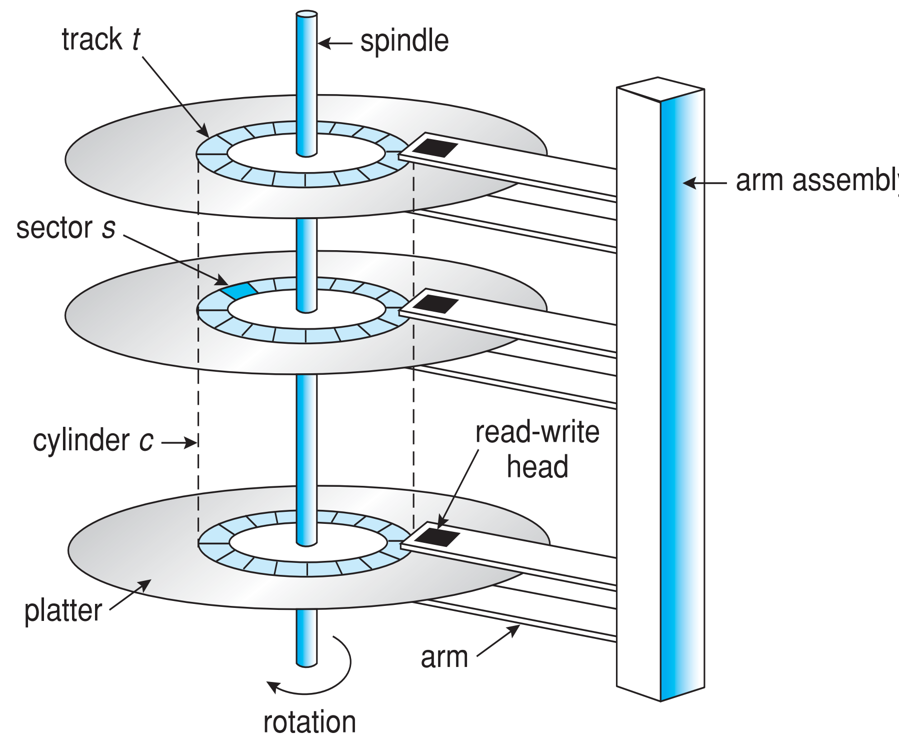
Storage Devices

- Magnetic disks
  - Storage that rarely becomes corrupted
  - Large capacity at low cost
  - Block level random access (except for Shingled Magnetic Recording (SMR))
  - Slow performance for random access
  - Better performance for sequential access

- Flash memory
  - Storage that rarely becomes corrupted
  - Capacity at intermediate cost (5-20x disk)
  - Block level random access
  - Good performance for reads; worse for random writes
  - Erasure requirement in large blocks
  - Wear patterns issue

The Amazing Magnetic Disk

- Unit of transfer: Sector
  - Ring of sectors form track
  - Stack of tracks form cylinder
  - Heads position on cylinders



- Disk tracks ~ 1µm (micron) wide
  - Wavelength of light is ~ 0.5µm
  - Resolution of human eye: 50µm
  - 100K tracks on a typical 2.5" disk
- Separated by unused guard regions
  - Reduces likelihood neighboring tracks are corrupted during writes (still small non-zero chance)

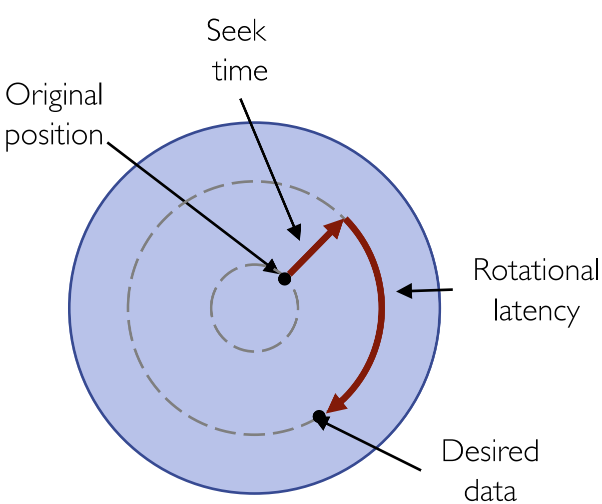
The Amazing Magnetic Disk (cont.)

- Track length varies across disk
  - Outside: More sectors per track, higher bandwidth
  - Disk is organized into regions of tracks with same # of sectors/track
  - Only outer half of radius is used
    - Most of disk area in outer regions of disk
- Disks are so big that some companies (like Google) reportedly only use part of disk for active data
  - Rest is archival data

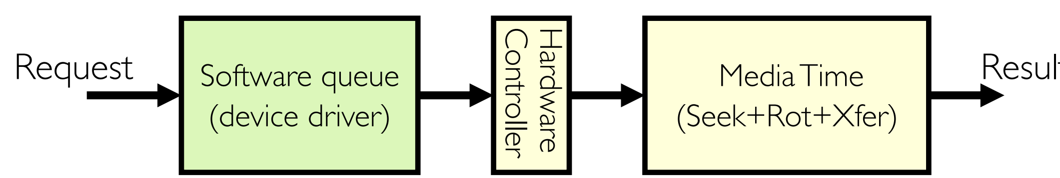
www.lorextechnology.com

Magnetic Disks

- Recall: Cylinder is all tracks under head at any given point on all surface
- Read/write data includes three stages
  - Seek time: position r/w head over proper track
  - Rotational latency: wait for desired sector to rotate under r/w head
  - Transfer time: transfer block of bits (sector) under r/w head



Disk Latency = Queuing Time + Controller time +
Seek Time + Rotation Time + Xfer Time



Disk Performance Example

- Assumptions
  - Ignoring queuing and controller times for now
  - Average seek time of 5 ms
  - 7200 RPM $\Rightarrow$ Time for rotation: 60000 (ms/minute) / 7200 (rotation/minute) $\cong$ 8ms
  - Transfer rate of 4 MB/s, sector size of 1 KB $\Rightarrow$ 2^{10} B / 2^{22} (B/s) = 2^{-12} s $\cong$ 0.24 ms
- Read sector from random place on disk
  - Seek (5 ms) + Rotational delay (4 ms) + Transfer (0.24 ms)
  - Approximately 10ms to fetch/put data: 100 KB/s
- Read sector from random place in same cylinder
  - Rotational delay (4 ms) + Transfer (0.24 ms)
  - Approximately 5 ms to fetch/put data: 200 KB/s
- Read next sector on same track
  - Transfer (0.24 ms): 4 MB/s
- Key to using disk effectively (especially for file systems) is to minimize seek & rotational delays

(Lots of) Intelligence in Controller

- Sectors contain sophisticated error correcting codes
  - Disk head magnet has field wider than track
  - Hide corruptions due to neighboring track writes
- Sector sparing
  - Remap bad sectors transparently to spare sectors on the same surface
- Slip sparing
  - Remap all sectors (when there is a bad sector) to preserve sequential behavior
- Track skewing
  - Offset sector numbers to allow for disk head movement to achieve sequential ops
- …

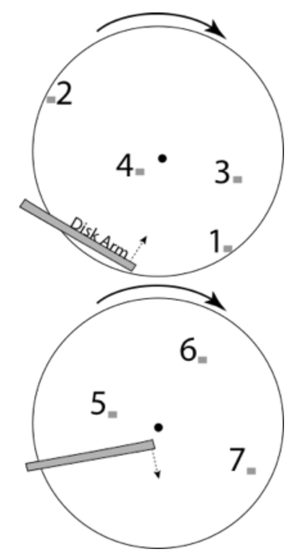
Example of Current HDDs

- Seagate EXOS X14 (2018)
  - 14TB hard disk
  - 8 platters, 16 heads
  - 4.16 ms average seek time
  - 4 KB physical sectors
  - 7200 RPMs
  - 6 Gbps SATA / 12Gbps SAS interface
  - 261 MB/s MAX transfer rate
  - Cache size: 256 MB

- IBM Personal Computer/AT (1986)
  - 30 MB hard disk
  - 30-40 ms seek time
  - 0.7-1 MB/s (est.)

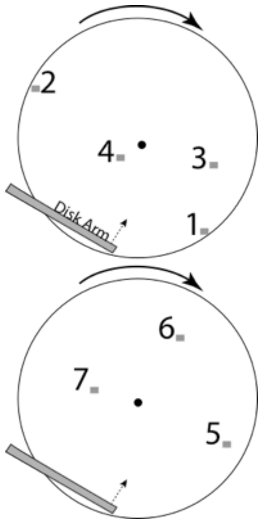
Disk Scheduling

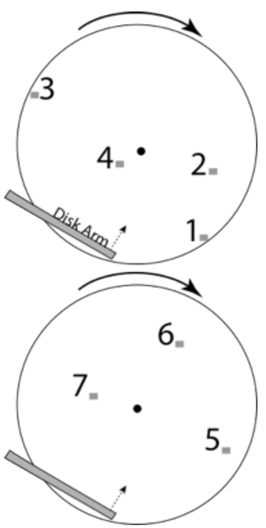
- FCFS: Schedule disk operations in order they arrive
  - Downsides?
    - Poor performance for sequence of requests that alternate between outer and inner tracks
- Shortest Seek Time First (SSTF)
  - Downsides?
    - Not optimal!
    - Starvation!
- SCAN: move disk arm in one direction, until all requests satisfied, then reverse direction
  - Also called "elevator scheduling"



Disk Scheduling (cont.)

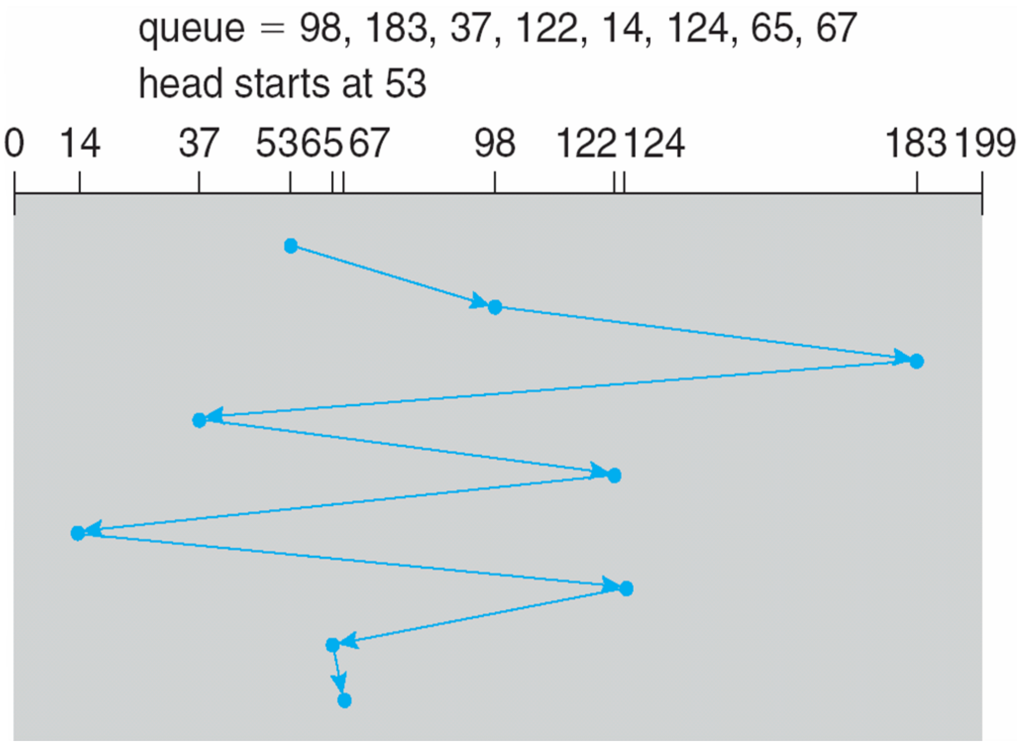
- CSCAN: move disk arm in one direction, until all requests satisfied, then start again from farthest request



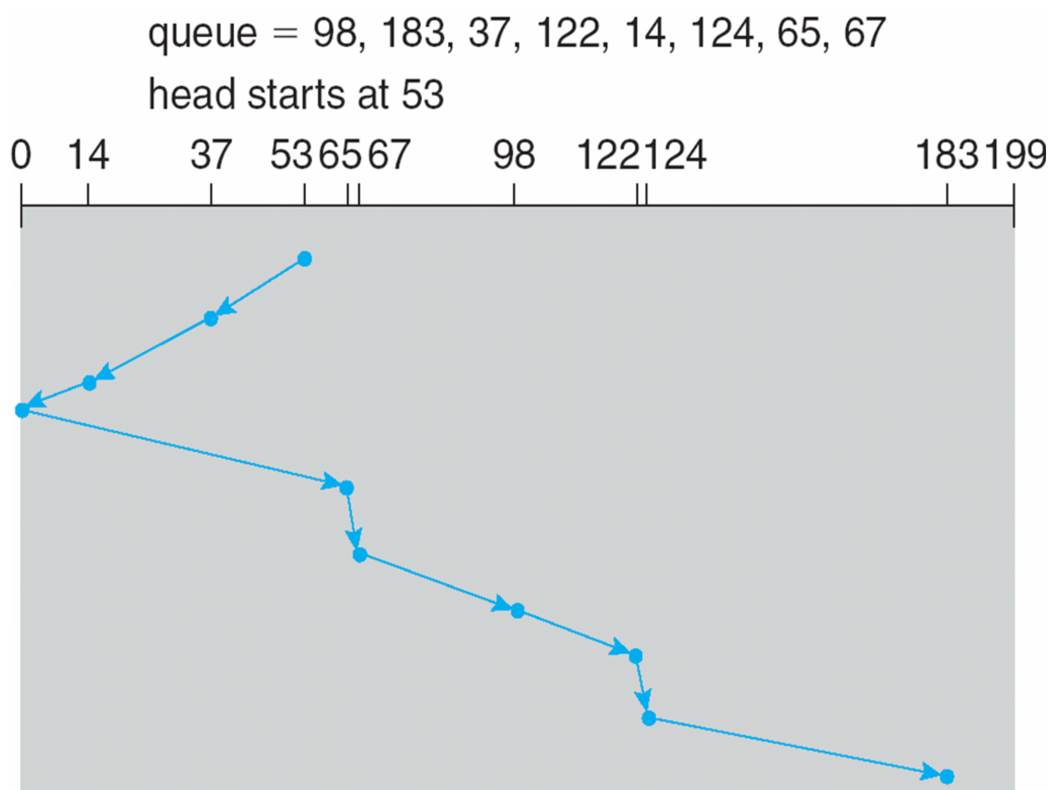


- R-CSCAN: CSCAN but consider that short track switch has rotational delay

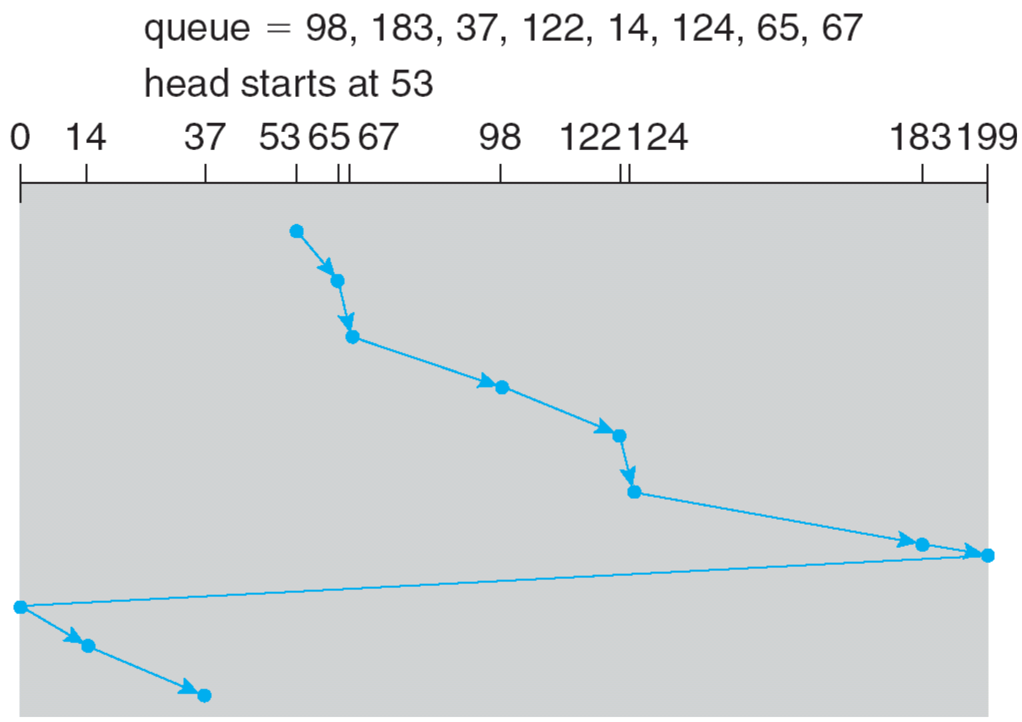
FCFS Example



SCAN Example



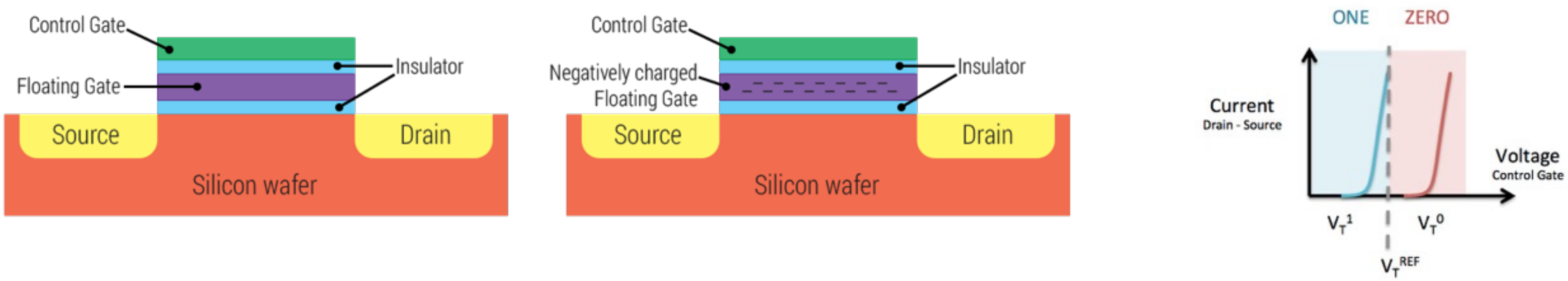
C-SCAN Example



Disk Performance

- When is disk performance highest?
  - When there are big sequential reads, or
  - When there is so much work to do that they can be piggy backed (reordering queues)
- OK to be inefficient when things are mostly idle
- Bursts are both a threat and an opportunity
- Other techniques:
  - Reduce overhead through user level drivers
  - Reduce the impact of I/O delays by doing other useful work in the meantime

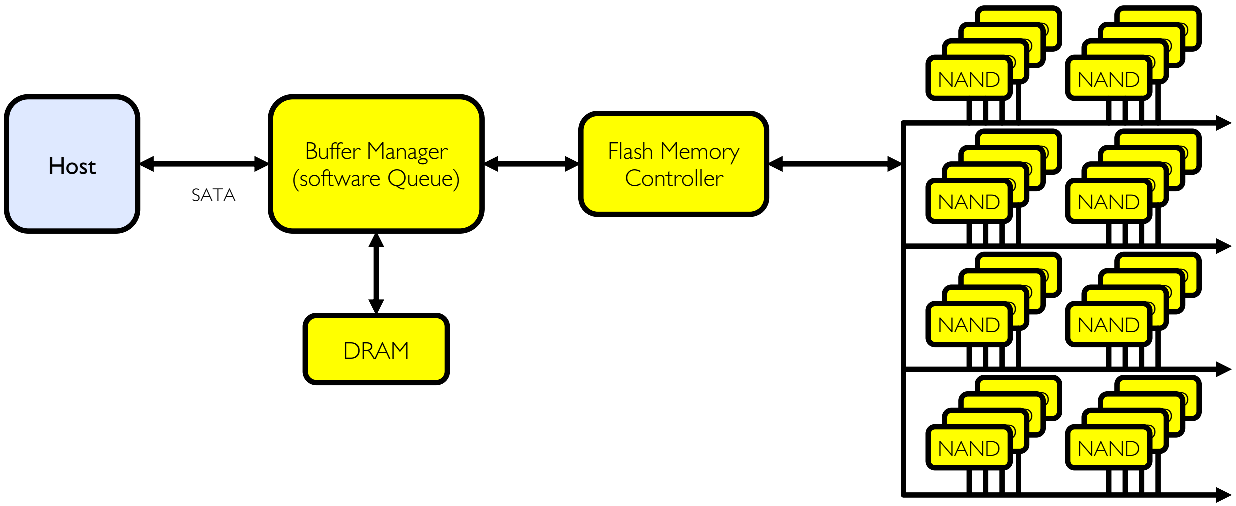
Flash Memory



- 1995: Replace rotating magnetic media with battery backed DRAM
- 2009: Use NAND multi-level cell (2 or 3-bit/cell) flash memory
  - Trapped electrons distinguish between `1` (no charge on FG) and `0` (negative charge on FG)
  - Data can be addressed, read, and modified in pages, typically between 4 KB and 16 KB in size
  - But … data can only be erased at level of entire blocks consisting of multiple pages (MB in size)
  - When block is erased all cells are logically set to `1`
- No moving parts (no rotate/seek motors)
  - Eliminates seek and rotational delay
  - Very low power and lightweight
  - Limited "write cycles"

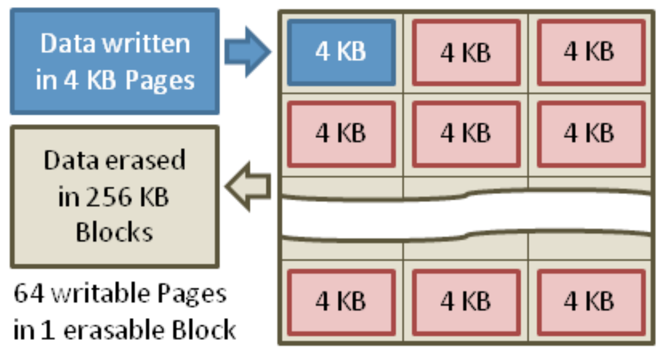
Figures: www.androidcentral.com and flashdba.com

Flash Memory – Reads



- No seek or rotational latency
- Transfer time: transfer 4 KB page
  - SATA: 300-600 MB/s ⇒ 4 KB / 400 MB/s ~ 10 us
- Latency = Queuing time + Controller time + Transfer time
- Highest Bandwidth: Sequential OR Random reads

Flash Memory – Writes



Typical NAND Flash Pages and Blocks

- Writing data is complex!
- Data can only be written into erased pages in each block
- Pages cannot be erased individually, erasing entire block takes time
- Rule of thumb
  - Write take 10x more time than reads
  - Erasure takes 10x more time than writes

Flash Memory Controller

- Maps logical page numbers to physical locations
- Maintains pool of empty blocks by coalescing used pages
  - Garbage collects blocks by copying live pages to new location, then erase
  - More efficient if blocks stored at the same time are deleted at the same time (e.g., keep blocks of file together)
- Wear-levels by only writing each physical page a limited number of times
- Remaps pages that no longer work (sector sparing)
- How does flash device know which blocks are live?
  - File system tells device when blocks are no longer in use (*Trim command*)

Example of Current SSDs

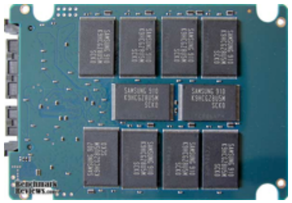
Samsung SSD 970 EVO Plus						
Usage Application	Client PCs					
Interface	PCIe Gen 3.0 x4, NVMe 1.3					
Hardware Information	Capacity[1]		250GB	500GB	1TB	2TB
	Controller		Samsung Phoenix Controller			
	NAND Flash Memory		Samsung V-NAND 3bit MLC			
	DRAM Cache Memory		512MB LPDDR4		1GB LPDDR4	2GB LPDDR4
	Dimension		Max 80.15 x Max 22.15 x Max 2.38 (mm)			
	Form Factor		M.2 (2280)[2]			
Performance (Up to.)[3][4]	Sequential Read		3500 MB/s			
	Sequential Write		2300 MB/s	3200 MB/s	3300 MB/s	
	QD 1 Thread 1	Ran. Read	17K IOPS	19K IOPS		
		Ran. Write	60K IOPS			62K IOPS
	QD 32 Thread 4	Ran. Read	250K IOPS	480K IOPS	600K IOPS	620K IOPS
		Ran. Write	550K IOPS			560K IOPS

Is full Kindle Heavier than Empty One?

- Actually, "Yes", but not by much
- Flash works by trapping electrons:
  - So, erased state lower energy than written state
- Assuming that:
  - Kindle has 4 GB flash
  - ½ of all bits in full Kindle are in high-energy state
  - High-energy state about 10-15 joules higher
  - Then: Full Kindle is 1 attogram (10^{-18} gram) heavier (Using E = mc2)
- Of course, this is less than most sensitive scale can measure (10^{-9} grams)
- This difference is overwhelmed by battery discharge, weight from getting warm, …

According to John Kubiatowicz (New York Times, Oct 24, 2011)

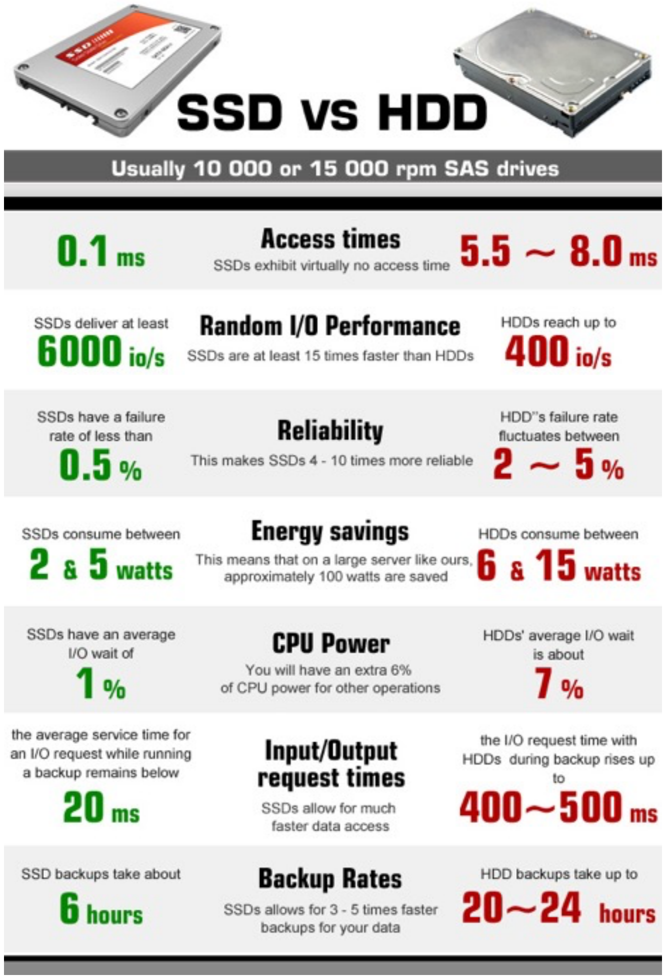
SSD Summary

- Pros (vs. hard disk drives)
  - Low latency, high throughput (eliminate seek/rotational delay)
  - No moving parts
    - Very light weight, low power, silent, very shock insensitive
  - Read at memory speeds (limited by controller and I/O bus)
- Cons
  - Expensive
  - Asymmetric block write performance
    - Controller garbage collection (GC) algorithms have major effect on performance
  - Limited drive lifetime
    - 1-10K writes/page for MLC NAND
    - Average failure rate is 6 years, life expectancy is 9–11 years

(These are changing rapidly!)

HDD vs. SSD


SSD vs HDD
Usually 10 000 or 15 000 rpm SAS drives
0.1 ms
Access times
SSDs exhibit virtually no access time
5.5 ~ 8.0 ms
SSDs deliver at least
6000 io/s
Random I/O Performance
SSDs are at least 15 times faster than HDDs
HDDs reach up to
400 io/s
SSDs have a failure rate of less than
0.5 %
Reliability
This makes SSDs 4 - 10 times more reliable
HDD''s failure rate fluctuates between
2 ~ 5 %
SSDs consume between
2 & 5 watts
Energy savings
This means that on a large server like ours, approximately 100 watts are saved
HDDs consume between
6 & 15 watts
SSDs have an average I/O wait of
1 %
CPU Power
You will have an extra 6% of CPU power for other operations
HDDs' average I/O wait is about
7 %
the average service time for an I/O request while running a backup remains below
20 ms
Input/Output request times
SSDs allow for much faster data access
the I/O request time with HDDs during backup rises up to
400~500 ms
SSD backups take about
6 hours
Backup Rates
SSDs allows for 3 - 5 times faster backups for your data
HDD backups take up to
20~24 hours

Summary (1/2)

- I/O device types
  - Many different speeds (0.1 B/s to GB/s)
  - Different access patterns
    - Block devices, character devices, network devices
  - Different access timing
    - Blocking, non-blocking, asynchronous

- I/O controllers
  - Hardware that controls actual device
  - Processor accesses through I/O instructions, load/store to special physical memory

- I/O notification mechanisms
  - Interrupts
  - Polling: Report results through status register that processor looks at periodically

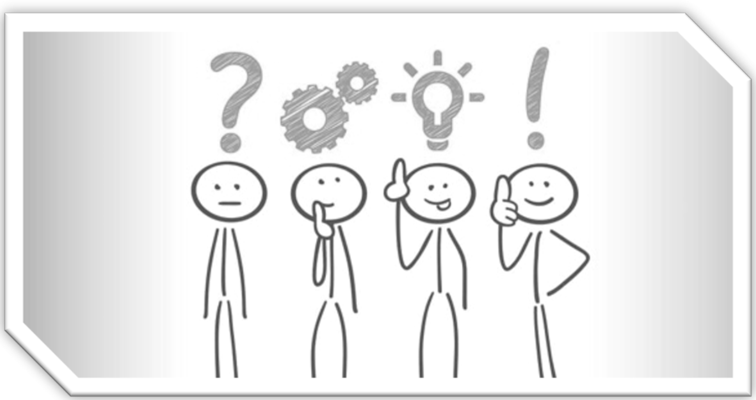
Summary (2/2)

- Disk performance
  - Rotational latency: on average ½ rotation
  - Transfer time: spec of disk depends on rotation speed and bit storage density

- Devices have complex interaction and performance characteristics
  - Response time (Latency) = Queue + Overhead + Transfer
    - Effective BW = BW * T/(S+T)
  - HDD: Queuing time + controller + seek + rotation + transfer
  - SDD: Queuing time + controller + transfer (erasure & wear)

- Disk scheduling
  - FIFO, SSTF, SCAN, CSCAN, R-CSCAN

Questions?



Acknowledgment

- Slides by courtesy of Anderson, Culler, Stoica, Silberschatz, Joseph, and Canny