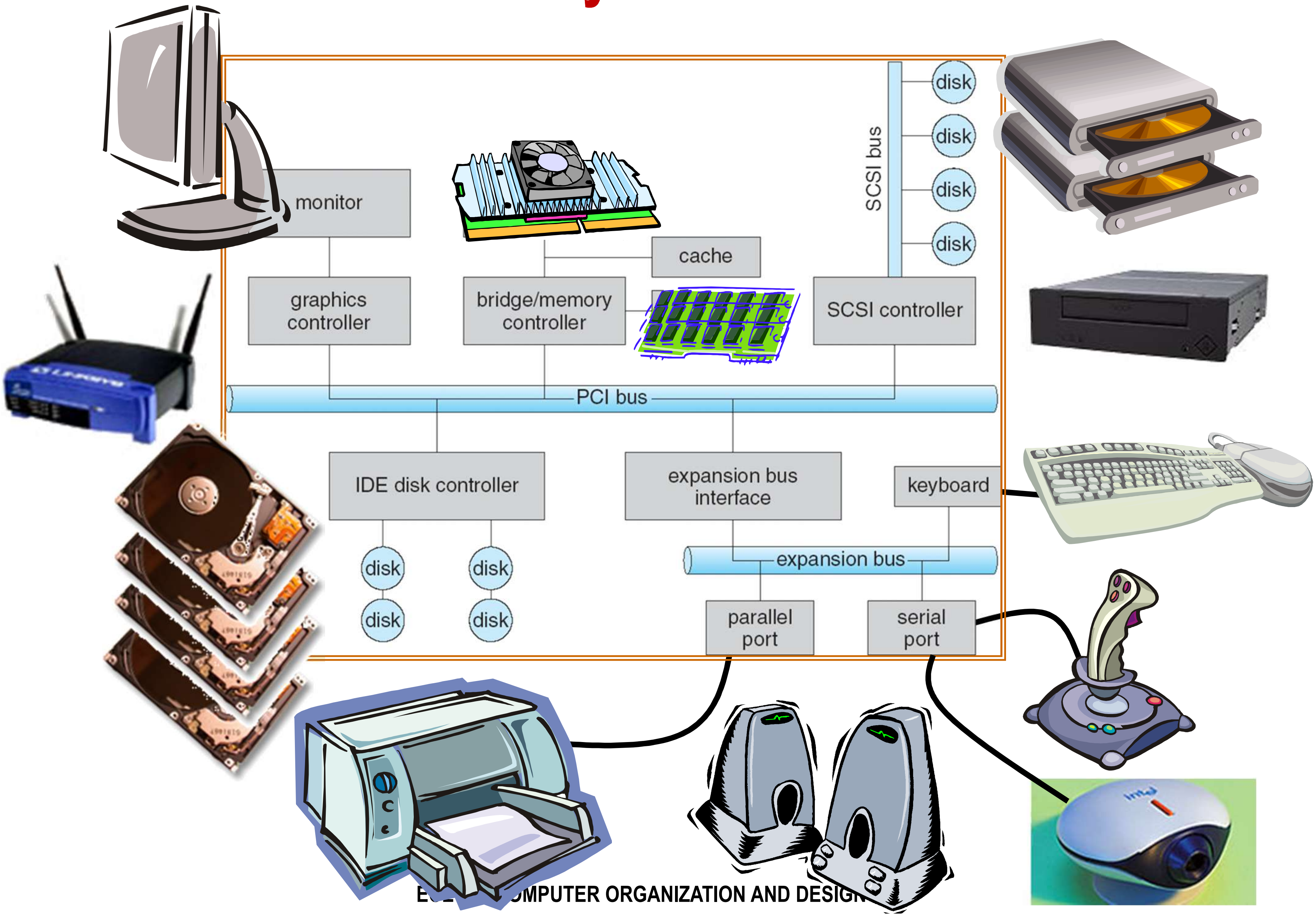
Lecture 16:
I/O Subsystems – Storage System

Ack: Acknowledgement: Many slides were adapted from Prof. Anthony D. Joseph's CS162 Operating Systems and Systems Programming.

Review: Modern I/O Systems



Review: Modern I/O Systems

- how to talk to devices
  - ✓ memory-mapped I/O or programmed I/O
- how to get events
  - ✓ polling or interrupt
- how to transfer lots of data?

Communication Interface

Q: How does ~~program~~ ~~OS~~ code talk to device?

A: special instructions to talk over special busses

Programmed I/O ← Interact with cmd, status, and data device registers directly

- inb $a, 0x64 ← kbd status register
- outb $a, 0x60 ← kbd data register
- Specifies: device, data, direction
- Protection: only allowed in kernel mode

Kernel boundary crossinging is expensive

*x86: $a implicit; also inw, outw, inh, outh, …

Communication Interface

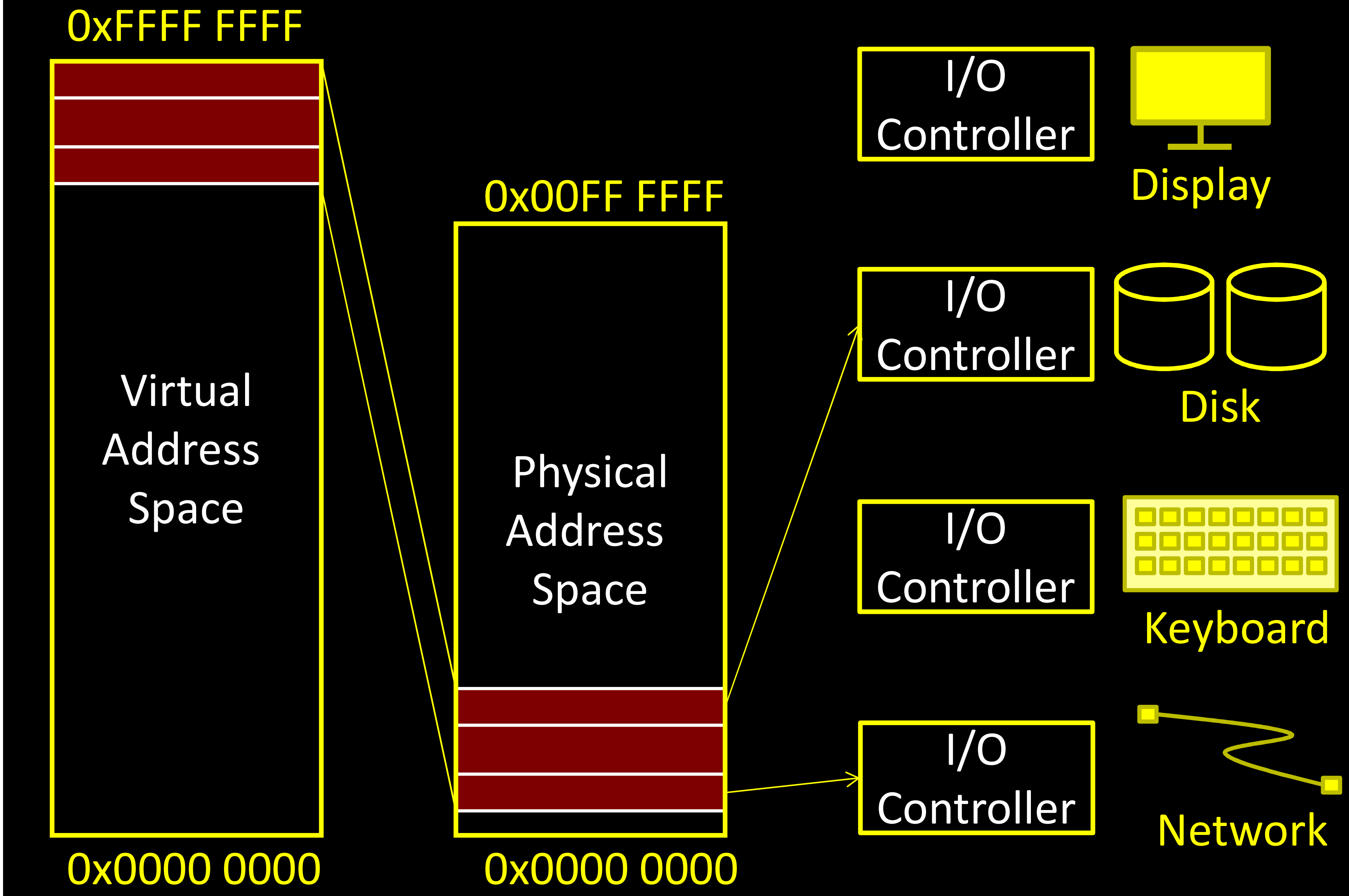
Q: How does ~~program~~ ~~OS~~ code talk to device?

A: Map registers into virtual address space

Memory-mapped I/O ← Faster. Less boundary crossing

- Accesses to certain addresses redirected to I/O devices
- Data goes over the memory bus
- Protection: via bits in pagetable entries
- OS+MMU+devices configure mappings

Memory-Mapped I/O



Comparing Programmed I/O vs Memory Mapped I/O

Programmed I/O

- Requires special instructions
- Can require dedicated hardware interface to devices
- Protection enforced via kernel mode access to instructions
- Virtualization can be difficult

Memory-Mapped I/O

- Re-uses standard load/store instructions
- Re-uses standard memory hardware interface
- Protection enforced with normal memory protection scheme
- Virtualization enabled with normal memory virtualization scheme

Communication Method

Q: How does program learn device is ready/done?

A: Polling: Periodically check I/O status register

- If device ready, do operation
- If device done, …
- If error, take action

```
char read_kbd()
{
do {
  sleep();
  status = inb(0x64);
 } while(!(status & 1));

 return inb(0x60);  }
```

Pro? Con?

- Predictable timing & inexpensive
- But: wastes CPU cycles if nothing to do
- Efficient if there is always work to do (e.g. 10Gbps NIC)

Common in small, cheap, or real-time embedded systems

Sometimes for very active devices too…

Communication Method

Q: How does program learn device is ready/done?

A: Interrupts: Device sends interrupt to CPU

- Cause register identifies the interrupting device
- interrupt handler examines device, decides what to do

Priority interrupts

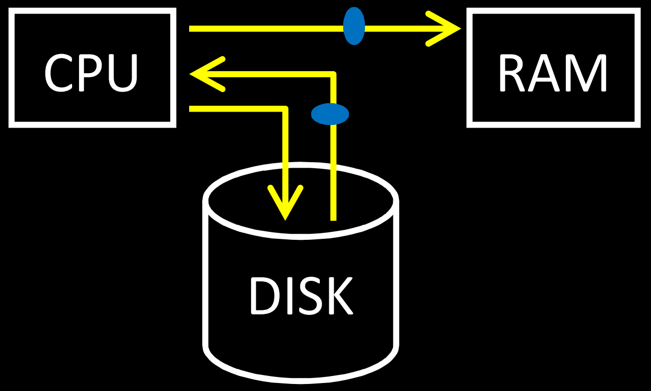
- Urgent events can interrupt lower-priority interrupt handling
- OS can ~~disable~~ defer interrupts

Pro? Con?

- More efficient: only interrupt when device ready/done
- Less efficient: more expensive since save CPU context
  - CPU context: PC, SP, registers, etc
- Con: unpredictable b/c event arrival depends on other devices' activity

DMA: Direct Memory Access

Programmed I/O xfer: Device ←→ CPU ←→ RAM



CPU

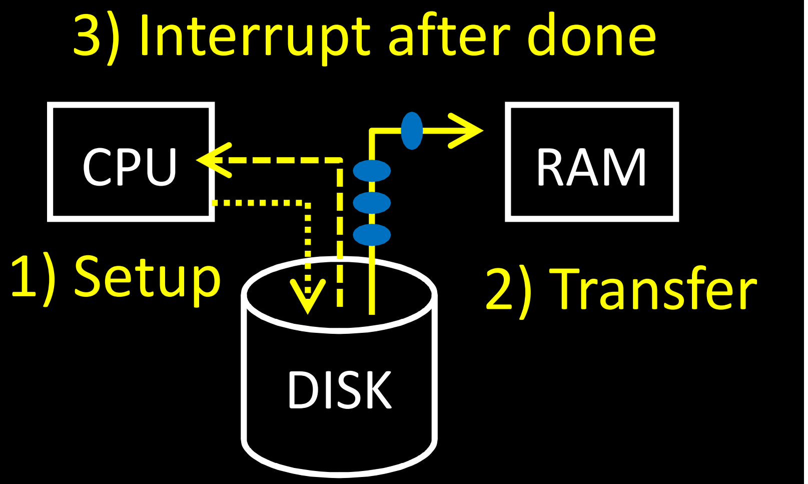
RAM

DISK

for (i = 1 .. n)

- CPU issues read request
- Device puts data on bus
  & CPU reads into registers
- CPU writes data to memory

DMA xfer: Device ←→ RAM



3) Interrupt after done

CPU

RAM

1) Setup

2) Transfer

DISK

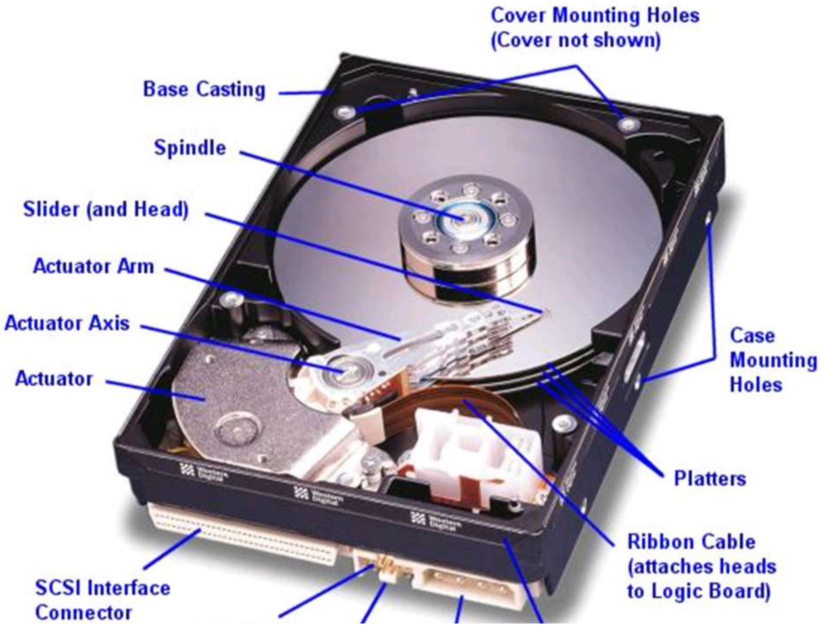
- CPU sets up DMA request
- for (i = 1 ... n)
  Device puts data on bus
  & RAM accepts it
- Device interrupts CPU after done

Storage Devices as Limiting Factor

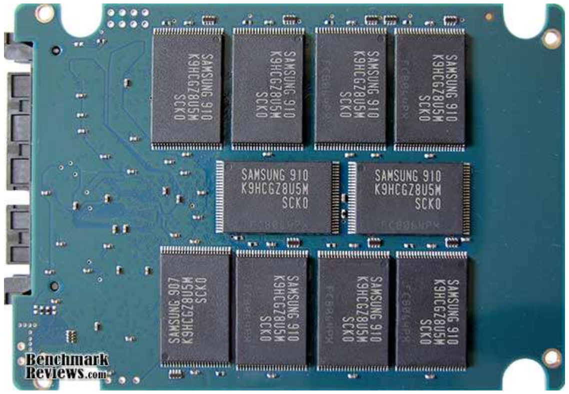
- CPU performance
  - ✓ 60% improvement every year
- storage I/O subsystem performance
  - ✓ less than10% improvement each year due to limitations imposed by the mechanical components
- storage I/O bottleneck w/ Amdahl's Law
  - ✓ diminished improvement introduced by increased RAM and CPU speed to the overall system performance –"only as strong as your weakest link"
- full understanding of SSDs
  - ✓ one must be knowledgeable of the current generation of platter based HDDs
  - ✓ SSDs mirror the functionality of the existing standard of HDDs

SSD versus HDD

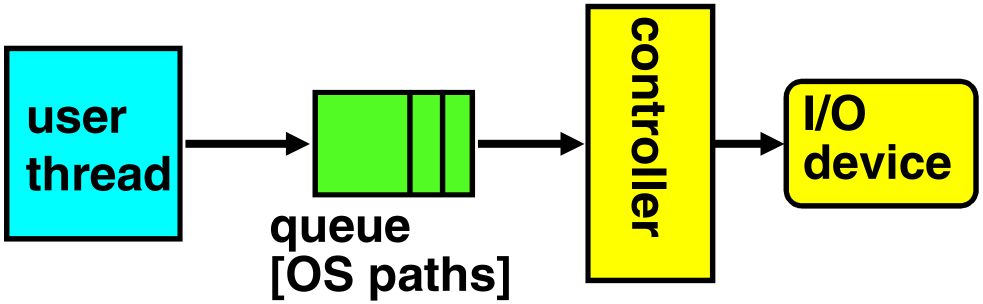
- same form factors
  - ✓ 1.8'', 2.5'', and 3.5''
- interface –SATA, PCIe, SCSI
  - ✓ computer doesn't necessarily know which is connected

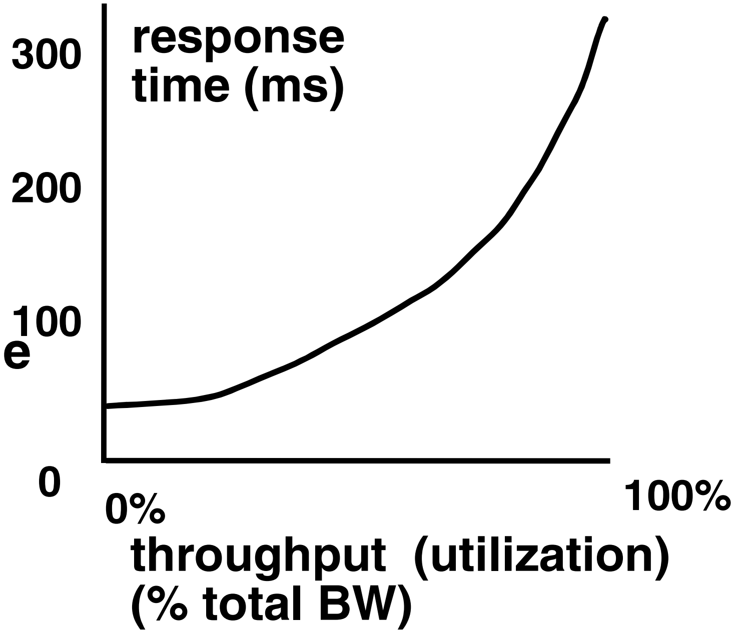


Western Digital Drive
http://www.storagereview.com/guide/



Storage I/O Performance

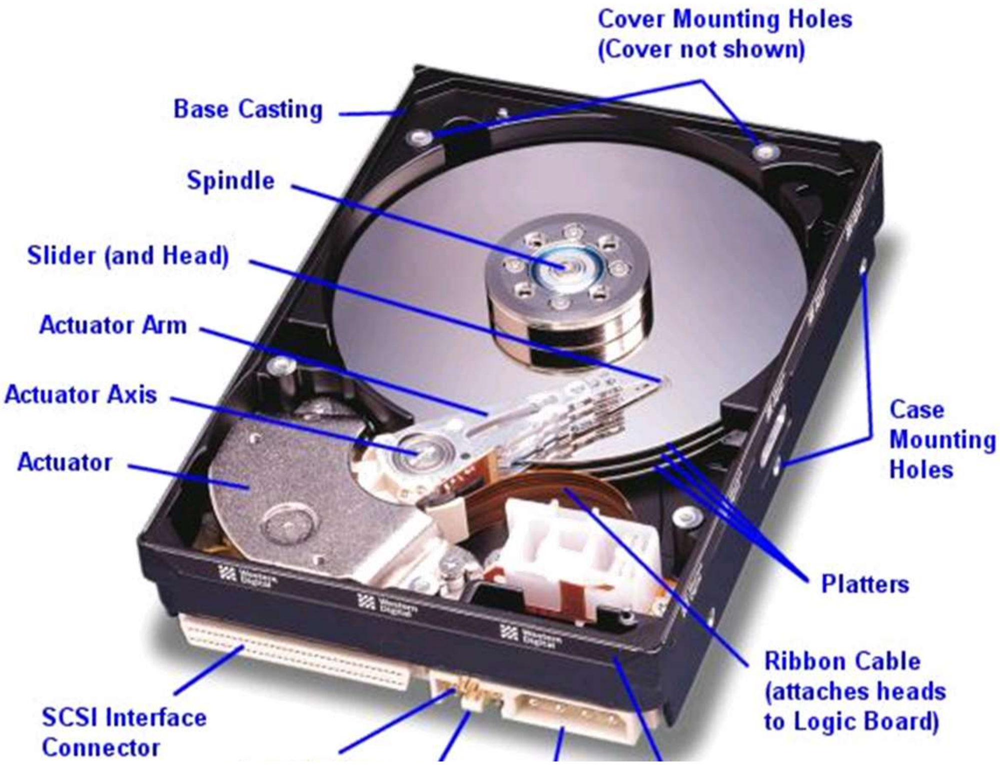




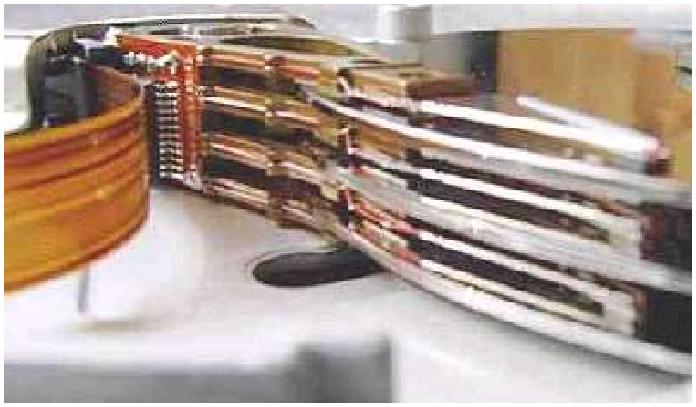
response time = queue + I/O device service time

- performance of I/O subsystem
  - ✓ metrics: response time, throughput
  - ✓ contributing factors to latency:
    - ○ software paths (can be loosely modeled by a queue)
    - ○ hardware controller
    - ○ I/O device service time
- queuing behavior:
  - ✓ can lead to big increases of latency as utilization approaches 100%
  - ✓ solutions?

Hard Disk Drives (HDDs)

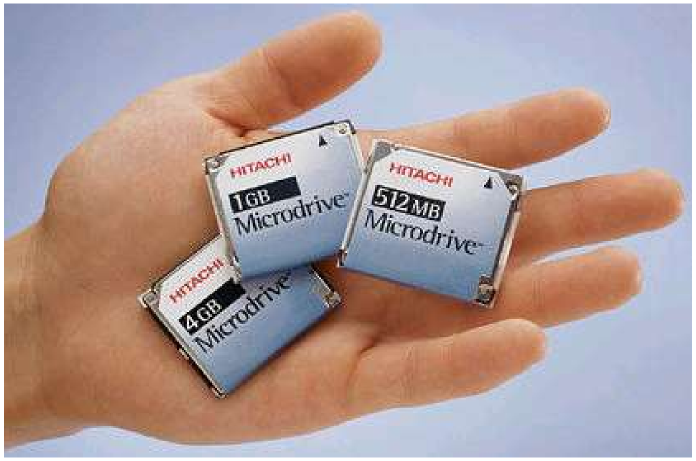


Western Digital Drive
http://www.storagereview.com/guide/



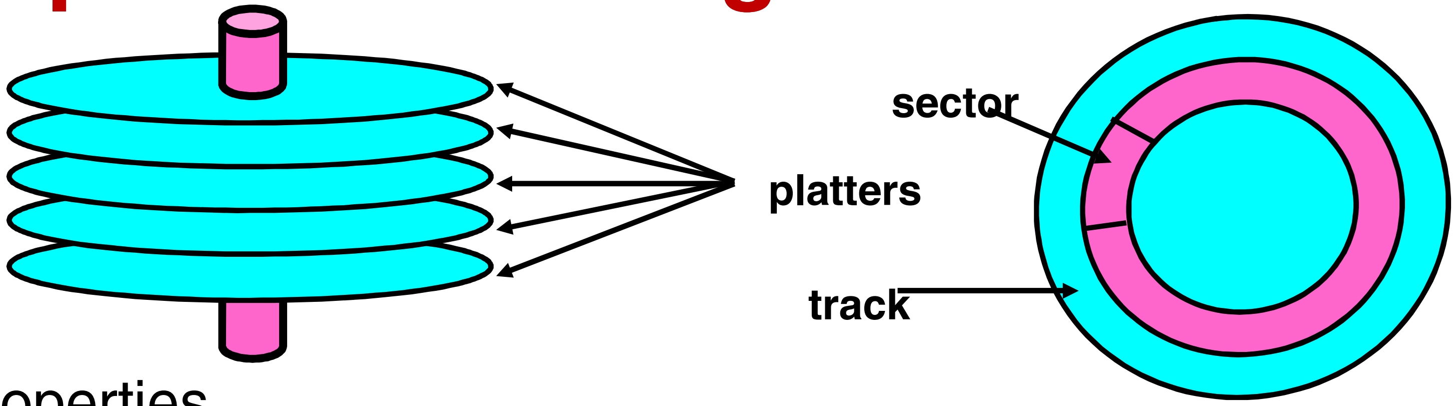
Read/Write Head
Side View

IBM Personal Computer/AT (1986)
- 30 MB hard disk - $500
- 30-40ms seek time
- 0.7-1 MB/s (est.)



IBM/Hitachi Microdrive

Properties of a Magnetic Hard Disk

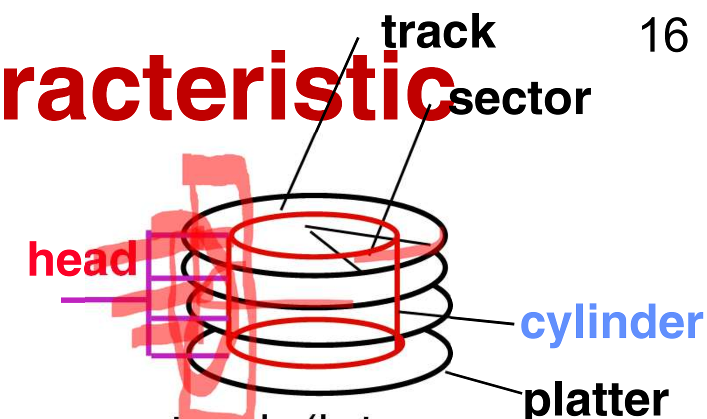


- properties
  - ✓ independently addressable element: sector
    - ○ OS always transfers groups of sectors together—“blocks”
  - ✓ a disk can access directly any given block either sequentially or randomly.

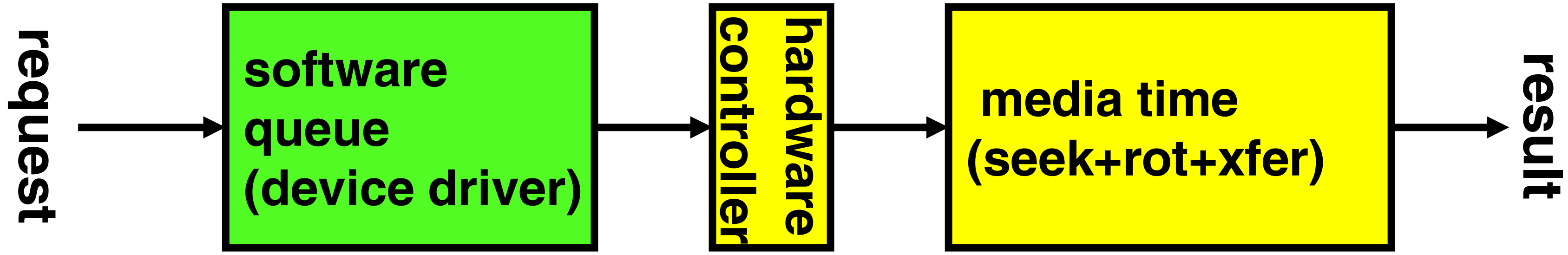
- typical numbers (depending on the disk size):
  - ✓ 500 to more than 20,000 tracks per surface
  - ✓ 32 to 800 sectors per track

- zoned bit recording
  - ✓ constant bit density: more bits (sectors) on outer tracks
  - ✓ speed varies with track location

Magnetic Disk Characteristic



- cylinder: all the tracks under the head at a given point on all surfaces
- read/write: three-stage process:
  - ✓ **seek time**: position the head/arm over the proper track (into proper cylinder)
  - ✓ **rotational latency**: wait for the desired sector to rotate under the read/write head
  - ✓ **transfer time**: transfer a block of bits (sector) under the read-write head
- disk latency = queuing time + controller time + seek time + rotation time + xfer time



- highest bandwidth:
  - ✓ transfer large group of blocks sequentially from one track SATA

Typical Numbers of a Magnetic Disk

parameter	info / range
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 yielding corresponding times of **8-4 milliseconds**
controller time	depends on controller hardware
transfer time	**typically 50 to 100 MB/s.** PCIe 1Gb/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	drops by a factor of two every 1.5 years (or even faster). **$0.03-0.07/GB in 2013**

Disk Performance Examples

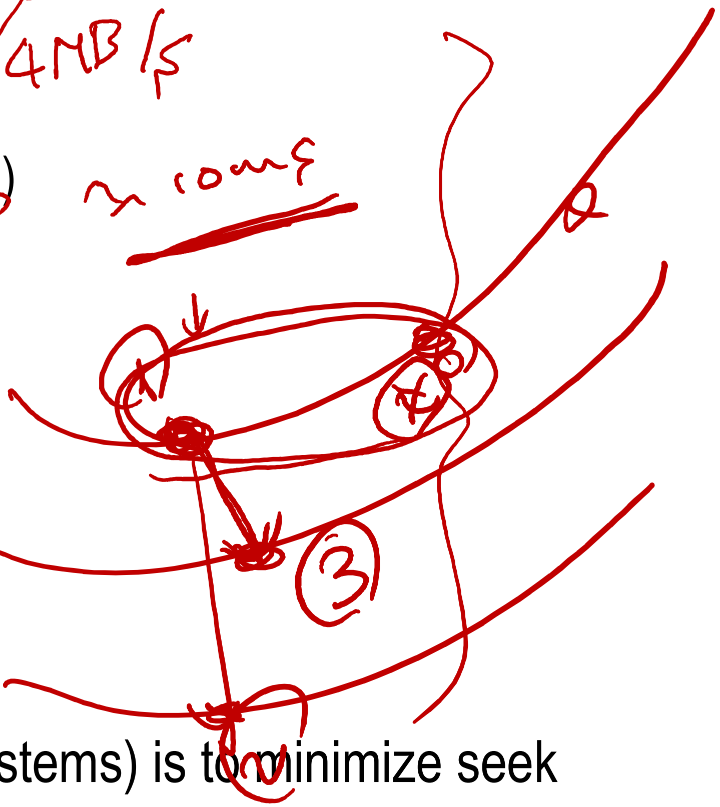
- assumptions:
  - ✓ ignoring queuing and controller times for now
  - ✓ avg seek time of 5ms,
  - ✓ 7200RPM ⇒ time for one rotation: 60000ms/7200 ~= 8ms
  - ✓ transfer rate of 4MByte/s, sector size of 1 KByte

4MB/s

- read sector from random place on disk:
  - ✓ seek (5ms) + rot. delay (4ms) + transfer (0.25ms)
  - ✓ approx 10ms to fetch/put data: 100 KByte/sec

~ 10ms

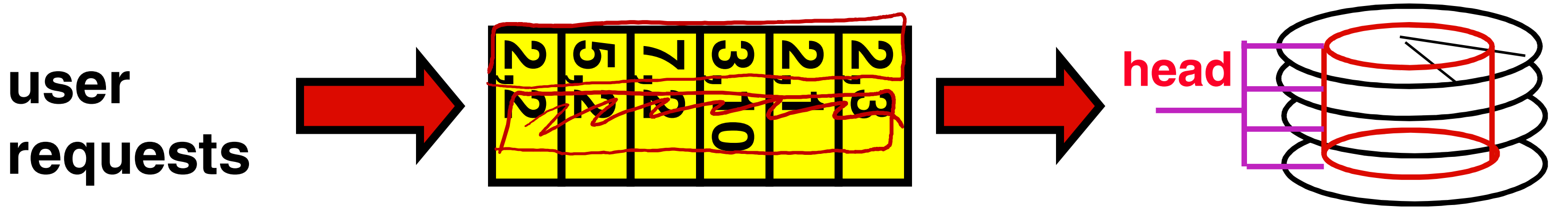
- read sector from random place in same cylinder:
  - ✓ rot. delay (4ms) + transfer (0.25ms)
  - ✓ approx 5ms to fetch/put data: 200 KByte/sec
- read next sector on same track:
  - ✓ transfer (0.25ms): 4 MByte/sec



- key to using disk effectively (especially for file systems) is to minimize seek and rotational delays

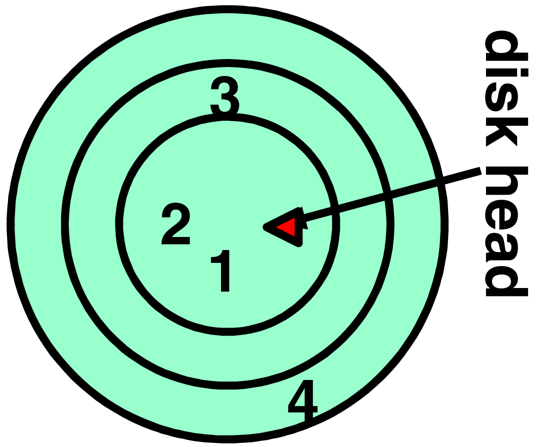
Disk Scheduling

- disk can do only one request at a time; what order do you choose to do queued requests?
  - ✓ request denoted by (track, sector)



- scheduling algorithms:
  - ✓ First In First Out (FIFO)
  - ✓ Shortest Seek Time First
  - ✓ SCAN
  - ✓ C-SCAN

- In our examples we will ignore the sector
  - ✓ consider only track #



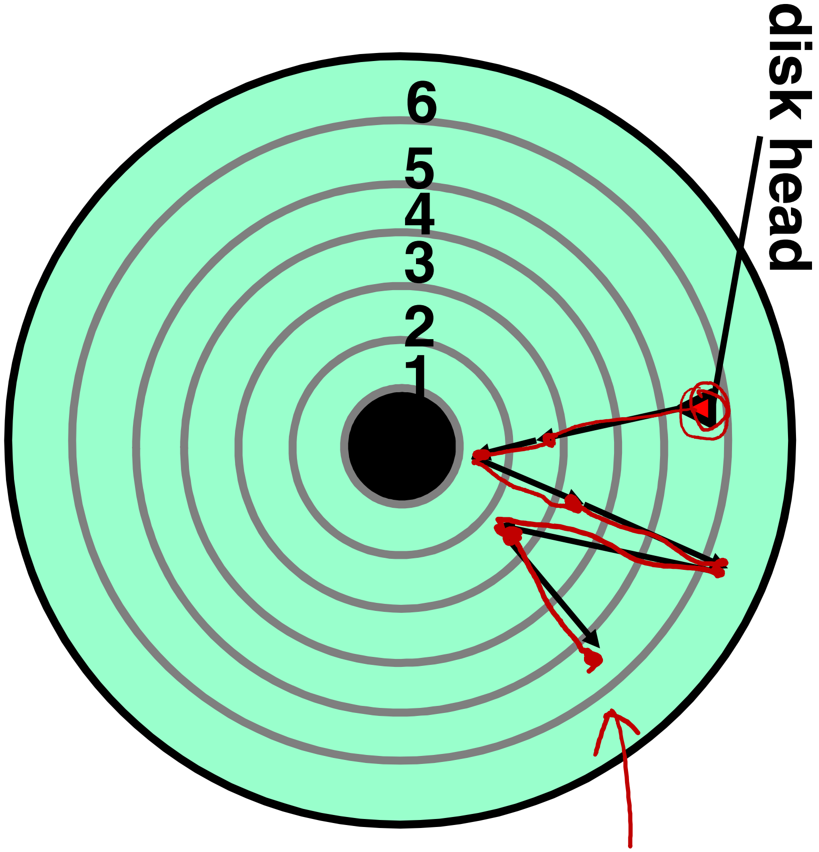
FIFO: First In First Out

- schedule request in the order they arrive in the queue

- example:
  - ✓ request queue: 2, 1, 3, 6, 2, 5
  - ✓ scheduling order: 2, 1, 3, 6, 2, 5

  - pros: fair among requesters

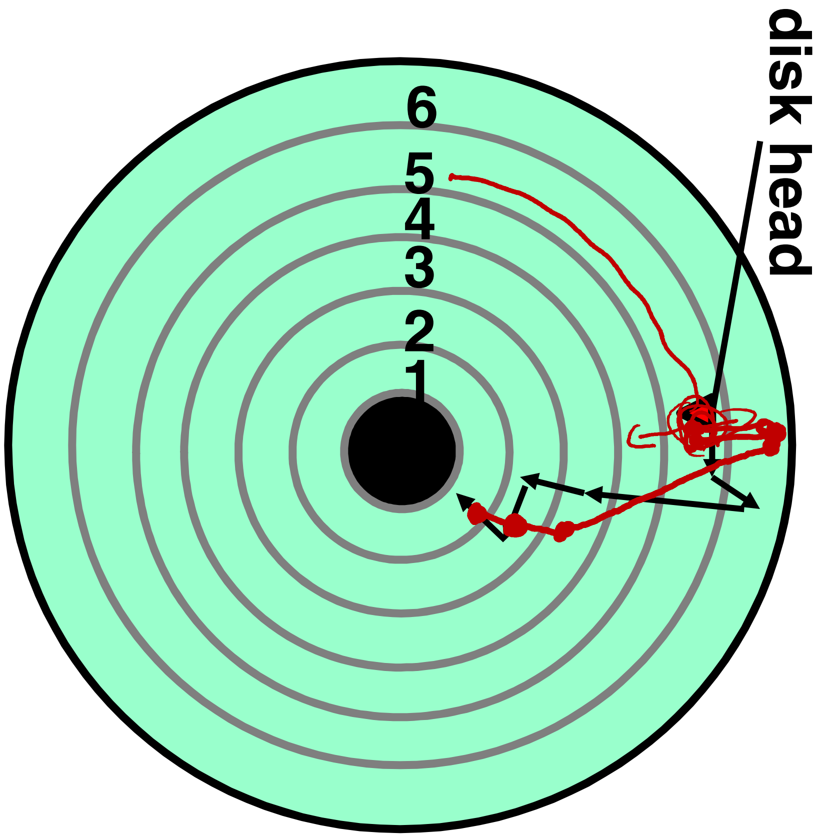
  - cons: order of arrival may be to random spots on the disk ⇒ very long seeks



SSTF: Shortest Seek Time First

- pick the request that's closest to the head on the disk
  - ✓ although called SSTF, include rotational delay in calculation, as rotation can be as long as seek
- example:
  - ✓ request queue: 2, 1, 3, 6, 2, 5
  - ✓ scheduling order: 5, 6, 3, 2, 2, 1
- pros: reduce seeks
- cons: may lead to starvation

fairness



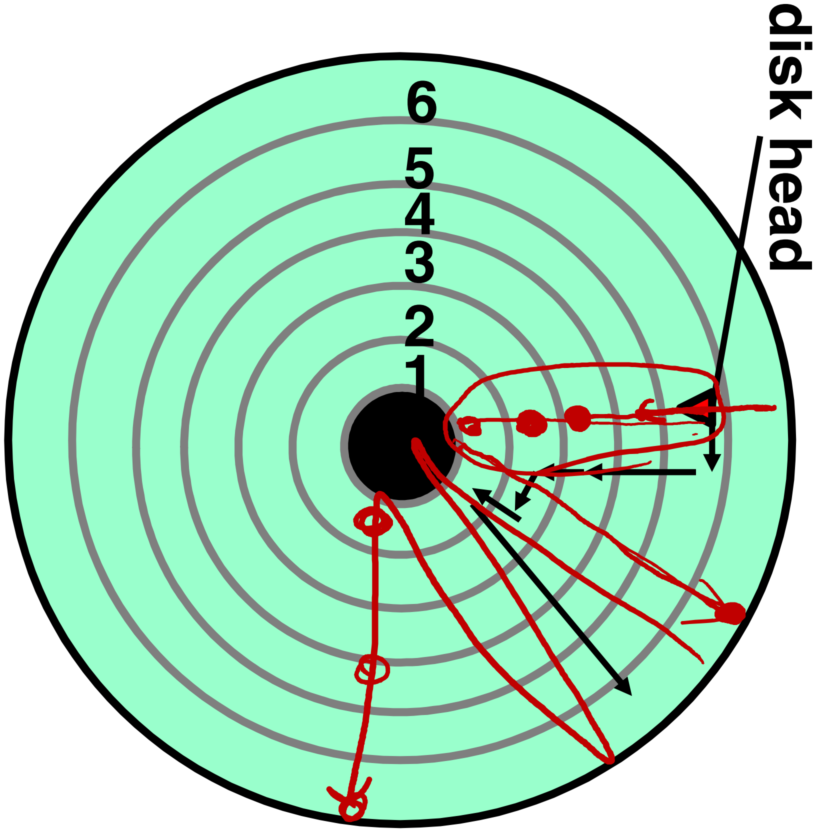
SCAN

- implements an elevator algorithm: take the closest request in the direction of travel

- example:
  - ✓ request queue: 2, 1, 3, 6, 2, 5
  - ✓ head is moving towards center
  - ✓ scheduling order: 5, 3, 2, 2, 1, 6

- pros:
  - ✓ no starvation
  - ✓ low seek

- cons: favor middle tracks



Logical Block Addressing (LBA)

- a way of addressing blocks that simply numbers them linearly rather than providing a cylinder, head, and sector number
- HDD
  - ✓ sectors of concentric circles in HDDs
- SSD
  - ✓ are actually inside of the flash media in SSD
  - ✓ represent individual addresses

Interface

- Serial ATA (SATA)
  - ✓ serial: 4 Pin + grounds
  - ✓ reduced 40 pins of parallel ATA (PATA) to 4 pins to increase speed
- Speeds (Maximum)
  - ✓ ~~PATA: 133 MB/s~~
  - ✓ ~~SATA-I: 125 MB/s~~
  - ✓ ~~SATA-II: 250 MB/s~~
  - ✓ ~~SATA-III: 500 MB/s~~

SSD

SSh

PCIe

NVMe

Interface

- PCI Express (PCIe) v2.0 (3.0)
  - ✓ 500 MB/s (1GB/s) per lane
  - ✓ up to 16 lanes w/ very low power and broad hardware support
- PCIe SSD
  - ✓ only for extremely high performance applications due to their enormous cost
- NVMe
  - ✓ …

Solid State Disks (SSDs)

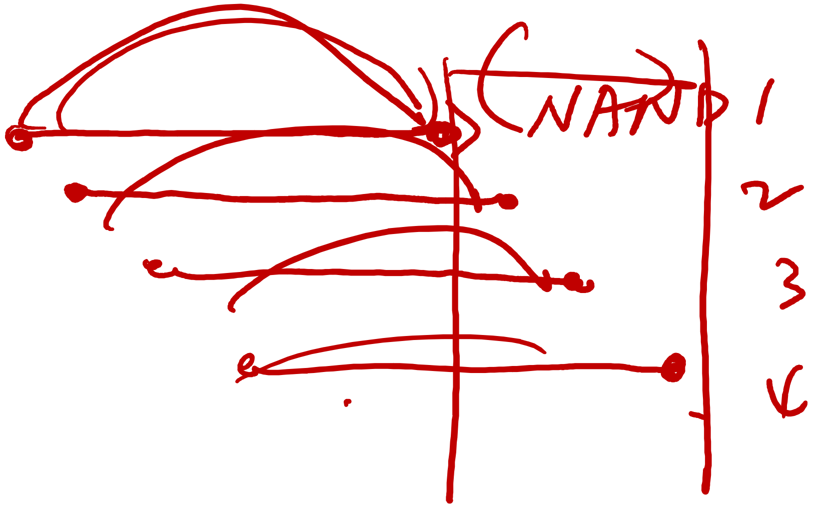
- 1995 – replace rotating magnetic media with non-volatile memory (battery backed DRAM)

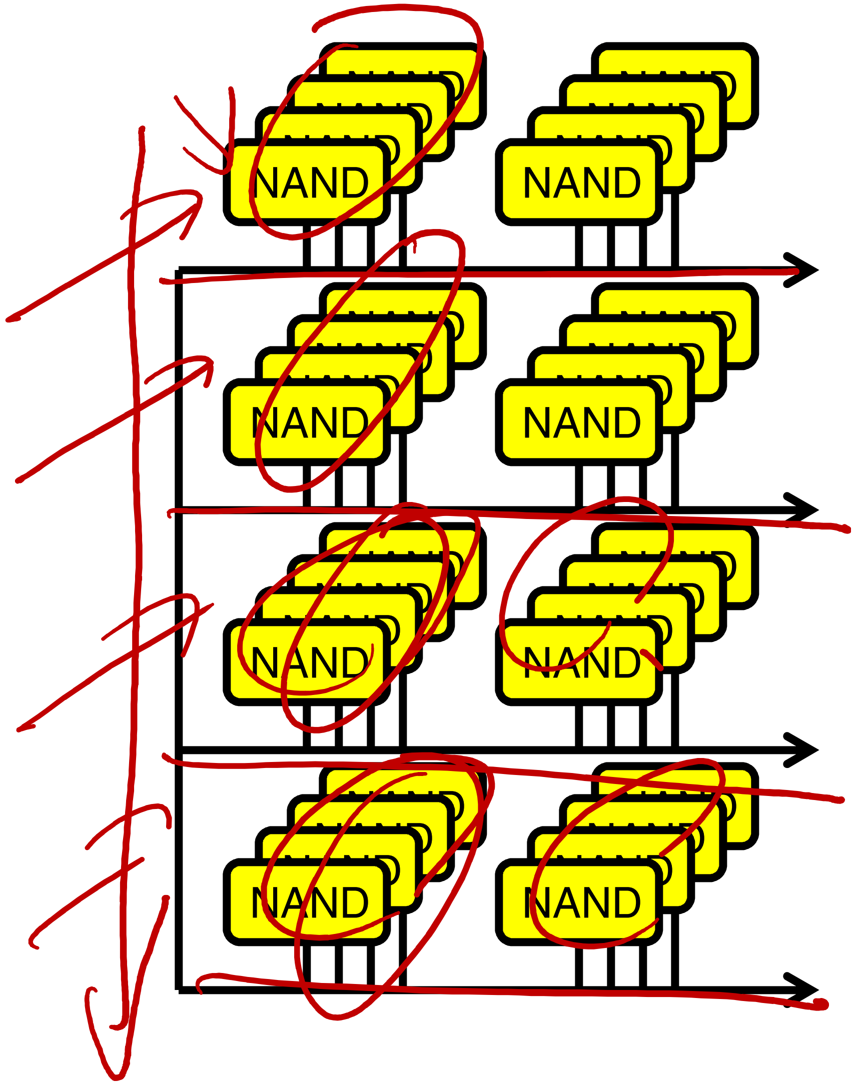
MLC

- 2009 – use NAND multi-level cell (2-bit/cell) flash memory
  - ✓ sector (4 KB page) addressable, but stores 4-64 “pages” per memory block
- no moving parts (no rotate/seek motors)
  - ✓ eliminates seek and rotational delay (0.1-0.2ms access time)
  - ✓ very low power and lightweight

SSD Architecture

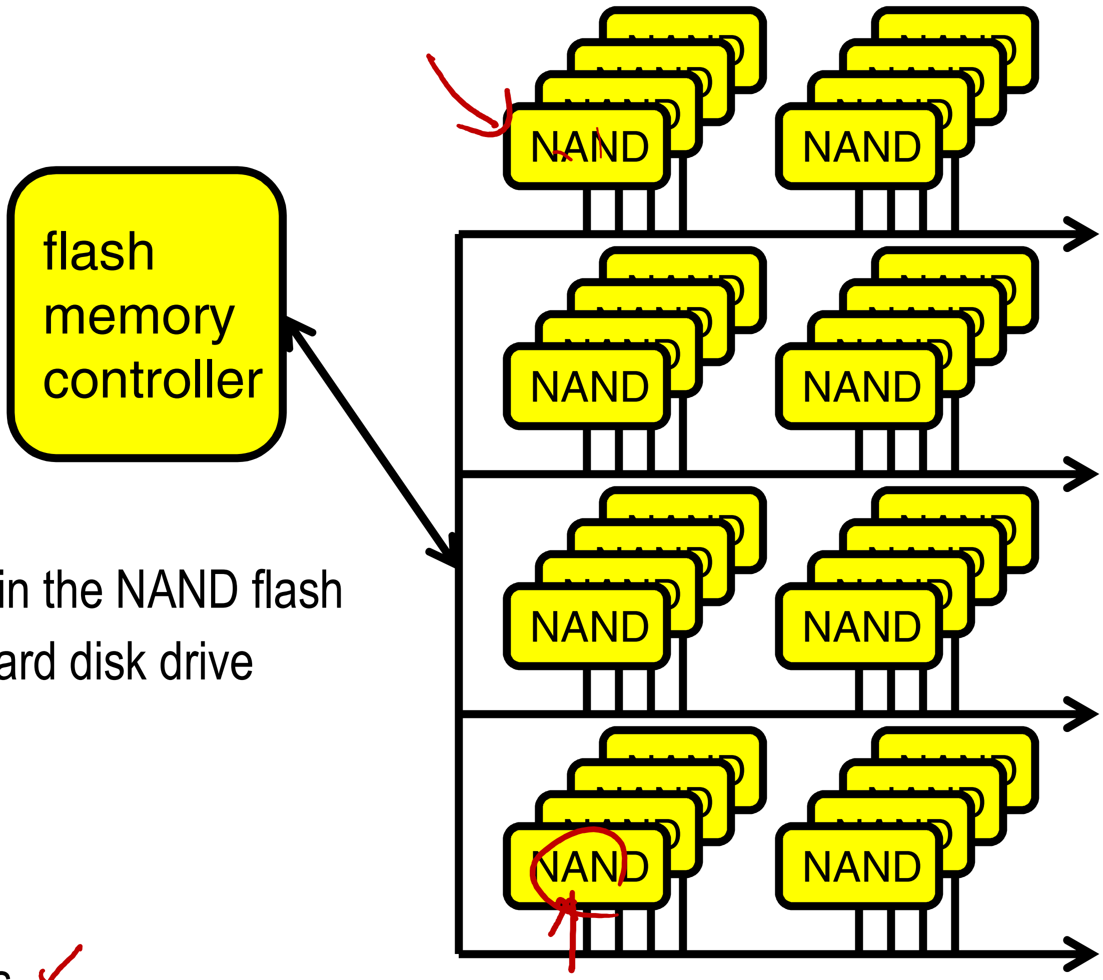
- many NAND flash devices
  - ✓ 10 to upwards of 60 or 70





SSD Architecture

- many NAND flash devices
  - ✓ 10 to upwards of 60 or 70
- controller
  - ✓ takes the raw data storage in the NAND flash
  - ✓ makes it look and act like hard disk drive
  - ✓ contains
    - ○ micro controller
    - ○ DRAM buffer
    - ○ error correction
    - ○ flash interface modules



SSD Controller Components

- flash interface modules (FIMs)
  - ✓ physically and logically connect the controller to the NAND flash devices
  - ✓ communicate w/ multiple NAND flash devices
    - ○ higher performance w/ more FIMs
- micro-controller, a processor inside the controller
  - ✓ takes the incoming data and manipulates it
    - ○ ECC
    - ○ LBA to physical address mapping
    - ○ putting it into the flash or retrieving it from the flash
- DRAM cache
  - ✓ reasonable amount of very low latency memory that gives the processor some r oom to work
    - ○ data buffer
    - ○ address translation buffer

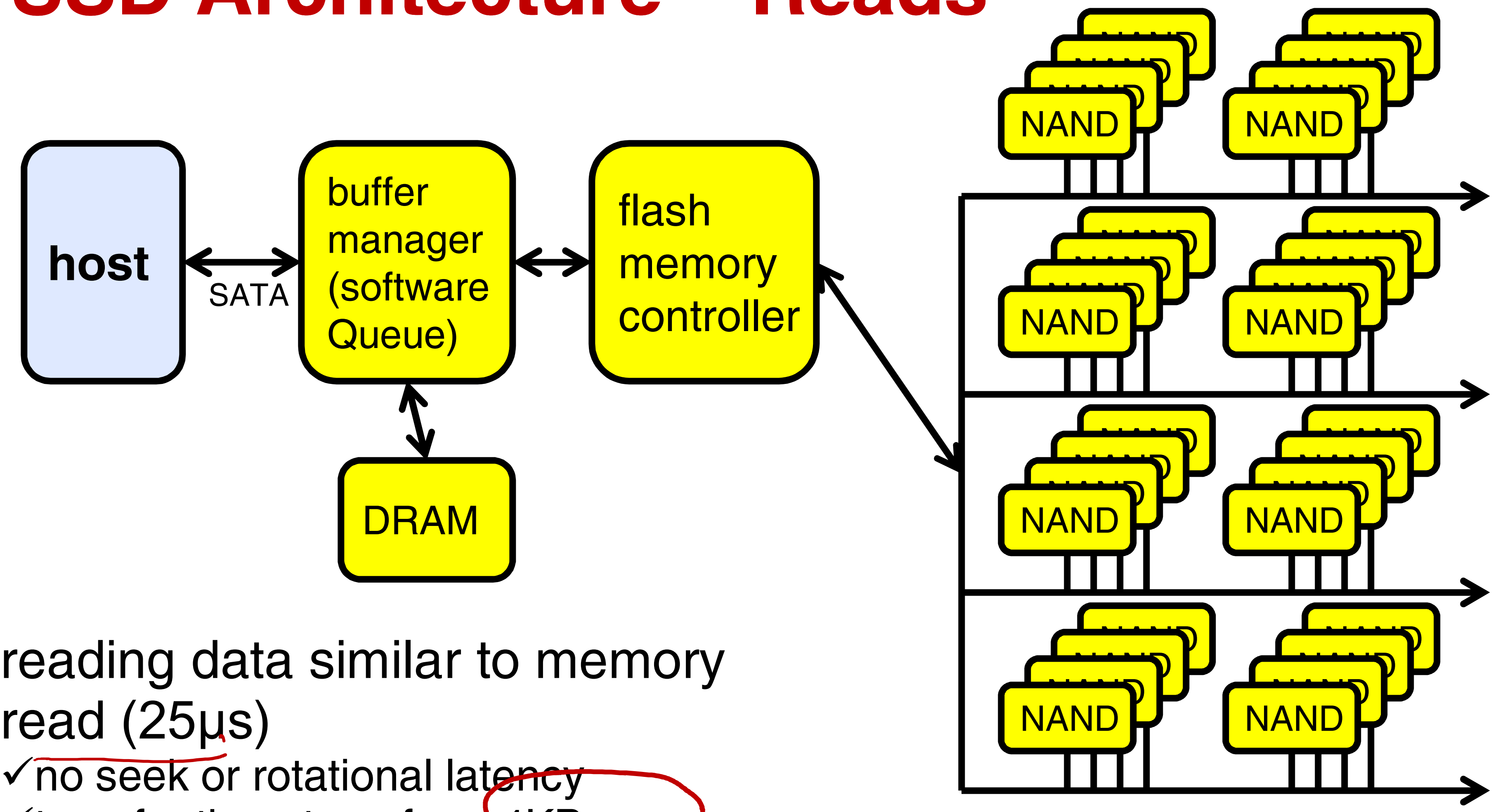
SSD Controller Components

- NAND flash media
  - ✓ NAND cells arranged in multiple planes allowing
    - parallel access to NAND
    - interleaving
  - ✓ data moves in and out through a cache element

SLC, MLC and TLC

- MLC (multi-level cell)
  - ✓ cheaper yet slower and slightly less reliable
- SLC (single level cell)
  - ✓ faster and more reliable, yet more expensive than SLC
  - ✓ advantages of SLC are dwindling due to advancements in controller design, whi ch mitigate the disadvantages of MLC

SSD Architecture – Reads



reading data similar to memory read (25μs)

✓no seek or rotational latency

✓transfer time: transfer a 4KB page

- limited by controller and disk interface (SATA: 300-600MB/s)

✓latency = queuing time + controller time + xfer time

✓highest bandwidth: sequential or random reads

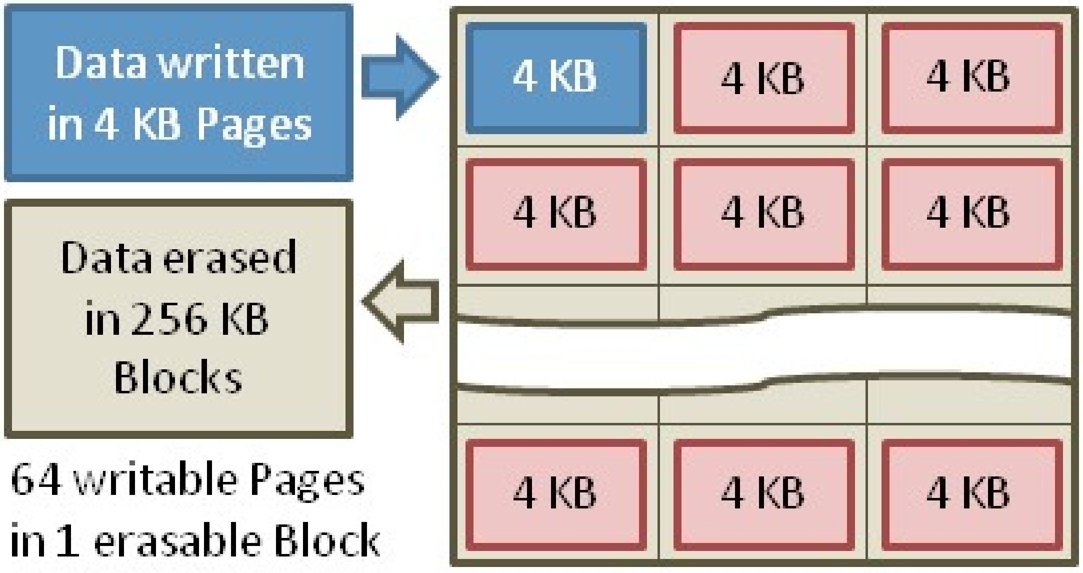
SSD Architecture – Writes (I)

- Writing data is complex! (~200μs – 1.7ms)

✓ Can only write empty pages in a block

✓ Erasing a block takes ~1.5ms

✓ Controller maintains pool of empty blocks, also reserves some % of capacity


Data written in 4 KB Pages
Data erased in 256 KB Blocks
64 writable Pages in 1 erasable Block
4 KB
4 KB
4 KB
4 KB
4 KB
4 KB
4 KB
4 KB
4 KB

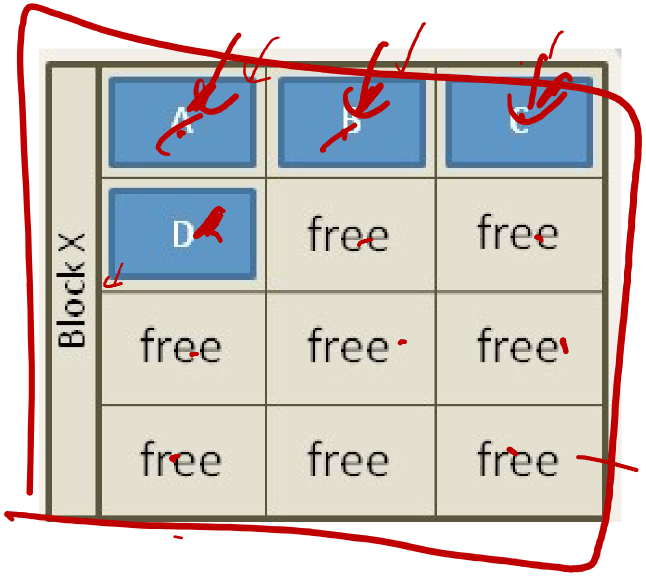
Typical NAND Flash Pages and Blocks

https://en.wikipedia.org/wiki/Solid-state_drive

SSD Architecture – Writes (II)

- Write A, B, C, D

A B C D



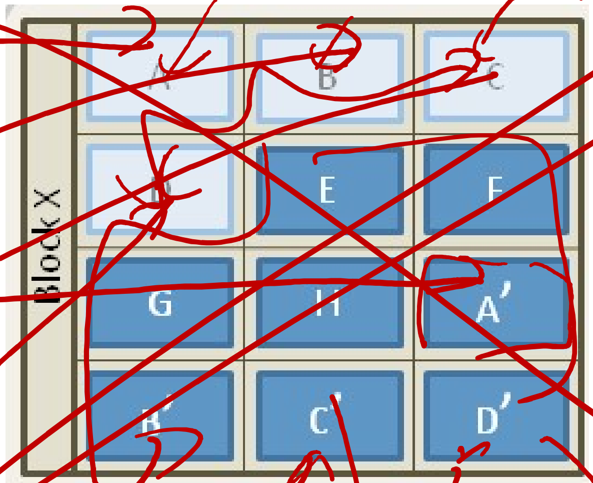
Block X		
A	B	C
D	free	free
free	free	free
free	free	free

https://en.wikipedia.org/wiki/Solid-state_drive

SSD Architecture – Writes (II)

- Write A, B, C, D

- Write E, F, G, H and A’, B’, C’, D’
  - ✓ Record A, B, C, D as obsolete

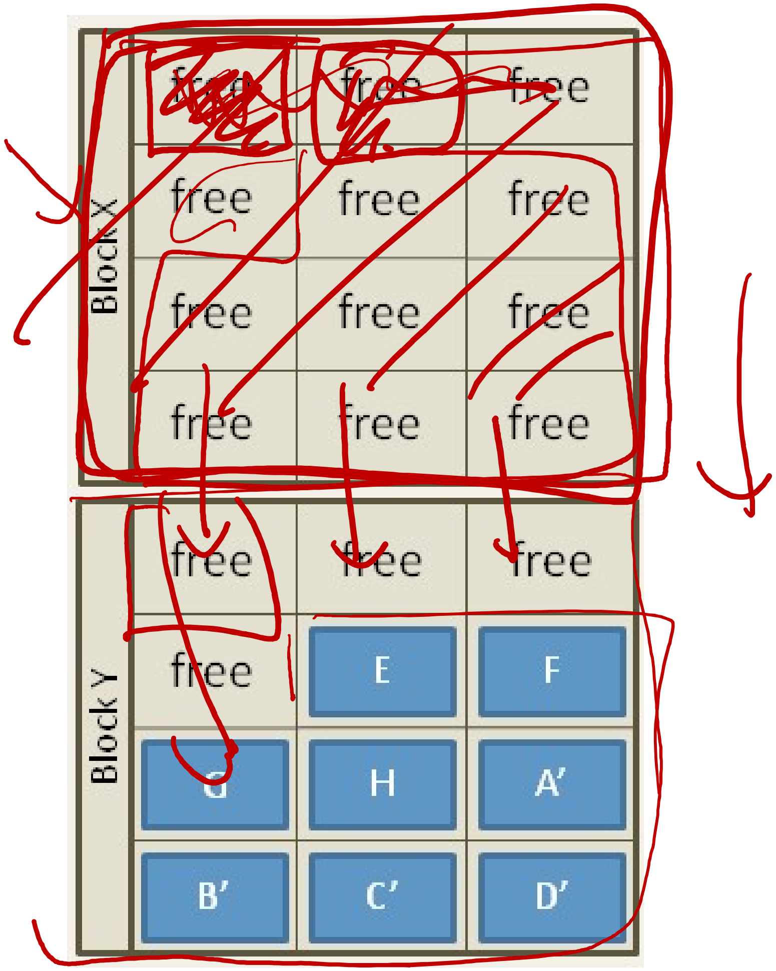


1 2 3 4

https://en.wikipedia.org/wiki/Solid-state_drive

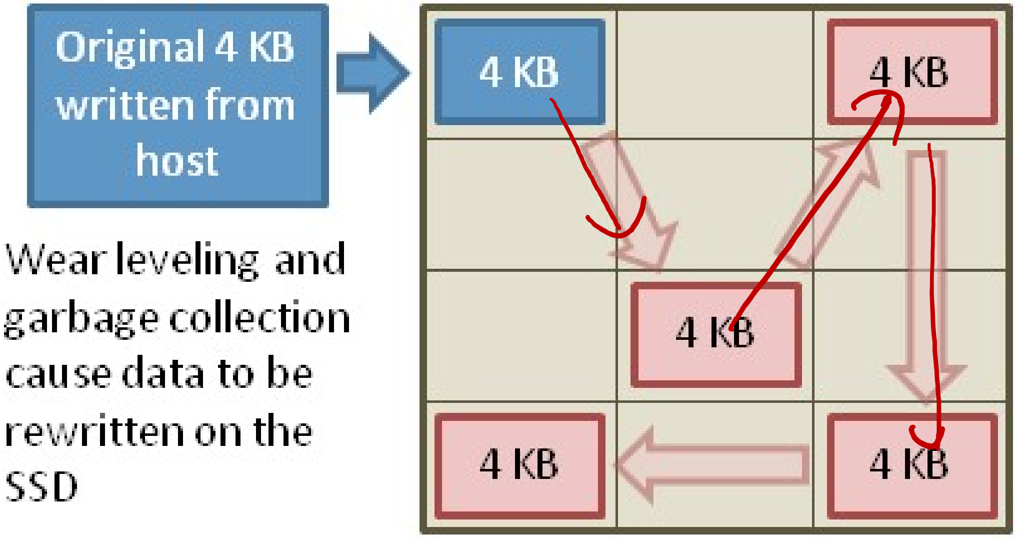
SSD Architecture – Writes (II)

- Write A, B, C, D
- Write E, F, G, H and A', B', C', D'
  - ✓ Record A, B, C, D as obsolete
- Controller *garbage collects* obsolete pages by copying valid pages to new (erased) block
- Typical steady state behavior when SSD is almost full



SSD Architecture – Writes (III)

- Write and erase cycles require “high” voltage
  - ✓ Damages memory cells, limits SSD lifespan
  - ✓ Controller uses ECC, performs wear leveling



Wear leveling and garbage collection cause data to be rewritten on the SSD

- Result i
  - ✓ Latency = Queuing Time + Controller time (Find Free Block) + Xfer Time
  - ✓ Highest BW: Seq. OR Random writes (limited by empty pages)

Rule of thumb: writes 10x more expensive than reads, and erases 10x more expensive than writes

Storage Performance & Price

	Bandwidth (Sequential R/W)	Cost/GB	Size
HDD[2]	50-100 MB/s	$0.03-0.07/GB	2-4 TB
SSD[1,2]	200-550 MB/s (SATA) 6 GB/s (read PCI) 4.4 GB/s (write PCI)	$0.87-1.13/GB	200GB-1TB
DRAM[2]	10-16 GB/s	$4-14*/GB *SK Hynix 9/4/13 fire	64GB-256GB

[1] http://www.fastestssd.com/featured/ssd-rankings-the-fastest-solid-state-drives/

[2] http://www.extremetech.com/computing/164677-storage-pricewatch-hard-drive-and-ssd-prices-drop-making-for-a-good-time-to-buy

BW: SSD up to x10 than HDD, DRAM > x10 than SSD

Price: HDD x20 less than SSD, SSD x5 less than DRAM

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
  - ✓ small storage (0.1-0.5x disk), very expensive (20x disk)
    - ○ Hybrid alternative: combine small SSD with large HDD
  - ✓ asymmetric block write performance: read pg/erase/write pg
    - ○ Controller garbage collection (GC) algorithms have major effect on performance
  - ✓ limited drive lifetime
    - ○ 1-10K writes/page for MLC NAND
    - ○ avg failure rate is 6 years, life expectancy is 9–11 years

Announcement

- today's lecture: storage
  - ✓ Ch. 5.6 – 5.7 (HP1)
- next lecture on 4/5
  - ✓ mid-term 2 review

Virtual memory

to