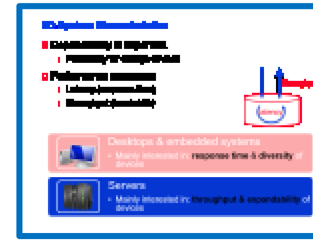
IT4272E-COMPUTER SYSTEMS

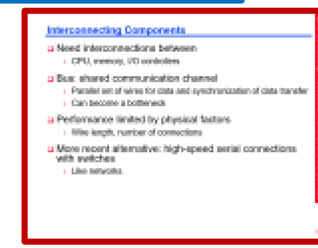
Chapter 6: Storage and Other I/O Topics

[with materials from *Computer Organization and Design, 4th Edition*, Patterson & Hennessy, © 2008, MK]

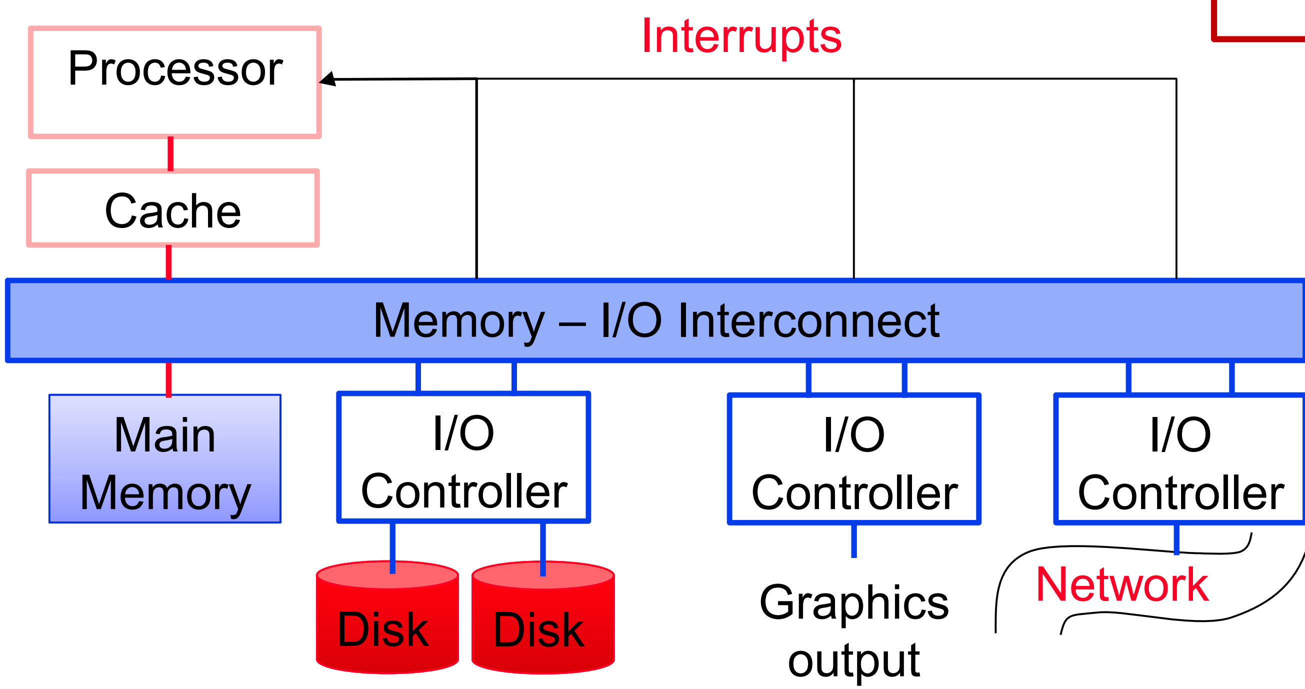
Introduction

- I/O devices can be characterized by
  - Behaviour: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec, transfers/sec



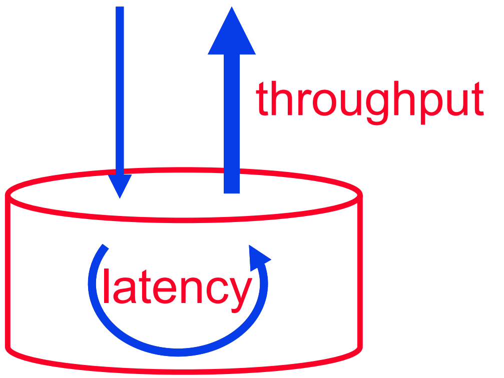


- I/O bus connections



I/O System Characteristics

- Dependability is important
  - Particularly for storage devices
- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)



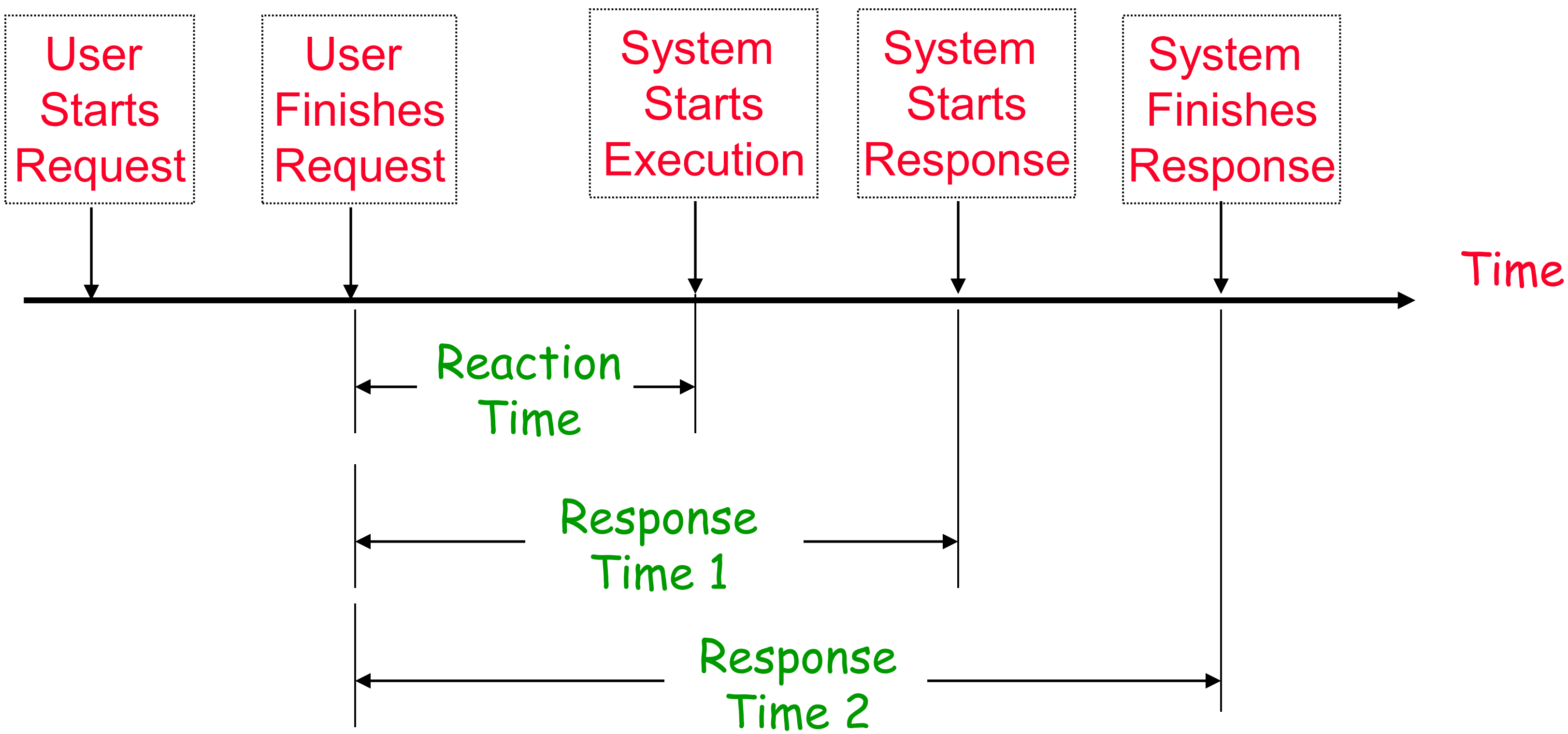
Desktops & embedded systems

- Mainly interested in: **response time** & **diversity** of devices

Servers

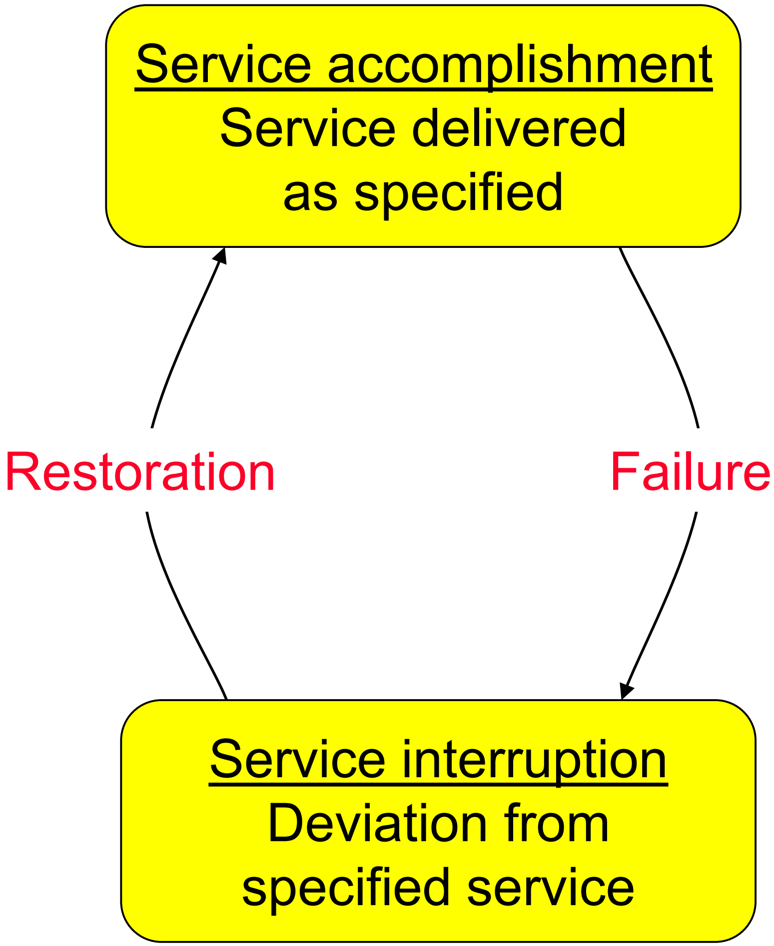
- Mainly interested in: **throughput & expandability** of devices

Respond Time



- Can have two measures of response time
  - Both ok, but 2 preferred if execution long

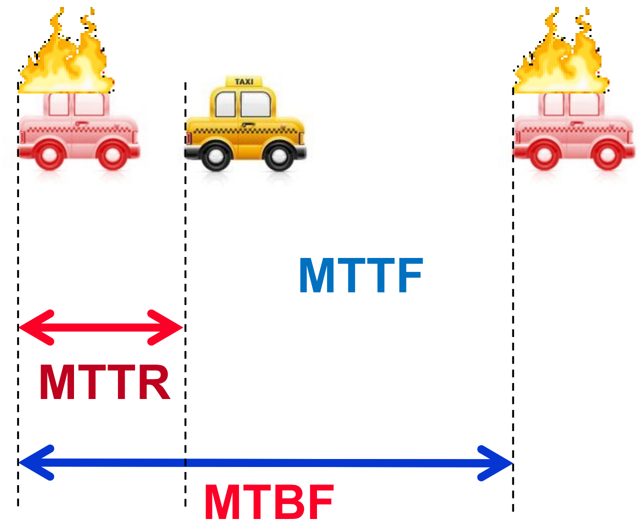
Dependability



- 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)



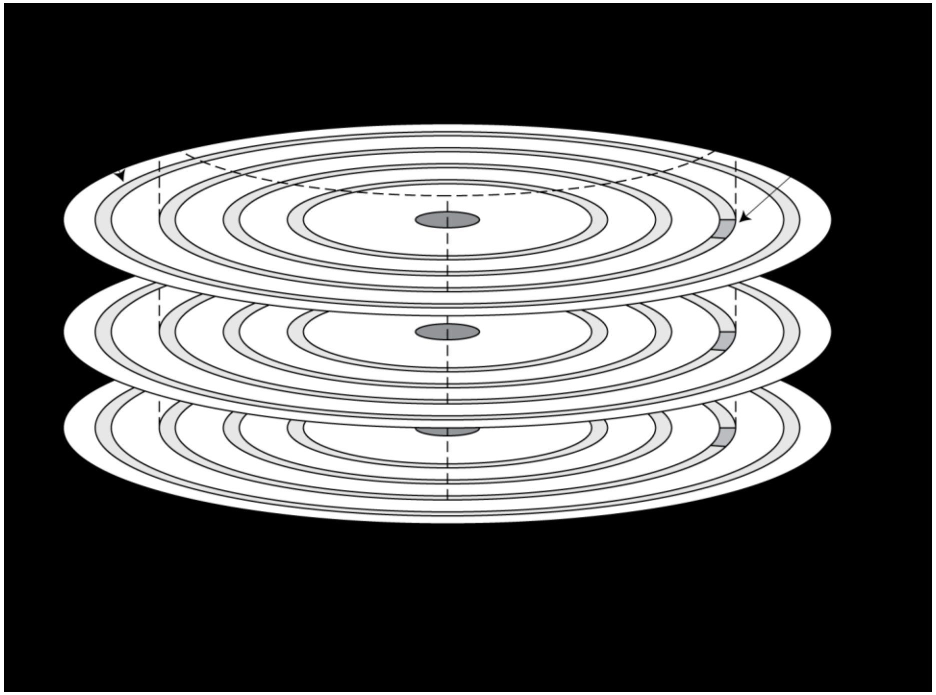
Improving Availability

Increase MTTF: fault avoidance, fault tolerance, fault forecasting

Reduce MTTR: improved tools and processes for diagnosis and repair

Disk Storage

- Nonvolatile, rotating magnetic storage

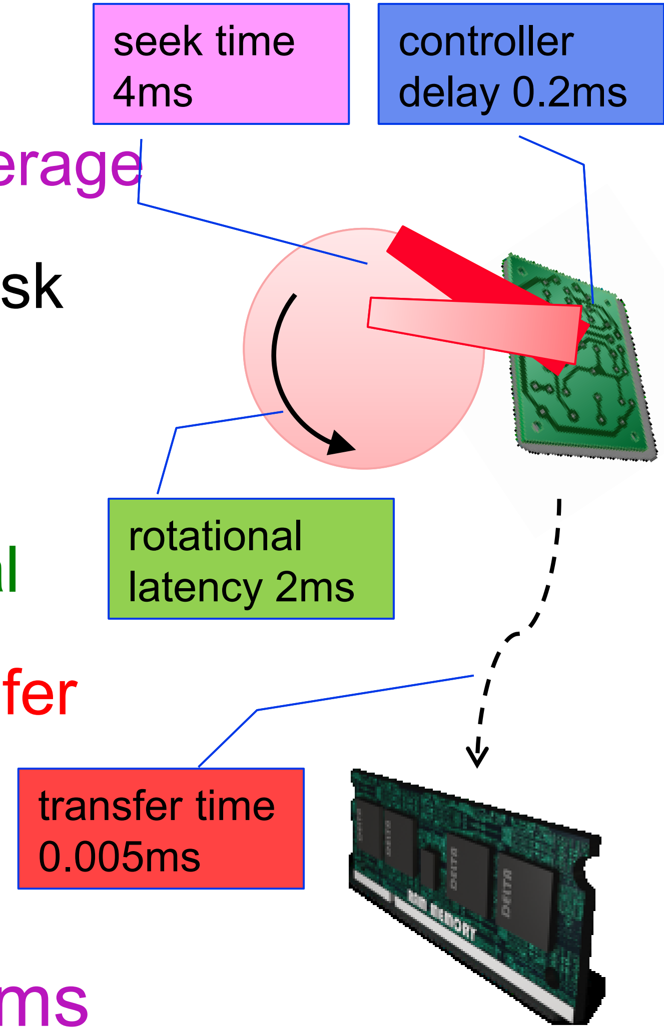


Disk Sectors and Access

- 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



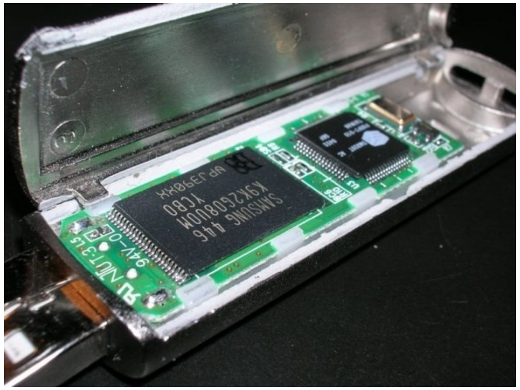
Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times (25%~33%)
- 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

anticipation /æn,tisi'peiʃn/: sự đoán trước

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 1000’s of accesses
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks

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

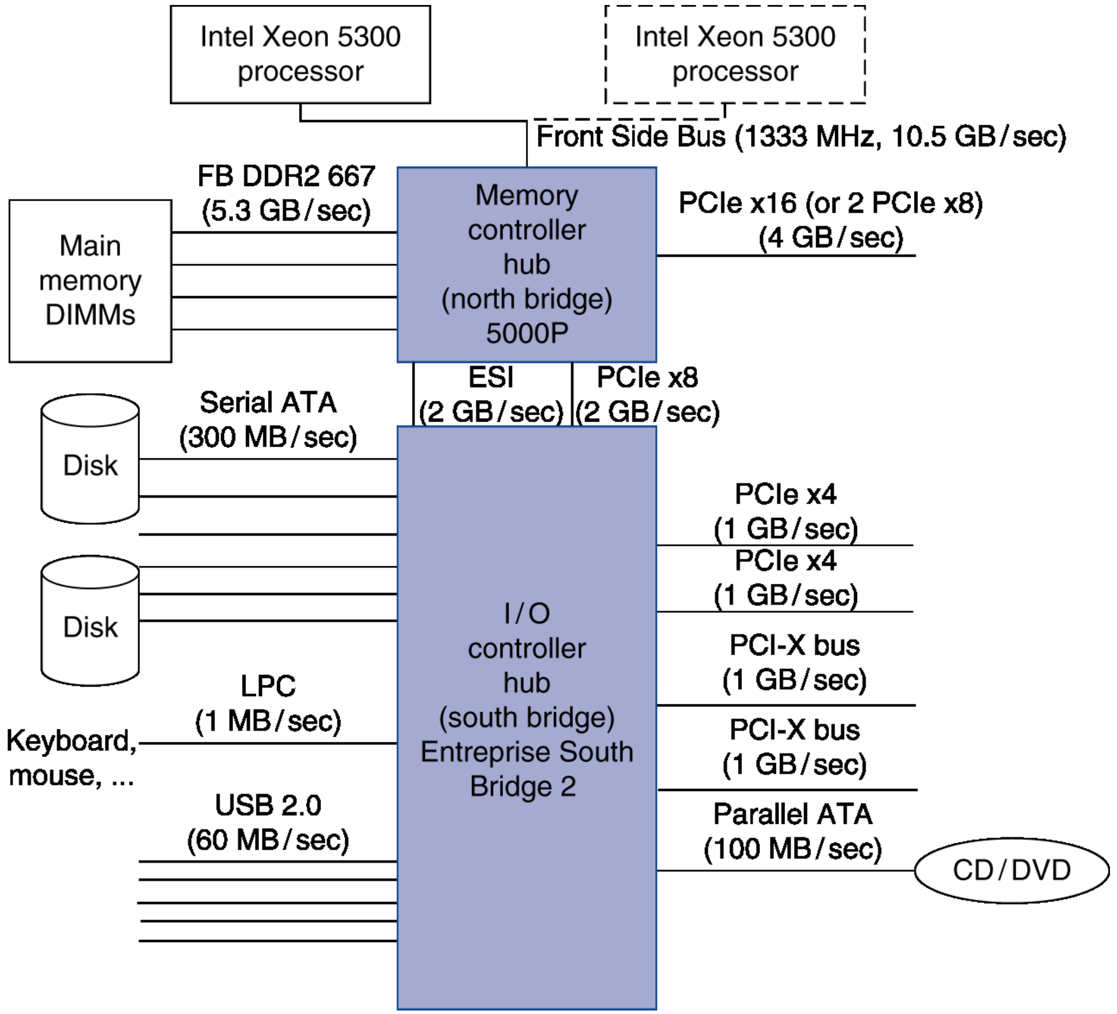
Bus Signals and Synchronization

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions
- Synchronous
  - Uses a bus clock
- Asynchronous
  - Uses request/acknowledge control lines for handshaking

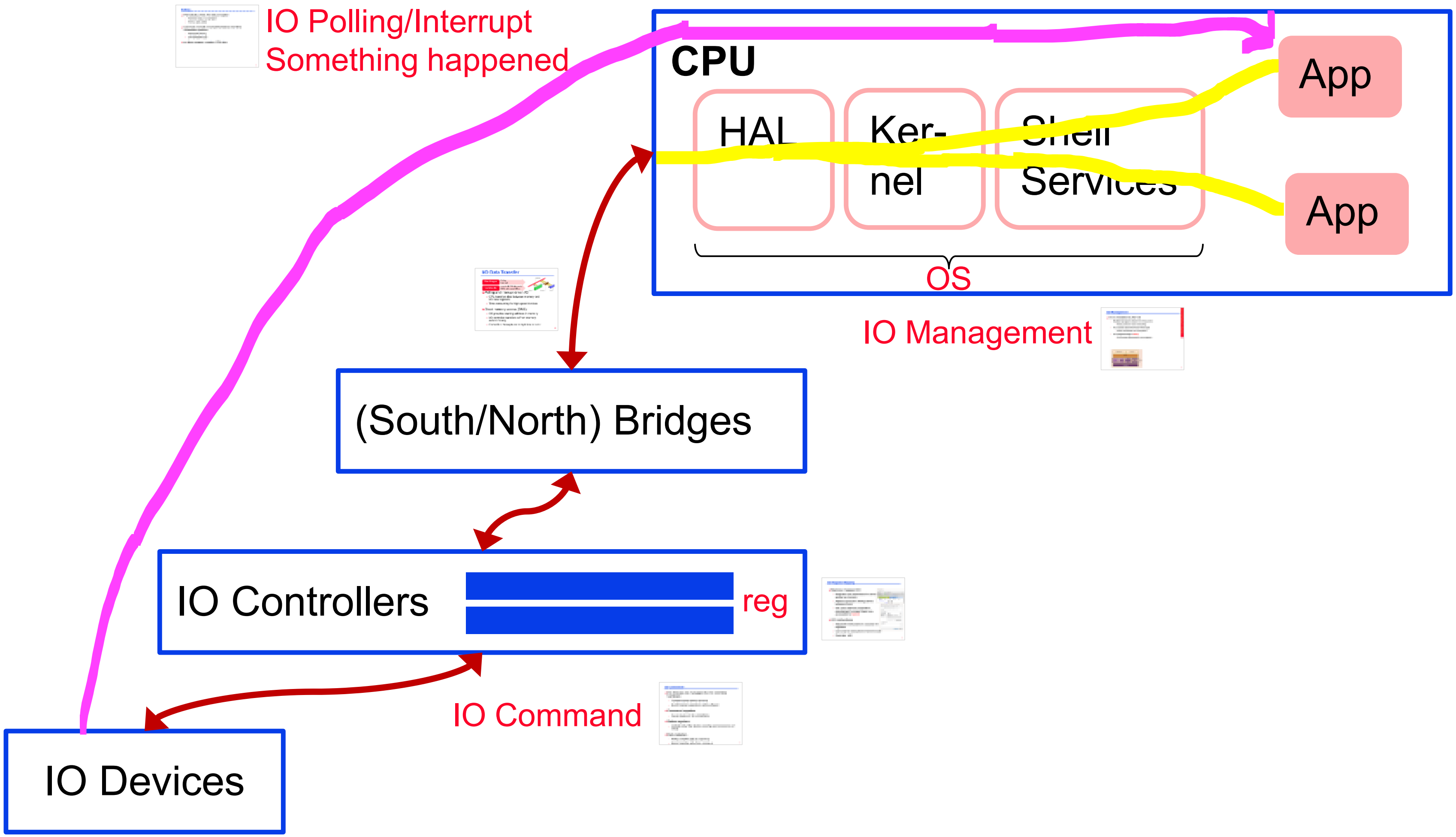
I/O Bus Examples

	Firewire	USB 2.0	PCI Express	Serial ATA	Serial Attached SCSI
Intended use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Data width	4	2	2/lane	4	4
Peak bandwidth	50MB/s or 100MB/s	0.2MB/s, 1.5MB/s, or 60MB/s	250MB/s/lane 1×, 2×, 4×, 8×, 16×, 32×	300MB/s	300MB/s
Hot pluggable	Yes	Yes	Depends	Yes	Yes
Max length	4.5m	5m	0.5m	1m	8m
Standard	IEEE 1394	USB Implementers Forum	PCI-SIG	SATA-IO	INCITS TC T10

Typical x86 PC I/O System



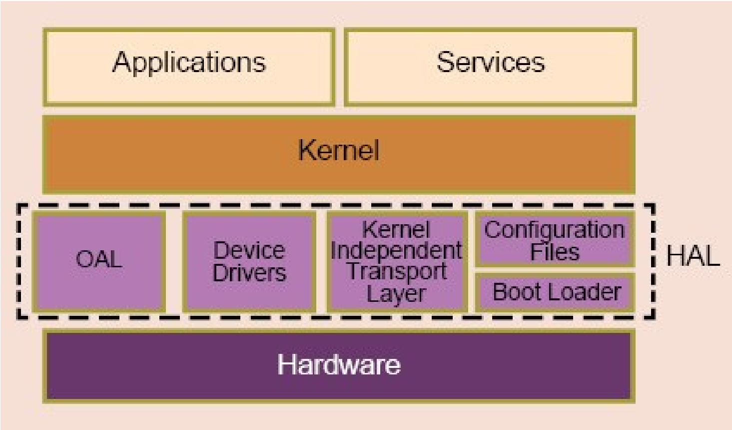
IO Model



IO Polling/Interrupt
Something happened
CPU
HAL
Ker-
nel
Shell
Services
App
App
OS
IO Management
(South/North) Bridges
IO Controllers
reg
IO Command
IO Devices

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

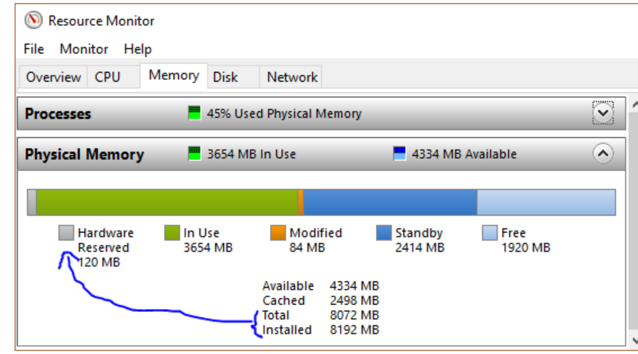


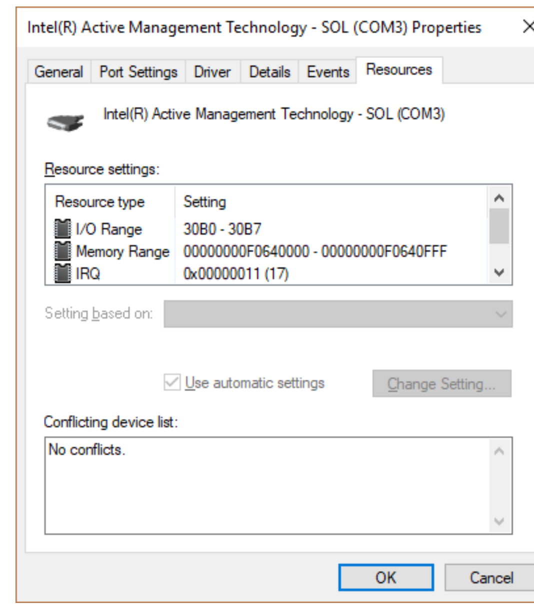
I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software
- Command registers
  - Cause device to do something
- Status registers
  - Indicate what the device is doing and occurrence of errors
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device

I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel
- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86





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

Interrupts

- 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
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

Interrupts: Examples

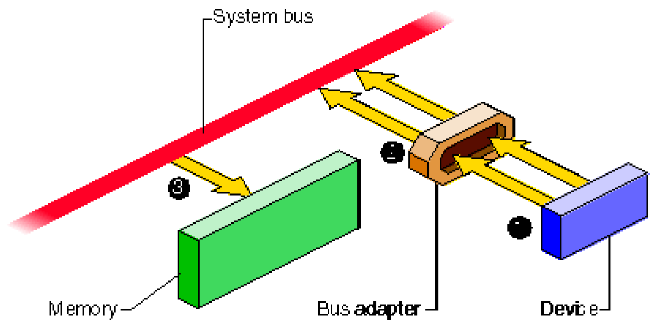
```
tiennd@tiennd:~$ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:         76          0          0          0   IO-APIC-edge      timer
  1:      13090          3          0          0   IO-APIC-edge      i8042
  8:          1          0          0          0   IO-APIC-edge      rtc0
  9:       1064          0          0          0   IO-APIC-fasteoi   acpi
 12:     213112          2          0          0   IO-APIC-edge      i8042
 16:     915415          0          0          0   IO-APIC-fasteoi   ehci_hcd:usb1, nvidia
 17:     282256        347          0      22193   IO-APIC-fasteoi   ath9k, snd_hda_intel
 23:      74438          0          0          0   IO-APIC-fasteoi   ehci_hcd:usb2
 41:      81104          0          0          0   PCI-MSI-edge      ahci
 42:          0          0          0          0   PCI-MSI-edge      eth0
 43:         10          0          0          0   PCI-MSI-edge      mei
 44:        286          0          0          0   PCI-MSI-edge      snd_hda_intel
NMI:         39        609        372        359   Non-maskable interrupts
LOC:    1377006    1000228     742879     758045   Local timer interrupts
SPU:          0          0          0          0   Spurious interrupts
```

IRQ Number | the number of interrupt handled by CPU Core | Interrupt Type | Device Name

- Example with Asus K43SJ
- Each CPU in the system has its own column and its own number of interrupts per IRQ.
- IRQ0: system timer; IRQ1&12: keyboard&mouse.

I/O Data Transfer

When it happen	• Polling • Interrupt
How to transfer	• mem/io → CPU → mem/io • DMA, with cache/VMem



- Polling and interrupt-driven 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

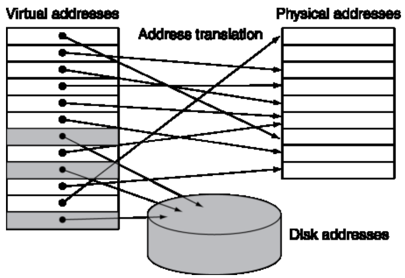
DMA/Cache Interaction

- 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

stale /steil/ (adj): cũ rích, mất hiệu lực
coherence /kou'hiərəns/: tính nhất quán

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
  - C
    for DMA

contiguous /kən'tigjuəs/: liền kề, bên cạnh

Measuring I/O Performance

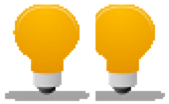
- I/O performance depends on

Hardware
- CPU
- Memory
- Controllers
- Buses

Software
- OS
- DBMS
- Application

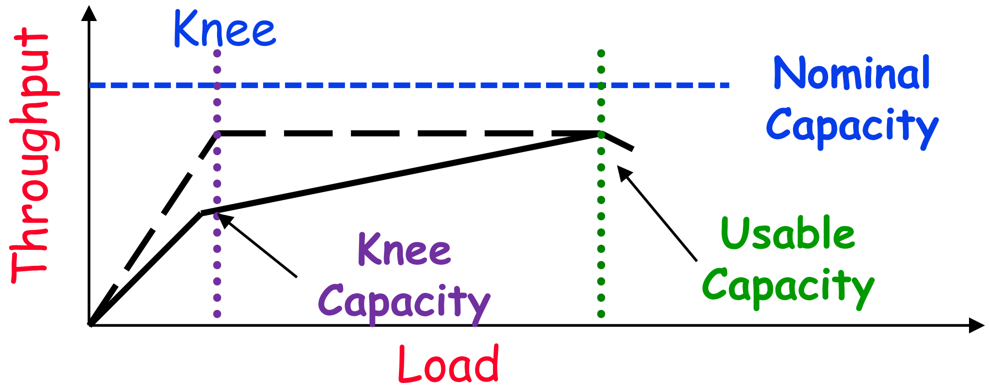
Workload
- Request rates
- Patterns

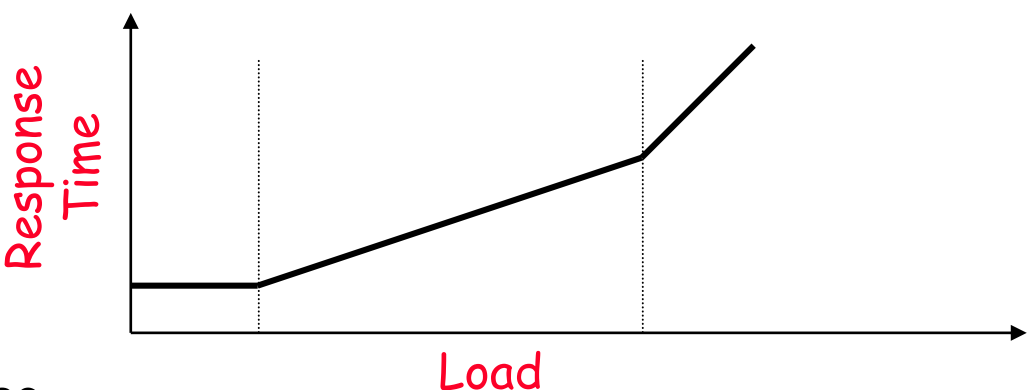
- I/O system design can trade-off between response time and throughput
  - Measurements of throughput often done with constrained response-time



Throughput vs Respond Time

- *Throughput* increases as load increases, to a point





- **Nominal capacity** is ideal (ex: 10 Mbps)
- **Usable capacity** is achievable (ex: 9.8 Mbps)
- **Knee** is where response time goes up rapidly for small increase in throughput

Transaction Processing Benchmarks

- Transactions
  - Small data accesses to a DBMS
  - **Interested in I/O rate**, not data rate
- Measure throughput
  - Subject to response time limits and failure handling
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Overall cost per transaction
- Transaction Processing Council (TPC) benchmarks (www.tcp.org)
  - TPC-APP: B2B application server and web services
  - TCP-C: on-line order entry environment
  - TCP-E: on-line transaction processing for brokerage firm
  - TPC-H: decision support — business oriented ad-hoc queries

File System & Web Benchmarks

- SPEC System File System (SFS)
  - Synthetic workload for NFS server, based on monitoring real systems
  - Results
    - Throughput (operations/sec)
    - Response time (average ms/operation)
- SPEC Web Server benchmark
  - Measures simultaneous user sessions, subject to required throughput/session
  - Three workloads: Banking, Ecommerce, and Support

I/O vs. CPU Performance

- Amdahl's Law
  - Don't neglect I/O performance as parallelism increases compute performance
- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O 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%

Amdahl and Gustafson's Laws

- **Amdahl's Law:** The speed up achieved through parallelization of **a program** is limited by the percentage of its workload that is inherently serial.
  Speedup(N)= 1 / (S + (1-S)/N) < 1/S
  N: processors, S: proportion of none-parallelization

- **Gustafson's Law:** With increasing data size, the speedup obtained though parallelization increases, because the parallel work increases with data size.
  Speedup(N) = N – S(N-1) = N(1-S) + S

denominator /di'nɔmineitə/ mẫu số; mẫu thức
numerator /'nju:məreitə/: tử số *fraction /'frækʃn/: phân số*

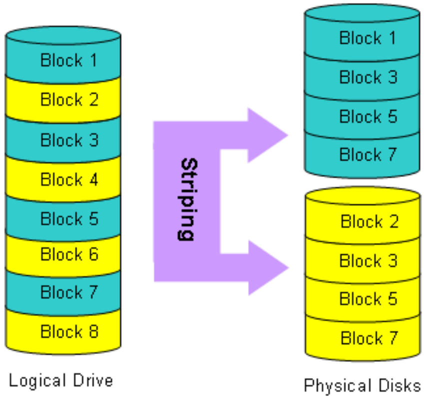
Exercise

- Assume accelerating a machine by adding a vector mode to it. When a computation is run in vector mode, it is 20 times faster than the normal mode of execution.
  However, the software program cannot be parallized absolutely and CPU's speedup of this program is only 2. So how many per cent the software cannot run in vector mode?

Answer:

RAID

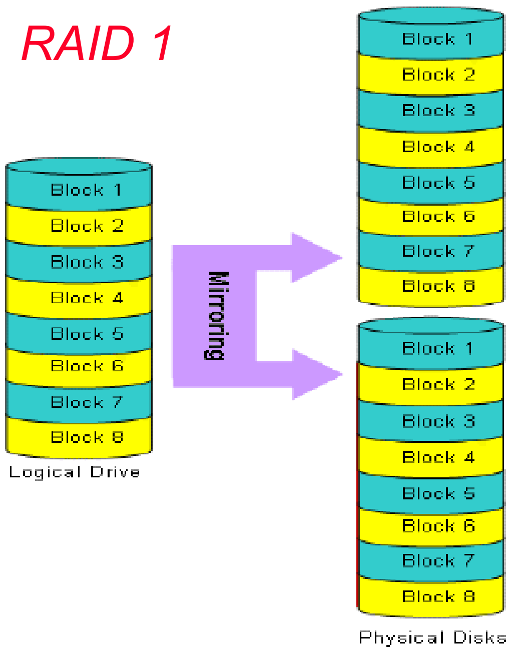
- Redundant Array of Inexpensive (Independent) Disks
  - Use multiple smaller disks (c.f. one large disk)
  - Parallelism improves performance
  - Plus extra disk(s) for redundant data storage
- Provides fault tolerant storage system
  - Especially if failed disks can be "hot swapped"
- RAID 0, stripping
  - No redundancy ("AID"?)
    - Just stripe data over multiple disks
  - But it does improve performance



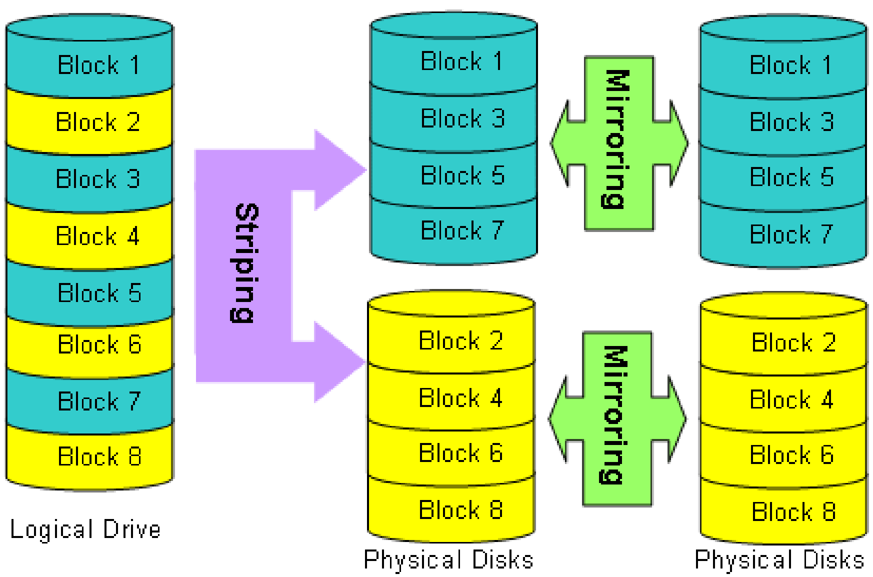
RAID 1 & 0+1

- RAID 1: Mirroring
  - N + N disks, replicate data
    - Write data to both data disk and mirror disk
    - On disk failure, read from mirror

RAID 1

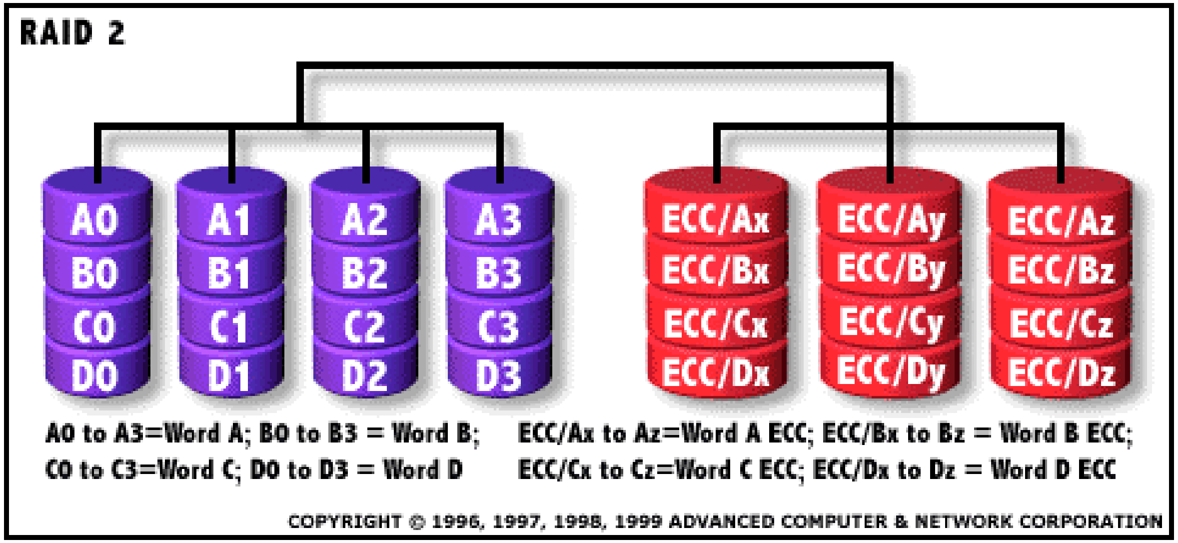


RAID 0+1: Stripping+ Mirroring



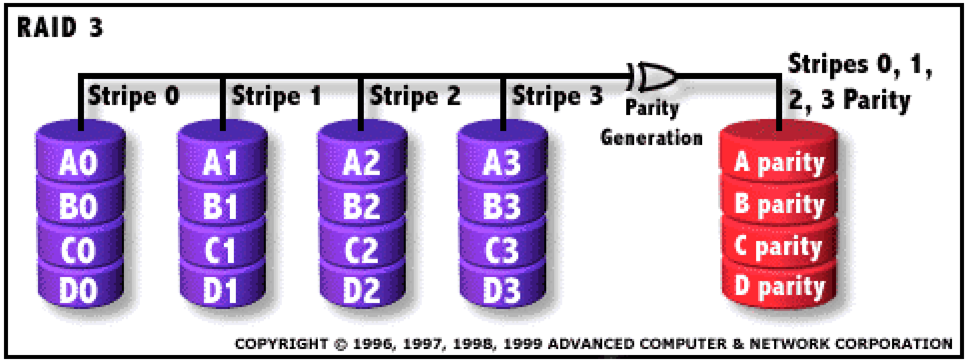
RAID 2, bit stripped

- 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



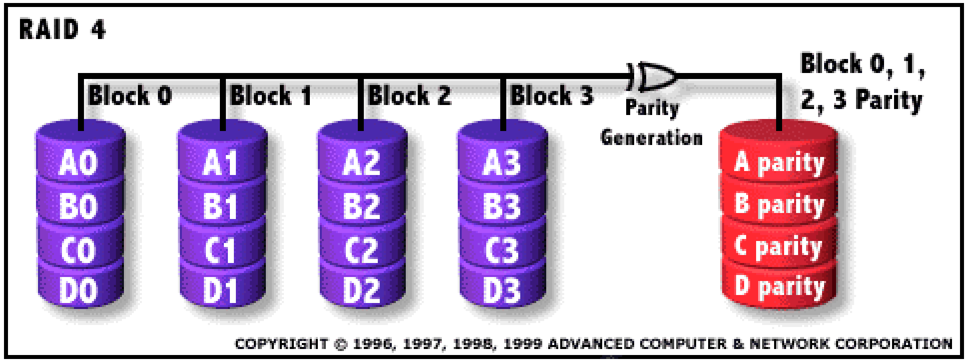
RAID 3: Bit-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at **byte level**
  - Redundant disk stores parity (dedicated parity disk)
  - **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

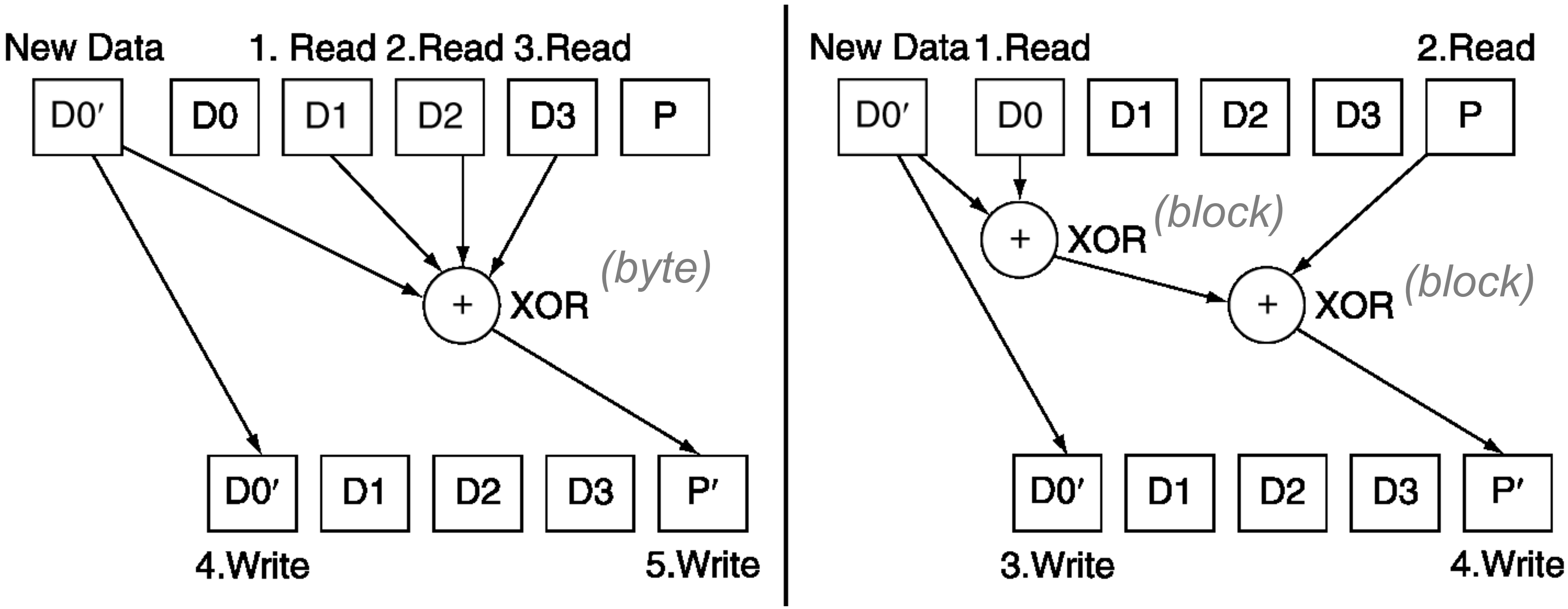


RAID 4: Block-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at **block level** *(16, 32, 64,128 kB)*
  - Redundant disk stores parity for a group of blocks
  - Read access
    - Read only the disk holding the required block
  - 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

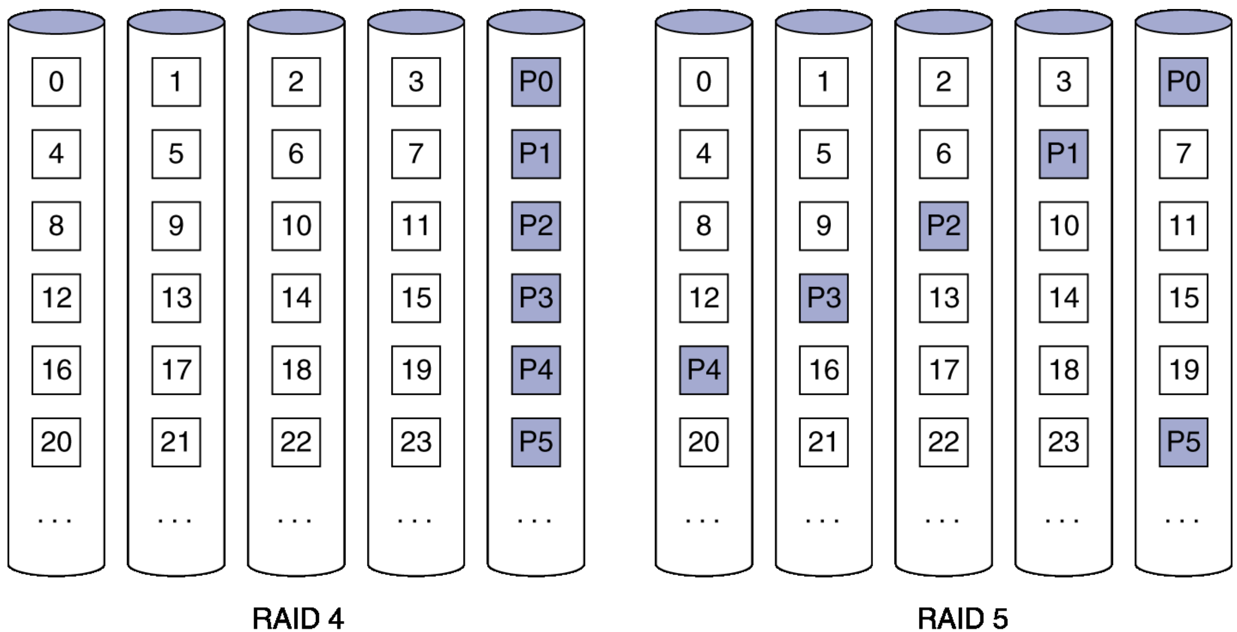


Read 3 disks to get 3 bytes, and then create parity byte

Read 2 disks to get 2 blocks (include parity block), and then create new parity block

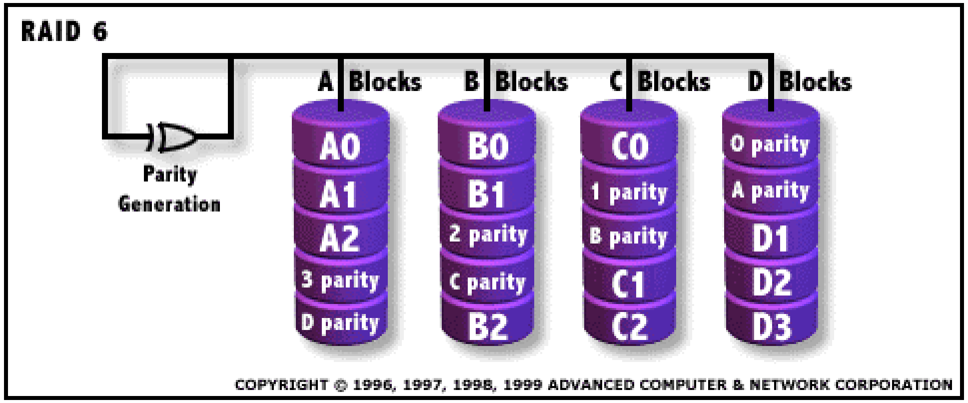
RAID 5: Distributed Parity

- 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

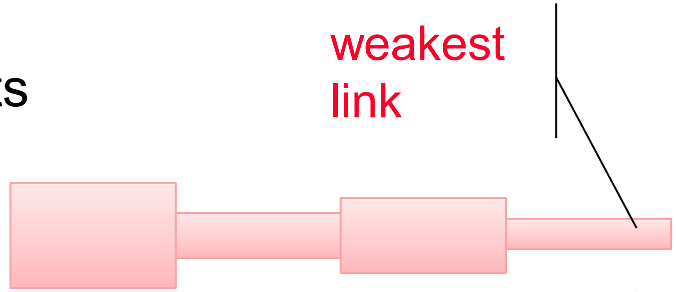


RAID Summary

- RAID can improve performance and availability
  - High availability requires hot swapping
- Assumes independent disk failures
  - Too bad if the building burns down!
- See “Hard Disk Performance, Quality and Reliability”
  - http://www.pcguide.com/ref/hdd/perf/index.htm

I/O System Design

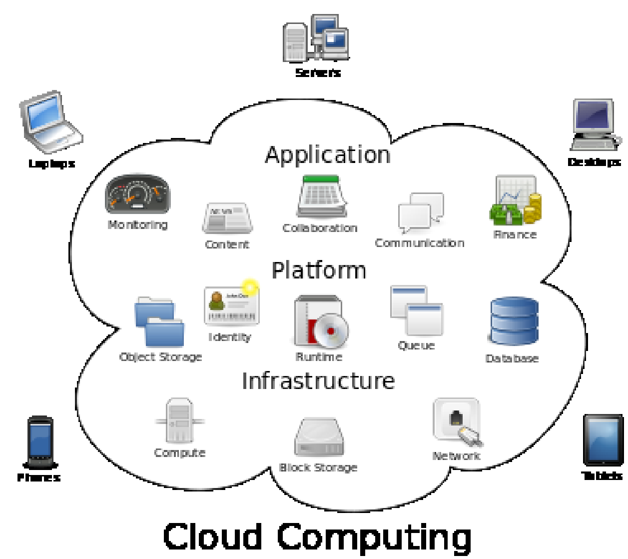
- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components
- Maximizing throughput



  - Find "weakest link" (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system
- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation

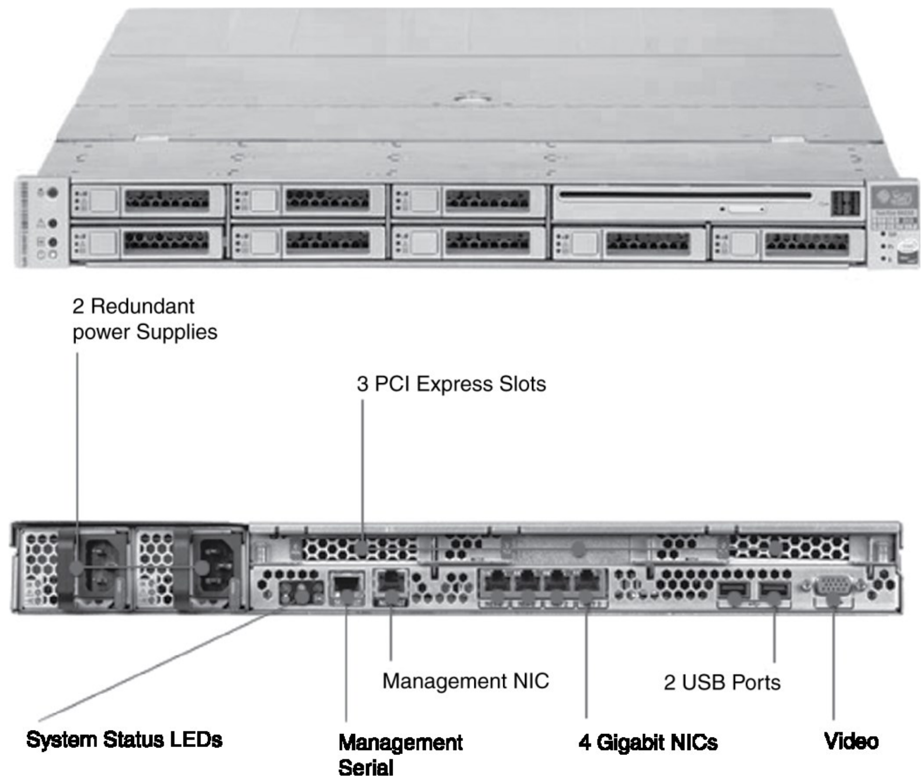
Server Computers

- Applications are increasingly run on servers
  - Web search, office apps, virtual worlds, cloud…
- Requires large data centre servers
  - Multiple processors, networks connections, massive storage
  - Space and power constraints
- Server equipment built for 19” ra
  - Multiples of 1.75” (1U) high

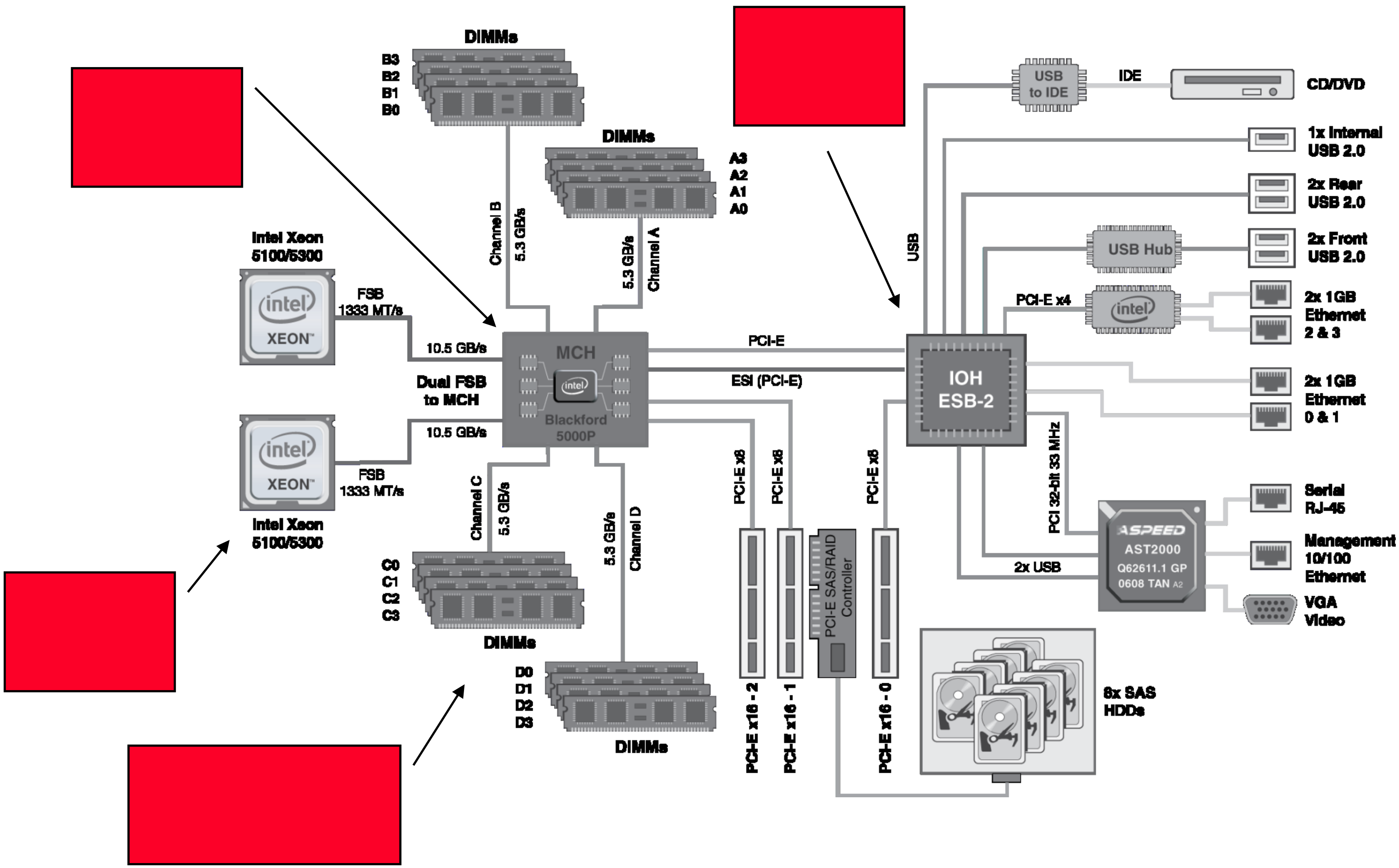


Rack-Mounted Servers

Sun Fire x4150 1U server



Sun Fire x4150 1U server



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: 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 IOs/sec
  - 8 cores: 3,333 x 8 = 26,667 IOs/sec
- Random reads, I/O rate **for disks**
  - Assume actual seek time is average/4
  - Time/op = seek + latency + transfer ~~(+ control time~0)~~ = 2.9ms/4 + 4ms/2 + 64KB/(112MB/s)=3.3ms(per IOs)
  - 1000*1/3.3=303 IOs/sec per disk, 2424 IOs/sec for 8 disks

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

Design Example (cont)

- PCI-E I/O rate
  - 2GB/sec / 64KB = 31,250 IOs/sec
- DRAM I/O rate
  - 5.336 GB/sec / 64KB = 83,375 IOs/sec
- FSB I/O rate
  - Assume we can sustain half the peak rate
  - 10.6 GB/sec /2 / 64KB = 81,540 IOs/sec per FSB
  - 163,080 IOs/sec for 2 FSBs (2 Intel Xeon)
- Weakest link: disks
  - 2424 IOs/sec random, 14,000 IOs/sec sequential
  - Other components have ample headroom to accommodate these rates

ample /'æmpl/ (adj): nhiều, phong phú
headroom: không gian trống

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?

$$\text{Failed Disks} = \frac{1000\text{ disks} \times (24 * 365)\text{ hrs/disk}}{1200000\text{ hrs/failure}} = 7.3$$

$$\text{Annual Failure Rate (AFR)} = \frac{7.3}{1000} = 0.73\%$$

fallacy /ˈfæləsi/ ảo tưởng; ý kiến sai lầm

Fallacies


Prof. Bianca Schroeder

- Disk failure rates are as specified
  - Studies of failure rates in the field
    - Schroeder and Gibson: 2% to 4% vs. 0.6% to 0.8%
    - Pinheiro, *et al.*: 1.7% (first year) to 8.6% (third year) vs. 1.5%
  - Why?
- A 1GB/s interconnect transfers 1GB in one sec
  - But what's a GB?
  - For bandwidth, use 1GB = 10^9 B
  - For storage, use 1GB = 2^{30} B = 1.075×10^9 B
  - So 1GB/sec is 0.93GB in one second
    - About 7% error

Pitfall: Offloading to I/O Processors

- Overhead of managing I/O processor request may dominate
  - Quicker to do small operation on the CPU
  - But I/O architecture may prevent that
- I/O processor may be slower
  - Since it's supposed to be simpler
- Making it faster makes it into a major system component
  - Might need its own coprocessors!

pitfall /'pitfɔ:l/ cạm bẫy
Offload: đẩy dữ liệu ra ngoại vi

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

Tape

IBM System Storage TS1130 Tape Drive

erode /i'roud/: xói mòn, suy giảm

Fallacy: Disk Scheduling

- Best to let the OS schedule disk accesses
  - But modern drives deal with **L**ogical **B**lock **A**ddresses
    - 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

$$LBA = ((C \times HPC) + H) \times SPT + S - 1$$

$$C = LBA \div (SPT \times HPC)$$
$$H = (LBA \div SPT) \bmod HPC$$
$$S = (LBA \bmod SPT) + 1$$

unaware /'ʌbə'weə/: không hay biết

Example: Disk Management

```
tiennd@tiennd:~$ sudo fdisk -l

Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x404ccd9b

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1       356530606   976773119   310121257    f  W95 Ext'd (LBA)
/dev/sda2   *   146801970   356530544   104864287+   7  HPFS/NTFS/exFAT
/dev/sda3            2048   130070527    65034240   83  Linux
/dev/sda4       130070528   146800639     8365056   82  Linux swap / Solaris
/dev/sda5       356530608   482351624    62910508+   7  HPFS/NTFS/exFAT
/dev/sda6       482367488   976773119   247202816   83  Linux
```

- Disk size = <sector num>*<sector size>
  = 976773168 * 512 = 500107862016=465GB

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
    - Collision with DRAM refresh
    - Arbitration contention with other bus masters
  - 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
- I/O benchmarks
  - TPC, SPECSFS, SPECWeb
- RAID
  - Improves performance and dependability