Storage and Other I/O Topics

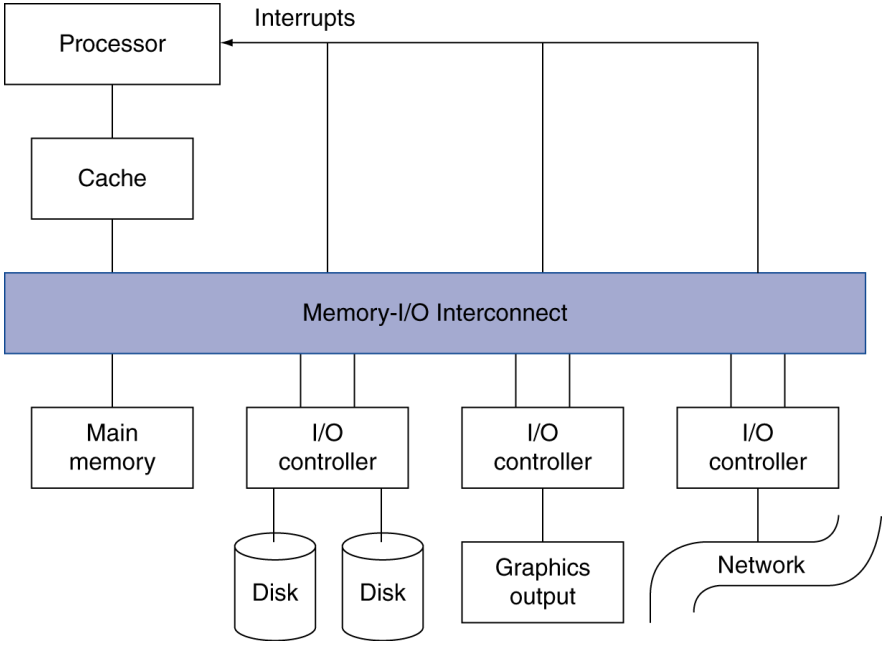
Jo, Heeseung

Introduction

I/O devices can be characterized by

- Behavior: input, output, storage
- Partner: human or machine
- Data rate: bytes/sec, transfers/sec

I/O bus connections



I/O System Characteristics

Dependability is important

- Particularly for storage devices

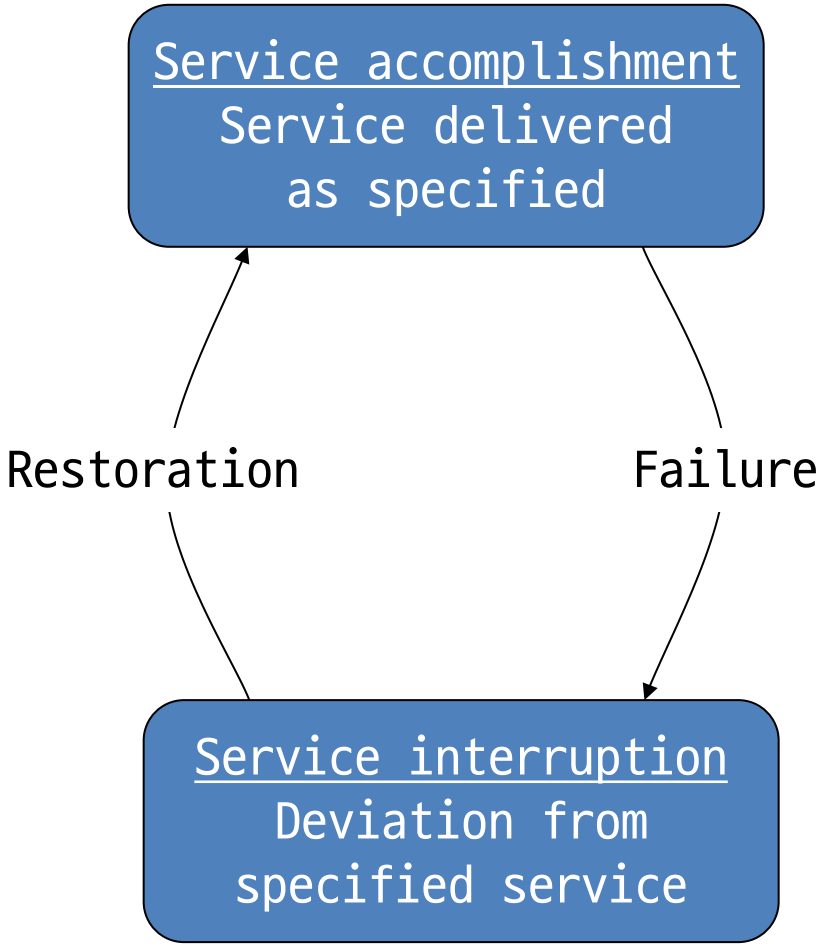
Performance measures

- Latency (response time)
- Throughput (bandwidth)
- Desktops & embedded systems
  - Mainly interested in response time & diversity of devices
- Servers
  - Mainly interested in throughput & expandability of devices

Dependability

Fault: failure of a component

- May or may not lead to system failure



Dependability Measures

Reliability: mean time to failure (MTTF)

Service interruption: mean time to repair (MTTR)

Mean time between failures

- MTBF = MTTF + MTTR

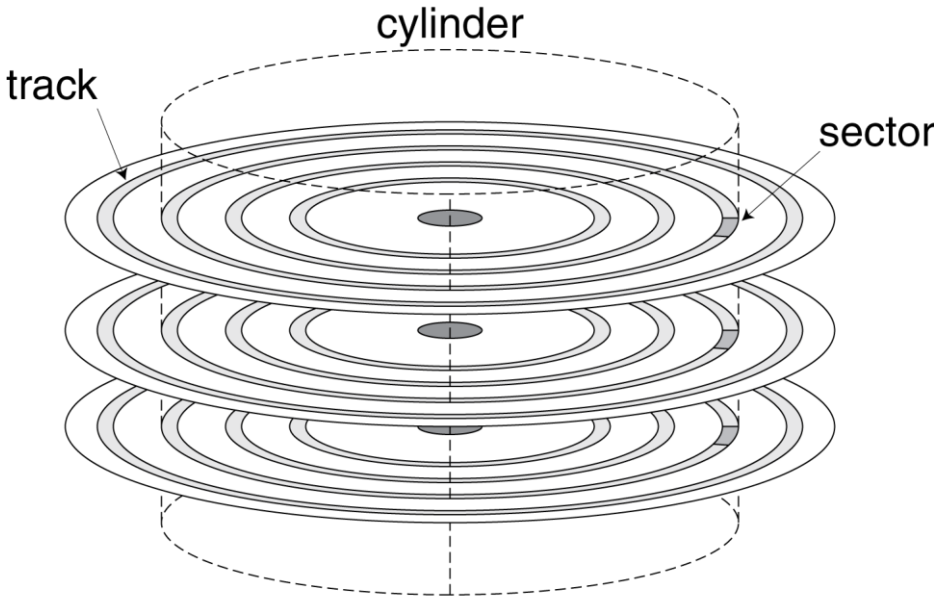
Availability = MTTF / (MTTF + MTTR)

Improving Availability

- Increase MTTF:
  - Fault avoidance
  - Fault tolerance
  - Fault forecasting
- Reduce MTTR
  - Improved tools
  - Processes
  - Diagnosis

Disk Storage

Nonvolatile, rotating magnetic storage



Disk Sectors and Access

Each sector records

- Sector ID
- Data (512 bytes, 4096 bytes proposed)
- Error correcting code (ECC)
  - Used to hide defects and recording errors
- Synchronization fields and gaps

Access to a sector involves

- Queuing delay if other accesses are pending
- Seek: move the heads
- Rotational latency
- Data transfer
- Controller overhead

Disk Access Example

Given

- 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk

Average read time

- 4ms seek time
  + ½ / (15,000/60) = 2ms rotational latency
  + 512 / 100MB/s = 0.005ms transfer time
  + 0.2ms controller delay
  = 6.2ms

If actual average seek time is 1ms

- Average read time = 3.2ms

Disk Performance Issues

Manufacturers quote average seek time

- Based on all possible seeks
- Locality and OS scheduling lead to smaller actual average seek times

Smart disk controller allocate physical sectors on disk

- Present logical sector interface to host
- SCSI, ATA, SATA

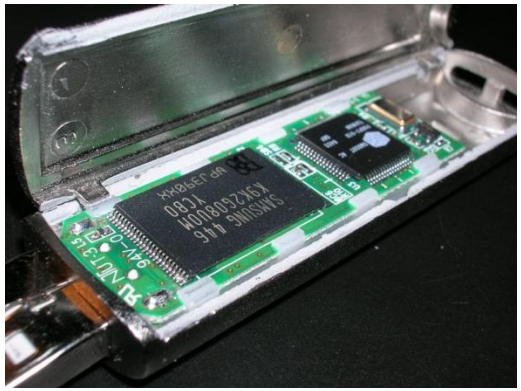
Disk drives include caches

- Prefetch sectors in anticipation of access
- Avoid seek and rotational delay

Flash Storage

Nonvolatile semiconductor storage

- 100× – 1000× faster than disk
- Smaller, lower power, more robust
- But more $/GB (between disk and DRAM)



Flash Types

NOR flash: bit cell like a NOR gate

- Random read/write access
- Used for instruction memory in embedded systems

NAND flash: bit cell like a NAND gate

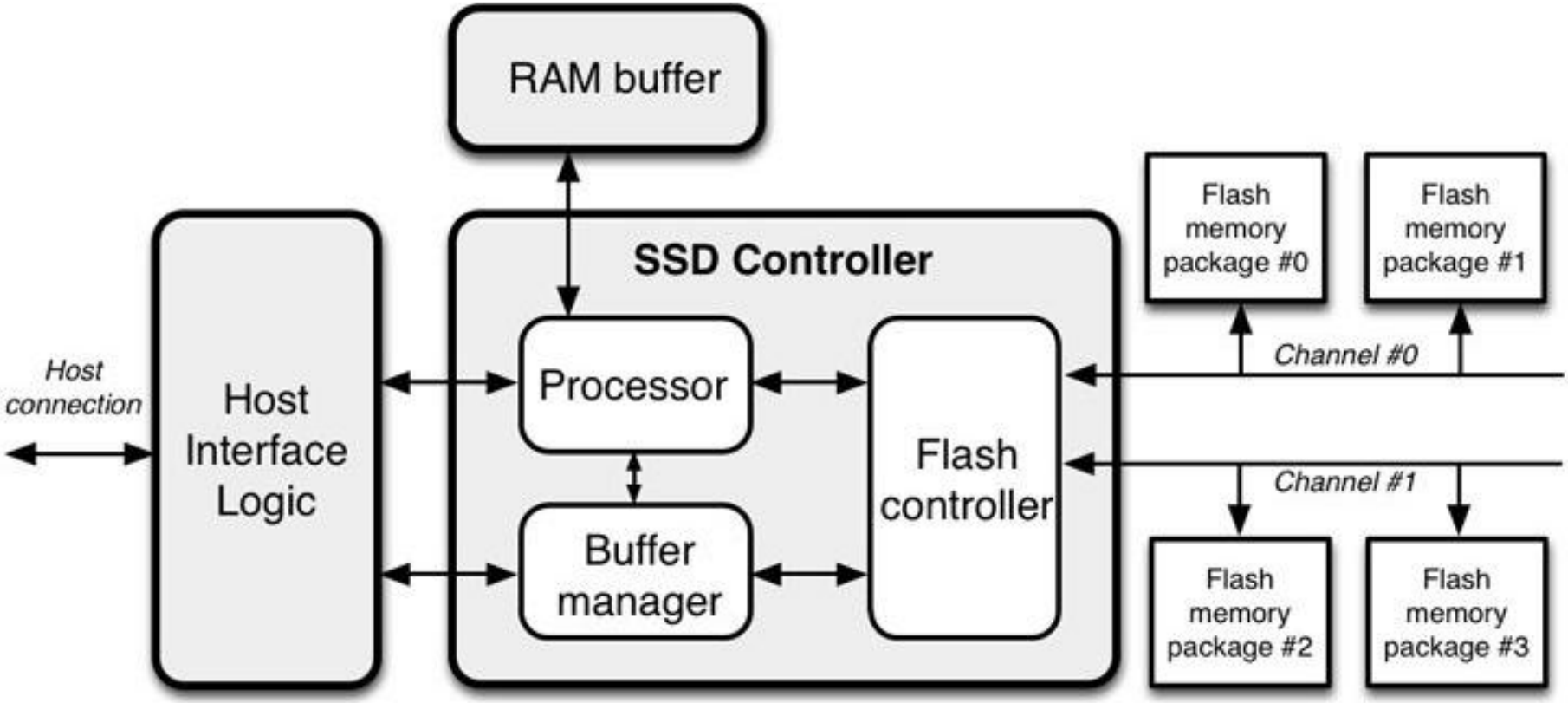
- Denser (bits/area), but block-at-a-time access
- Cheaper per GB
- Used for USB keys, media storage, …

Flash bits wears out after 10000's of erases

- Not suitable for direct RAM or disk replacement
- Wear leveling: remap data to less used blocks

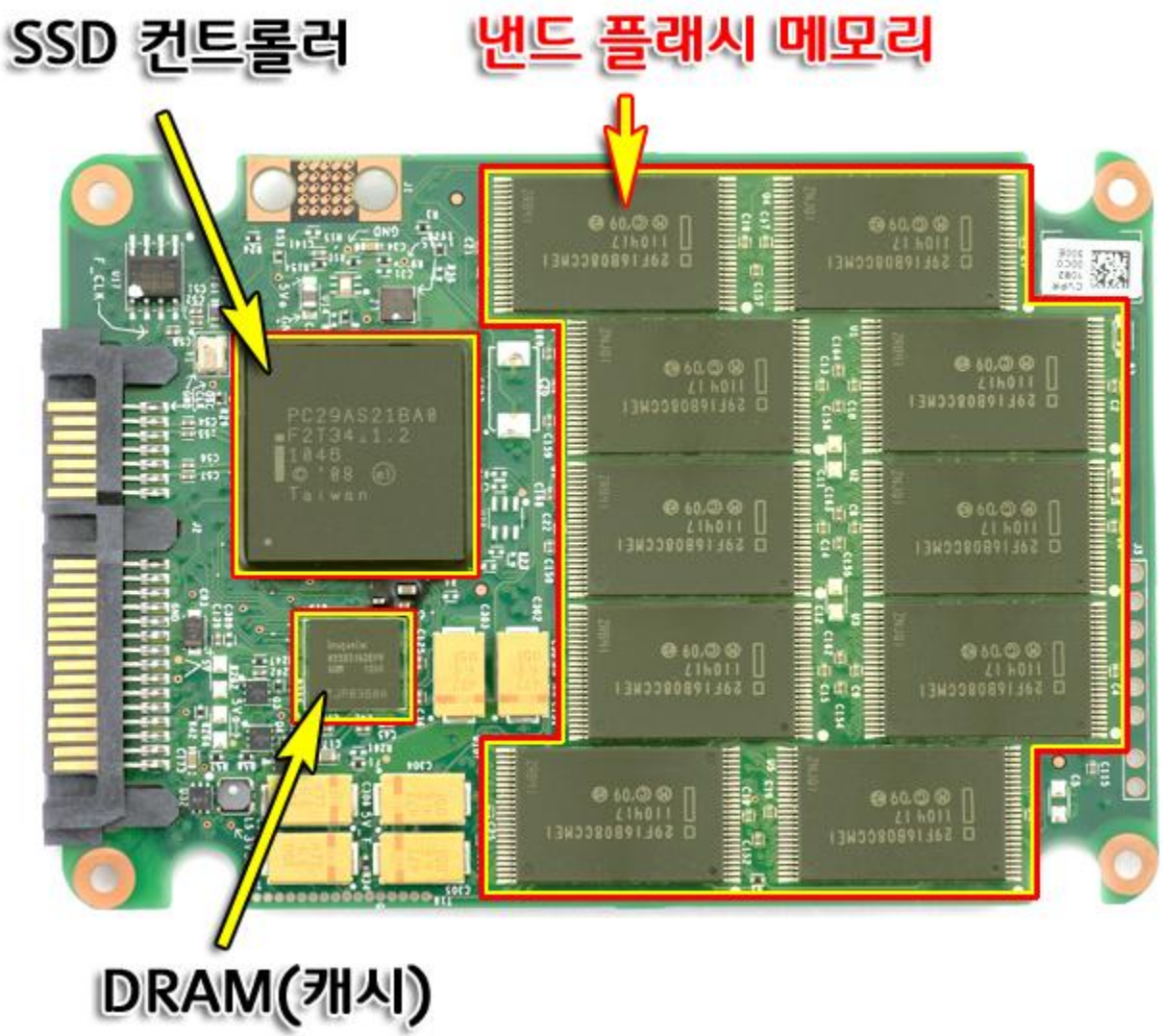
Solid state drive (SSD)

Architecture of a SSD

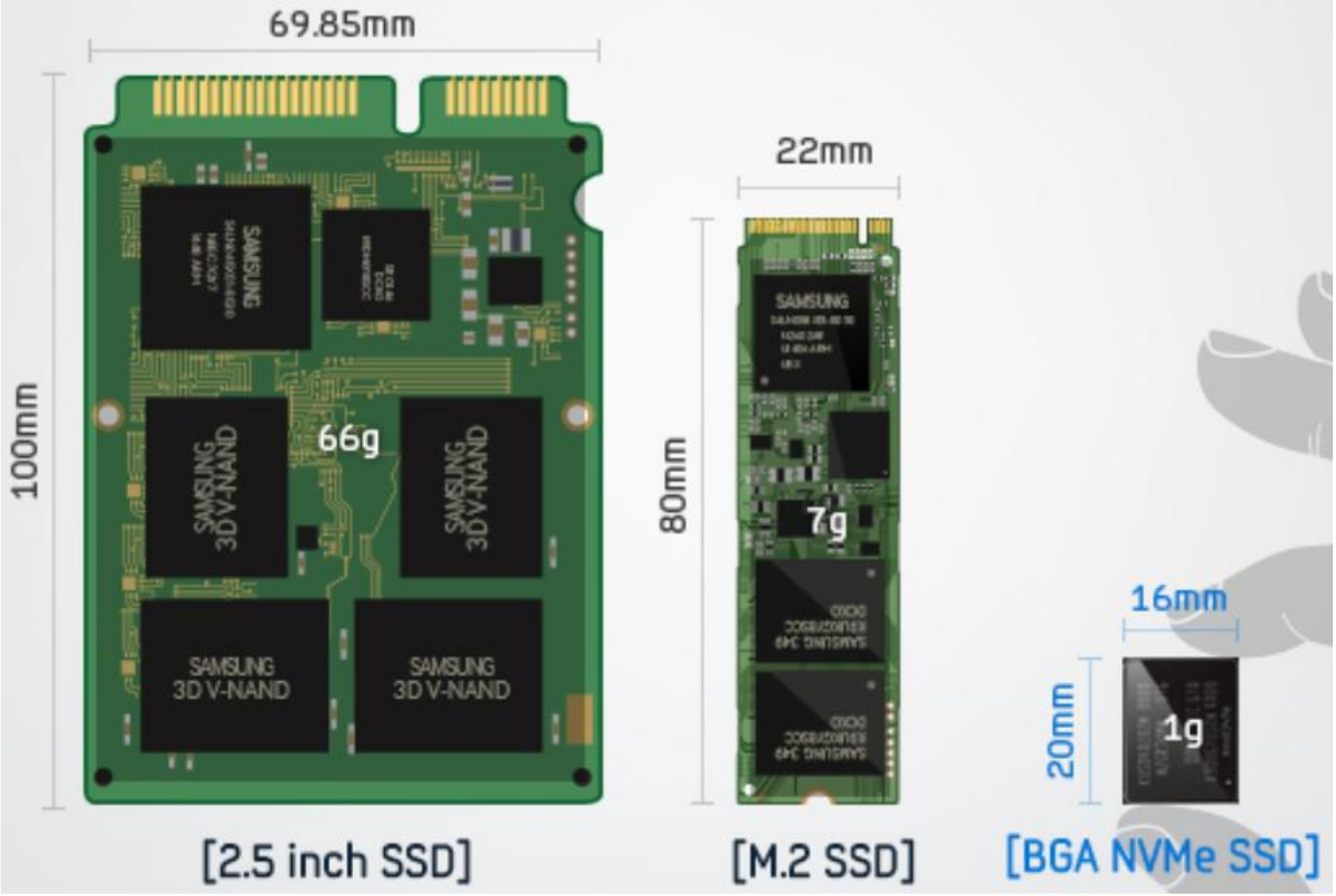


Solid state drive (SSD)

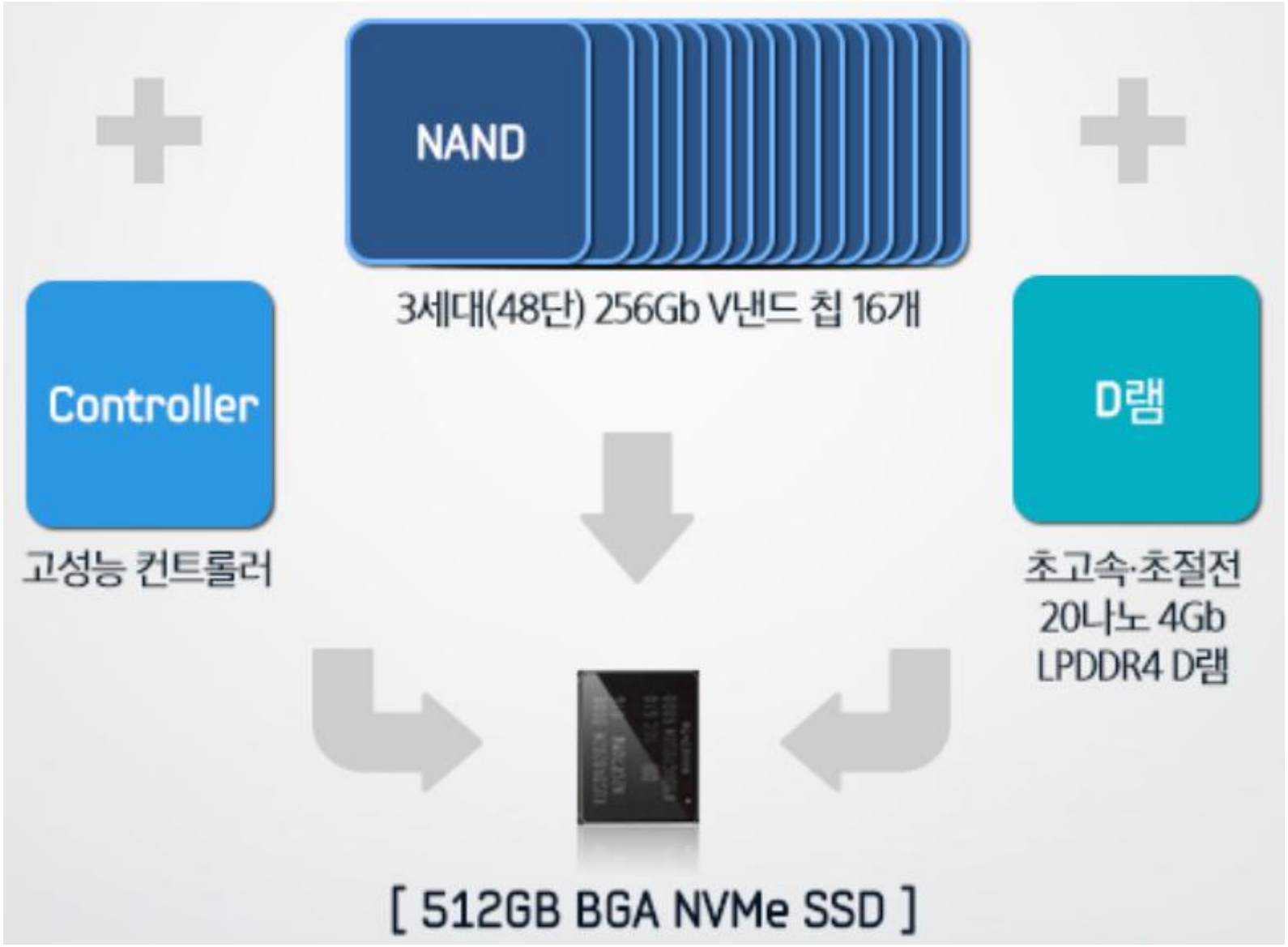
Architecture of a SSD



Solid state drive (SSD)

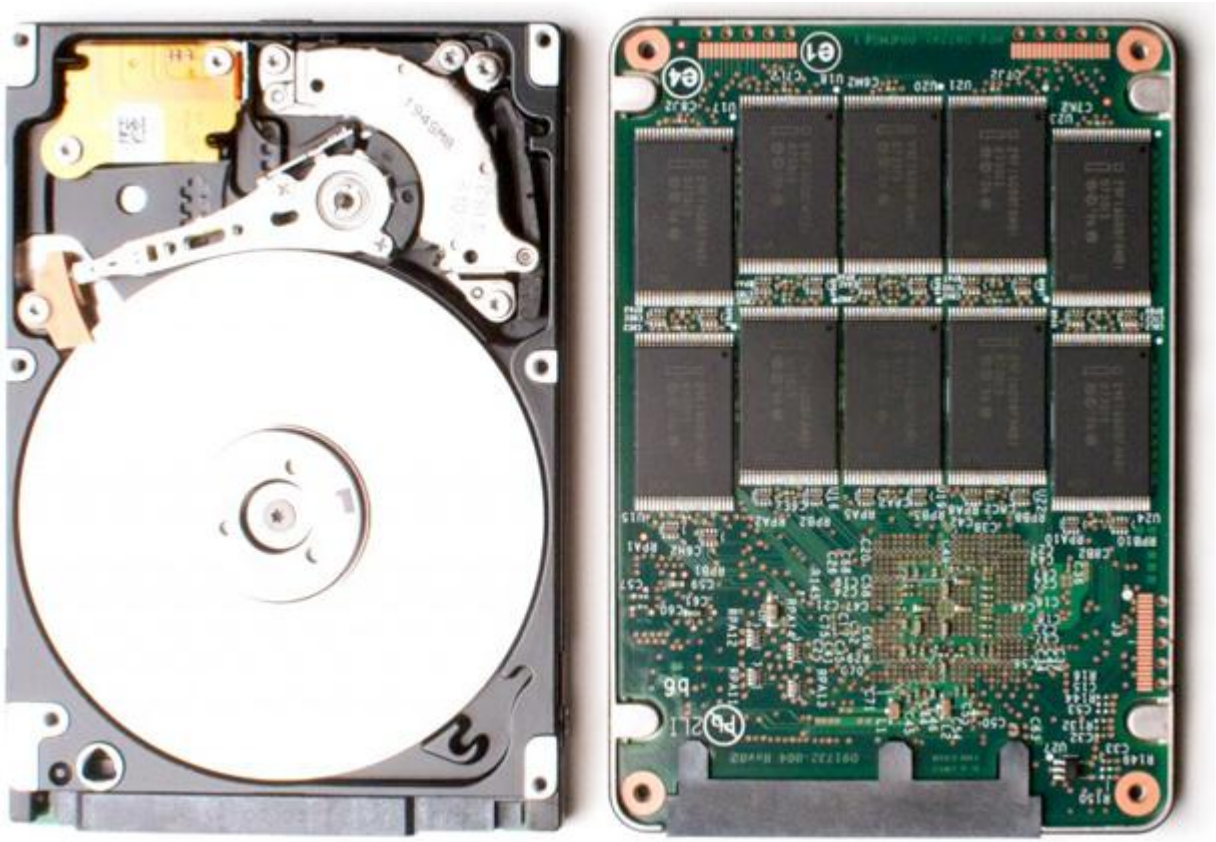


Solid state drive (SSD)



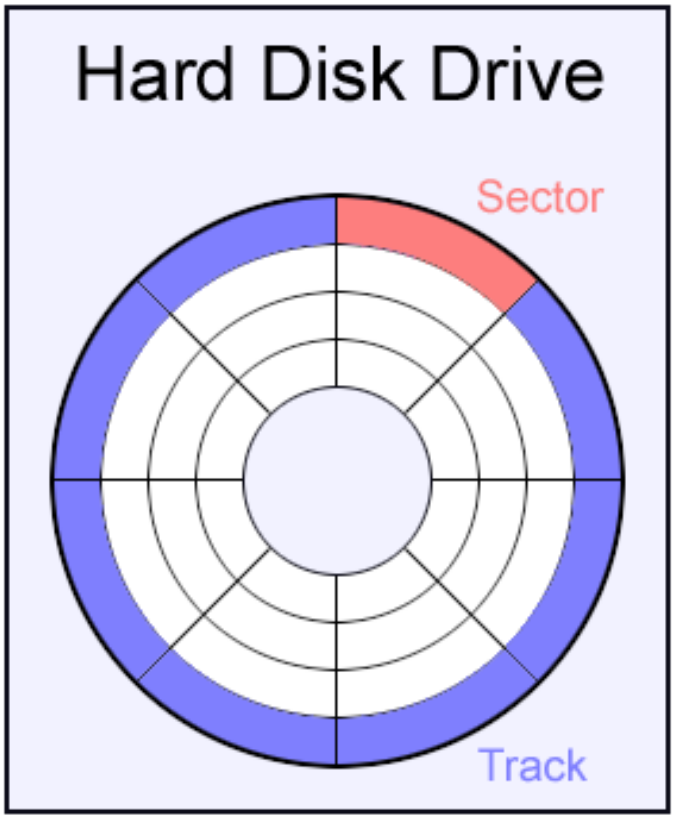
Solid state drive (SSD)

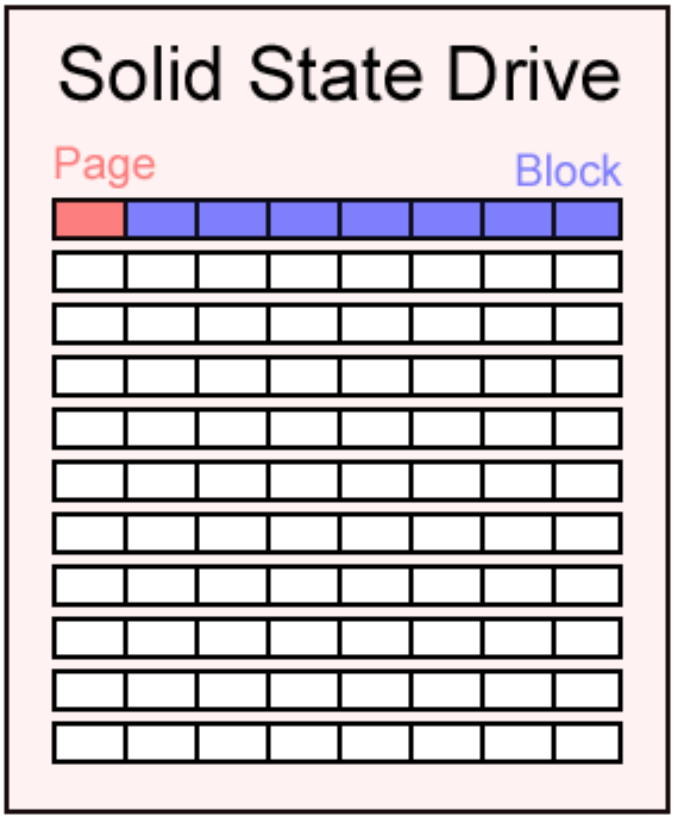
HDD vs. SSD



Solid state drive (SSD)

HDD vs. SSD





Interconnecting Components

Need interconnections between

- CPU, memory, I/O controllers

Bus: shared communication channel

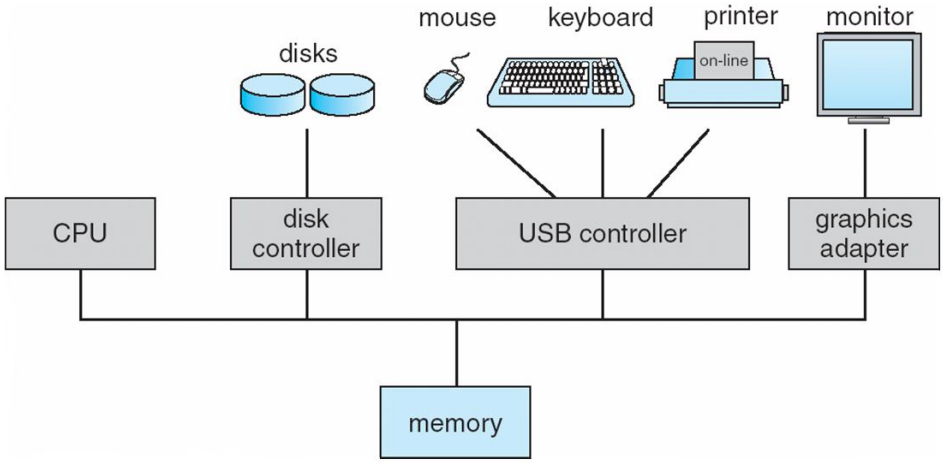
- Parallel set of wires for data and synchronization of data transfer
- Can become a bottleneck

Performance limited by physical factors

- Wire length, number of connections

More recent alternative: high-speed serial connections with switches

- Like networks



Bus Types

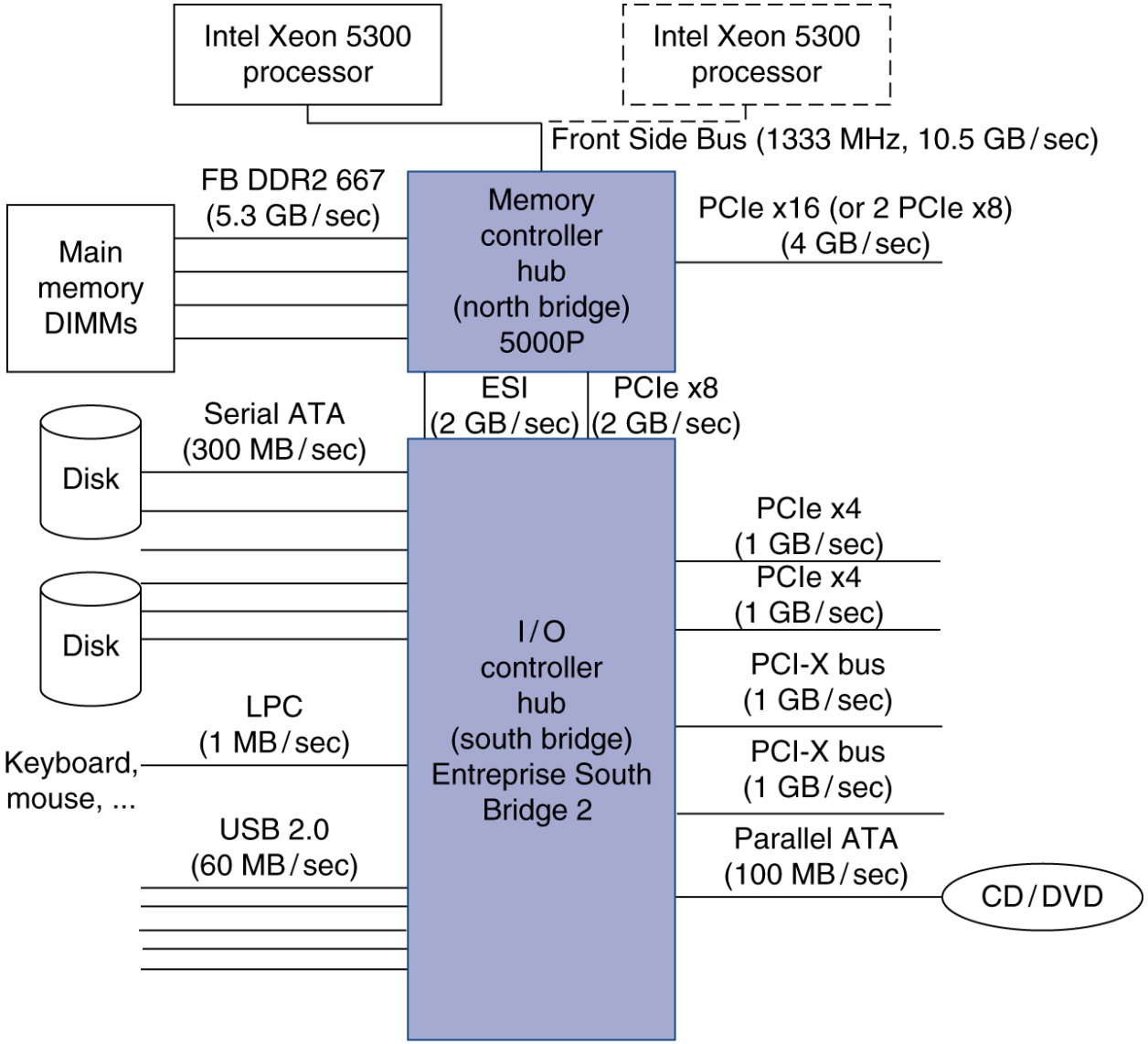
Processor-Memory buses

- Short, high speed
- Design is matched to memory organization

I/O buses

- Longer, allowing multiple connections
- Specified by standards for interoperability
- Connect to processor-memory bus through a bridge

Typical x86 PC I/O System



I/O Management

I/O is mediated by the OS

- Multiple programs share I/O resources
  - Need protection and scheduling
- I/O causes asynchronous interrupts
  - Same mechanism as exceptions
- I/O programming is fiddly
  - OS provides abstractions to programs

I/O Register Mapping

Isolated I/O (= Dedicated I/O instructions or Direct I/O)

- Separate instructions to access I/O registers
- I/O instruction: specifying both the device number and the command word
- Protection: can only be executed in kernel mode
- Intel IA-32, IBM 370

Memory mapped I/O

- Registers are addressed in same space as memory
- OS uses address translation mechanism to make them only accessible to kernel

Example: A Simple Printer

Status register

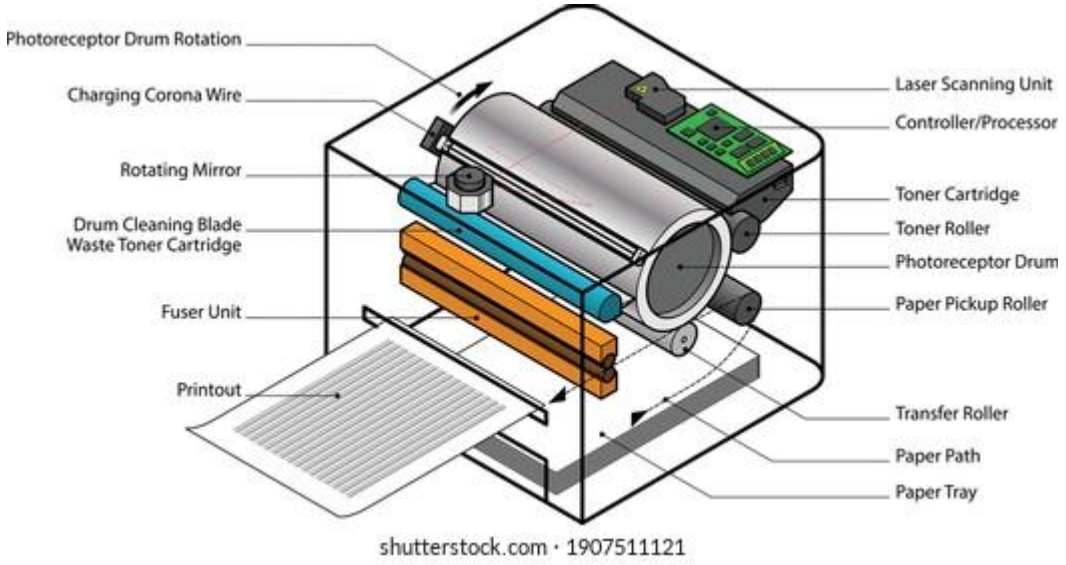
- Done bit
  - Set by the printer when it has printed a character
- Error bit
  - Paper jam or out of paper

Data register

- Data to be printed

Operations of processor

- Must wait until the done bit set by the printer
- Must check the error bit



Communicating with the Processor

How does the kernel notice an I/O has finished?

- Polling vs. Hardware interrupt

Polling : periodically check I/O status register

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

Common in small or low-performance real-time embedded systems

- Predictable timing
- Low hardware cost

In other systems, wastes CPU time

Communicating with the Processor

Interrupt-Driven I/O

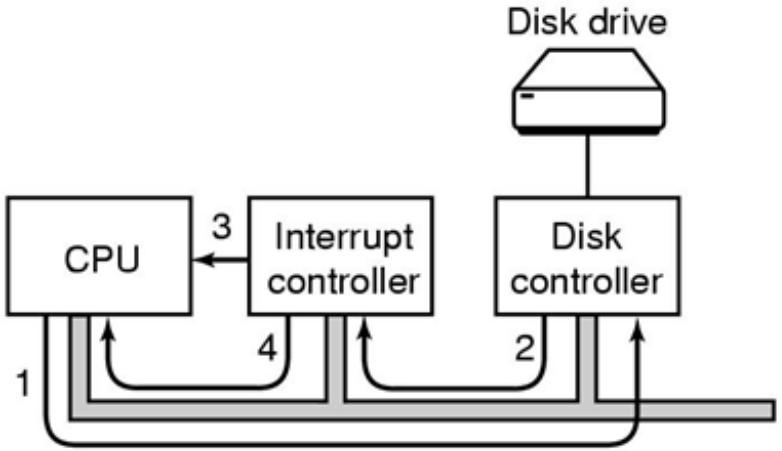
- When a device is ready or error occurs controller interrupts CPU

Interrupt is like an exception

- But not synchronized to instruction execution
- Can invoke handler between instructions

Priority interrupts

- Devices needing more urgent attention get higher priority
- Higher priority interrupt can interrupt handler for a lower priority interrupt



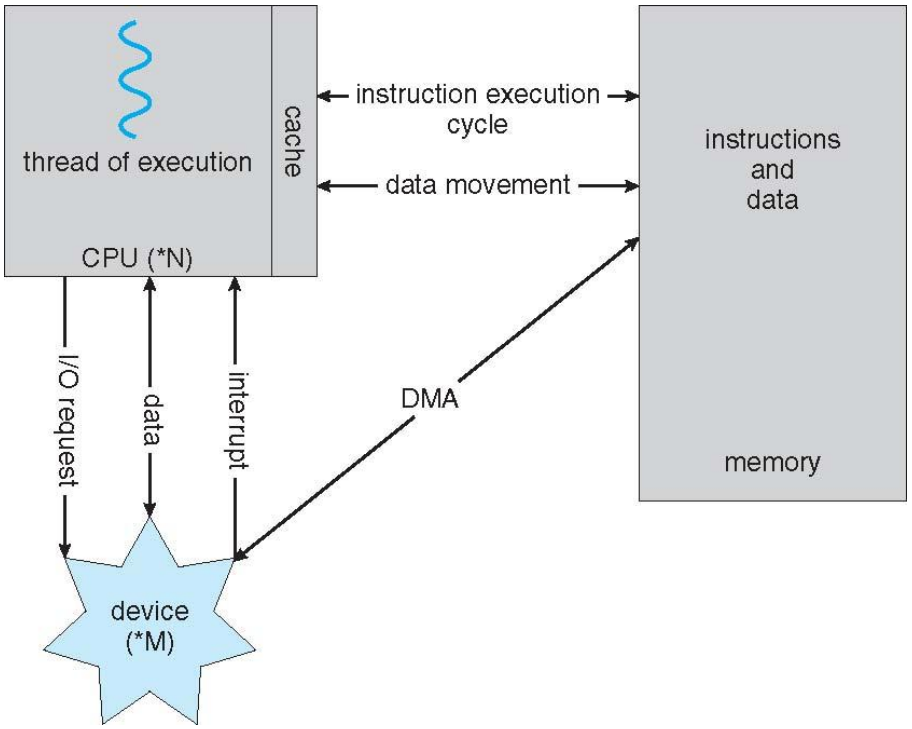
I/O Data Transfer

Programmed I/O

- CPU transfers data between memory and I/O data registers
- Time consuming for high-speed devices

Direct memory access (DMA)

- OS provides starting address in memory
- I/O controller transfers to/from memory autonomously
- Controller interrupts on completion or error



DMA/Cache Interaction

If DMA writes to a memory block that is cached

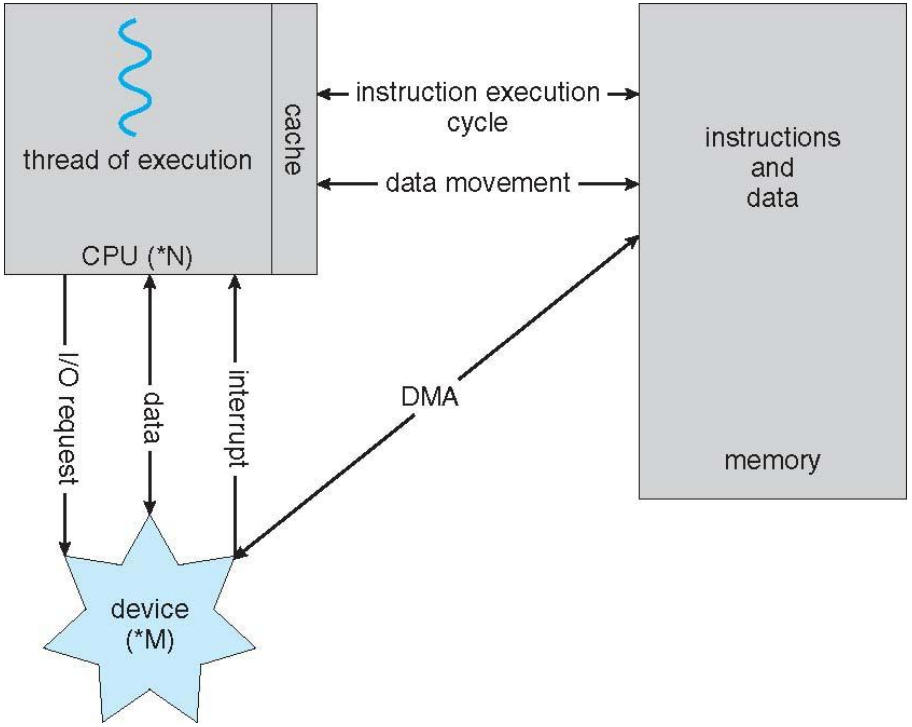
- Cached copy becomes stale

If write-back cache has dirty block, and DMA reads memory block

- Reads stale data

Need to ensure cache coherence

- Flush blocks from cache if they will be used for DMA
- Or use non-cacheable memory locations for I/O



DMA/VM Interaction

OS uses virtual addresses for memory

- DMA blocks may not be contiguous in physical memory

Should DMA use virtual addresses?

- Would require controller to do translation

If DMA uses physical addresses

- May need to break transfers into page-sized chunks
- Or chain multiple transfers
- Or allocate contiguous physical pages for DMA (common case)

I/O vs. CPU Performance

Amdahl's Law

- "The overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used"
- = Bottleneck part is critical

Parallelism increases compute performance

- -> Don't neglect I/O performance

I/O vs. CPU Performance

Example

- Benchmark takes 90s CPU time, 10s I/O time
- Double the number of CPUs/2 years
- I/O speed unchanged

Year	CPU time	I/O time	Elapsed time	% I/O time
now	90s	10s	100s	10%
+2	45s	10s	55s	18%
+4	23s	10s	33s	31%
+6	11s	10s	21s	47%

RAID

Redundant Array of Inexpensive (Independent) Disks

- Use multiple smaller disks (c.f. one large disk)
- Parallelism improves performance
- Plus extra disk(s) for redundant data storage

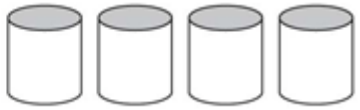
Provides fault tolerant storage system

- Especially if failed disks can be "hot swapped“
- RAID 0, 1, 2, 3, 4, 5, 6 level

RAID 0

- No redundancy
- Just stripe data over multiple disks
- But it does improve performance

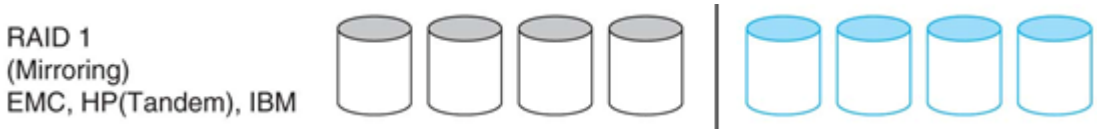
RAID 0
(No redundancy)
Widely used



RAID 1 & 2

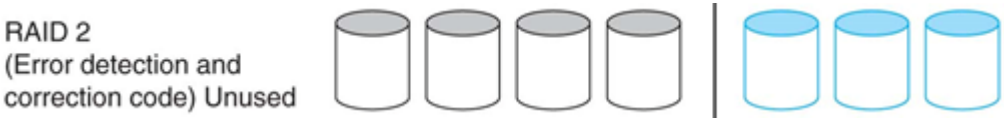
RAID 1: Mirroring

- N + N disks, replicate data
  - Write data to both data disk and mirror disk
  - On disk failure, read from mirror



RAID 2: Error correcting code (ECC)

- N + E disks (e.g., 10 + 4)
- Split data at bit level across N disks
- Generate E-bit ECC
- Too complex, not used in practice

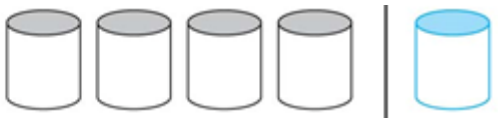


RAID 3: Bit-Interleaved Parity

N + 1 disks

- Data striped across N disks at byte level
- Redundant disk stores parity
- Read access
  - Read all disks
- Write access
  - Generate new parity and update all disks
- On failure
  - Use parity to reconstruct missing data

Not widely used

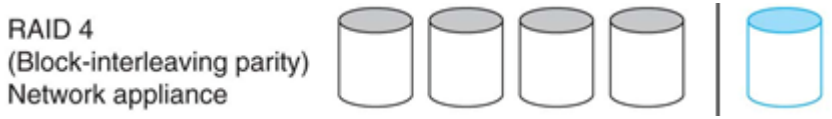


RAID 4: Block-Interleaved Parity

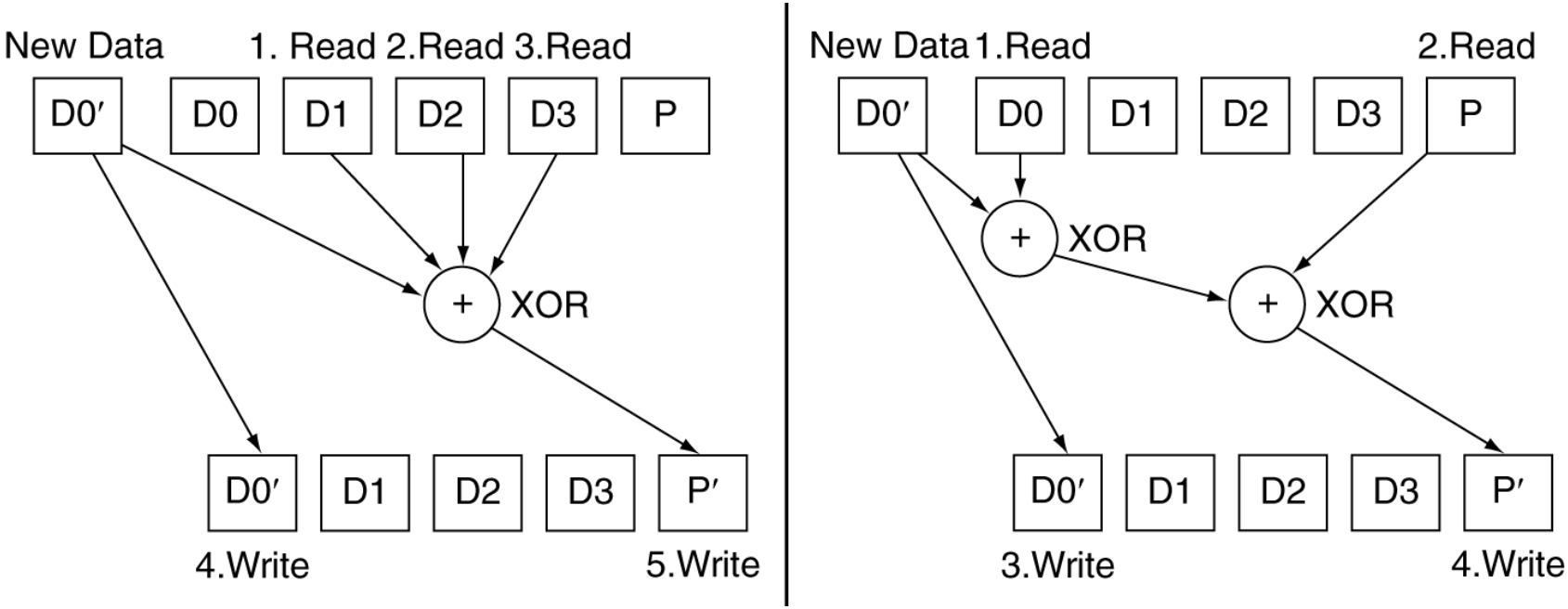
N + 1 disks

- Data striped across N disks at block level
- Redundant disk stores parity for a group of blocks
- Read access
  - Read only the disk holding the required block
  - vs. Read all disk in RAID 3
- Write access
  - Just read disk containing modified block, and parity disk
  - Calculate new parity, update data disk and parity disk
- On failure
  - Use parity to reconstruct missing data

Not widely used



RAID 3 vs RAID 4

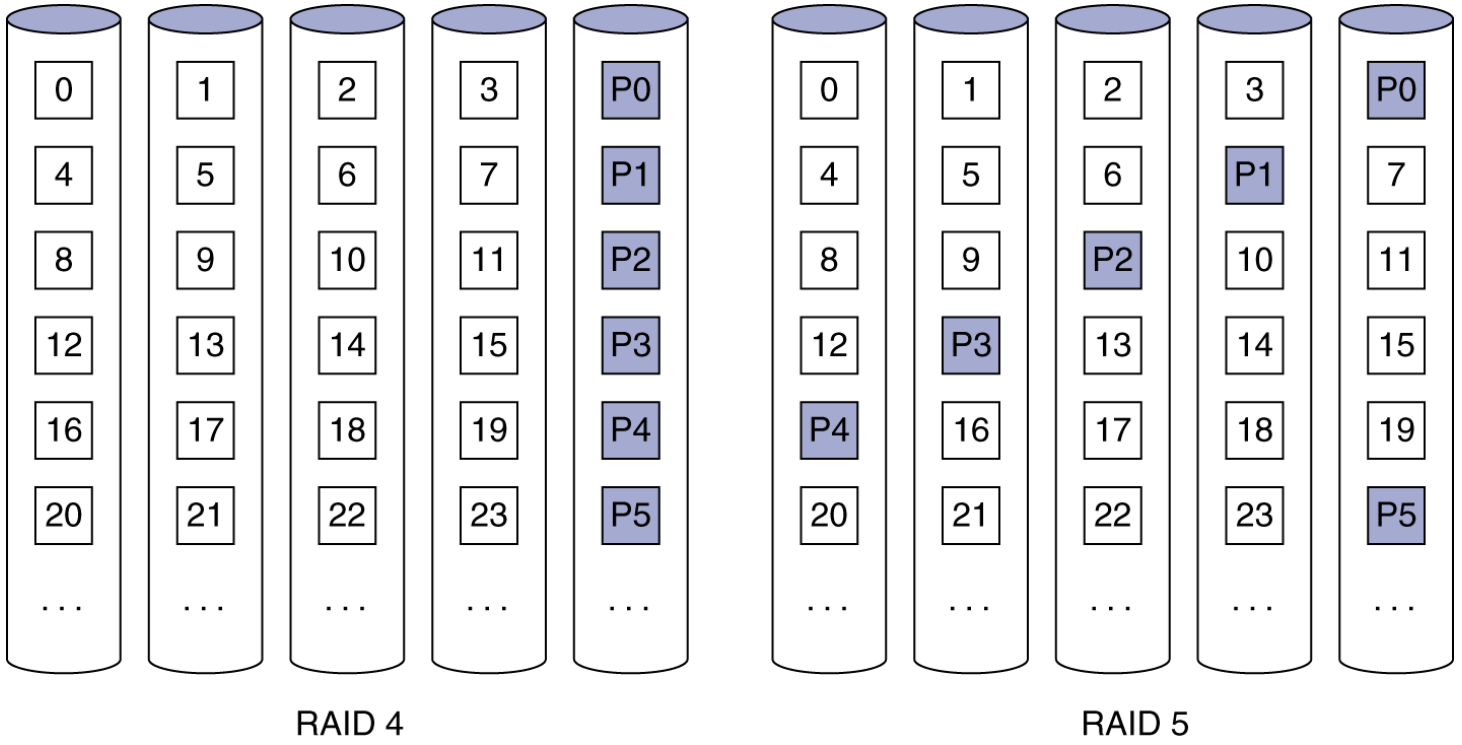


RAID 5: Distributed Parity

N + 1 disks

- Like RAID 4, but parity blocks distributed across disks
  - Avoids parity disk being a bottleneck

Widely used



RAID 6: P + Q Redundancy

N + 2 disks

- Like RAID 5, but two lots of parity
- Greater fault tolerance through more redundancy

Multiple RAID

- More advanced systems give similar fault tolerance with better performance

RAID Summary

RAID can improve performance and availability

- High availability requires hot swapping

Assumes independent disk failures

- Too bad if the building burns down!

I/O System Design

Satisfying latency requirements

- For time-critical operations

Maximizing throughput

- Find "weakest link" (lowest-bandwidth component)
- Configure to operate at its maximum bandwidth

If system is under heavy load

- Simple analysis is insufficient
- Need to use queuing models or simulation

Server Computers

Applications are increasingly run on servers

- Web search, office apps, virtual worlds, ...

Requirements for large data center servers

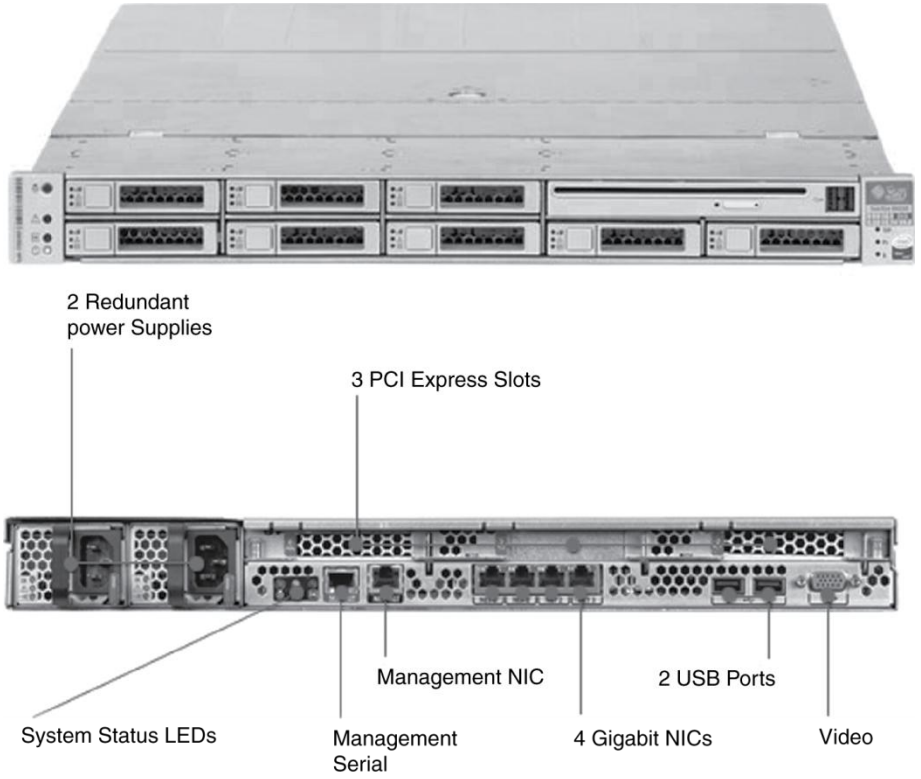
- Multiple processors, networks connections, massive storage
- Space and power constraints

Server equipment built for 19inch racks

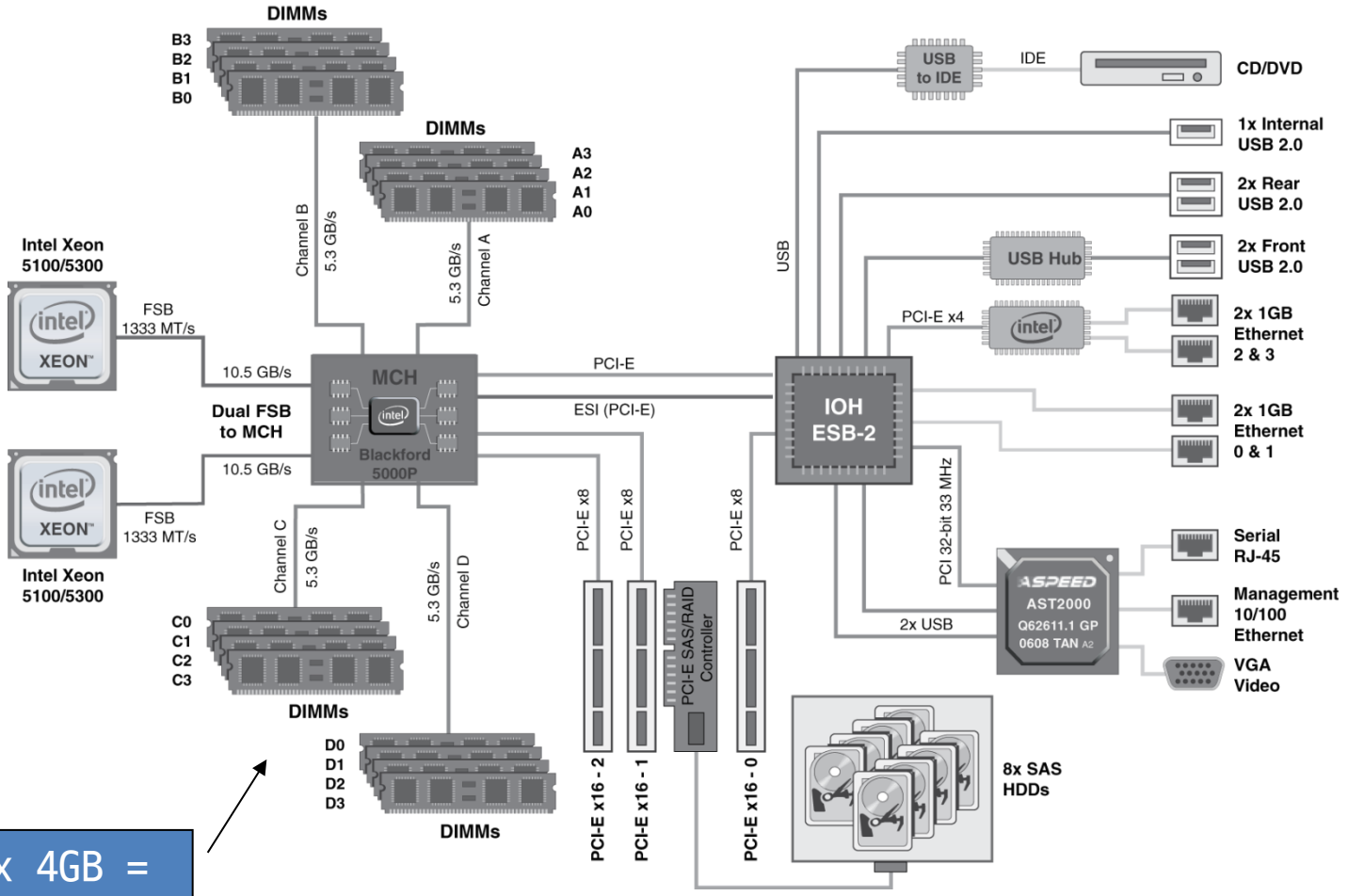
- Multiples of 1.75inch (1U) high

Rack-Mounted Servers

Sun Fire x4150 1U server



Sun Fire x4150 1U server



4 cores each

16 x 4GB = 64GB DRAM

I/O System Design Example

Given a Sun Fire x4150 system with

- Workload: 64KB disk reads
  - Each I/O op requires 200,000 user-code instructions and 100,000 OS instructions
- Each CPU: 10^9 instructions/sec
- FSB(front-side bus): 10.6 GB/sec peak
- DRAM DDR2 667MHz: 5.336 GB/sec
- PCI-E 8× bus: 8 × 250MB/sec = 2GB/sec
- Disks: 15,000 rpm, 2.9ms avg. seek time, 112MB/sec transfer rate

What I/O rate can be sustained?

- For random reads, and for sequential reads

Design Example (cont)

I/O rate for CPUs

- Per core: $10^9/(100{,}000 + 200{,}000) = 3{,}333$
- 8 cores: 26,667 ops/sec

Random reads, I/O rate for disks

- Assume actual seek time is 2.9ms/4 and latency is 4ms/2
- Time/op = seek + latency + transfer
  = 2.9ms/4 + 4ms/2 + 64KB/(112MB/s) = 3.3ms
- 303 ops/sec per disk, 2424 ops/sec for 8 disks

Sequential reads

- 112MB/s / 64KB = 1750 ops/sec per disk
- 14,000 ops/sec for 8 disks

Design Example (cont)

PCI-E I/O rate

- 2GB/sec / 64KB = 31,250 ops/sec

DRAM I/O rate

- 5.336 GB/sec / 64KB = 83,375 ops/sec

FSB I/O rate

- Assume we can sustain half the peak rate
- 5.3 GB/sec / 64KB = 81,540 ops/sec per FSB
- 163,080 ops/sec for 2 FSBs

Weakest link: disks

- 2424 ops/sec random, 14,000 ops/sec sequential
- Other components have ample headroom to accommodate these rates

Fallacy: Disk Dependability

If a disk manufacturer quotes MTTF as 1,200,000hr (140yr)

- A disk will work that long

Wrong: this is the mean time to failure

- What is the distribution of failures?
- What if you have 1000 disks

How many will fail per year?

- 1year = 8760hrs

$$\text{Annual Failure Rate (AFR)} = \frac{1000 \text{ disks} \times 8760 \text{ hrs/disk}}{1200000 \text{ hrs/failure}} = 0.73\%$$

Fallacies

Disk failure rates are as specified

- Studies of failure rates in the field say NO
- Schroeder and Gibson
  - 0.6% to 0.8%
  - 2% to 4% (real world)
- Pinheiro, et al.
  - 1.5%
  - 1.7% (first year) to 8.6% (third year)

Pitfall: Backing Up to Tape

Magnetic tape used to have advantages

- Removable, high capacity

Advantages eroded by disk technology developments

Makes better sense to replicate data

- E.g., RAID, remote mirroring

Fallacy: Disk Scheduling

Best to let the OS schedule disk accesses

- But modern drives deal with logical block addresses
  - Map to physical track, cylinder, sector locations
  - Also, blocks are cached by the drive
- OS is unaware of physical locations
  - Reordering can reduce performance
  - Depending on placement and caching

Pitfall: Peak Performance

Peak I/O rates are nearly impossible to achieve

- Usually, some other system component limits performance
- E.g., transfers to memory over a bus
- E.g., PCI bus: peak bandwidth ~133 MB/sec
  - In practice, max 80MB/sec sustainable

Concluding Remarks

I/O performance measures

- Throughput, response time
- Dependability and cost also important

Buses used to connect CPU, memory, I/O controllers

- Polling, interrupts, DMA

RAID

- Improves performance and dependability