182.092 Computer Architecture

Chapter 6A: Disk Systems

Adapted from

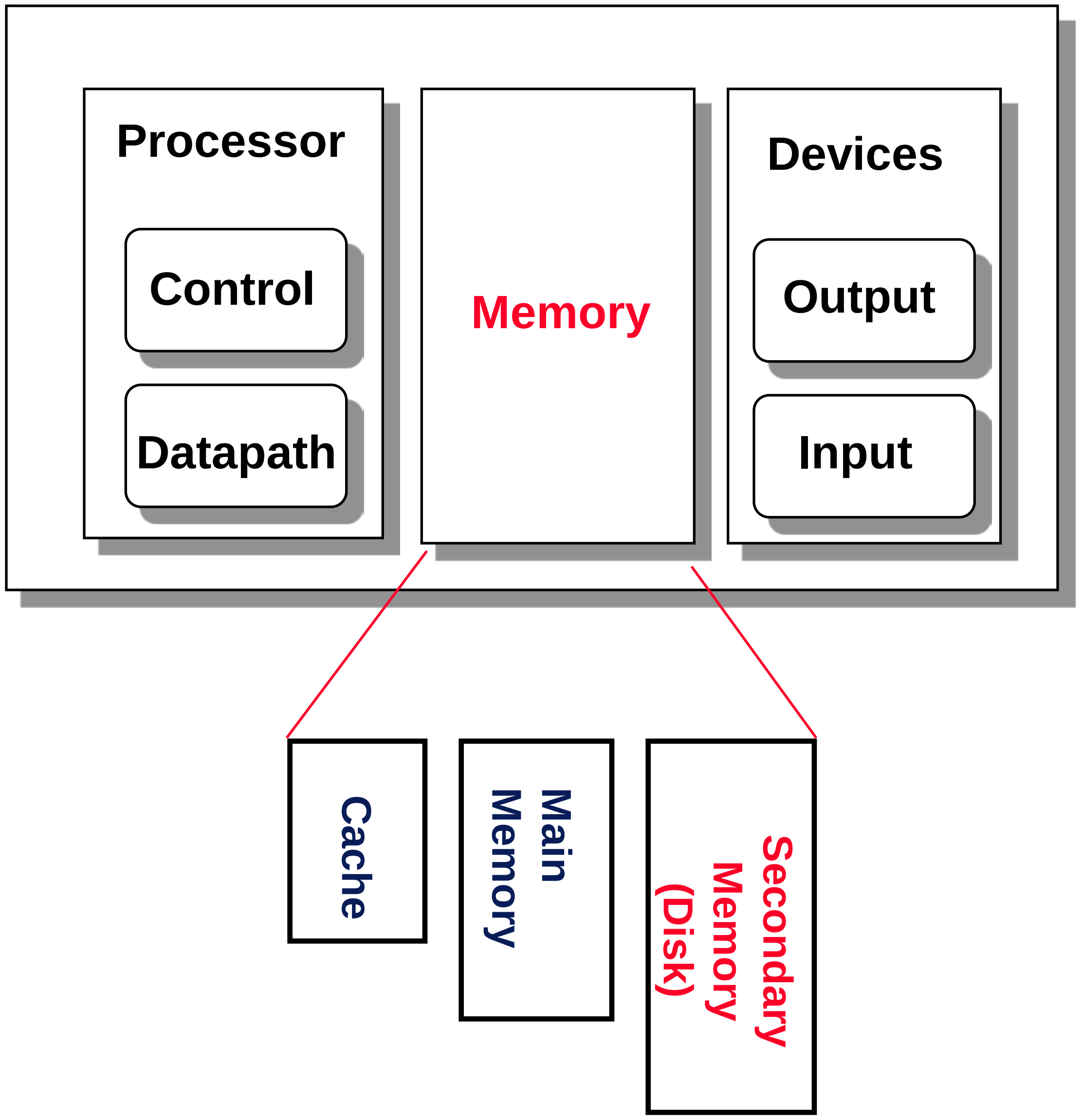
Computer Organization and Design, 4th Edition,

Patterson & Hennessy, © 2008, Morgan Kaufmann Publishers

and

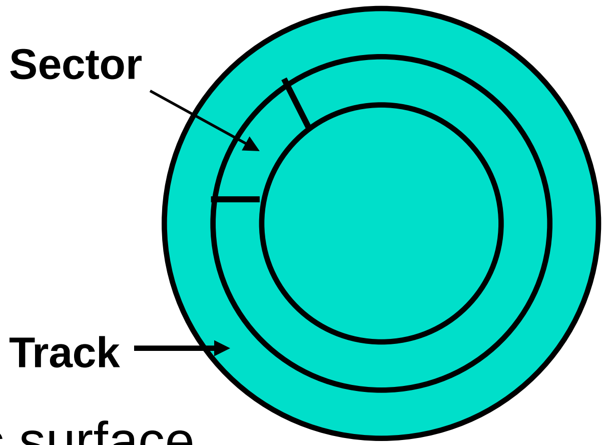
Mary Jane Irwin (www.cse.psu.edu/research/mdl/mji)

Review: Major Components of a Computer



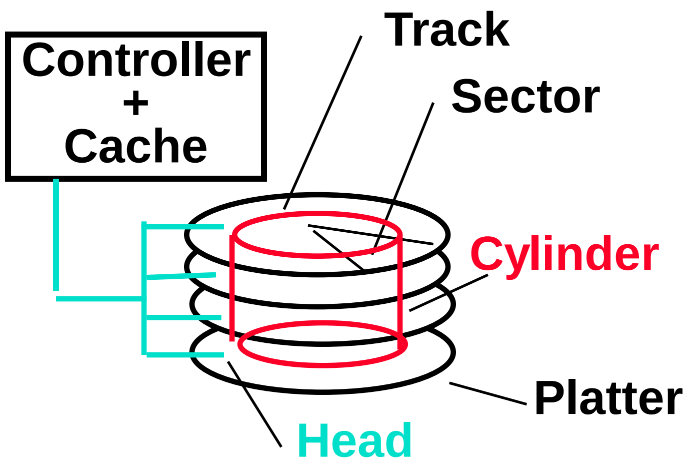
Magnetic Disk

- Purpose
  - Long term, nonvolatile storage
  - Lowest level in the memory hierarchy
    - slow, large, inexpensive
- General structure
  - A rotating platter coated with a magnetic surface
  - A moveable read/write head to access the information on the disk
- Typical numbers
  - 1 to 4 platters (each with 2 recordable surfaces) per disk of 1” to 3.5” in diameter
  - Rotational speeds of 5,400 to 15,000 RPM
  - 10,000 to 50,000 tracks per surface
    - cylinder - all the tracks under the head at a given point on all surfaces
  - 100 to 500 sectors per track
    - the smallest unit that can be read/written (typically 512B)



Magnetic Disk Characteristic

- Disk read/write components
  1. Seek time: position the head over the proper track (3 to 13 ms avg)
     - due to locality of disk references the actual average seek time may be only 25% to 33% of the advertised number



  2. Rotational latency: wait for the desired sector to rotate under the head (½ of 1/RPM converted to ms)
     - 0.5/5400RPM = 5.6ms to 0.5/15000RPM = 2.0ms
  3. Transfer time: transfer a block of bits (one or more sectors) under the head to the disk controller’s cache (70 to 125 MB/s are typical disk transfer rates in 2008)
     - the disk controller’s “cache” takes advantage of spatial locality in disk accesses
       – cache transfer rates are much faster (e.g., 375 MB/s)
  4. Controller time: the overhead the disk controller imposes in performing a disk I/O access (typically < .2 ms)

Typical Disk Access Time

- The average time to read or write a 512B sector for a disk rotating at 15,000 RPM with average seek time of 4 ms, a 100MB/sec transfer rate, and a 0.2 ms controller overhead

  Avg disk read/write = 4.0 ms + 0.5/(15,000RPM/(60sec/min))+ 0.5KB/(100MB/sec) + 0.2 ms = 4.0 + 2.0 + 0.01 + 0.2 = 6.2 ms

  If the measured average seek time is 25% of the advertised average seek time, then

  Avg disk read/write = 1.0 + 2.0 + 0.01 + 0.2 = 3.2 ms

- The rotational latency is usually the largest component of the access time

Disk Interface Standards

- Higher-level disk interfaces have a microprocessor disk controller that can lead to performance optimizations
  - ATA (Advanced Technology Attachment) – An interface standard for the connection of storage devices such as hard disks, solid-state drives, and CD-ROM drives. Parallel ATA has been largely replaced by serial ATA.
  - SCSI (Small Computer Systems Interface) – A set of standards (commands, protocols, and electrical and optical interfaces) for physically connecting and transferring data between computers and peripheral devices. Most commonly used for hard disks and tape drives.
- In particular, disk controllers have SRAM disk caches which support fast access to data that was recently read and often also include prefetch algorithms to try to anticipate demand

Magnetic Disk Examples (www.seagate.com)

Feature	Seagate ST31000340NS	Seagate ST973451SS	Seagate ST9160821AS
Disk diameter (inches)	3.5	2.5	2.5
Capacity (GB)	1000	73	160
# of surfaces (heads)	4	2	2
Rotation speed (RPM)	7,200	15,000	5,400
Transfer rate (MB/sec)	105	79-112	44
Minimum seek (ms)	0.8r-1.0w	0.2r-0.4w	1.5r-2.0w
Average seek (ms)	8.5r-9.5w	2.9r-3.3w	12.5r-13.0w
MTTF (hours@25^{o}C)	1,200,000	1,600,000	??
Dim (inches), Weight (lbs)	1x4x5.8, 1.4	0.6x2.8x3.9, 0.5	0.4x2.8x3.9, 0.2
GB/cu.inch, GB/watt	43, 91	11, 9	37, 84
Power: op/idle/sb (watts)	11/8/1	8/5.8/-	1.9/0.6/0.2
Price in 2008, $/GB	~$0.3/GB	~$5/GB	~$0.6/GB

Disk Latency & Bandwidth Milestones

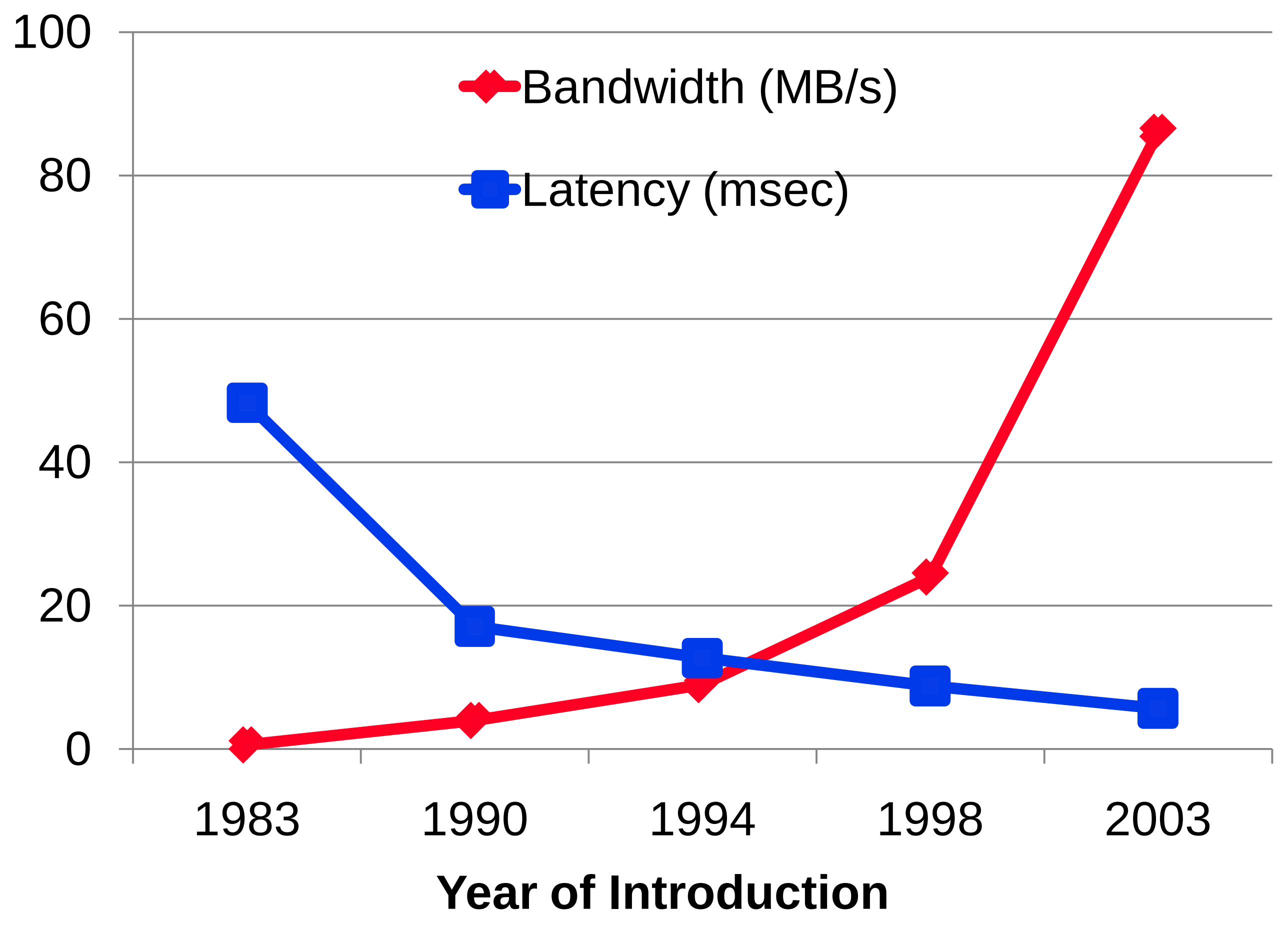
	CDC Wren	SG ST41	SG ST15	SG ST39	SG ST37
RSpeed (RPM)	3600	5400	7200	10000	15000
Year	1983	1990	1994	1998	2003
Capacity (Gbytes)	0.03	1.4	4.3	9.1	73.4
Diameter (inches)	5.25	5.25	3.5	3.0	2.5
Interface	ST-412	SCSI	SCSI	SCSI	SCSI
Bandwidth (MB/s)	0.6	4	9	24	86
Latency (msec)	48.3	17.1	12.7	8.8	5.7

Patterson, CACM Vol 47, #10, 2004

- Disk latency is one average seek time plus the rotational latency.
- Disk bandwidth is the peak transfer time of formatted data from the media (not from the cache).

Latency & Bandwidth Improvements

- In the time that the disk bandwidth doubles the latency improves by a factor of only 1.2 to 1.4



Flash Storage

- Flash memory is the first credible challenger to disks. It is semiconductor memory that is nonvolatile like disks, but has latency 100 to 1000 times faster than disk and is smaller, more power efficient, and more shock resistant.
  - In 2008, the price of flash is $4 to $10 per GB or about 2 to 10 times higher than disk and 5 to 10 times lower than DRAM.
  - Flash memory bits wear out (unlike disks and DRAMs), but wear leveling can make it unlikely that the write limits of the flash will be exceeded

Feature	Kingston	Transend	RiDATA
Capacity (GB)	8	16	32
Bytes/sector	512	512	512
Transfer rates (MB/sec)	4	20r-18w	68r-50w
MTTF	>1,000,000	>1,000,000	>4,000,000
Price (2008)	~ $30	~ $70	~ $300

Dependability, Reliability, Availability

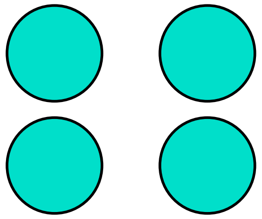
- Reliability – measured by the mean time to failure (MTTF). Service interruption is measured by mean time to repair (MTTR)
- Availability – a measure of service accomplishment

  Availability = MTTF/(MTTF + MTTR)

- To increase MTTF, either improve the quality of the components or design the system to continue operating in the presence of faulty components
  1. Fault avoidance: preventing fault occurrence by construction
  2. Fault tolerance: using redundancy to correct or bypass faulty components (hardware)
     - Fault detection versus fault correction
     - Permanent faults versus transient faults

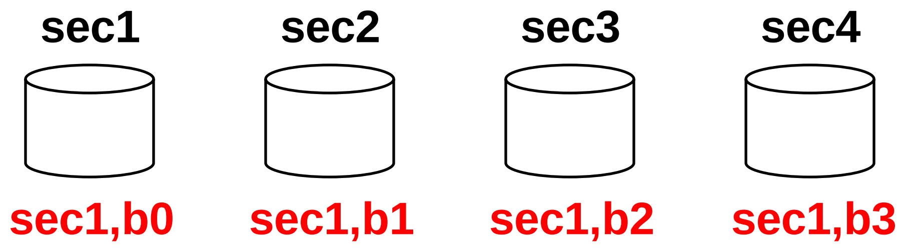
RAIDs: Disk Arrays

Redundant Array of
Inexpensive Disks



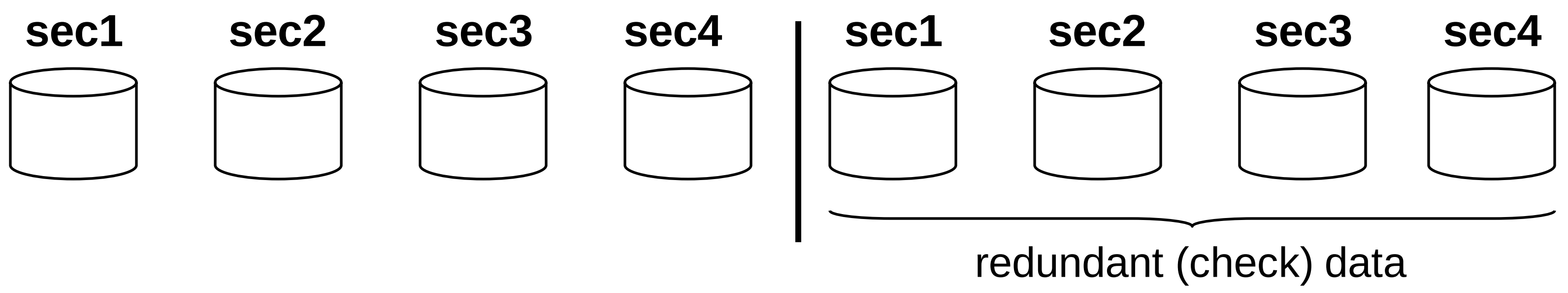
- Arrays of small and inexpensive disks
  - Increase potential throughput by having many disk drives
    - Data is spread over multiple disk
    - Multiple accesses are made to several disks at a time
- Reliability is lower than a single disk
- But availability can be improved by adding redundant disks (RAID)
  - Lost information can be reconstructed from redundant information
  - MTTR: mean time to repair is in the order of hours
  - MTTF: mean time to failure of disks is tens of years

RAID: Level 0 (No Redundancy; Striping)



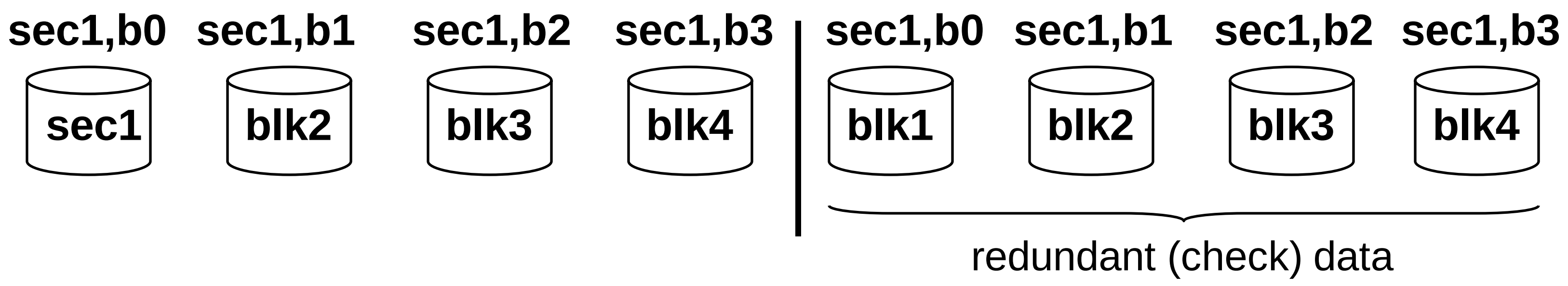
- Multiple smaller disks as opposed to one big disk
  - Spreading the sector over multiple disks – striping – means that multiple blocks can be accessed in parallel increasing the performance
    - A 4 disk system gives four times the throughput of a 1 disk system
  - Same cost as one *big* disk – assuming 4 small disks cost the same as one big disk
- No redundancy, so what if one disk fails?
  - Failure of one or more disks is more likely as the number of disks in the system increases

RAID: Level 1 (Redundancy via Mirroring)



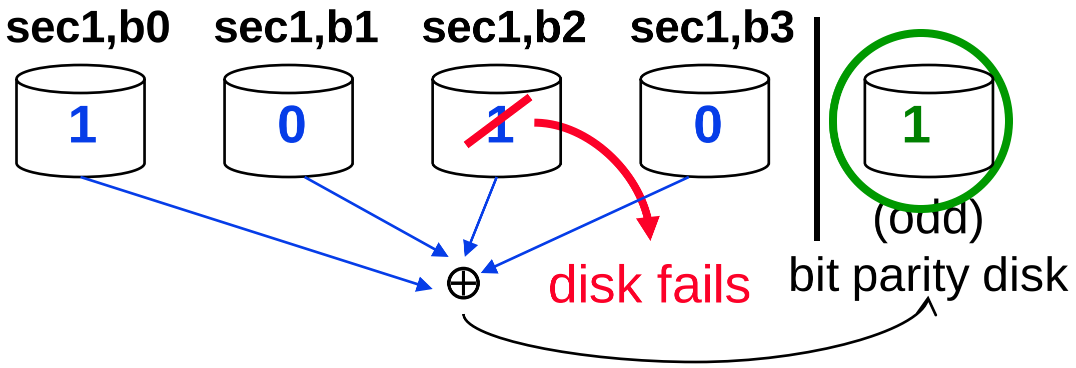
- Uses twice as many disks as RAID 0 (e.g., 8 smaller disks with the second set of 4 duplicating the first set) so there are always two copies of the data
  - # redundant disks = # of data disks so twice the cost of one big disk
    - writes have to be made to both sets of disks, so writes would be only 1/2 the performance of a RAID 0
- What if one disk fails?
  - If a disk fails, the system just goes to the “mirror” for the data

RAID: Level 0+1 (Striping with Mirroring)



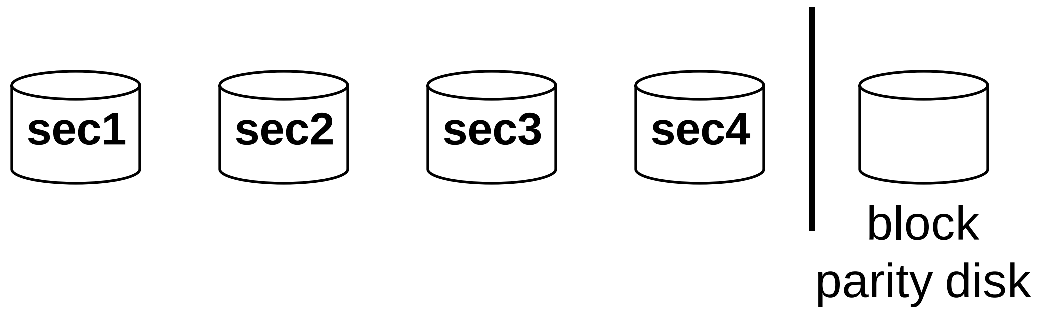
- Combines the best of RAID 0 and RAID 1, data is striped across four disks and mirrored to four disks
  - Four times the throughput (due to striping)
  - # redundant disks = # of data disks   so twice the cost of one big disk
    - writes have to be made to both sets of disks, so writes would be only 1/2 the performance of RAID 0
- What if one disk fails?
  - If a disk fails, the system just goes to the “mirror” for the data

RAID: Level 3 (Bit-Interleaved Parity)



- ❑ Cost of higher availability is reduced to 1/N where N is the number of disks in a protection group
  - # redundant disks = 1 # of protection groups
    - writes require writing the new data to the data disk as well as computing the parity, meaning reading the other disks, so that the parity disk can be updated
- ❑ Can tolerate *limited* (single) disk failure, since the data can be reconstructed
    - reads require reading all the operational data disks as well as the parity disk to calculate the missing data that was stored on the failed disk

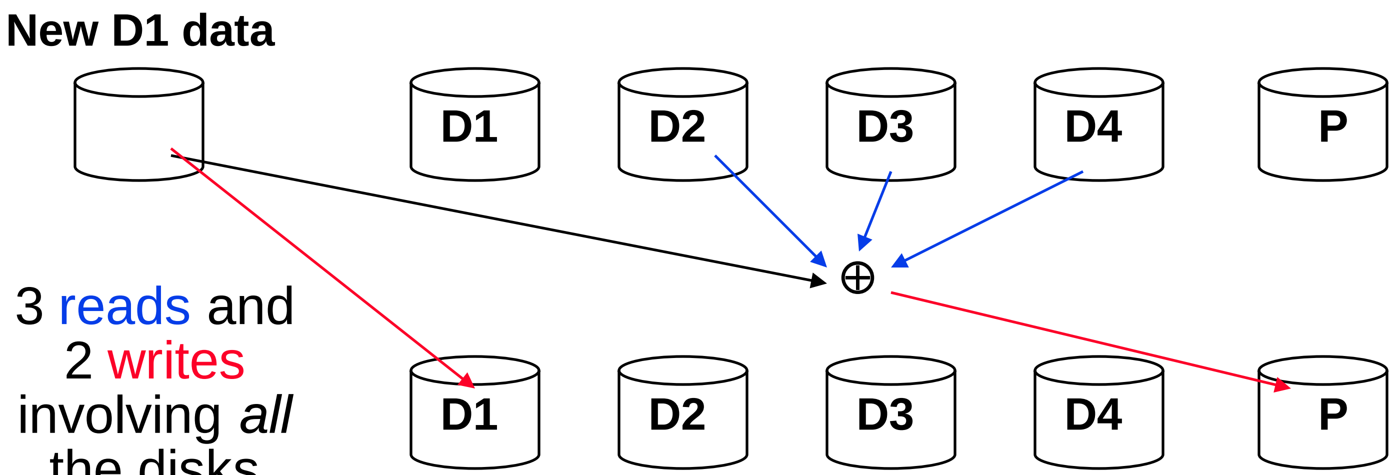
RAID: Level 4 (Block-Interleaved Parity)



- Cost of higher availability still only 1/N but the parity is stored as blocks associated with sets of data blocks
  - Four times the throughput (striping)
  - # redundant disks = 1 # of protection groups
  - Supports "small reads" and "small writes" (reads and writes that go to just one (or a few) data disk in a protection group)
    - by watching which bits change when writing new information, need only to change the corresponding bits on the parity disk
    - the parity disk must be updated on every write, so it is a bottleneck for back-to-back writes
- Can tolerate *limited* disk failure, since the data can be reconstructed

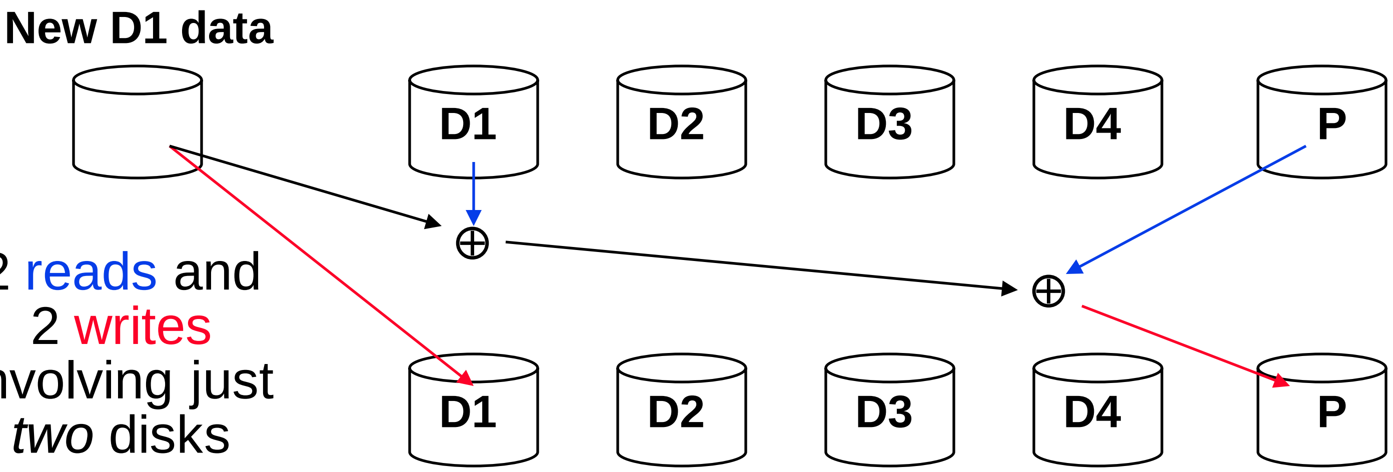
Small Writes

- RAID 3 writes



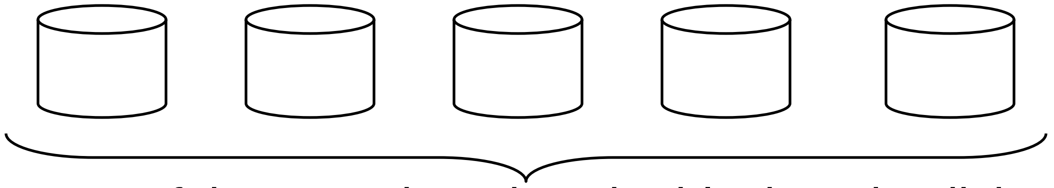
3 reads and
2 writes
involving *all*
the disks

- RAID 4 *small* writes



2 reads and
2 writes
involving just
two disks

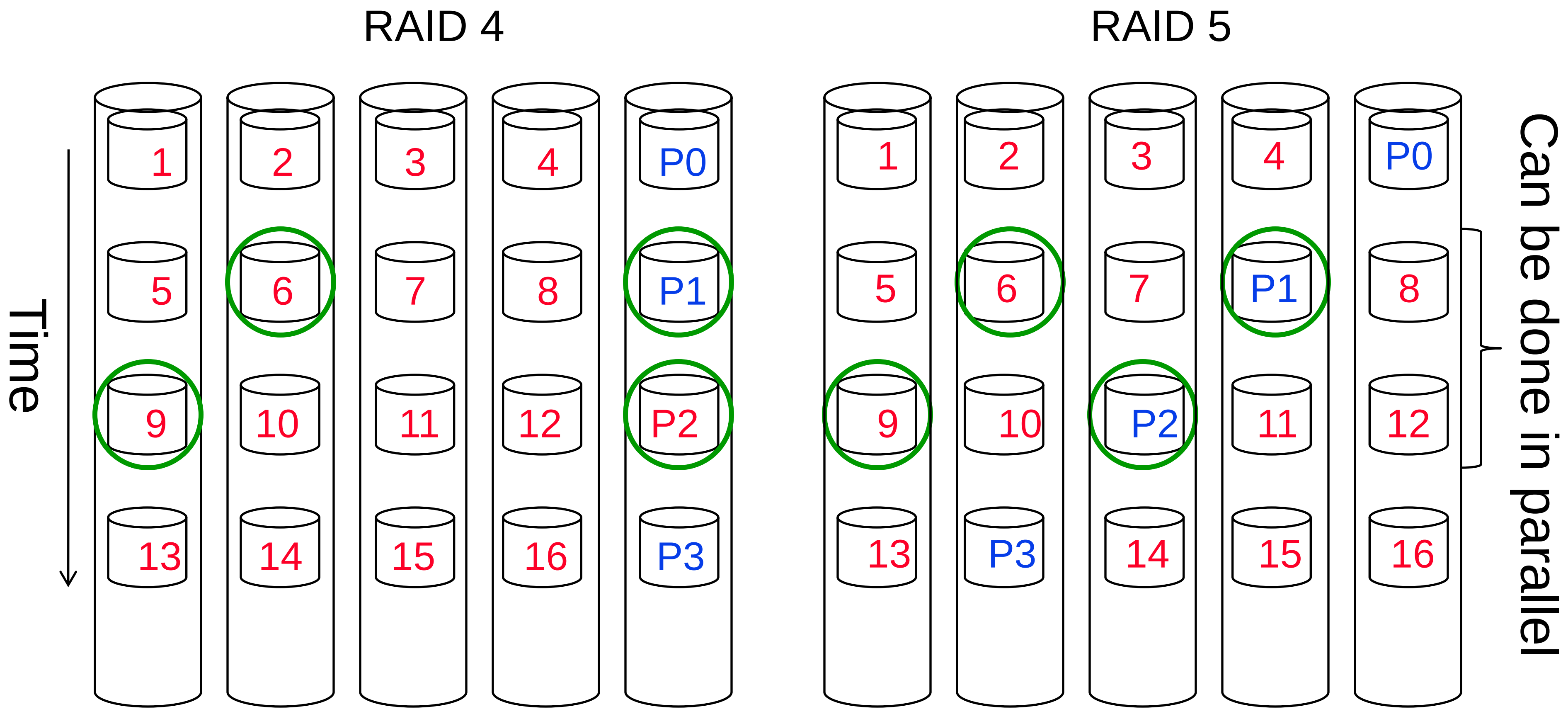
RAID: Level 5 (Distributed Block-Interleaved Parity)



one of these assigned as the block parity disk

- Cost of higher availability still only 1/N but the parity block can be located on any of the disks so there is no single bottleneck for writes
  - Still four times the throughput (striping)
  - # redundant disks = 1   # of protection groups
  - Supports "small reads" and "small writes" (reads and writes that go to just one (or a few) data disk in a protection group)
  - Allows multiple simultaneous writes as long as the accompanying parity blocks are not located on the same disk
- Can tolerate *limited* disk failure, since the data can be reconstructed

Distributing Parity Blocks



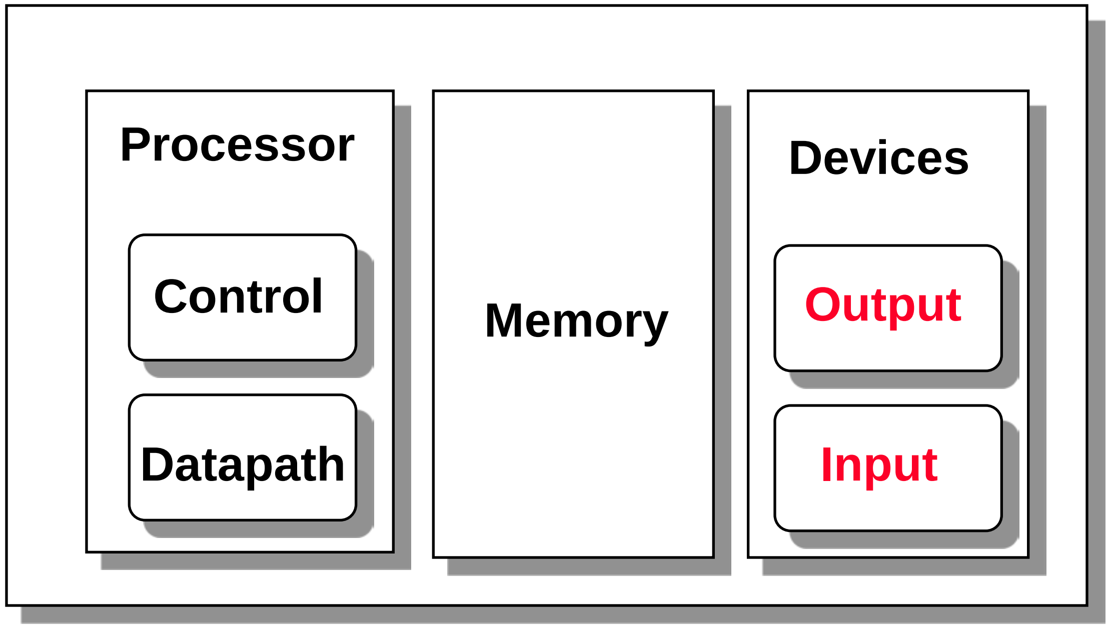
- By distributing parity blocks to all disks, some small writes can be performed in parallel

Summary

- Four components of disk access time:
  - Seek Time: advertised to be 3 to 14 ms but lower in real systems
  - Rotational Latency: 5.6 ms at 5400 RPM and 2.0 ms at 15000 RPM
  - Transfer Time: 30 to 80 MB/s
  - Controller Time: typically less than .2 ms
- RAIDS can be used to improve availability
  - RAID 1 and RAID 5 – widely used in servers, one estimate is that 80% of disks in servers are RAIDs
  - RAID 0+1 (mirroring) – EMC, Tandem, IBM
  - RAID 3 – Storage Concepts
  - RAID 4 – Network Appliance
- RAIDS have enough redundancy to allow continuous operation, but not hot swapping

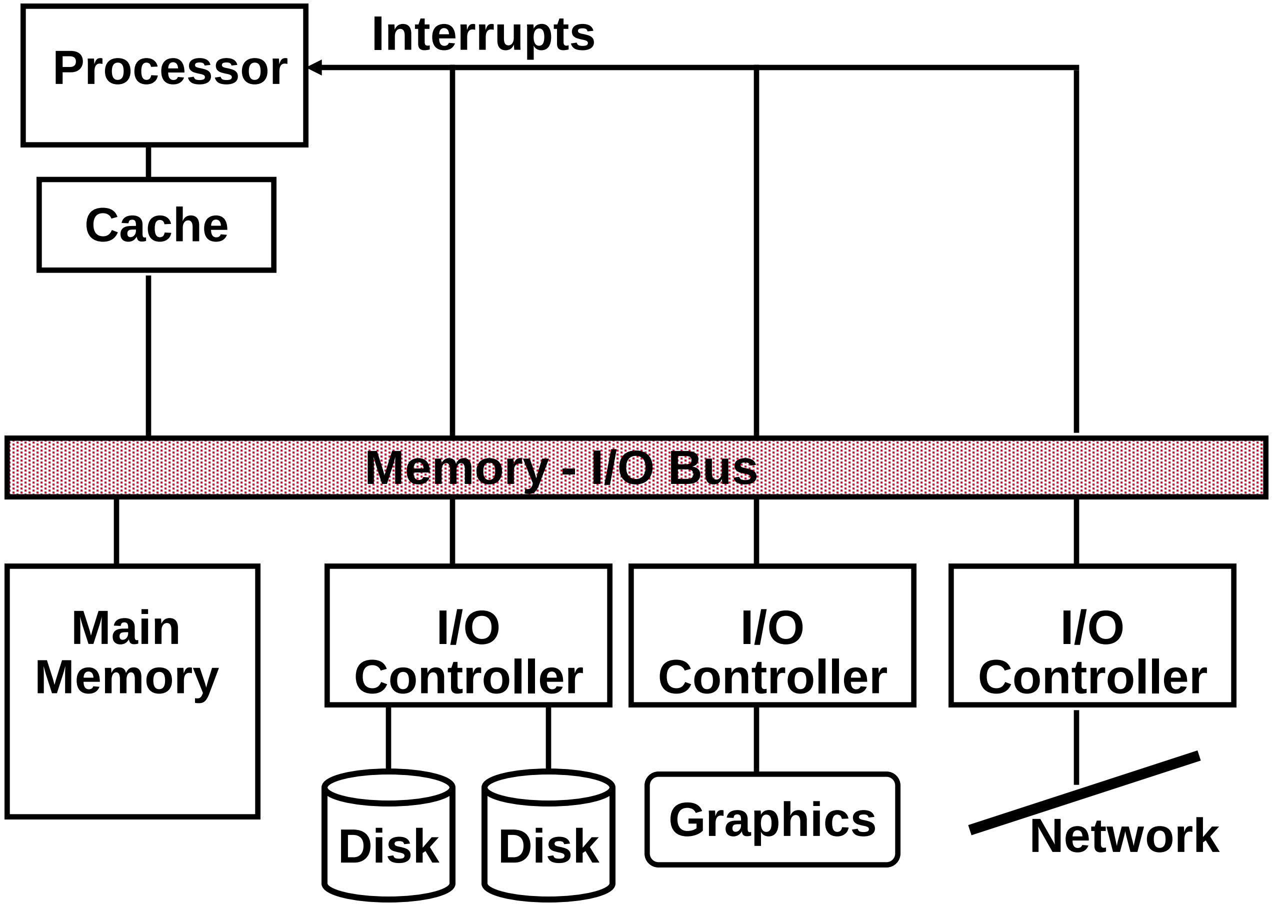
Chapter 6B: I/O Systems

Review: Major Components of a Computer



- Important metrics for an I/O system
  - Performance
  - Expandability
  - Dependability
  - Cost, size, weight
  - Security

A Typical I/O System



Input and Output Devices

- I/O devices are incredibly diverse with respect to
  - Behavior – input, output or storage
  - Partner – human or machine
  - Data rate – the peak rate at which data can be transferred between the I/O device and the main memory or processor

Device	Behavior	Partner	Data rate (Mb/s)
Keyboard	input	human	0.0001
Mouse	input	human	0.0038
Laser printer	output	human	3.2000
Magnetic disk	storage	machine	800.0000-3000.0000
Graphics display	output	human	800.0000-8000.0000
Network/LAN	input or output	machine	100.0000-10000.0000

8 orders of magnitude range

I/O Performance Measures

- I/O bandwidth (throughput) – amount of information that can be input (output) and communicated across an interconnect (e.g., a bus) to the processor/memory (I/O device) per unit time
  1. How much data can we move through the system in a certain time?
  2. How many I/O operations can we do per unit time?
- I/O response time (latency) – the total elapsed time to accomplish an input or output operation
  - An especially important performance metric in real-time systems
- Many applications require *both* high throughput and short response times

I/O System Interconnect Issues

- A bus is a shared communication link (a single set of wires used to connect multiple subsystems) that needs to support a range of devices with widely varying latencies and data transfer rates
  - Advantages
    - Versatile – new devices can be added easily and can be moved between computer systems that use the same bus standard
    - Low cost – a single set of wires is shared in multiple ways
  - Disadvantages
    - Creates a communication bottleneck – bus bandwidth limits the maximum I/O throughput

- The maximum bus speed is largely limited by
  - The length of the bus
  - The number of devices on the bus

Types of Buses

- Processor-memory bus (“Front Side Bus”, proprietary)
  - Short and high speed
  - Matched to the memory system to maximize the memory-processor bandwidth
  - Optimized for cache block transfers
- I/O bus (industry standard, e.g., SCSI, USB, Firewire)
  - Usually is lengthy and slower
  - Needs to accommodate a wide range of I/O devices
  - Use either the processor-memory bus or a backplane bus to connect to memory
- Backplane bus (industry standard, e.g., ATA, PCIexpress)
  - Allow processor, memory and I/O devices to coexist on a single bus
  - Used as an intermediary bus connecting I/O busses to the processor-memory bus

I/O Transactions

- An I/O transaction is a sequence of operations over the interconnect that includes a request and may include a response either of which may carry data. A transaction is initiated by a single request and may take *many* individual bus operations. An I/O transaction typically includes two parts
  1. Sending the address
  2. Receiving or sending the data

- Bus transactions are defined by what they do to memory

output
  - A read transaction reads data from memory (to either the processor or an I/O device)

input
  - A write transaction writes data to the memory (from either the processor or an I/O device)

Synchronous and Asynchronous Buses

- Synchronous bus (e.g., processor-memory buses)
  - Includes a clock in the control lines and has a fixed protocol for communication that is relative to the clock
  - Advantage: involves very little logic and can run very fast
  - Disadvantages:
    - Every device communicating on the bus must use same clock rate
    - To avoid clock skew, they cannot be long if they are fast
- Asynchronous bus (e.g., I/O buses)
  - It is not clocked, so requires a handshaking protocol and additional control lines (ReadReq, Ack, DataRdy)
  - Advantages:
    - Can accommodate a wide range of devices and device speeds
    - Can be lengthened without worrying about clock skew or synchronization problems
  - Disadvantage: slow(er)

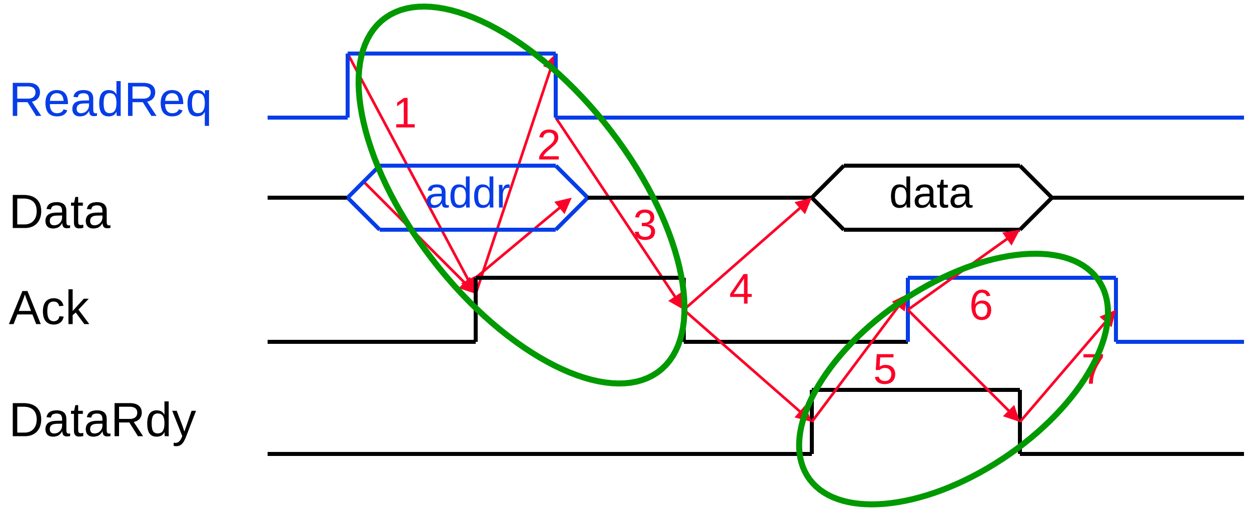
ATA Cable Sizes

- Companies have transitioned from synchronous, parallel wide buses to asynchronous narrow buses
  - Reflection on wires and clock skew makes it difficult to use 16 to 64 parallel wires running at a high clock rate (e.g., ~400MHz) so companies have moved to buses with a few one-way wires running at a very high "clock" rate (~2GHz)

  - Serial ATA cables (red) are much thinner than parallel ATA cables (green)

Asynchronous Bus Handshaking Protocol

- Output (read) data from memory to an I/O device



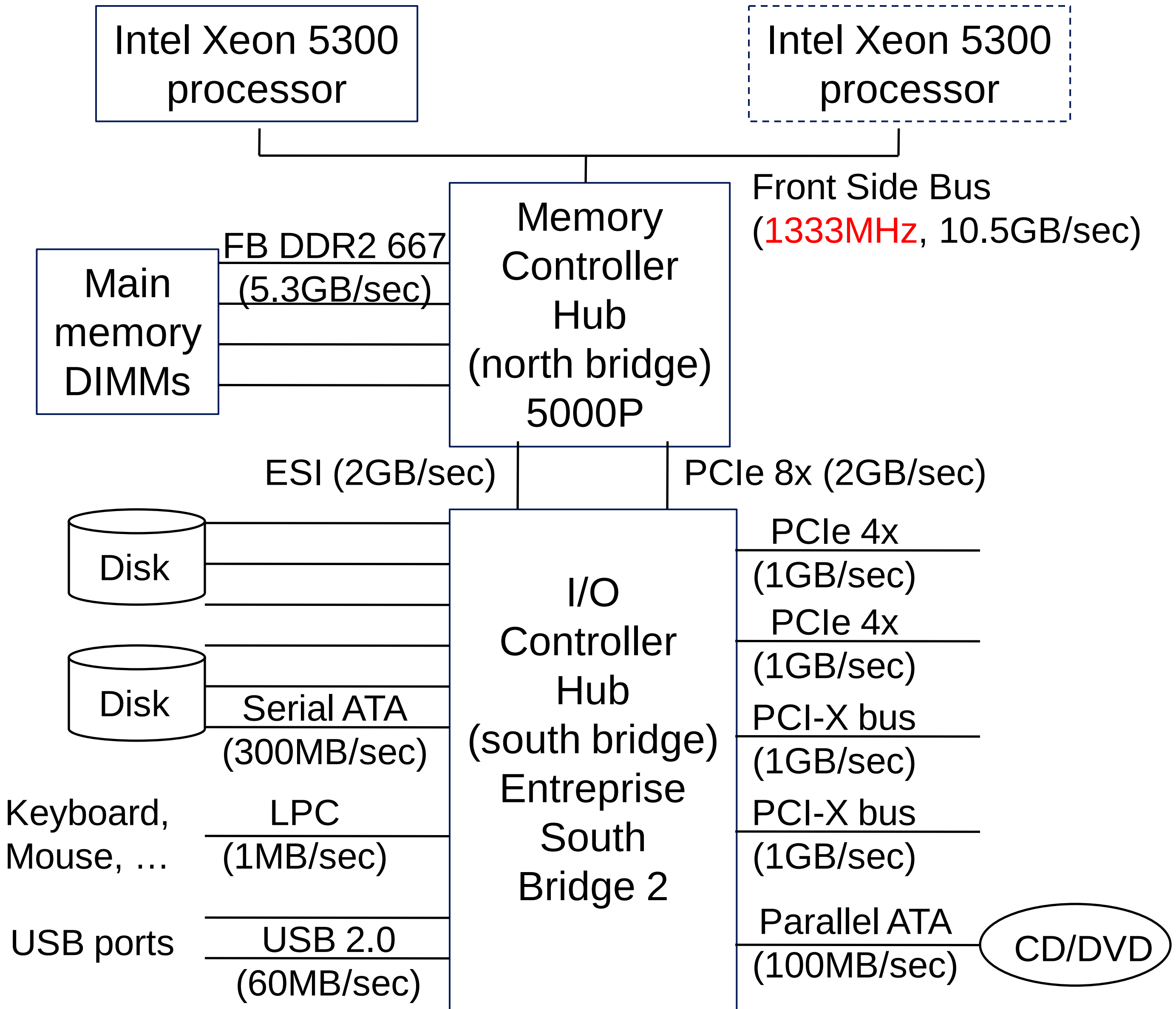
I/O device signals a request by raising ReadReq and putting the addr on the data lines

1. Memory sees ReadReq, reads addr from data lines, and raises Ack
2. I/O device sees Ack and releases the ReadReq and data lines
3. Memory sees ReadReq go low and drops Ack
4. When memory has data ready, it places it on data lines and raises DataRdy
5. I/O device sees DataRdy, reads the data from data lines, and raises Ack
6. Memory sees Ack, releases the data lines, and drops DataRdy
7. I/O device sees DataRdy go low and drops Ack

Key Characteristics of I/O Standards

	Firewire	**USB 2.0**	**PCIe**	**Serial ATA**	**SA SCSI**
Use	External	External	Internal	Internal	External
Devices per channel	63	127	1	1	4
Max length	4.5 meters	5 meters	0.5 meters	1 meter	8 meters
Data Width	4	2	2 per lane	4	4
Peak Bandwidth	50MB/sec (400) 100MB/sec (800)	0.2MB/sec (low) 1.5MB/sec (full) 60MB/sec (high)	250MB/sec per lane (1x) Come as 1x, 2x, 4x, 8x, 16x, 32x	300MB/sec	300MB/sec
Hot pluggable?	Yes	Yes	Depends	Yes	Yes

A Typical I/O System



Interfacing I/O Devices to the Processor, Memory, and OS

- The operating system acts as the interface between the I/O hardware and the program requesting I/O since
  - Multiple programs using the processor share the I/O system
  - I/O systems usually use interrupts which are handled by the OS
  - Low-level control of an I/O device is complex and detailed
- Thus OS must handle interrupts generated by I/O devices and supply routines for low-level I/O device operations, provide equitable access to the shared I/O resources, protect those I/O devices/activities to which a user program doesn't have access, and schedule I/O requests to enhance system throughput
  - OS must be able to give commands to the I/O devices
  - I/O device must be able to notify the OS about its status
  - Must be able to transfer data between the memory and the I/O device

Communication of I/O Devices and Processor

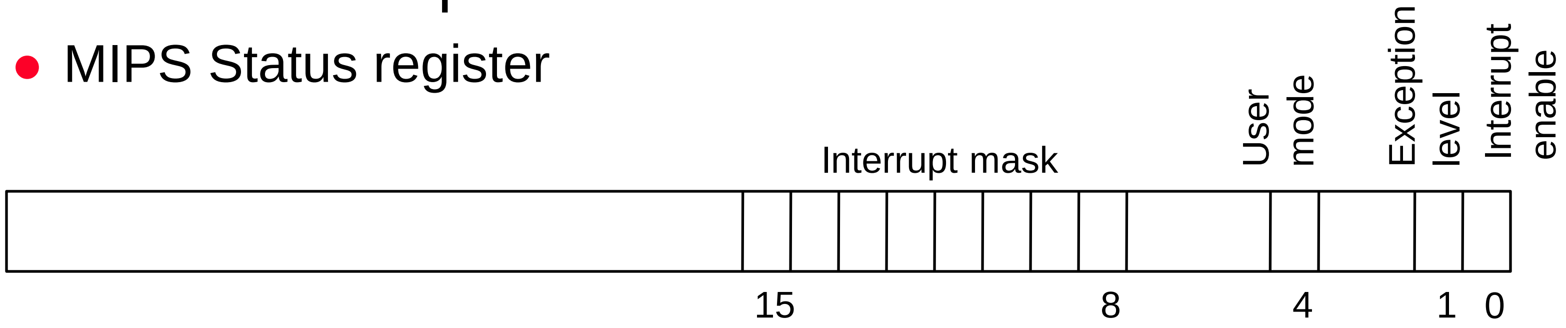
- How the processor directs the I/O devices
  - Special I/O instructions
    - Must specify both the device and the command
  - Memory-mapped I/O
    - Portions of the high-order memory address space are assigned to each I/O device
    - Read and writes to those memory addresses are interpreted as commands to the I/O devices
    - Load/stores to the I/O address space can *only* be done by the OS
- How I/O devices communicate with the processor
  - Polling – the processor periodically checks the status of an I/O device (through the OS) to determine its need for service
    - Processor is totally in control – but does all the work
    - Can waste a lot of processor time due to speed differences
  - Interrupt-driven I/O – the I/O device issues an interrupt to indicate that it needs attention

Interrupt Driven I/O

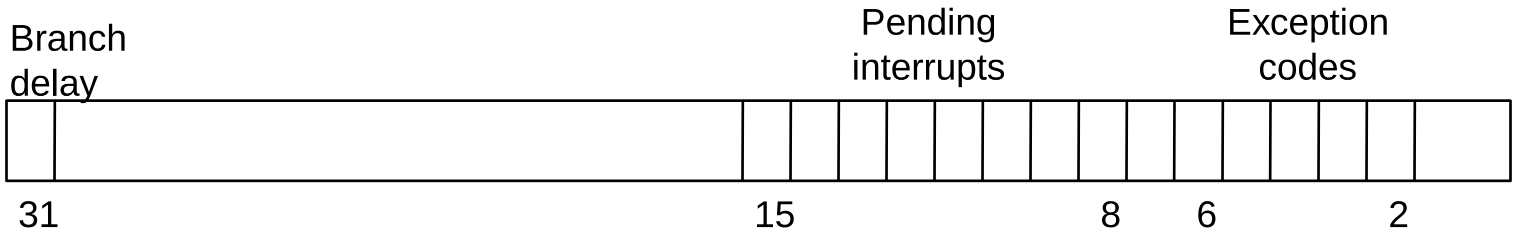
- An I/O interrupt is asynchronous wrt instruction execution
  - Is not associated with any instruction so doesn't prevent any instruction from completing
    - You can pick your own convenient point to handle the interrupt
- With I/O interrupts
  - Need a way to identify the device generating the interrupt
  - Can have different urgencies (so need a way to prioritize them)
- Advantages of using interrupts
  - Relieves the processor from having to continuously poll for an I/O event; user program progress is only suspended during the actual transfer of I/O data to/from user memory space
- Disadvantage – special hardware is needed to
  - Indicate the I/O device causing the interrupt and to save the necessary information prior to servicing the interrupt and to resume normal processing after servicing the interrupt

Interrupt Priority Levels

- Priority levels can be used to direct the OS the order in which the interrupts should be serviced
  - MIPS Status register



    - Determines who can interrupt the processor (if Interrupt enable is 0, none can interrupt)
  - MIPS Cause register



    - To enable a Pending interrupt, the correspond bit in the Interrupt mask must be 1
    - Once an interrupt occurs, the OS can find the reason in the Exception codes field

Interrupt Handling Steps

1. Logically AND the Pending interrupt field and the Interrupt mask field to see which enabled interrupts could be the culprit. Make copies of both Status and Cause registers.
2. Select the higher priority of these interrupts (leftmost is highest)
3. Save the Interrupt mask field
4. Change the Interrupt mask field to disable all interrupts of equal or lower priority
5. Save the processor state prior to "handling" the interrupt
6. Set the Interrupt enable bit (to allow higher-priority interrupts)
7. Call the appropriate interrupt handler routine
8. Before returning from interrupt, set the Interrupt enable bit back to 0 and restore the Interrupt mask field

- Interrupt priority levels (IPLs) assigned by the OS to each process can be raised and lowered via changes to the Status's Interrupt mask field

Direct Memory Access (DMA)

- For high-bandwidth devices (like disks) interrupt-driven I/O would consume a *lot* of processor cycles
- With DMA, the DMA controller has the ability to transfer large blocks of data directly to/from the memory without involving the processor
    1. The processor initiates the DMA transfer by supplying the I/O device address, the operation to be performed, the memory address destination/source, the number of bytes to transfer
    2. The DMA controller manages the entire transfer (possibly thousand of bytes in length), arbitrating for the bus
    3. When the DMA transfer is complete, the DMA controller interrupts the processor to let it know that the transfer is complete
- There may be multiple DMA devices in one system
    - Processor and DMA controllers contend for bus cycles and for memory

The DMA Stale Data Problem

- In systems with caches, there can be two copies of a data item, one in the cache and one in the main memory
  - For a DMA input (from disk to memory) – the processor will be using stale data if that location is also in the cache
  - For a DMA output (from memory to disk) and a write-back cache – the I/O device will receive stale data if the data is in the cache and has not yet been written back to the memory
- The coherency problem can be solved by
  1. Routing all I/O activity through the cache – expensive and a large negative performance impact
  2. Having the OS invalidate all the entries in the cache for an I/O input or force write-backs for an I/O output (called a cache flush)
  3. Providing hardware to *selectively* invalidate cache entries – i.e., need a snooping cache controller

DMA and Virtual Memory Considerations

- Should the DMA work with virtual addresses or physical addresses?
- If working with physical addresses
  - Must constrain all of the DMA transfers to stay within one page because if it crosses a page boundary, then it won’t necessarily be contiguous in memory
  - If the transfer won’t fit in a single page, it can be broken into a series of transfers (each of which fit in a page) which are handled individually and *chained* together
- If working with virtual addresses
  - The DMA controller will have to translate the virtual address to a physical address (i.e., will need a TLB structure)
- Whichever is used, the OS must cooperate by not remapping pages while a DMA transfer involving that page is in progress