CS162
Operating Systems and Systems Programming
Lecture 16

General I/O

March 20th, 2017

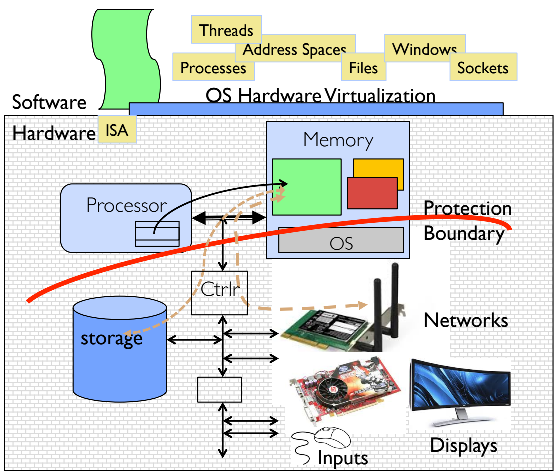
Prof. Ion Stoica

http://cs162.eecs.Berkeley.edu

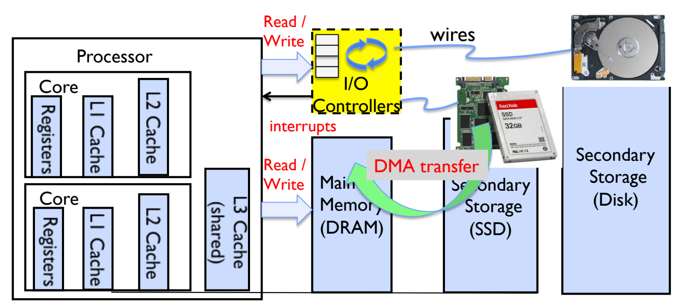
The Requirements of I/O

- 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 the 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?

OS Basics: I/O



In a Picture

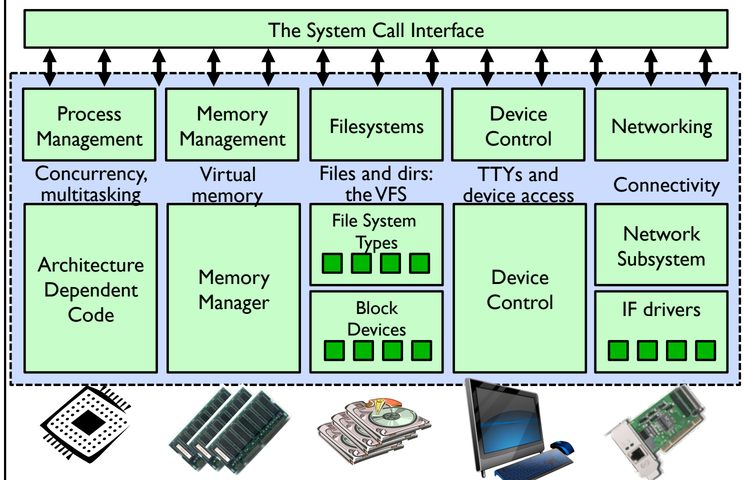


- I/O devices you recognize are supported by I/O Controllers
- Processors accesses them by reading and writing IO registers as if they were memory
  - Write commands and arguments, read status and results

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
  - Some devices require continual monitoring
  - Others generate interrupts when they need service
- Transfer Mechanism: Programmed IO and DMA

Kernel Device Structure



The Goal of the 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 code that controls devices ("device driver") implements standard interface
- We will try to get a flavor for what is involved in actually controlling devices in rest of lecture
  - Can only scratch surface!

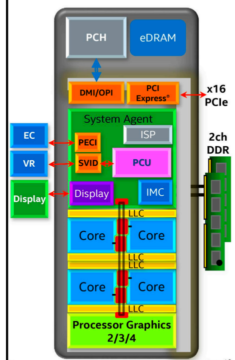
Want Standard Interfaces to Devices

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

How Does User Deal with Timing?

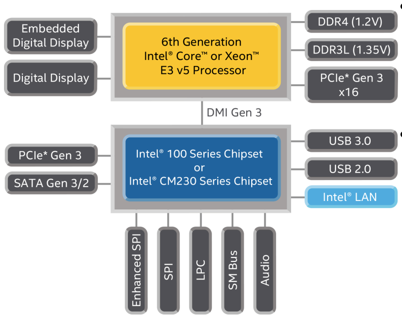
- Blocking Interface: "Wait"
  - When request data (e.g. `read()` system call), put process to sleep until data is ready
  - When write data (e.g. `write()` system call), put process to sleep until device is ready for data
- Non-blocking Interface: "Don't Wait"
  - Returns quickly from read or write request 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

Chip-scale Features of 2015 x86 (Sky Lake)



- Significant pieces:
  - Four OOO cores with deeper buffers
    - » New Intel MPX (Memory Protection Extensions)
    - » New Intel SGX (Software Guard Extensions)
    - » Issue up to 6 μ-ops/cycle
  - Integrated GPU, System Agent (Mem, Fast I/O)
  - Large shared L3 cache with on-chip ring bus
    - » 2 MB/core instead of 1.5 MB/core
    - » High-BW access to L3 Cache
- Integrated I/O
  - Integrated memory controller (IMC)
    - » Two independent channels of DDR3L/DDR4 DRAM
  - High-speed PCI-Express (for Graphics cards)
  - Direct Media Interface (DMI) Connection to PCH (Platform Control Hub)

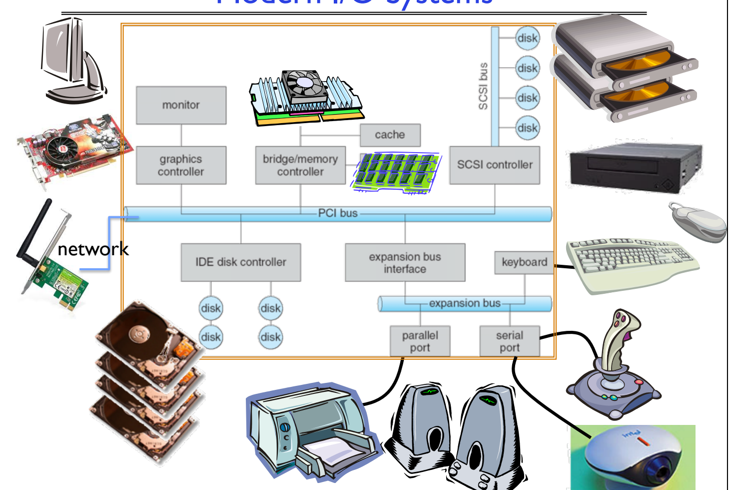
Sky Lake I/O: PCH



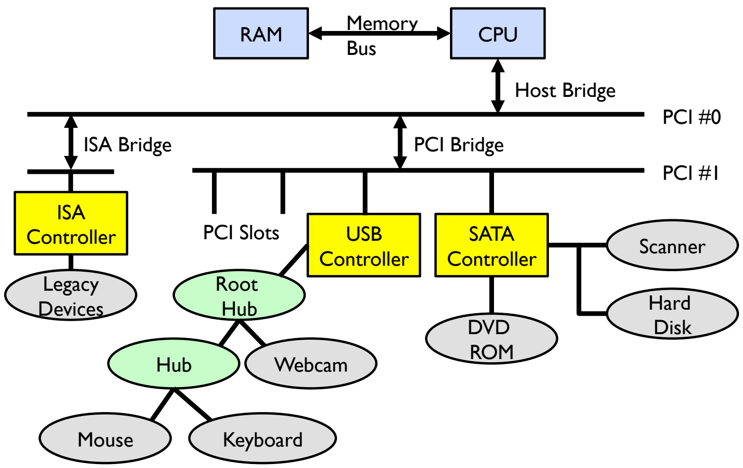
Sky Lake
System Configuration

- Platform Controller Hub
  - Connected to processor with proprietary bus
    - » Direct Media Interface
- Types of I/O on PCH:
  - USB, Ethernet
  - Thunderbolt 3
  - Audio, BIOS support
  - More PCI Express (lower speed than on Processor)
  - SATA (for Disks)

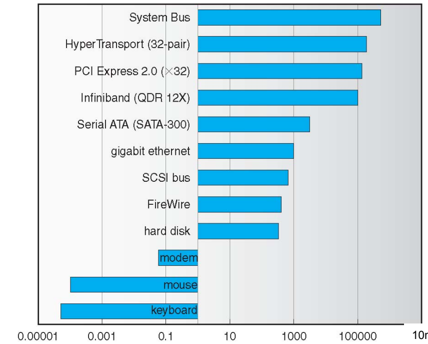
Modern I/O Systems



Example: PCI Architecture



Example Device-Transfer Rates in Mb/s (Sun Enterprise 6000)



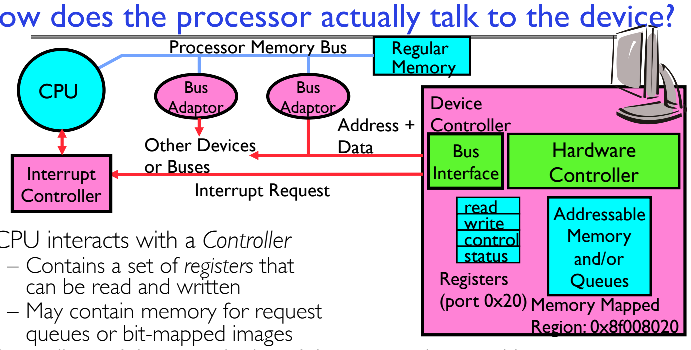
- Device Rates vary over 12 orders of magnitude !!!
  - System better be able to handle this wide range
  - Better not have high overhead/byte for fast devices!
  - Better not waste time waiting for slow devices

Administrivia

- Midterm 2 TOMORROW on Tue 3/21 7:00-8:30PM
  - All topics up to and including Lecture 15
    - » Focus will be on Lectures 11 – 15 and associated readings
    - » Projects 1 & 2, Homework 0 – 2
  - Closed book with 2 pages of hand-written notes both sides
  - Room assignments by **last name**:
    - » A-H 1 100 Genetics and Plant Biology Building, I-Z 1 Pimentel

BREAK

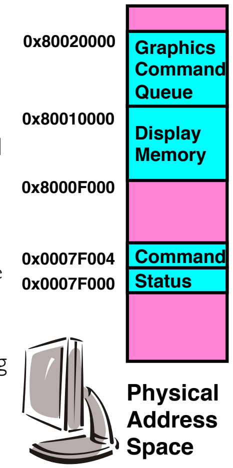
How does the processor actually talk to the device?



- CPU interacts with a *Controller*
  - Contains a set of *registers* that can be read and written
  - May contain memory for request queues or bit-mapped images
- Regardless of the complexity of the connections and buses, processor accesses registers in two ways:
  - I/O instructions: in/out instructions
    - » Example from the Intel architecture: `out 0x21,AL`
  - Memory mapped I/O: load/store instructions
    - » Registers/memory appear in physical address space
    - » I/O accomplished with load and store instructions

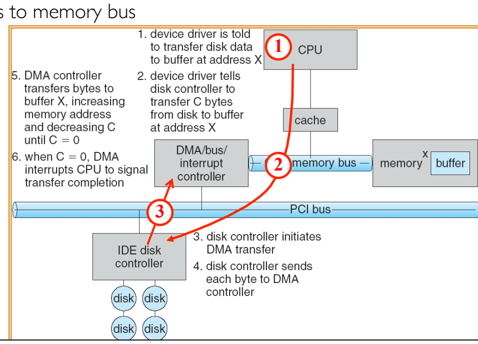
Example: Memory-Mapped Display Controller

- Memory-Mapped:
  - Hardware maps control registers and display memory into physical address space
    - » Addresses set by HW jumpers or at boot time
  - Simply writing to display memory (also called the "frame buffer") changes image on screen
    - » Addr: `0x8000F000 – 0x8000FFFF`
  - Writing graphics description to cmd queue
    - » Say enter a set of triangles describing some scene
    - » Addr: `0x80010000 – 0x8001FFFF`
  - Writing to the command register may cause on-board graphics hardware to do something
    - » Say render the above scene
    - » Addr: `0x0007F004`
- Can protect with address translation



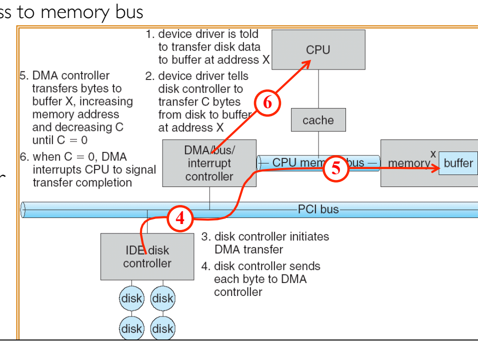
Transferring Data To/From Controller

- 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:
  - Give controller access to memory bus
  - Ask it to transfer data blocks to/from memory directly
- Sample interaction with DMA controller (from OSC book):



Transferring Data To/From Controller

- 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:
  - Give controller access to memory bus
  - Ask it to transfer data blocks to/from memory directly
- Sample interaction with DMA controller (from OSC book):



I/O Device Notifying the OS

- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error
- I/O Interrupt:
  - Device generates an interrupt whenever it needs service
  - Pro: handles unpredictable events well
  - Con: interrupts relatively high overhead
- Polling:
  - OS periodically checks a 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

Device Drivers

- Device Driver: Device-specific code in the kernel that interacts directly with the device hardware
  - Supports a standard, internal interface
  - Same kernel I/O system can interact easily with different device drivers
  - Special device-specific configuration supported with the `ioctl()` system call
- Device Drivers typically divided into two pieces:
  - Top half: accessed in call path from system calls
    - » implements a set of standard, cross-device calls like `open()`, `close()`, `read()`, `write()`, `ioctl()`, `strategy()`
    - » This is the kernel's interface to the device driver
    - » Top half will *start* I/O to device, may put thread to sleep until finished
  - Bottom half: run as interrupt routine
    - » Gets input or transfers next block of output
    - » May wake sleeping threads if I/O now complete

Life Cycle of An I/O Request

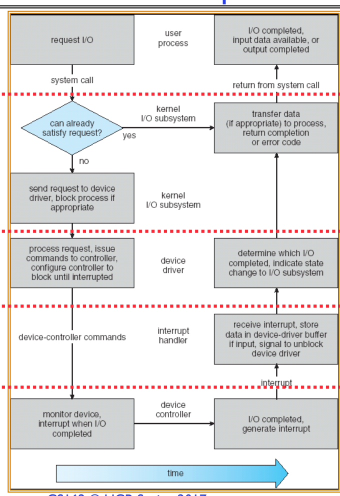
User Program

Kernel I/O Subsystem

Device Driver Top Half

Device Driver Bottom Half

Device Hardware

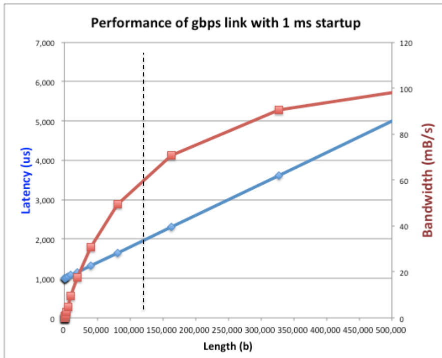


Basic Performance Concepts

- *Response Time or Latency*: Time to perform an operation(s)
- *Bandwidth or Throughput*: Rate at which operations are performed (op/s)
  - Files: MB/s, Networks: Mb/s, Arithmetic: GFLOP/s
- *Start up* or "Overhead": time to initiate an operation
- Most I/O operations are roughly linear in *n* bytes
  - Latency(n) = Overhead + n/TransferCapacity

Example (Fast Network)

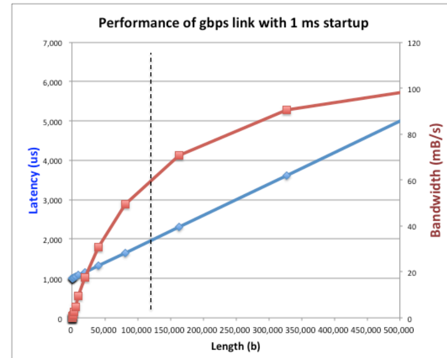
- Consider a 1 Gb/s link (B = 125 MB/s)
  - With a startup cost S = 1 ms



  - Latency(n) = S + n/B
  - Bandwidth = n/(S + n/B) = B*n/(B*S + n) = B/(B*S/n + 1)

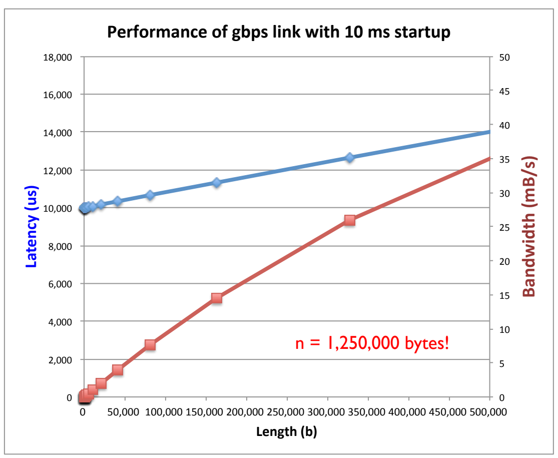
Example (Fast Network)

- Consider a 1 Gb/s link (B = 125 MB/s)
  - With a startup cost S = 1 ms



  - Bandwidth = B/(B*S/n + 1)
  - half-power point occurs at n=S*B → Bandwidth = B/2

Example: at 10 ms startup (like Disk)



What Determines Peak BW for I/O ?

- Bus Speed
  - PCI-X: 1064 MB/s = 133 MHz x 64 bit (per lane)
  - ULTRA WIDE SCSI: 40 MB/s
  - Serial Attached SCSI & Serial ATA & IEEE 1394 (firewire): 1.6 Gb/s full duplex (200 MB/s)
  - USB 3.0 – 5 Gb/s
  - Thunderbolt 3 – 40 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 SMR – later!)
  - Slow performance for random access
  - Better performance for sequential access

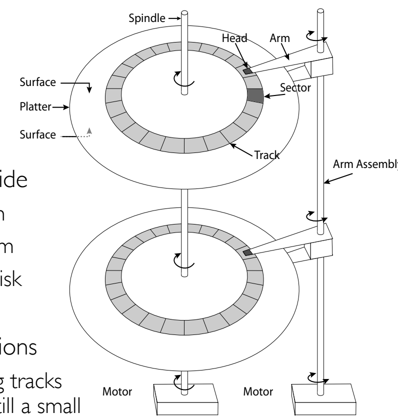
- Flash memory
  - Storage that rarely becomes corrupted
  - Capacity at intermediate cost (5-20x disk)
  - Block level random access
  - Good performance for reads; worse for random writes
  - Erasure requirement in large blocks
  - Wear patterns issue

The Amazing Magnetic Disk

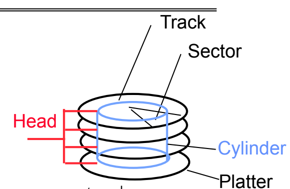
- Unit of Transfer: Sector
  - Ring of sectors form a track
  - Stack of tracks form a cylinder
  - Heads position on cylinders

- 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 a small non-zero chance)

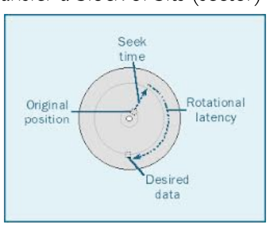


Review: Magnetic Disks



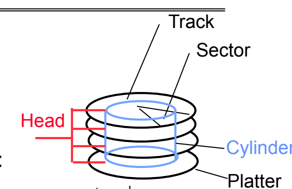
- Cylinders: all the tracks under the head at a given point on all surface

- Read/write data is a three-stage process:
  - Seek time: position the head/arm over the proper track
  - Rotational latency: wait for desired sector to rotate under r/w head
  - Transfer time: transfer a block of bits (sector) under r/w head



Seek time = 4-8m
One rotation = 1-2ms
(3600-7200 RPM)

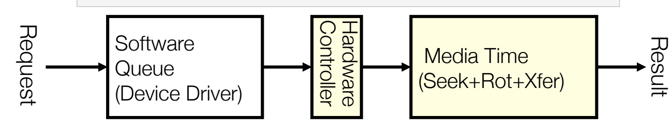
Review: Magnetic Disks



- Cylinders: all the tracks under the head at a given point on all surface

- Read/write data is a three-stage process:
  - Seek time: position the head/arm over the proper track
  - Rotational latency: wait for desired sector to rotate under r/w head
  - Transfer time: transfer a block of bits (sector) under r/w head

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



Disk Performance Example

- Assumptions:
  - Ignoring queuing and controller times for now
  - Avg seek time of 5ms,
  - 7200RPM ⇒ Time for rotation: 60000 (ms/minute) / 7200(rev/min) ~= 8ms
  - Transfer rate of 4MByte/s, sector size of 1 Kbyte ⇒ 1024 bytes/4×10^6 (bytes/s) = 256 × 10^{-6} sec ≅ .26 ms
- Read sector from random place on disk:
  - Seek (5ms) + Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 10ms to fetch/put data: **100 KByte/sec**
- Read sector from random place in same cylinder:
  - Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 5ms to fetch/put data: **200 KByte/sec**
- Read next sector on same track:
  - Transfer (0.26ms): **4 MByte/sec**
- Key to using disk effectively (especially for file systems) is to *minimize seek and rotational delays*

Typical Numbers for Magnetic Disk

Parameter	Info / Range
Space/Density	Space: 8TB (Seagate), 10TB (Hitachi) in 3½ inch form factor! Areal Density: ≥ 1Terabit/square inch! (SMR, Helium, …)
Average seek time	Typically 5-10 milliseconds. Depending on reference locality, actual cost may be 25-33% of this number.
Average rotational latency	Most laptop/desktop disks rotate at 3600-7200 RPM (16-8 ms/rotation). Server disks up to 15,000 RPM. Average latency is halfway around disk so 8-4 milliseconds
Controller time	Depends on controller hardware
Transfer time	Typically 50 to 100 MB/s. Depends on: • Transfer size (usually a sector): 512B – 1KB per sector • Rotation speed: 3600 RPM to 15000 RPM • Recording density: bits per inch on a track • Diameter: ranges from 1 in to 5.25 in
Cost	Used to drop by a factor of two every 1.5 years (or even faster); now slowing down

(Lots of) Intelligence in the Controller

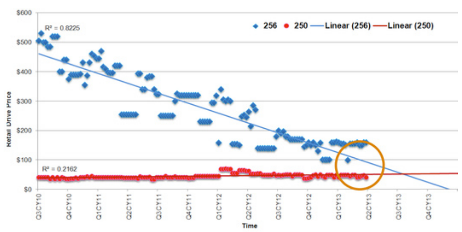
- Sectors contain sophisticated error correcting codes
  - Disk head magnet has a field wider than track
  - Hide corruptions due to neighboring track writes
- Sector sparing
  - Remap bad sectors transparently to spare sectors on the same surface
- Slip sparing
  - Remap all sectors (when there is a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

Seagate Enterprise

10 TB (2016)
- 7 platters, 14 heads
- 7200 RPMs
- 6 Gbps SATA /12Gbps SAS interface
- 220MB/s transfer rate, cache size: 256MB
- Helium filled: reduce friction and power usage
- Price: $500 ($0.05/GB)

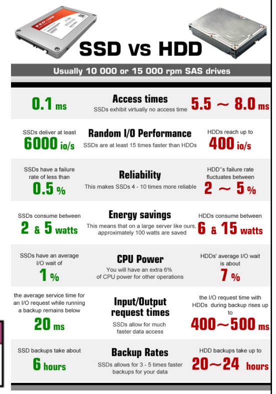
IBM Personal Computer/AT (1986)
- 30 MB hard disk
- 30-40ms seek time
- 0.7-1 MB/s (est.)
- Price: $500 ($17K/GB, 340,000x more expensive !!)

HDD vs SSD Comparison



Price Crossover Point for HDD and SSD

	2012	2013	2014	2015E	2016F	2017F
HDD	0.09	0.08	0.07	0.06	0.06	0.06
2.5" SSD	0.99	0.68	0.55	0.39	0.24	0.17



SSD prices drop much faster than HDD

Largest SSDs

- 60TB (2016)
- Dual port: 16Gbs
- Seq reads: 1.5GB/s
- Seq writes: 1GB/s
- Random Read Ops (IOPS): 150K
- Price: ~ $20K ($0.33/GB)

Summary

- I/O Devices Types:
  - Many different speeds (0.1 bytes/sec to GBytes/sec)
  - 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
- Notification mechanisms
  - Interrupts
  - Polling: Report results through status register that processor looks at periodically
- Device drivers interface to I/O devices
  - Provide clean Read/Write interface to OS above
  - Manipulate devices through PIO, DMA & interrupt handling
  - Three types: block, character, and network