ECE590-03
Enterprise Storage Architecture

Fall 2016

Hard disks, SSDs, and the I/O subsystem

Tyler Bletsch
Duke University

Slides include material from Vince Freeh (NCSU)

Hard Disk Drives (HDD)

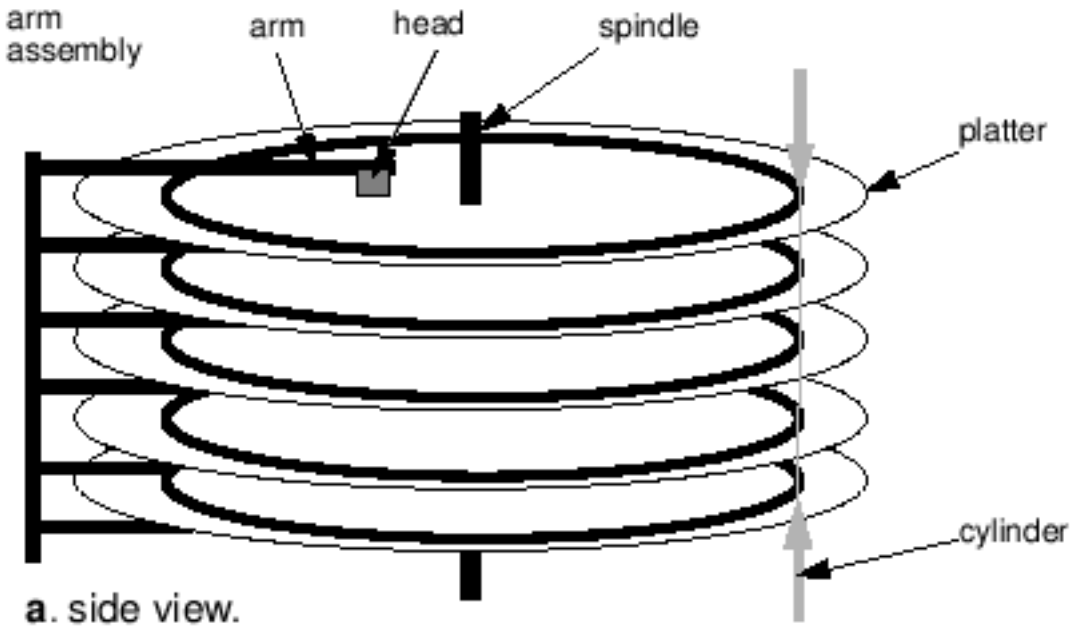
History

- First: IBM 350 (1956)
  - 50 platters (100 surfaces)
  - 100 tracks per surface (10,000 tracks)
  - 500 characters per track
  - 5 million characters
  - 24” disks, 20” high

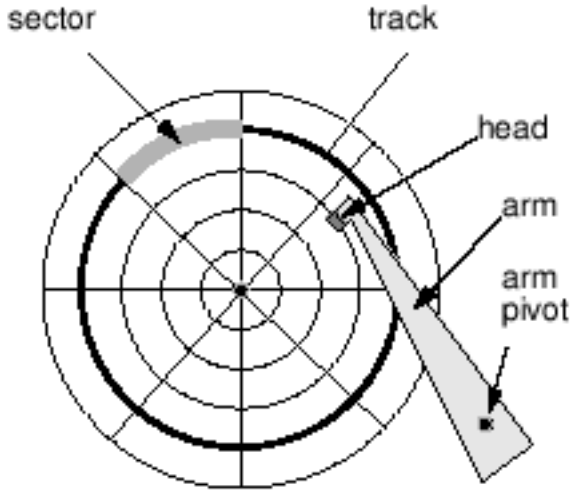
Overview

- Record data by magnetizing ferromagnetic material
- Read data by detecting magnetization
- Typical design
  - 1 or more platters on a spindle
  - Platter of non-magnetic material (glass or aluminum), coated with ferromagnetic material
  - Platters rotate past read/write heads
  - Heads ‘float’ on a cushion of air
  - Landing zones for parking heads

Basic schematic

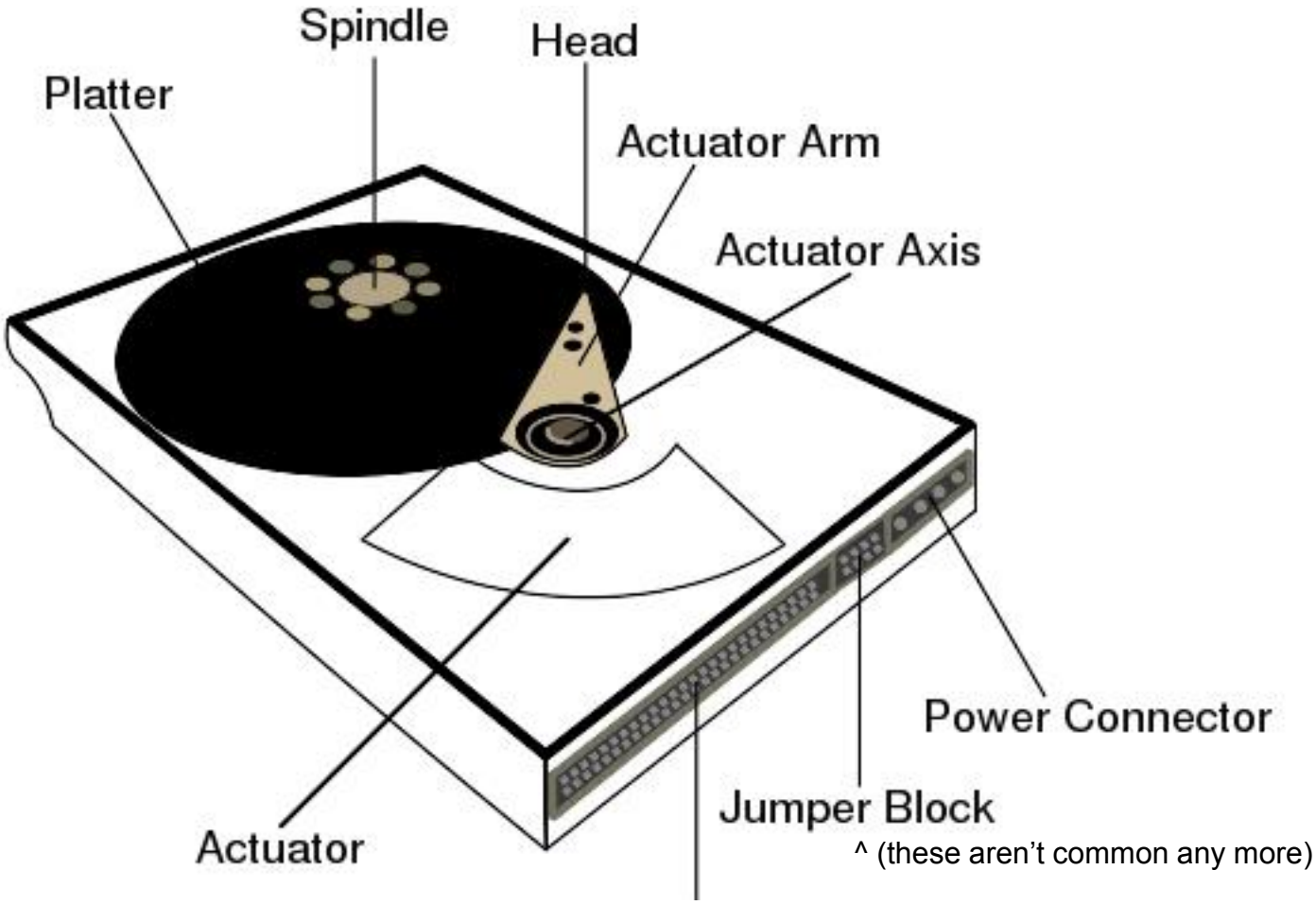


a. side view.



b. top view.

Generic hard drive



^ (these aren't common any more)

Types and connectivity (legacy)

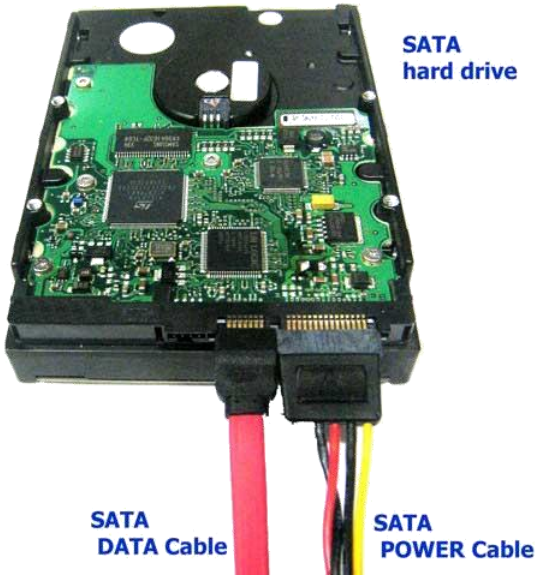
- SCSI (Small Computer System Interface):
  - Pronounced "Scuzzy"
  - One of the earliest small drive protocols
  - Many revisions to standard – many types of connectors!
  - The Standard That Will Not Die: the drives are gone, but most enterprise gear still speaks the SCSI protocol

- Fibre Channel (FC):
  - Used in some Fibre Channel SANs
  - Speaks SCSI on the wire
  - Modern Fibre Channel SANs can use any drives: back-end ≠ front-end

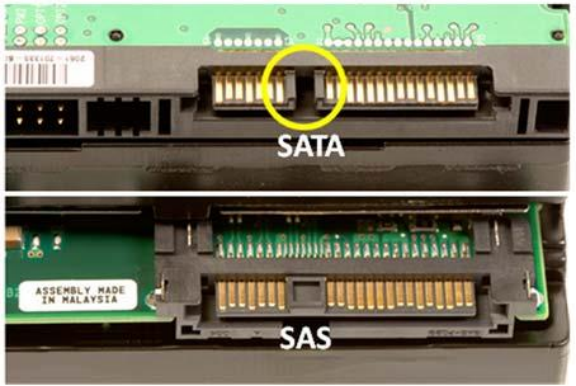
- IDE / ATA:
  - Older standard for consumer drives
  - Obsoleted by SATA in 2003

Types and connectivity (modern)

- SATA (Serial ATA):
  - Current consumer standard
  - Series of backward-compatible revisions
    SATA 1 = 1.5 Gbit/s, SATA 2 = 3 Gbit/s,
    SATA 3 = 6.0 Gbit/s, SATA 3.2 = 16 Gbit/s
  - Data and power connectors are hot-swap ready
  - Extensions for external drives/enclosures (eSATA), small all-flash boards (mSATA, M.2), multi-connection cables (SFF-8484), more
  - Usually in 2.5” and 3.5” form factors



- SAS (Serial-Attached-SCSI)
  - SCSI protocol over SATA-style wires
  - (Almost) same connector
  - Can use SATA drives on SAS controller, not vice versa

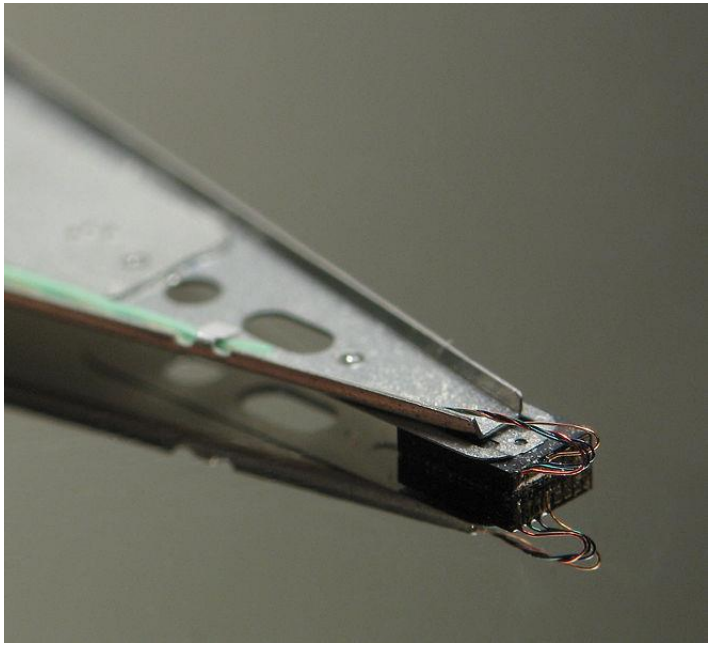


Inside hard drive

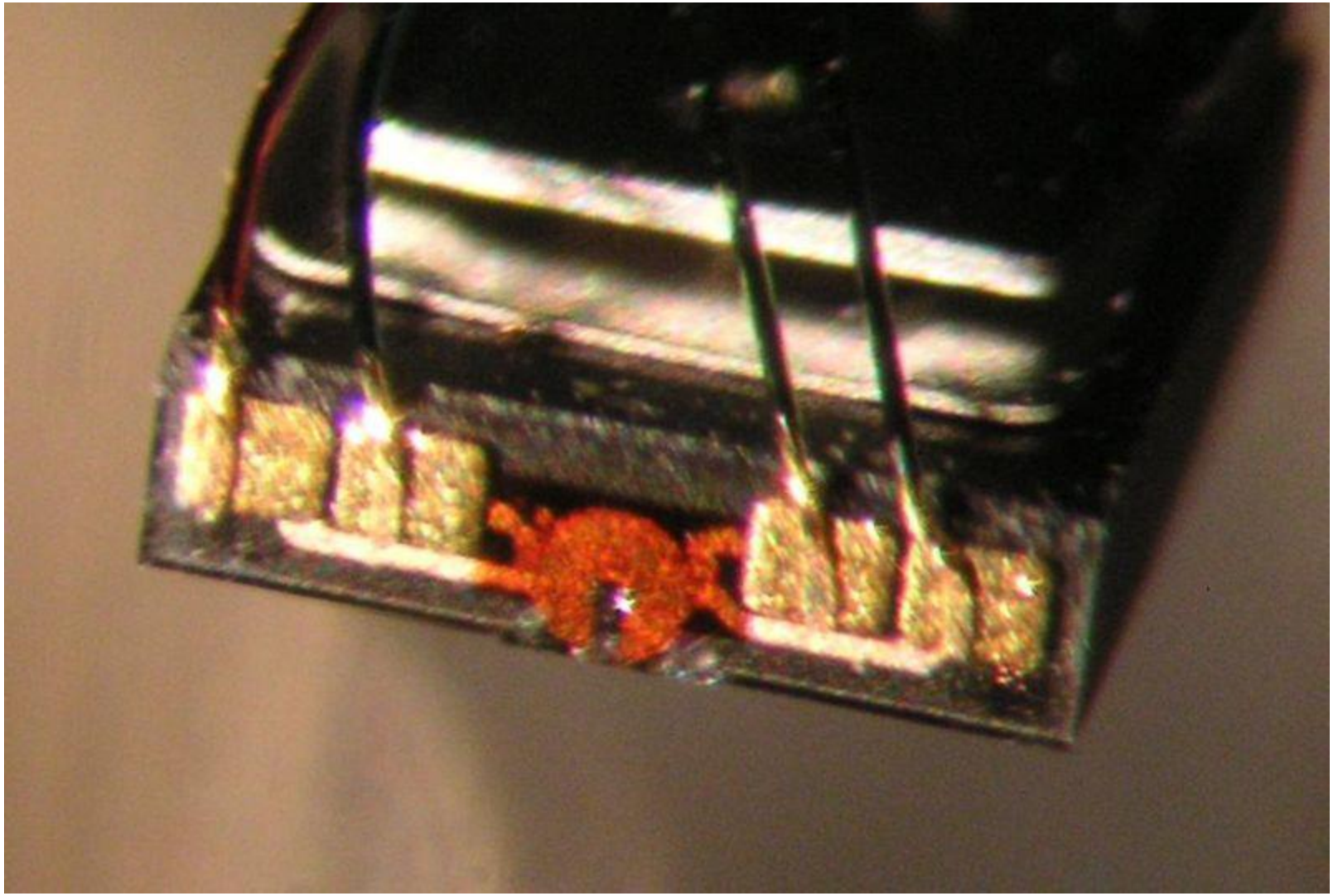
Anatomy


Anatomy of a Hard Disk Drive
Western Digital
Caviar 22100, 2111.8 MB
Closeup of the read-write head
Other side of the frame
Frame
Actuator magnet
Dessicant bag
Hole for the motor
Hole for actuator axis
Cover
(upside down)
Fasteners and other hardware
Disassembled platters
Actuator axis
Actuator
Motor contacts
Read-write head
Sankyo
Spindle motor
Controller
Platter Assembly
(two platters and the motor)

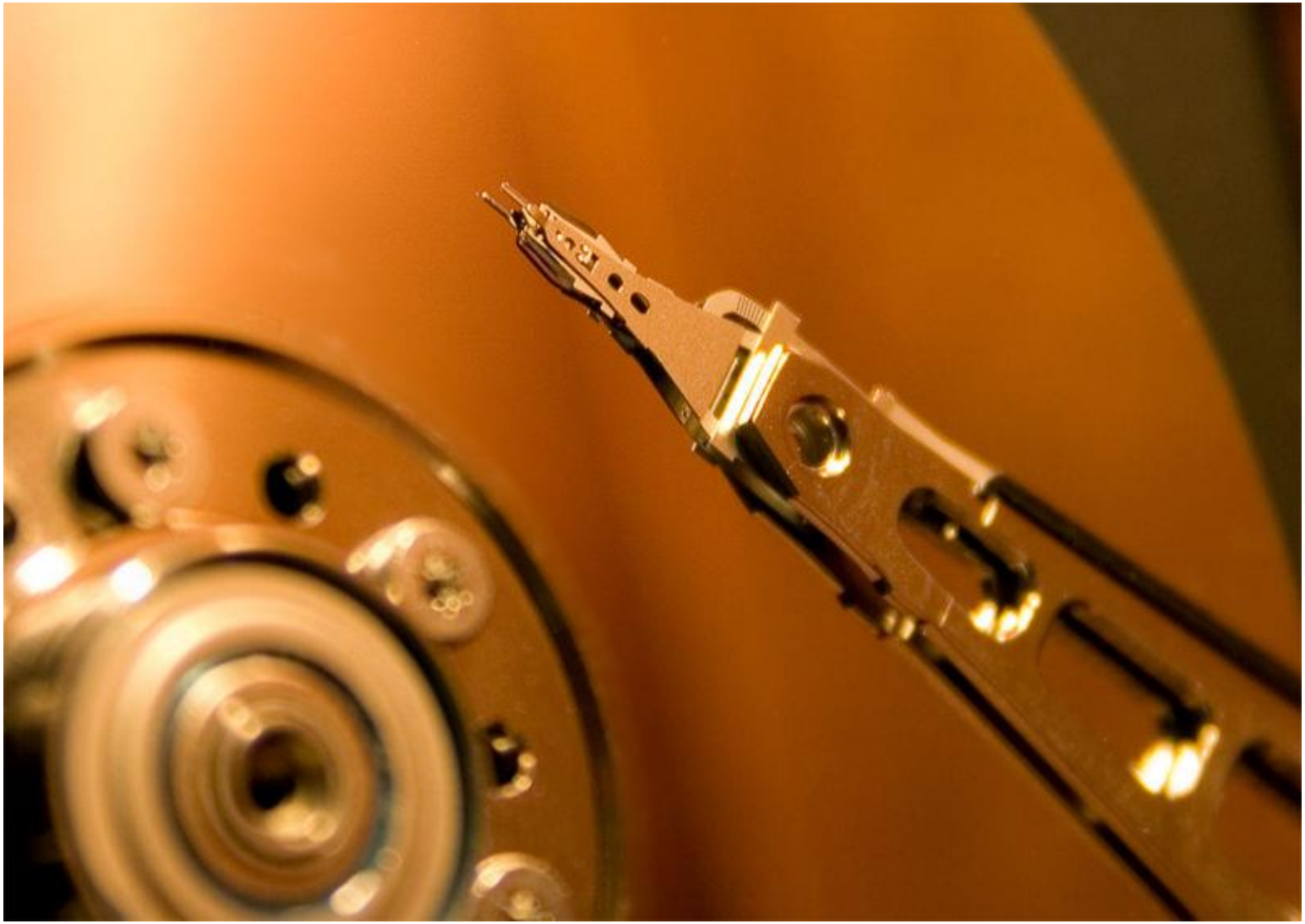
Read/write head



Head close-up



Arm

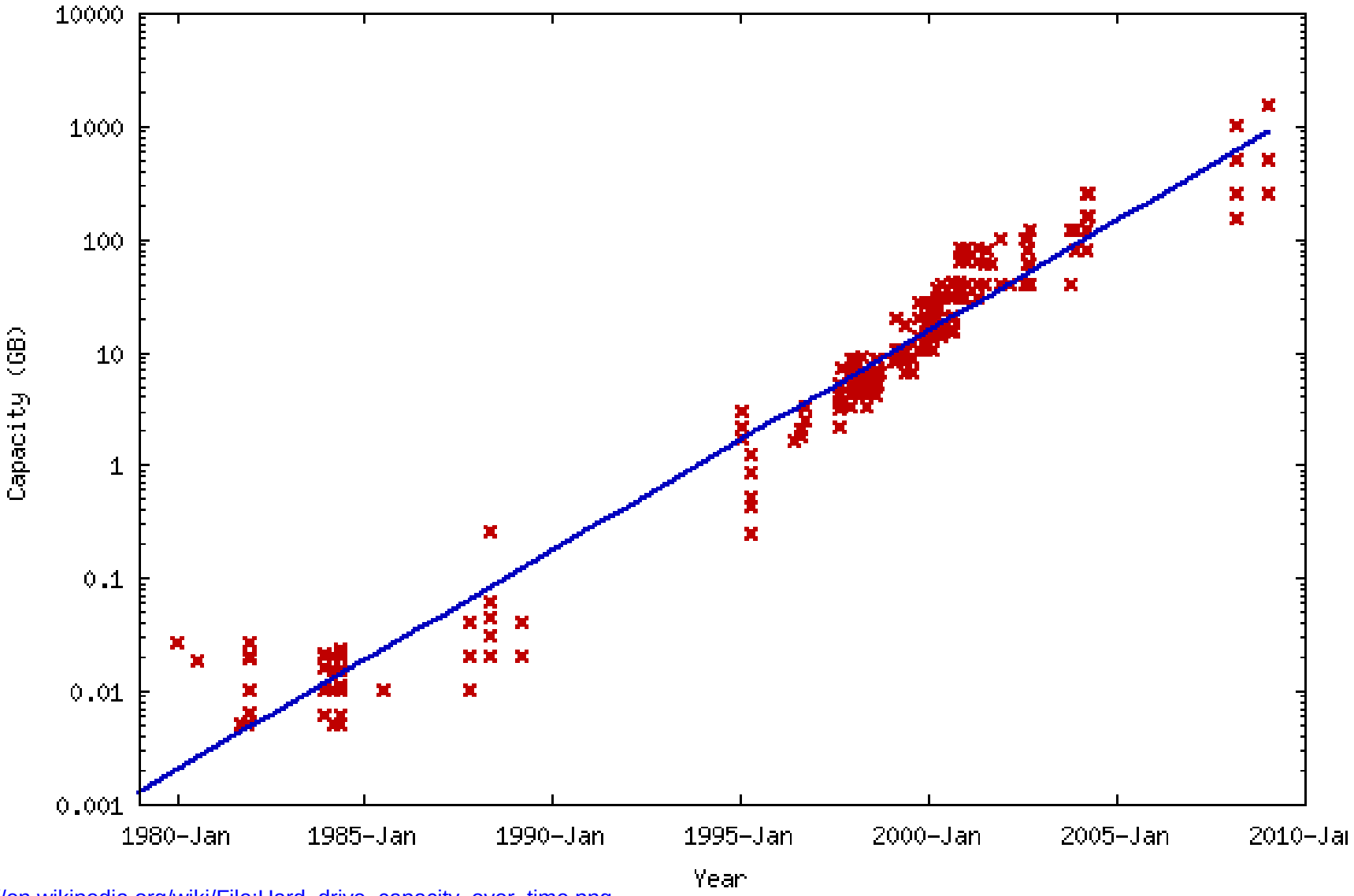


Video of hard disk in operation

https://www.youtube.com/watch?v=sG2sGd5XxM4

From: http://www.metacafe.com/watch/1971051/hard_disk_operation/

Hard drive capacity



http://en.wikipedia.org/wiki/File:Hard_drive_capacity_over_time.png

Seeking

- Steps
  - Speedup
  - Coast
  - Slowdown
  - Settle
- Very short seeks (2-4 tracks): dominated by settle time
- Short seeks (<200-400 tracks):
  - Almost all time in constant acceleration phase
  - Time proportional to square root of distance
- Long seeks:
  - Most time in constant speed (coast)
  - Time proportional to distance

Average seek time

- What is the "average" seek? If
  1. Seeks are fully independent and
  2. All tracks are populated:
  ➔ average seek = 1/3 full stroke
- But seeks are not independent
- Short seeks are common

- Using an average seek time for all seeks yields a poor model

Track following

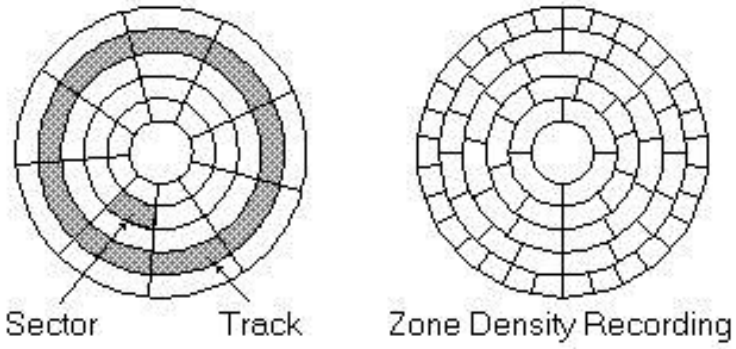
- Fine tuning the head position
  - At end of seek
  - Switching between last sector one track to first on another
  - Switching between head (irregularities in platters) [*]
- Time for full settle
  - 2-4ms; 0.24-0.48 revolutions
  - (7200RPM ➔ 0.12 revolutions/ms)
- Time for *
  - 1/3-1/2 settle time
  - 0.5-1.5 ms (0.06-0.18 revolutions @ 7200RPM)

Optimistic head settling on read

- Start reading when close
- Let error correcting come into play if not aligned

- Not feasible for writes

Zoning



- Note
  - More linear distance at edges then at center
  - Bits/track ~ R (circumference = $2\pi R$)
  - To maximize density, bits/inch should be the same
- How many bits per track?
  - Same number for all ➔ simplicity; lowest capacity
  - Different number for each ➔ very complex; greatest capacity
- Zoning
  - Group tracks into zones, with same number of bits
  - Outer zones have more bits than inner zones
  - Compromise between simplicity and capacity

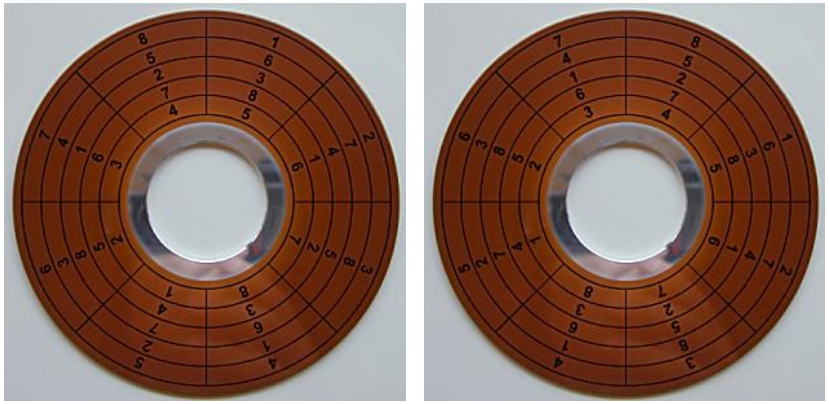
Example

IBM deskstar 40GV (ca. 2000)

Zone	Tracks in Zone	Sectors Per Track	Data Transfer Rate (Mbits/s)
0	624	792	372.0
1	1,424	780	366.4
2	1,680	760	357.0
3	1,616	740	347.6
4	2,752	720	338.2
5	2,880	680	319.4
6	1,904	660	310.0
7	2,384	630	295.9
8	3,328	600	281.8
9	4,432	540	253.6
10	4,528	480	225.5
11	2,192	440	206.7
12	1,600	420	197.3
13	1,168	400	187.9
14	18,15	370	173.8

Track skewing

- Why:
  - Imagine that sectors are numbered identically on each track, and we want to read all of two adjacent tracks (common!)
  - When we finish the last sector of the first track, we seek to the next track.
  - In that time, the platter has moved 0.24-0.48 revolutions
  - We have to wait almost a full rotation to start reading sector 1! Bad!
- What:
  - Offset first sector a small amount on each track
  - (Also offset it between platters due to head switch time)
- Effect:
  - Able to read data across tracks at full speed



From http://www.pcguide.com/ref/hdd/geom/tracksSkew-c.html

Sparing

- Reserve some sectors in case of defects
- Two mechanisms
  - Mapping
  - Slipping
- Mapping
  - Table that maps requested sector → actual sector
- Slipping
  - Skip over bad sector
- Combinations
  - Skip-track sparing at disk "low level" (factory) format
  - Remapping for defects found during operation

Caching and buffering

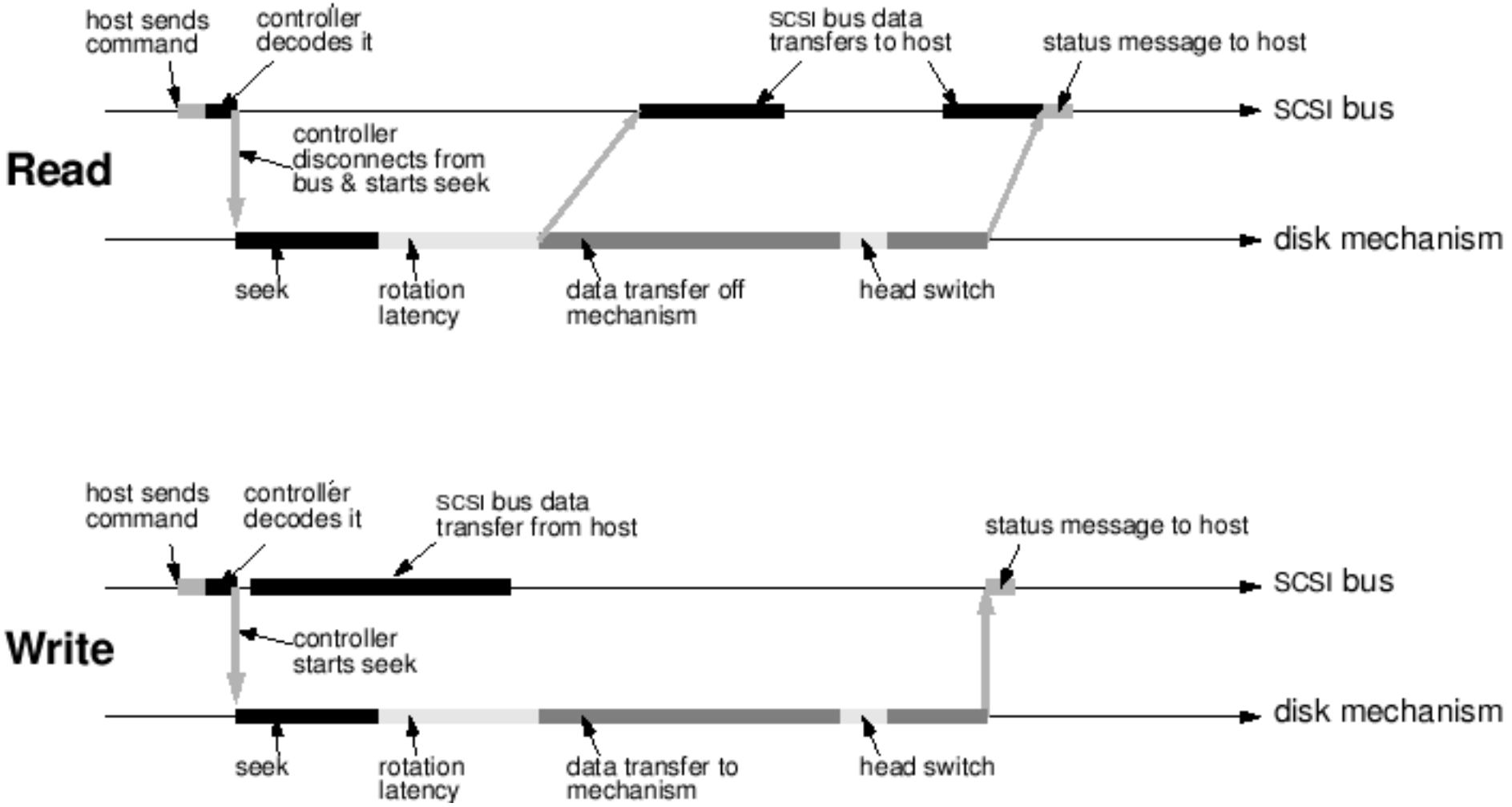
- Disks have caches
  - Caching (eg, optimistic read-ahead)
  - Buffering (eg, accommodate speed differences bus/disk)
- Buffering
  - Accept write from bus into buffer
  - Seek to sector
  - Write buffer
- Read-ahead caching
  - On demand read, fetch requested data and more
  - Upside: subsequent read may hit in cache
  - Downside: may delay next request; complex

Command queuing

- Send multiple commands (SCSI)
- Disk schedules commands
- Should be “better” because disk “knows” more

- Questions
  - How often are there multiple requests?
  - How does OS maintain priorities with command queuing?

Time line


Read
host sends command
controller decodes it
SCSI bus data transfers to host
status message to host
SCSI bus
controller disconnects from bus & starts seek
disk mechanism
seek
rotation latency
data transfer off mechanism
head switch
Write
host sends command
controller decodes it
SCSI bus data transfer from host
status message to host
SCSI bus
controller starts seek
disk mechanism
seek
rotation latency
data transfer to mechanism
head switch

Disk Parameters

	Seagate 6TB Enterprise HDD (2016)	Seagate Savvio (~2005)	Toshiba MK1003 (early 2000s)
Diameter	3.5″	2.5″	1.8″
Capacity	6 TB	73 GB	10 GB
RPM	7200 RPM	10000 RPM	4200 RPM
Cache	128 MB	8 MB	512 KB
Platters	~6	2	1
Average Seek	4.16 ms	4.5 ms	7 ms
Sustained Data Rate	216 MB/s	94 MB/s	16 MB/s
Interface	SAS/SATA	SCSI	ATA
Use	Desktop	Laptop	Ancient iPod

Disk Read/Write Latency

- Disk read/write latency has four components
  - **Seek delay (t_{seek})**: head seeks to right track
  - **Rotational delay ($t_{rotation}$)**: right sector rotates under head
    - On average: time to go halfway around disk
  - **Transfer time ($t_{transfer}$)**: data actually being transferred
  - **Controller delay ($t_{controller}$)**: controller overhead (on either side)
- Example: time to read a 4KB page assuming…
  - 128 sectors/track, 512 B/sector, 6000 RPM, 10 ms t_{seek}, 1 ms $t_{controller}$
  - 6000 RPM $\rightarrow$ 100 R/s $\rightarrow$ 10 ms/R $\rightarrow$ $t_{rotation}$ = 10 ms / 2 = 5 ms
  - 4 KB page $\rightarrow$ 8 sectors $\rightarrow$ $t_{transfer}$ = 10 ms * 8/128 = 0.6 ms
  - $t_{disk} = t_{seek} + t_{rotation} + \mathbf{t_{transfer}} + t_{controller}$
    $= 10 + 5 + \mathbf{0.6} + 1 = 16.6$ ms

Solid State Disks (SSD)

Introduction

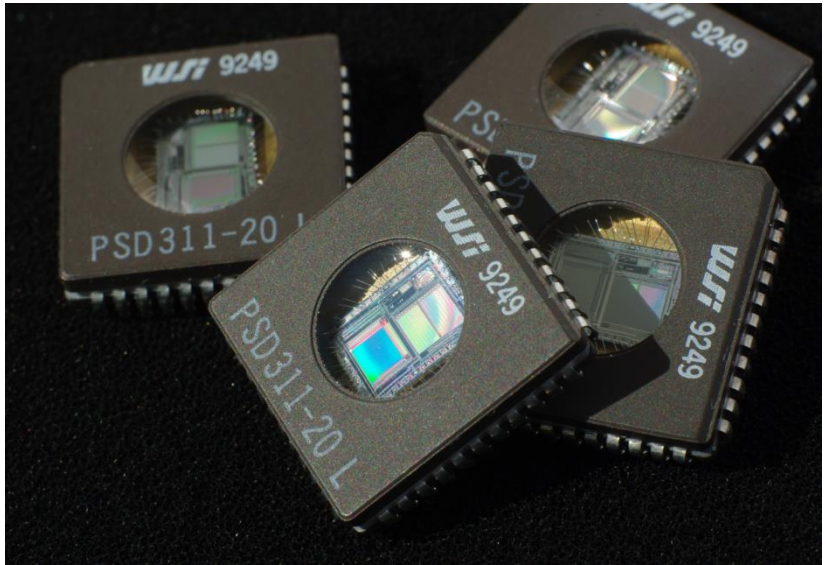
- Solid state drive (SSD)
  - Storage drives with no mechanical component
  - Available up to 4TB capacity
  - Usually 2.5” form factor

Source: wikipedia

Evolution of SSDs

- PROM – programmed once, non erasable
- EPROM – erased by UV lighting*, then reprogrammed
- EEPROM – electrically erase entire chip, then reprogram
- Flash – electrically erase and rerecord a single memory cell
- SSD - flash with a block interface emulating controller

* Obsolete, but totally awesome looking because they had a little window:



Flash memory primer

- Types: NAND and NOR
  - NOR allows bit level access
  - NAND allows block level access
    - For SSD, NAND is mostly used, NOR going out of favor
- Flash memory is an array of columns and rows
  - Each intersection contains a memory cell
    - Memory cell = floating gate + control gate
    - 1 cell = 1 bit

Memory cells of NAND flash

Single-level cell (SLC)	Multi-level cell (MLC)	Triple-level cell (TLC)
Single (bit) level cell	Two (bit) level cell	Three (bit) level cell
Fast: 25us read/100-300 us write	Reasonably fast: 50us read, 600-900us write	Decently fast: 75us read, 900-1350 us write
Write endurance - 100,000 cycles	Write endurance – 10000 cycles	Write endurance – 5000 cycles
Expensive	Less expensive	Least expensive

SSD internals

Package contains multiple dies (chips)

Die segmented into multiple planes

A plane with thousands(2048) of blocks + IO buffer pages

A block is around 64 or 128 pages

A page has a 2KB or 4KB data + ECC/additional information

SSD internals

- Logical pages striped over multiple packages
  - A flash memory package provides 40MB/s
  - SSDs use array of flash memory packages
- Interfacing:
  - Flash memory → Serial IO → SSD Controller → disk interface (SATA)
- SSD Controller implements Flash Translation Layer (FTL)
  - Emulates a hard disk
  - Exposes logical blocks to the upper level components
  - Performs additional functionality

SSD controller

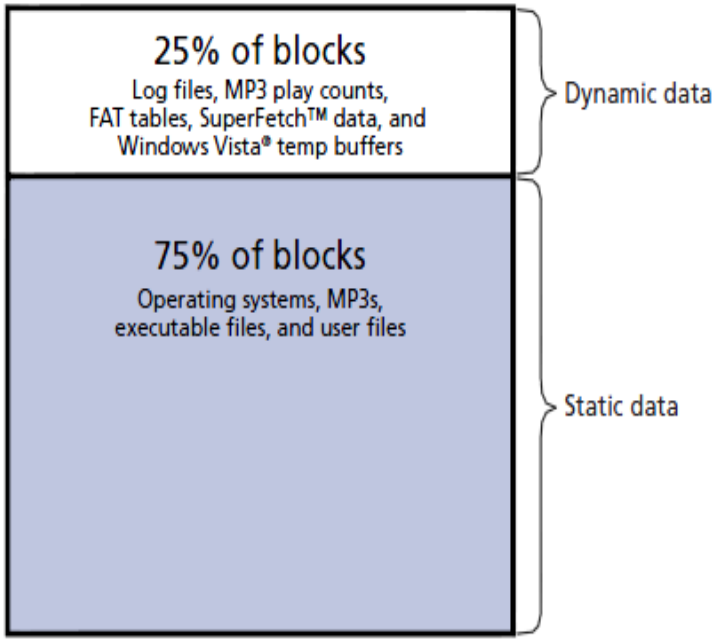
- Differences in SSD is due to controller
  - Performance loss if controller not properly implemented
- Has CPU, RAM cache, and may have battery/supercapacitor
- Dynamic logical block mapping
  - LBA to PBA
    - Page level mapping (uses large RAM space ~512MB)
    - Block level mapping (expensive read/write/modify)
  - Most use hybrid
    - Block level with log sized page level mapping

Wear levelling

- SSDs wear out
  - Each memory cell has finite flips
  - All storage systems have finite flips even HDD, CDs
  - SSD finite flips < HDD or CD
  - HDD failure modes are larger than SSD
- General method: over-provision unused blocks
  - Write on the unused block
  - Invalidate previous page
  - Remap new page

Dynamic wear leveling

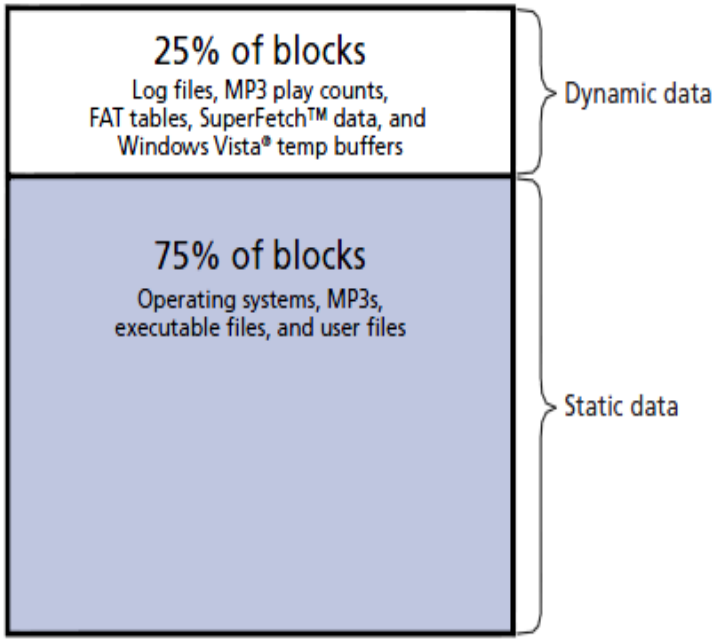
- Only pool unused blocks
- Only non-static portion is wear leveled
- Controller implementation easy
- Example: SSD lifespan dependent on 25% of SSD



Source: micron

Static wear leveling

- Pool all blocks
- All blocks are wear leveled
- Controller complicated
  - needs to track cycle # of all blocks
- Static data moved to blocks with higher cycle #
- Example: SSD lifespan dependent on 100% of SSD



Source: micron

Preemptive erasure

- Preemptive movement of cold data
- Recycle invalidated pages
  - Performed by garbage collector
  - Background operation
  - Triggered when close to having no more unused blocks

SSD operations

- Read
  - Page level granularity
  - 25us (SLC) to 60us (MLC)
- Write
  - Page level granularity
  - 250us (SLC) to 900us(MLC)
  - 10 x slower than read

SSD Operations

- Erase
  - Block level granularity, not page or word level
  - Erase must be done before writes
  - 3.5ms
  - 15 x slower than write

- TRIM
  - Command to notify SSD controller about deleted blocks
  - Sent by filesystem when a file is deleted
  - Avoids write amplification and improves SSD life

Using SSD (1)

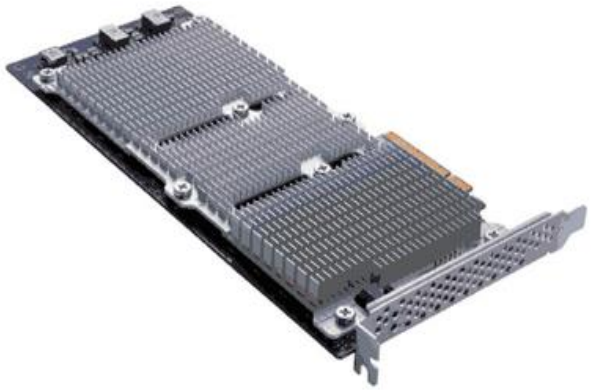
- Hybrid storage (tiering)
  - Server flash
    - Client cache to backend shared storage
    - Accelerates applications
    - Boosts efficiency of backend storage (backend demand decreases by upto 50%)
    - Example: NetApp Flash Accel acts as cache to storage controller
      - Maintains data coherency between the cache and backend storage
      - Supports data persistent for reboots

Using SSD (2)

- Hybrid storage
  - Flash array as cache (PCI-e cards flash arrays)
    - Example: NetApp Flash Cache in storage controller
    - Cache for reads
  - SSDs as cache
    - Example: NetApp Flash Pool in storage controller
    - Hot data tiered between SSDs and HDD backend storage
    - Cache for read and write
- SSD as main storage device
  - NetApp "All Flash" storage controllers
  - 300,000 read IOPS
  - < 1 ms response time
  - > 6Gbps bandwidth
  - Cost: $huge

NetApp flash cache

- Combined with HDD
- Upto 16 TB read cache



NetApp EF540 flash array

- 2U
- Target: transactional apps with high IOPS and low latency
- Equivalent to > 1000 15K RPM HDDs
- 95% reduction in space, power, and cooling
- Capacity: up to 38TB

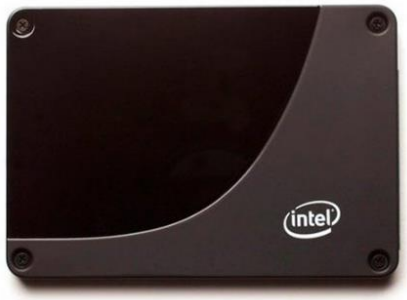
Source: NetApp

Differences between SSD and HDD

SSD	HDD
Uniform seek time	Different seek time for different sectors
Fast seek time – random read/writes as fast as sequential read/writes	Seek time dependent upon the RPM
Cost (Intel 530 Series 240GB – $209) • Capacity – $0.87/GB • Rate – $0.005/IOPS • Bandwidth - $0.38/Mbps	Cost (Seagate Constellation 1TB 7200rpm - $116) • Capacity – $0.11/GB • Rate – $0.55/IOPS • Bandwidth - $0.99/Mbps
Power: Active power: 195mW – 2W Idle power: 125mW – 0.5 W Low power consumption, No sleep mode	Power: Average operating power: 5.4W Higher power consumption, sleep mode zero power, higher wake up cost

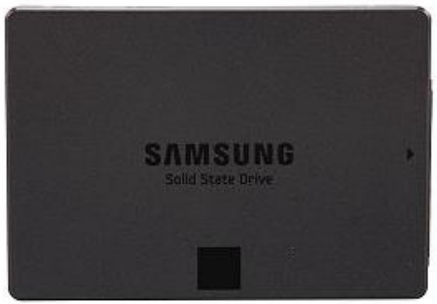
Differences between SSD and HDD

SSD	HDD
> 10,000 to > 1million IOPS	Hundreds of IOPS
Read/write in microseconds	Read/write in milliseconds
No mechanical part – no wear and tear	Moving part – wear and tear
MTBF ~ 2 million hours	MTBF ~ 1.2 million hours
Faster wear of a memory cell when it is written multiple times	Slower wear of the magnetic bit recording



Intel X-25E - $345
(older)
SLC
32 GB
SATA II
170-250MB/s
Latency 75-85us

Intel 530 - $209
(new)
MLC
240GB
SATA III
up to 540MB/s
Latency 80-85us



Samsung 840 EVO - $499
(new)
TLC
1TB
SATA III
up to 540MB/s

Which is cheaper?

HDD?

Yes!

Cheaper per gigabyte of capacity.

or

SSD?

Yes!

Cheaper per IOPS (performance).

Tradeoff!

Workloads

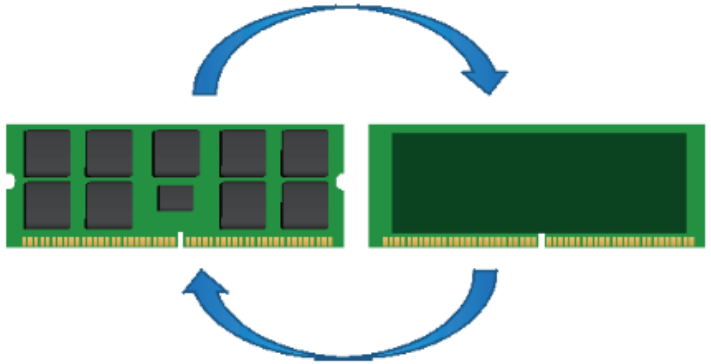
Workloads	SSD	HDD	Why ?
High write		Y	Wear for SSD
Sequential write	Y	Y	SSD: Seek time low HDD: Lower seek time
Log files (small writes)	Y		Faster seek time
Database read queries	Y		Faster seek time
Database write queries	Y		Faster seek time
Analytics – HDFS	Y	Y	SSD – Append operation faster HDD – higher capacity
Operating systems	Y	Y	SSD: Less changing files HDD: More read only data

Other Flash technologies - NVDIMMS

- Revisiting NVRAM
- DDR3 DIMMS + NAND Flash
  - Speed of DIMMS
  - extensive read/write cycles for DIMMS
  - Non volatile nature of NAND Flash
- Support added by BIOS
  - Backup to NAND Flash
  - Triggered by HW SAVE signal
- Stored charge
  - Super capacitors
  - Battery packs

How It Works

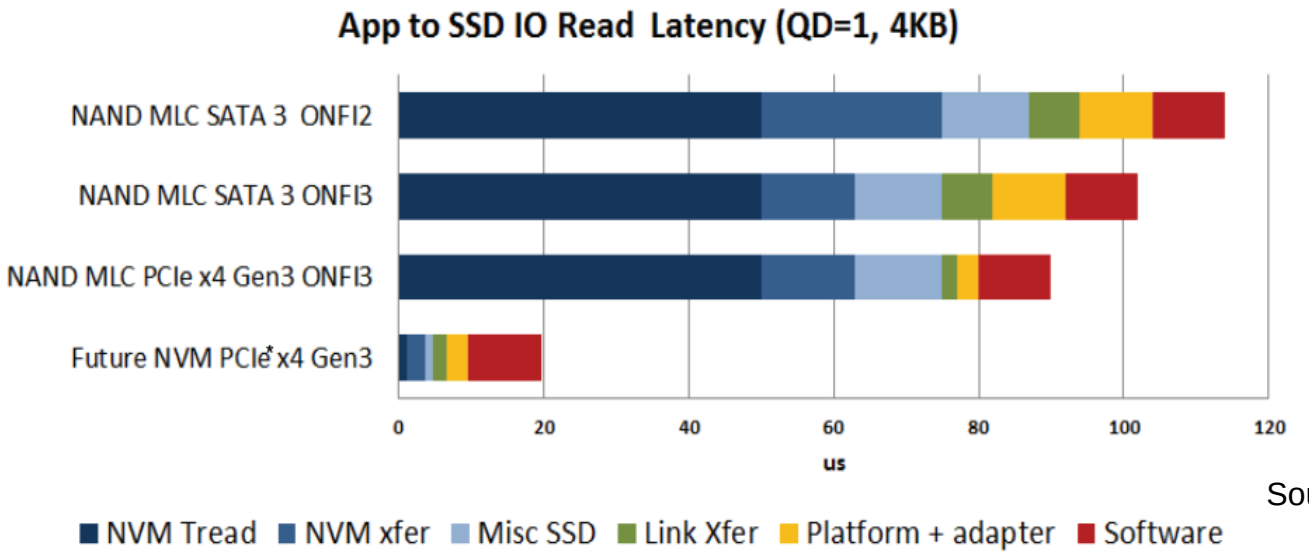
If there is a power failure, the supercap module powers NVDIMM while it copies all data from the DDR-3 to on-module flash



When power is restored NVDIMM copies all data from flash to DDR-3 and normal operation resumes

(SNIA - **NVDIMM Technical Brief**)

In future - persistent memory

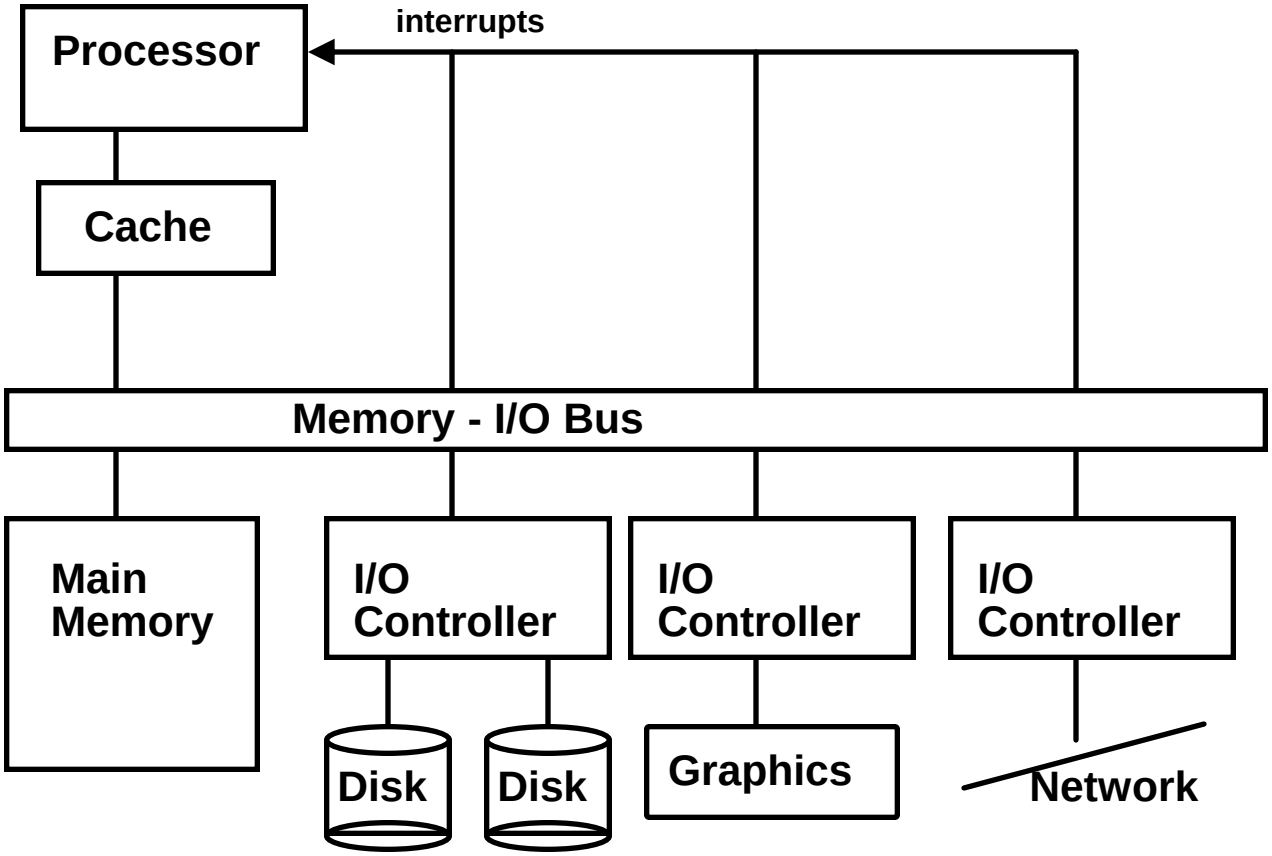


Source: Andy Rudoff, Intel

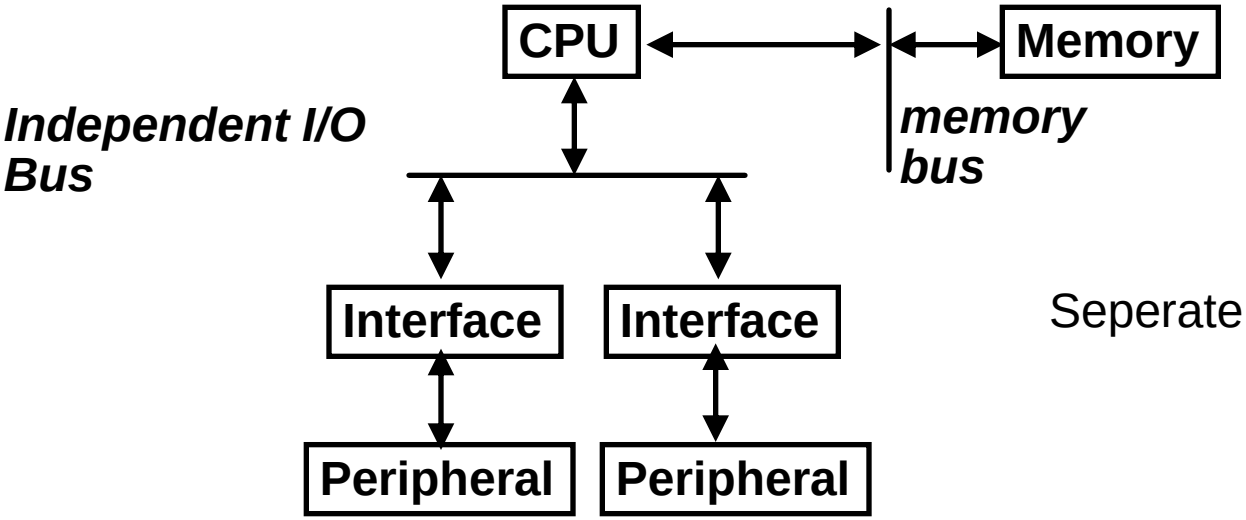
- NVM latency closer to DRAM
- Types
  - Battery-backed DRAM, NVM with caching, Next-gen NVM
- Attributes:
  - Bytes-addressable, LOAD/STORE access, memory-like, DMA
  - Data not persistent until flushed

The I/O Subsystem

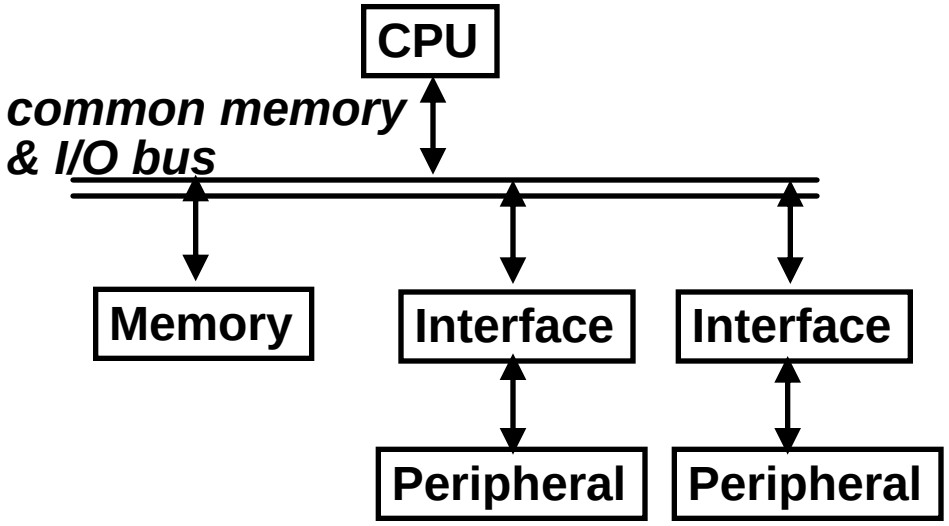
I/O Systems


Processor
interrupts
Cache
Memory - I/O Bus
Main Memory
I/O Controller
I/O Controller
I/O Controller
Disk
Disk
Graphics
Network

I/O Interface

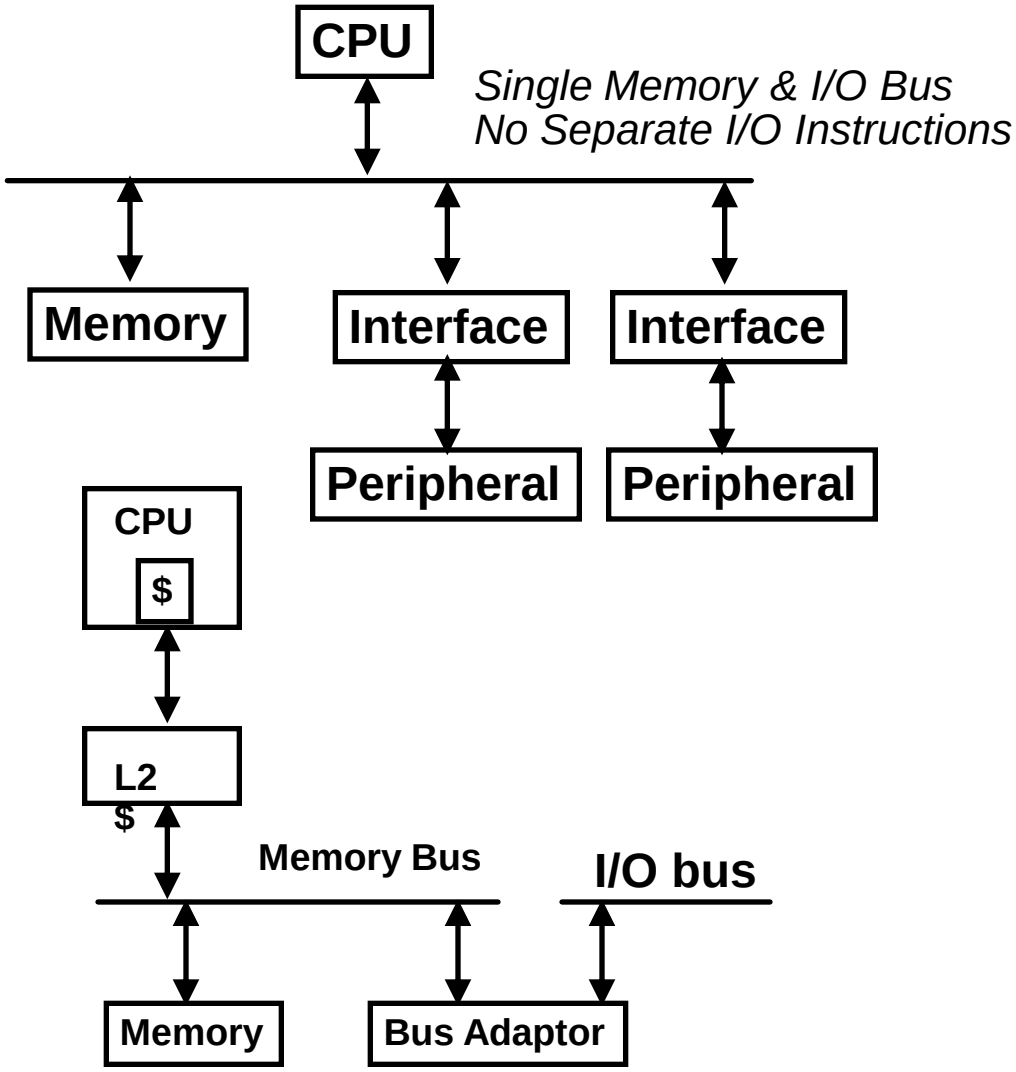


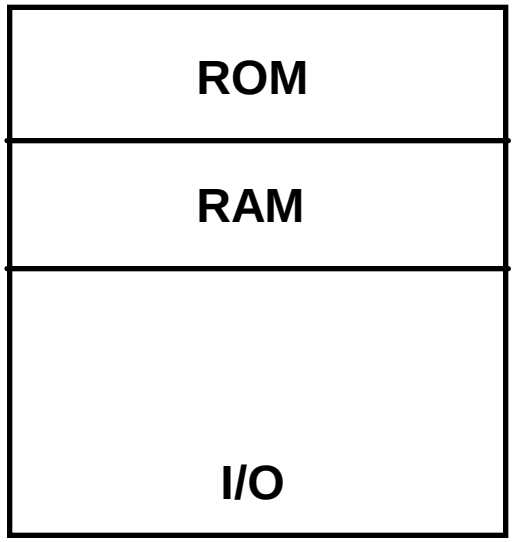
Seperate I/O instructions (in,out)



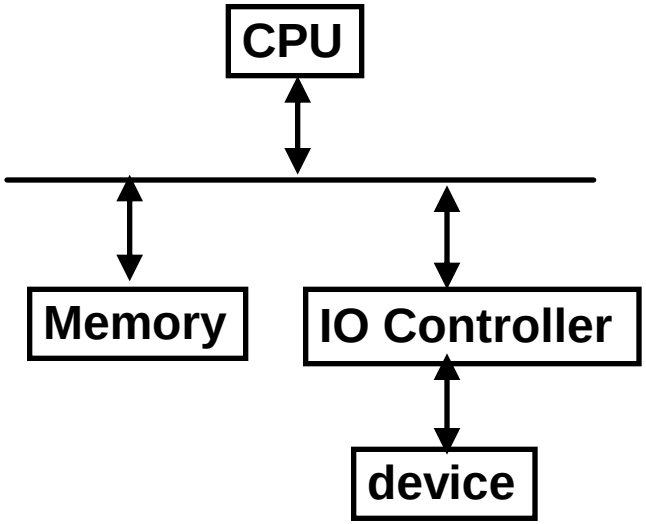
Lines distinguish between I/O and memory transfers

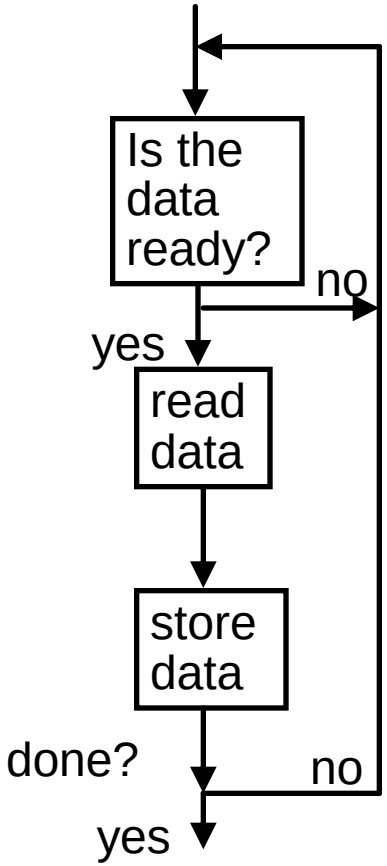
Memory Mapped I/O





Programmed I/O (Polling)

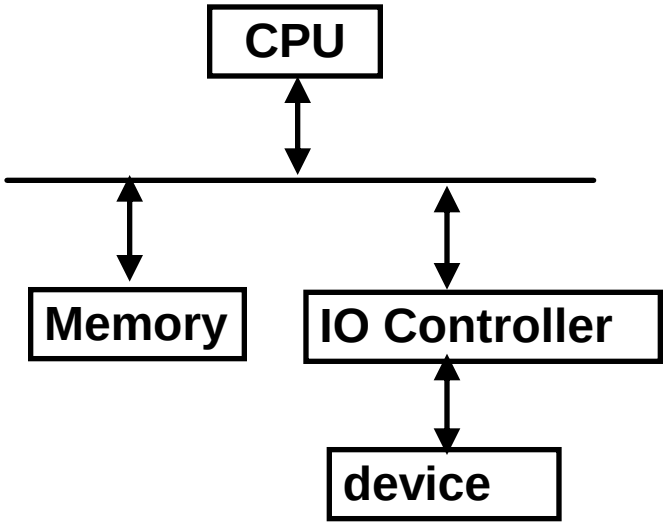


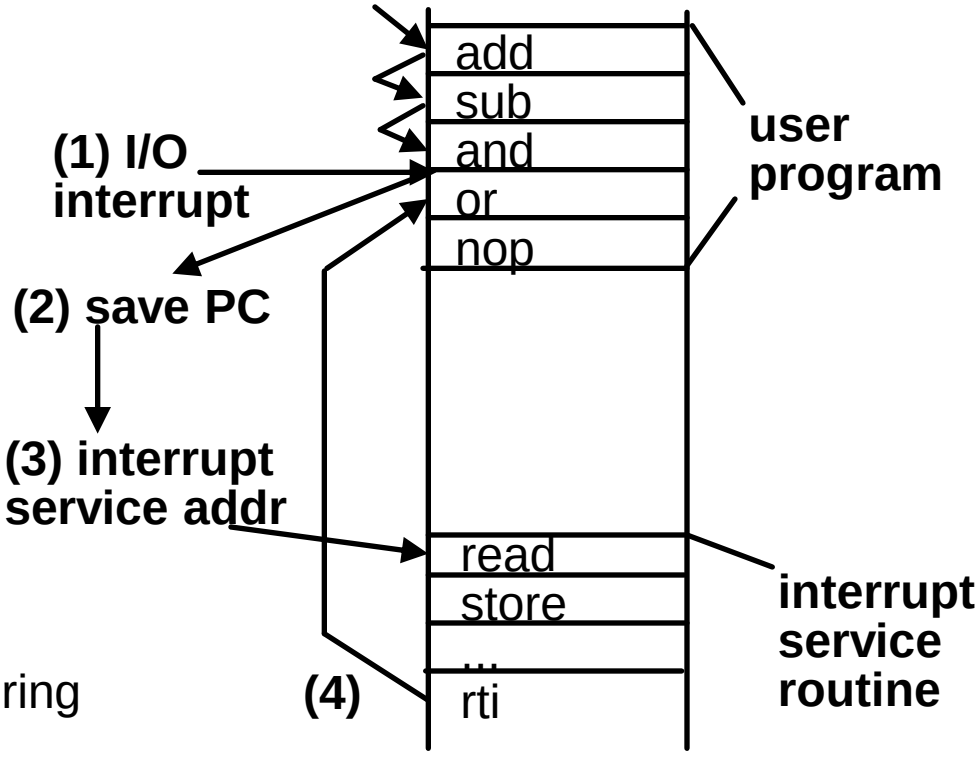


busy wait loop
not an efficient
way to use the CPU
unless the device
is very fast!

but checks for I/O
completion can be
dispersed among
computationally
intensive code

Interrupt Driven Data Transfer





User program progress only halted during actual transfer

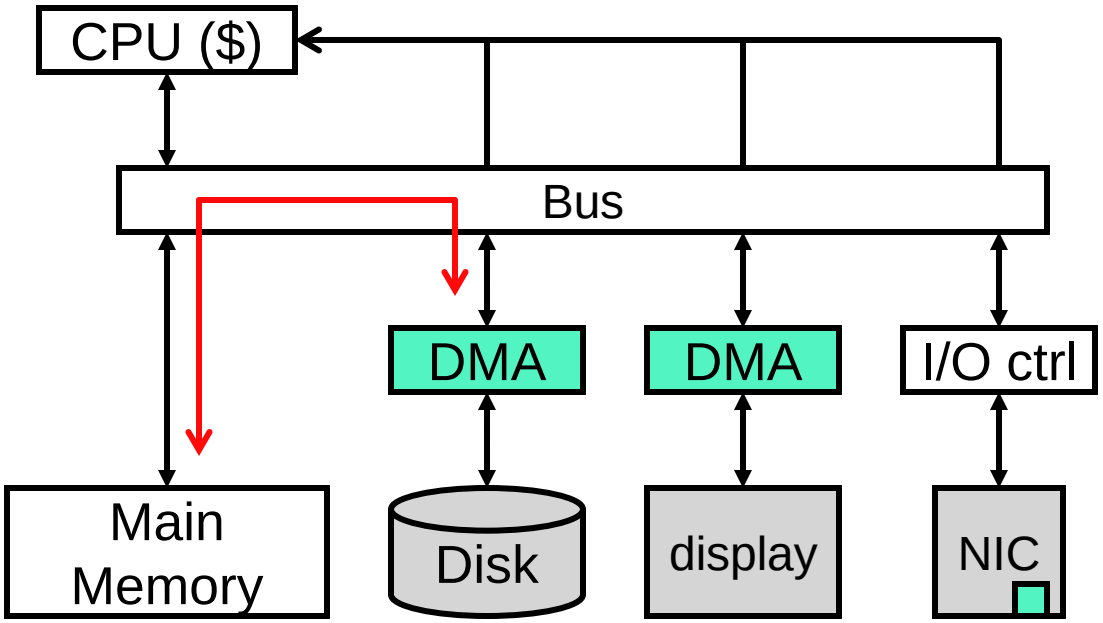
Direct Memory Access (DMA)

- Interrupts remove overhead of polling…
- But still requires OS to transfer data one word at a time
  - OK for low bandwidth I/O devices: mice, microphones, etc.
  - Bad for high bandwidth I/O devices: disks, monitors, etc.

- **Direct Memory Access (DMA)**
  - Transfer data between I/O and memory without processor control
  - Transfers entire blocks (e.g., pages, video frames) at a time
    - Can use bus "burst" transfer mode if available
  - Only interrupts processor when done (or if error occurs)

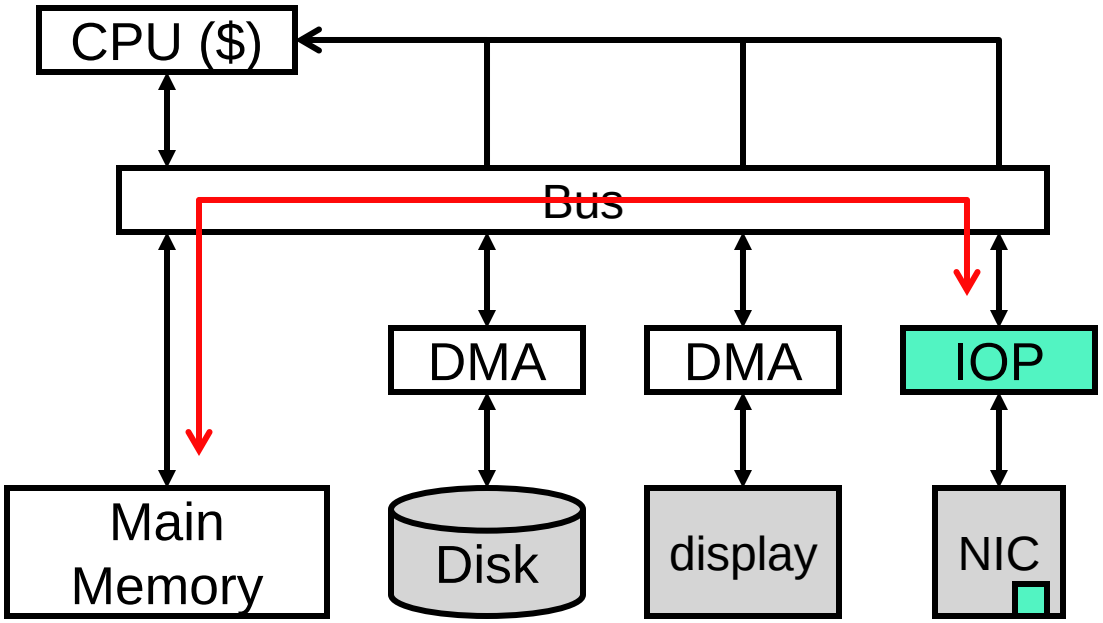
DMA Controllers

- To do DMA, I/O device attached to **DMA controller**
  - Multiple devices can be connected to one DMA controller
  - Controller itself seen as a memory mapped I/O device
    - Processor initializes start memory address, transfer size, etc.
  - DMA controller takes care of bus arbitration and transfer details
    - So that's why buses support arbitration and multiple masters!



I/O Processors

- A DMA controller is a very simple component
  - May be as simple as a FSM with some local memory
- Some I/O requires complicated sequences of transfers
  - **I/O processor**: heavier DMA controller that executes instructions
    - Can be programmed to do complex transfers
    - E.g., programmable network card



Summary: Fundamental properties of I/O systems

Top questions to ask about any I/O system:

- Storage device(s):
  - What kind of device (SSD, HDD, etc.)?
  - Performance characteristics?
- Topology:
  - What's connected to what (buses, IO controller(s), fan-out, etc.)?
  - What protocols in use (SAS, SATA, etc.)?
  - Where are the bottlenecks (PCI-E bus? SATA protocol limit? IO controller bandwidth limit?)
  - Protocol interaction: polled, interrupt, DMA?