Lecture for Storage System

Type of Storage Devices

Buses—*Connecting I/O Devices to CPU/Memory*

Reliability,Availability,and Dependability

RAID: *Redundant Arrays of Inexpensive Disks*

Error and Failures in Real Systems

I/O Performance Measures

A Little Queuing Theory

7.1 Introduction

- The prejudice
  - Historically neglected by CPU enthusiasts
    - CPU time which by definition ignores I/O
  - Citizenship of I/O is even apparent in the label *peripheral* applied to I/O devices.
- The fact
  - A computer without I/O devices is like a car without wheels
    - You can't get very far without them.
  - response time
    - The time between when the user types a command and when results appear—is surely a better measure of performance.

Does I/O Performance Matter?

- One argument : I/O speed doesn't matter
  - If a process waits for a peripheral, run another task
  - *Throughput does not descend*
  - I/O performance doesn't matter in a multiprogrammed environment.
- Several points to make in reply
  - if users didn't care about response time
    - Interactive software never would have been invented
    - Be no workstations or personal computers today;
  - Expensive to rely on running other processes
    - Paging traffic from process switching might actually increase I/O.
    - Mobile devices and desktop computing, there is only one person per computer and thus fewer processes than in timesharing.
  - Many times the only waiting process is the human being!

I/O's Revenge is at hand

- Amdahl's Law: system speed-up limited by the slowest part!

  10% IO & 10x CPU => Speedup= $\frac{1}{0.1+0.09}$ =5 (lose 50%)

  10% IO & 100x CPU => Speedup= $\frac{1}{0.1+0.009}$ =10 (lose 90%)

- I/O bottleneck:
  - Diminishing fraction of time in CPU
  - Diminishing value of faster CPUs
- I/O performance increasingly limits system performance and effectiveness
  - CPU Performance: 55% per year and I/O did not improve
  - Every task would become I/O bound.
  - There would be no reason to buy faster CPUs—and no jobs for CPU designers.

Does CPU Performance Matter?

- Why still important to keep CPUs busy vs. IO devices ("CPU time"), as CPUs not costly?
  - Moore's Law leads to both large, fast CPUs but also to very small, cheap CPUs
  - 2001 Hypothesis: 600 MHz PC is fast enough for Office Tools?
  - PC slowdown since fast enough unless games, new apps?
- People care more about storing information and communicating information than calculating
  - "Information Technology" vs. "Computer Science"
  - 1960s and 1980s: Computing Revolution
  - 1990s and 2000s: Information Age
- This shift in focus from computation to communication and storage of information
  - emphasizes reliability, availability and scalability as well as cost-performance.

Types of Storage Devices-1

- Device Providing Information

Sensor	Key	CRT
1~1000B/S	10B/S	2000B/S

Printer	Communication Cable
1800B/S	30~200000B/S

- Multimedia Data Device

high speed graphics	video display	Audio frequency
1MB/S	100MB/S	64KB/S

- Network Communication

DIX (Ethernet network standard Digital, Intel, Xerox)、TB2、RJ45

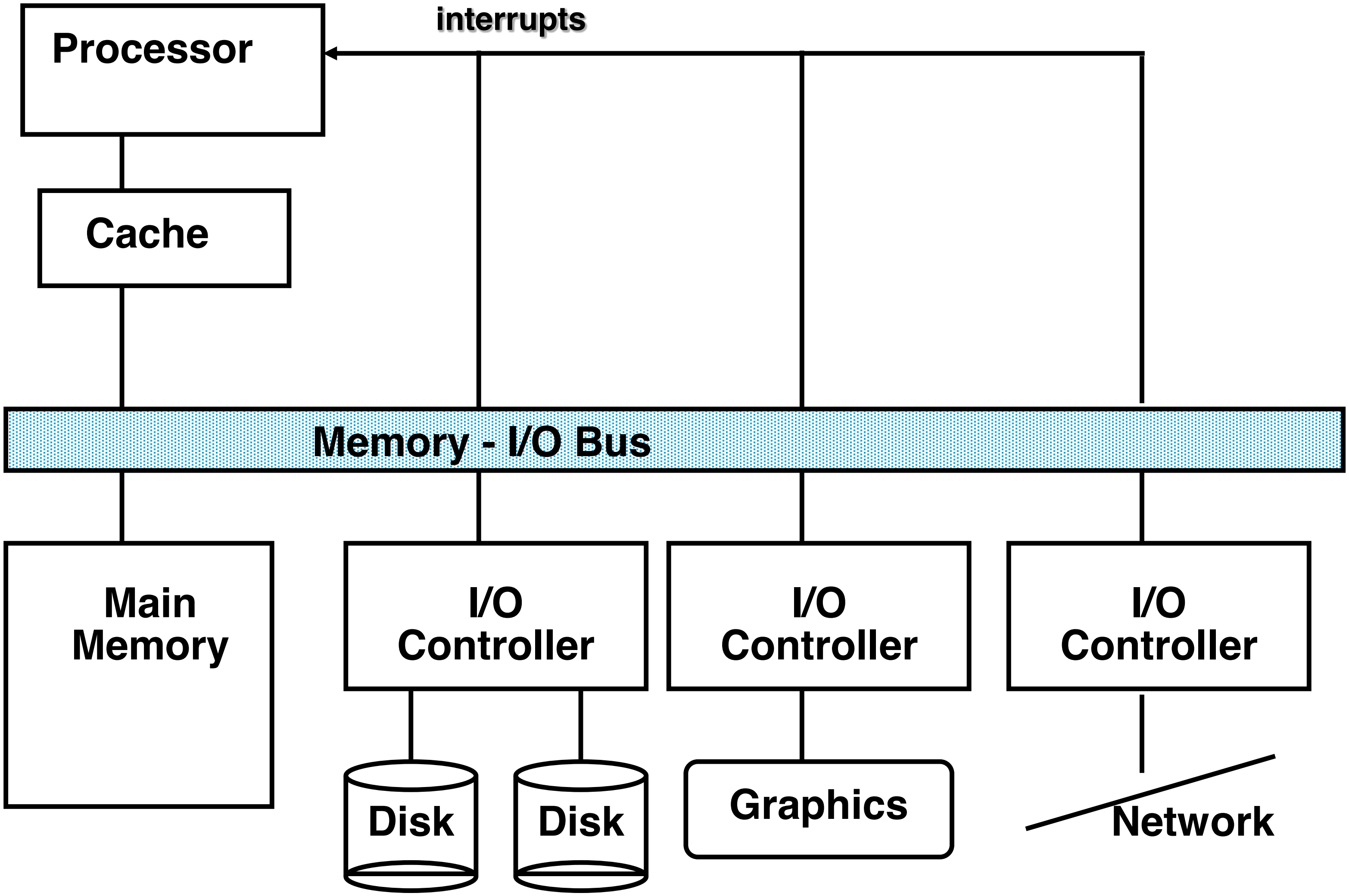
Types of Storage Devices-2

Device	Behavior	Partner	Data Rate (KB/sec)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Printer	Output	Human	3.00
Floppy disk	Storage	Machine	50.00
Laser Printer	Output	Human	100.00
Optical Disk	Storage	Machine	500.00
Magnetic Disk	Storage	Machine	5,000.00
Network-LAN	Input or Output	Machine	20 --1,000.00
Graphics Display	Output	Human	30,000.00

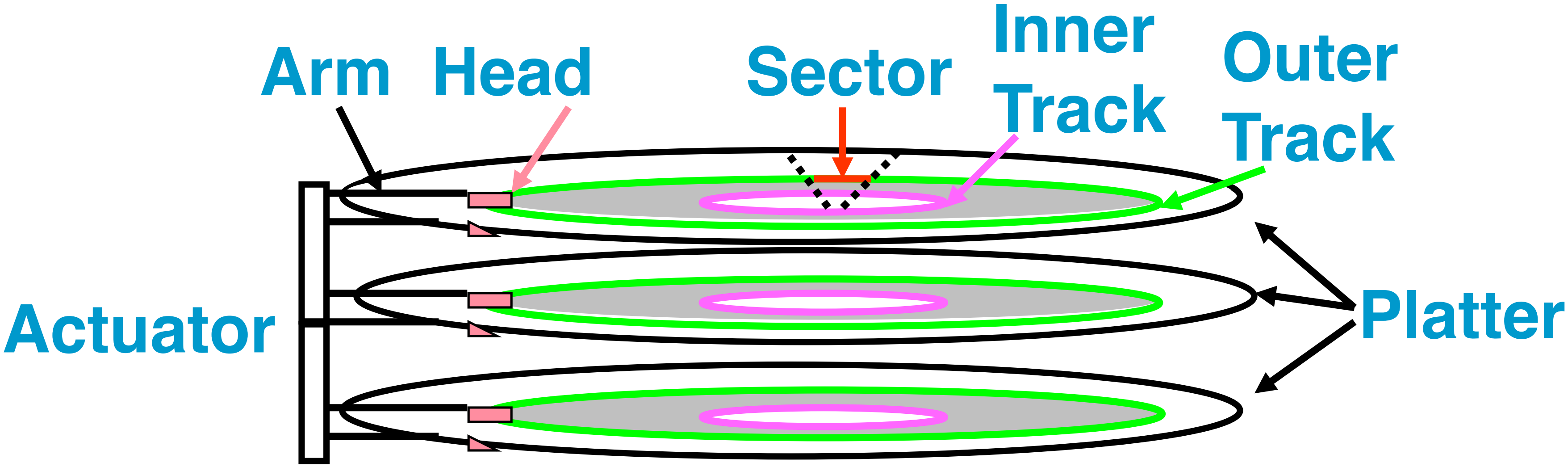
Storage Technology Drivers

- Driven by the prevailing computing paradigm
  - 1950s: migration from batch to on-line processing
  - 1990s: migration to ubiquitous computing
    - computers in phones, books, cars, video cameras, …
    - nationwide fiber optical network with wireless tails
- Effects on storage industry:
  - Embedded storage
    - smaller, cheaper, more reliable, lower power
  - Data utilities
    - high capacity, hierarchically managed storage

I/O Systems


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

Disk Device Terminology



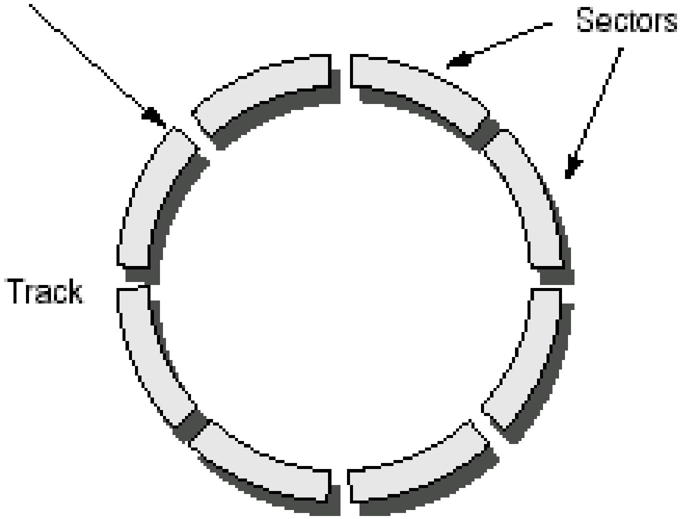
- Several platters, with information recorded magnetically on both surfaces (usually)

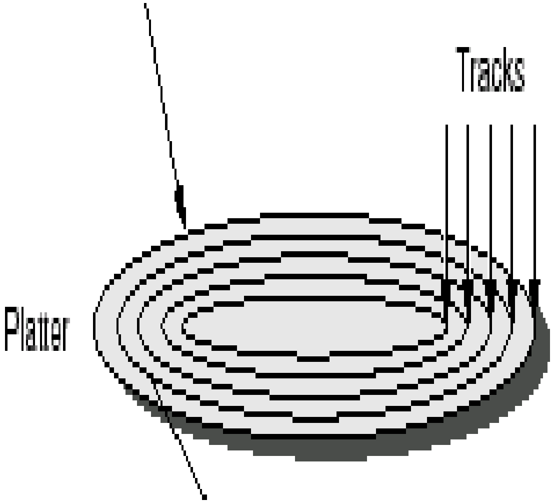
Disk Device Terminology

Bits recorded in tracks, which in turn divided into sectors (e.g., 512 Bytes)

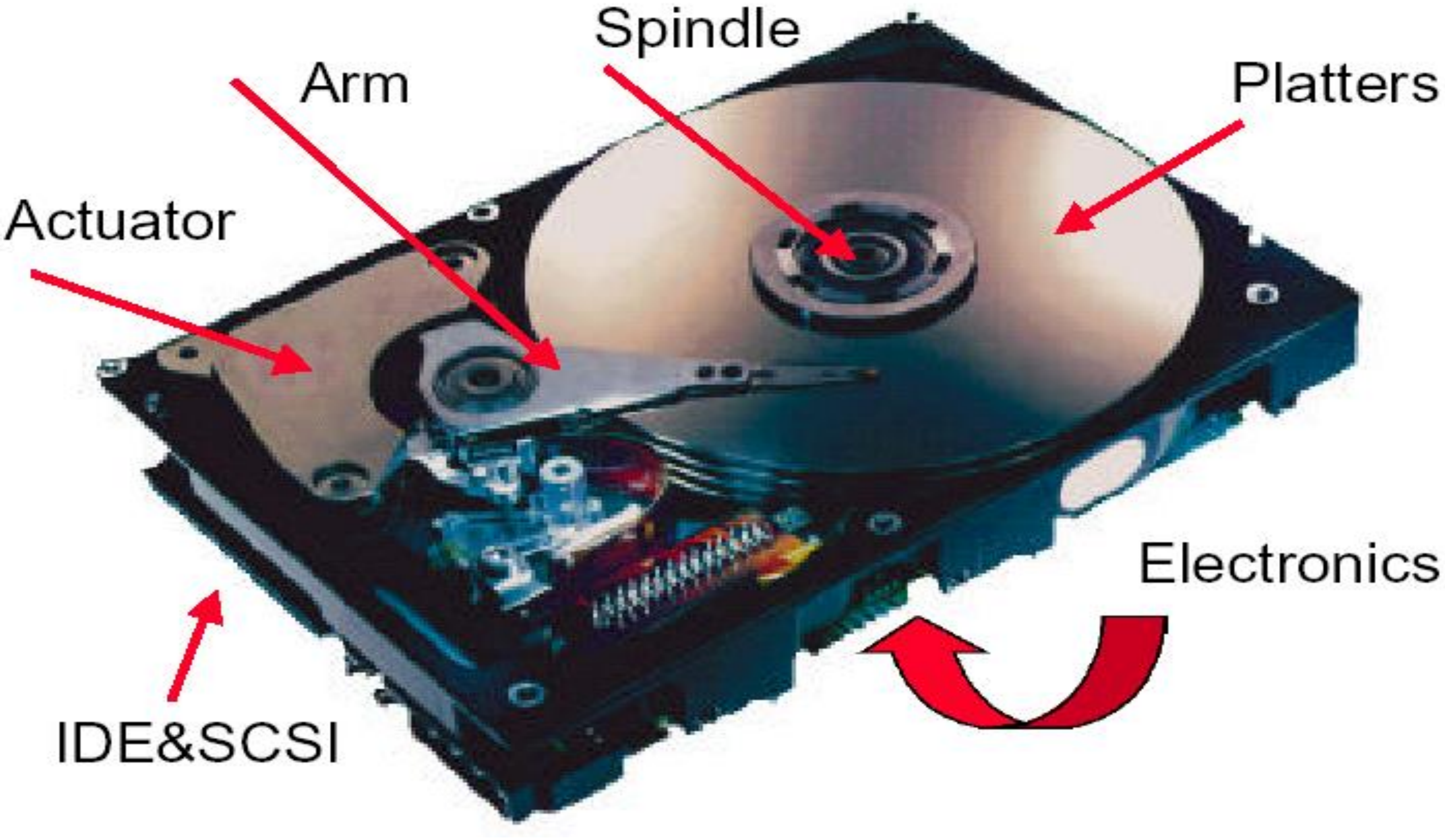
Actuator moves head (end of arm,1/surface) over track ("seek"), select surface, wait for sector rotate under head, then read or write

"Cylinder": all tracks under heads





What's Inside A Disk Drive?



Disk Device Performance

Disk Latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead

- Seek time: move head to the desired track
  - today's drives - 5 to 15 ms
  - average seek = time for all possible seeks/no. of possible seeks
  - actual average seek = 25% to 33% due to locality
- Rotational latency
  - today's drives - 5,400 to 12,000 RPM ;
  - approximately 12 ms to 5 ms
  - average rotational latency = (0.5)(rotational latency)
- Transfer time
  - time to transfer a sector (1 KB/sector)
  - function of rotation speed, recording density
  - today's drives - 10 to 40 MBytes/second
- Controller time
  - overhead on drive electronics adds to manage drive
  - but also gives prefetching and caching

Disk Device Performance-2

Average access time = (seek time) + (rotational latency) + (transfer) + (controller time)

- Track and cylinder skew
  - cylinder switch time
    - delay to change from one cylinder to the next
      - may have to wait an extra rotation
    - solution - drives incorporate skew
      - offset sectors between cylinders to account for switch time
  - head switch time
    - change heads to go from one track to next on same cylinder
      - incur additional settling time
- Prefetching
  - disks usually read an entire track at a time
  - assumes that request for the next sector will come soon
- Caching
  - limited amount of caching across requests, but prefetching is preferred

Disk Device Performance-3

- Average distance sector from head?
- 1/2 time of a rotation
  - 10000 Revolutions Per Minute   166.67 Rev/sec
  - 1 revolution = 1/ 166.67 sec   6.00 milliseconds
  - 1/2 rotation (revolution)   3.00 ms
- Average no. tracks move arm?
  - Sum all possible seek distances
    from all possible tracks / # possible
    - Assumes average seek distance is random
  - Disk industry standard benchmark

Data Rate: Inner vs. Outer Tracks

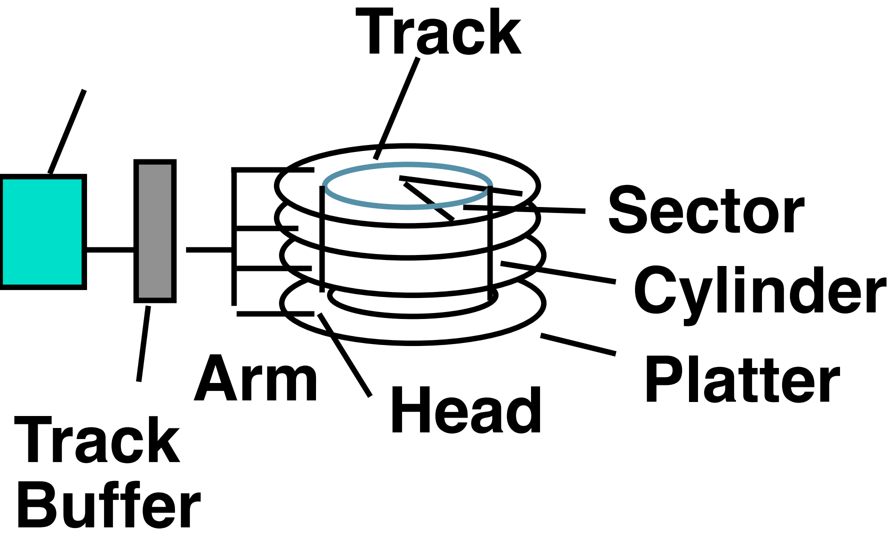
- To keep things simple, originally kept same number of sectors per track
  - Since outer track longer, lower bits per inch
- Competition decided to keep BPI the same for all tracks ("constant bit density")
  - More capacity per disk
  - More of sectors per track towards edge
  - Since disk spins at constant speed,
    outer tracks have faster data rate
- Bandwidth outer track 1.7X inner track!
  - Inner track highest density, outer track lowest, so not really constant
  - 2.1X length of track outer / inner, 1.7X bits outer / inner

Devices: Magnetic Disks

- Purpose:
  - Long-term, nonvolatile storage
  - Large, inexpensive, slow level in the storage hierarchy
- Characteristics:
  - Seek Time (~8 ms avg)
    - positional latency
    - rotational latency
  - Transfer rate
    - 10-40 MByte/sec
    - Blocks
  - Capacity
    - Gigabytes
    - Quadruples every 2 years (aerodynamics)

$$\text{Response time} = \text{Queue} + \underbrace{\text{Controller} + \text{Seek} + \text{Rot} + \text{Xfer}}_{\text{Service time}}$$

State of the Art: Barracuda 180



- 181.6 GB, 3.5 inch disk
- 12 platters, 24 surfaces
- 24,247 cylinders
- 7,200 RPM; (4.2 ms avg. latency)
- 7.4/8.2 ms avg. seek (r/w)
- 64 to 35 MB/s (internal)
- 0.1 ms controller time
- 10.3 watts (idle)

Latency =

$$\text{per access} + \text{per byte} \left\{ \begin{array}{l} \textbf{Queuing Time +} \\ \textbf{Controller time +} \\ \textbf{Seek Time +} \\ \textbf{Rotation Time +} \\ \textbf{Size / Bandwidth} \end{array} \right.$$

source: www.seagate.com

Disk Performance Example

- Disk characteristics
  - 512 byte sector, rotate at 5400 RPM, advertised seeks is 5 ms,
  - transfer rate is 40 MB/sec, it rotates at 10,000RPM, controller overhead is 0.1 ms, queue idle so no service time.
- Answer
  - Access Time = Seek time + Rotational Latency + Transfer time + Controller Time + Queuing Delay

$$= \mathbf{5ms} + \frac{\mathbf{0.5}}{\mathbf{10{,}000PRM}} + \frac{\mathbf{0.5KB}}{\mathbf{40MB/sec}} + \mathbf{0.1ms}$$

$$= \mathbf{5ms} + \mathbf{3.0} + \mathbf{0.013} + \mathbf{0.1} = \mathbf{8.11ms}$$

Disk Performance Example(cont.)

Assuming the measured seek time is 33% of the calculated average, the answer is

Access Time ＝33%×5ms ＋ 3.0 ms＋ 0.013ms ＋ 0.1ms ＝4.783ms

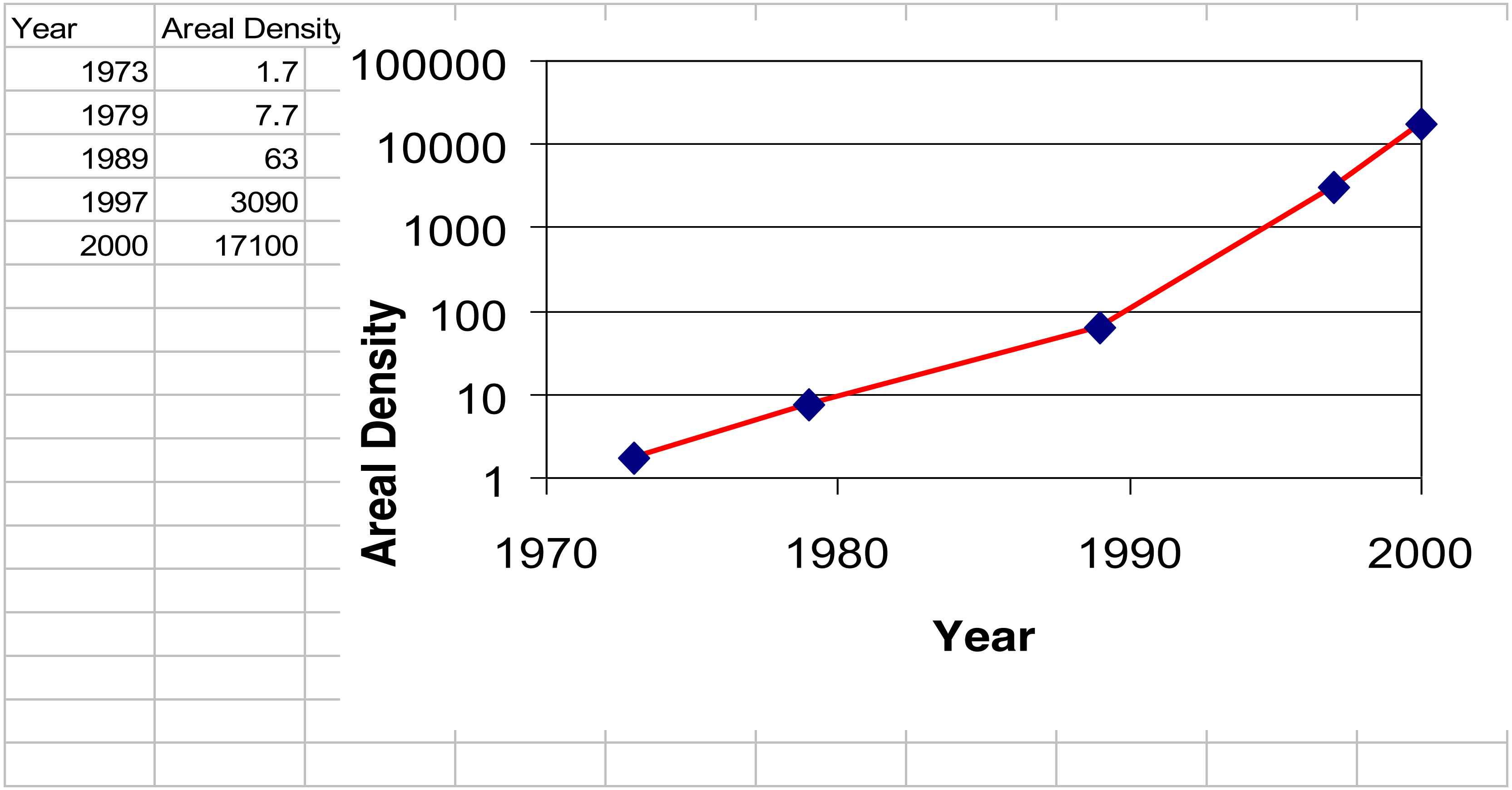
Note that **only 0.013 /4.783 or 0.3%** of the time is the disk transferring data in this example. Even page-sized transfers often take less than 5%, so disks normally spend most of their time **waiting for the head to get over the data** rather than reading or writing the data.

The Future of Magnetic Disks

- Bits recorded along a track
  - Metric is **Bits Per Inch** (BPI)
- Number of tracks per surface
  - Metric is **Tracks Per Inch** (TPI)
- Disk Designs Brag about bit density per unit area
  - Metric is **Bits Per Square** Inch
  - Called Areal Density
  - **Areal Density** = BPI x TPI

$$\text{Areal density} = \frac{\text{Tracks}}{\text{Inch}} \text{ on a disk surface} \times \frac{\text{Bits}}{\text{Inch}} \text{ on a track}$$

Areal Density



Year	Areal Density
1973	1.7
1979	7.7
1989	63
1997	3090
2000	17100

Areal Density = BPI x TPI

Change slope 29%/yr to 60%/yr about 1996

1 inch disk drive!

2000 IBM MicroDrive:

1.7" x 1.4" x 0.2"

1 GB, 3600 RPM,

5 MB/s, 15 ms seek

Digital camera, PalmPC?

2006 MicroDrive?

9 GB, 50 MB/s!

Assuming it finds a niche in a successful product

Assuming past trends continue

Optical Disks

One challenger

- High capacity、Low cost
- Read-only→Write once→ReWritable
- CD-DA、CD-ROM、CD-I、CD-R、VCD、DVD
- Pits(0.5μm)、lands、

Magnetic Tapes vs. Disk

- Longitudinal tape uses same technology as hard disk; tracks its density improvements
- Disk head flies above surface, tape head lies on surface
- Disk fixed, tape removable
- Inherent cost-performance based on geometries:
- fixed rotating platters with gaps
  - (random access, limited area, 1 media / reader)
- removable long strips wound on spool
  - (sequential access, "unlimited" length, multiple / reader)
- Helical Scan (VCR, Camcoder, DAT)
  - Spins head at angle to tape to improve density

Current Drawbacks to Tape

- Tape wear out:
  - Helical 100s of passes to 1000s for longitudinal
- Head wear out:
  - 2000 hours for helical
- Both must be accounted for in economic / reliability model
- Bits stretch
- Readers must be compatible with multiple generations of media
- Long rewind, eject, load, spin-up times; not inherent, just no need in marketplace
- Designed for archival

Automated Tape Libraries StorageTek Powderhorn 9310

7.7 feet

8200 pounds,
1.1 kilowatts

10.7 feet

- 6000 x 50 GB 9830 tapes = 300 TBytes in 2000 (uncompressed)
  - Library of Congress: all information in the world; in 1992, ASCII of all books = 30 TB
  - Exchange up to 450 tapes per hour (8 secs/tape)
- 1.7 to 7.7 Mbyte/sec per reader, up to 10 readers

Library vs. Storage

- Getting books today as quaint as the way I learned to program
  - punch cards, batch processing
  - wander thru shelves, anticipatory purchasing
- Cost $1 per book to check out
- $30 for a catalogue entry
- 30% of all books never checked out
- Write only journals?
- Digital library can transform campuses

Whither tape?

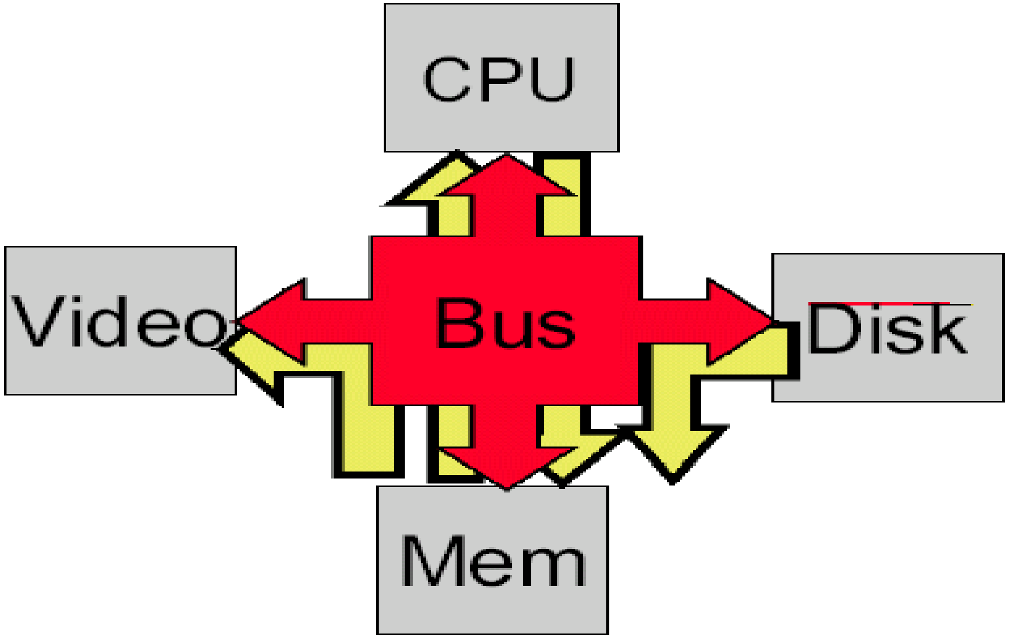
- Investment in research:
  - 90% of disks shipped in PCs; 100% of PCs have disks
  - ~0% of tape readers shipped in PCs; ~0% of PCs have disks
- Before, N disks / tape; today, N tapes / disk
  - 40 GB/DLT tape (uncompressed)
  - 80 to 192 GB/3.5" disk (uncompressed)
- Cost per GB:
  - In past, 10X to 100X tape cartridge vs. disk
  - Jan 2001: 40 GB for $53 (DLT cartridge), $2800 for reader
  - $1.33/GB cartridge, $2.03/GB 100 cartridges + 1 reader
  - ($10995 for 1 reader + 15 tape autoloader, $10.50/GB)
  - Jan 2001: 80 GB for $244 (IDE,5400 RPM), $3.05/GB
  - Will $/GB tape v. disk cross in 2001? 2002? 2003?
- Storage field is based on tape backup; what should we do? Discussion if time permits?

What about FLASH

- Compact Flash Cards
  - Intel Strata Flash
    - 16 Mb in 1 square cm. (.6 mm thick)
  - 100,000 write/erase cycles.
  - Standby current = 100uA, write = 45mA
  - Compact Flash 256MB~=$120 512MB~=$542
  - Transfer @ 3.5MB/s
- IBM Microdrive 1G~370
  - Standby current = 20mA, write = 250mA
  - Efficiency advertised in wats/MB
- Disks
  - Nearly instant standby wake-up time
  - Random access to data stored
  - Tolerant to shock and vibration (1000G of operating shock)

7.3 Buses--Connecting I/O Devices to CPU/Memory

Lots of sub-systems need to communicate



Busses: Shared wires for common communication

Bus Classifications

- CPU-memory busses

  Fast

  Proprietary

  Closed and controlled

  Support only memory transaction:

- IO busses

  Standardized (SCSI, PCI, AGP)

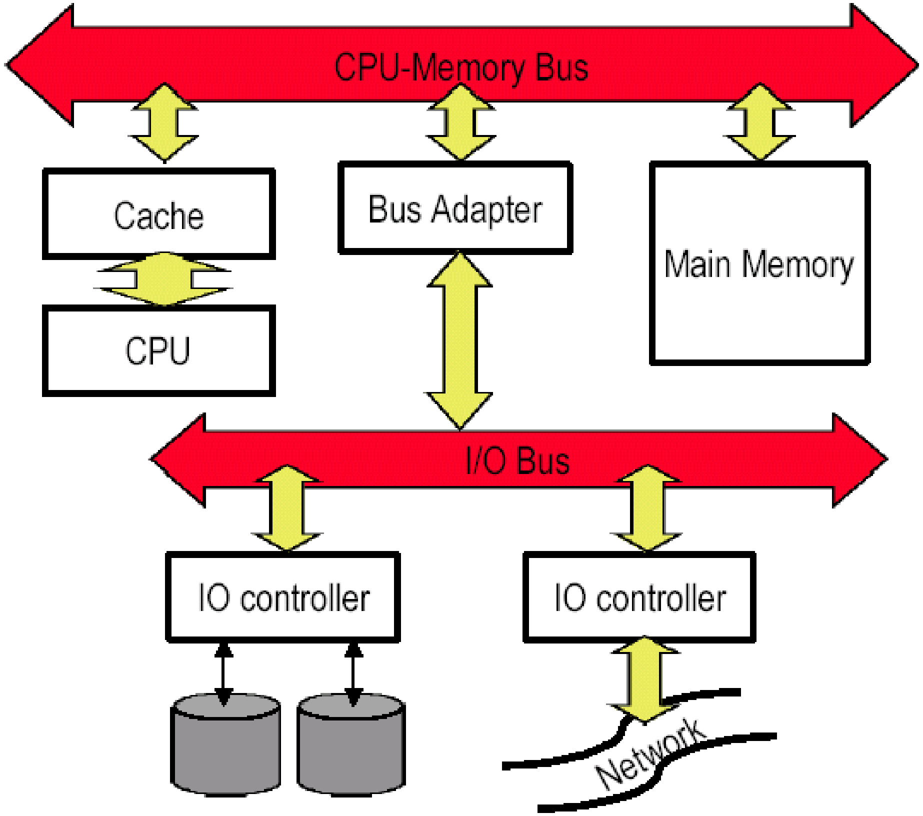
  More diversity

  More length

- Bus Bridges/Adapter

  Standardized (RS-232,)

  Cross from one bus to another



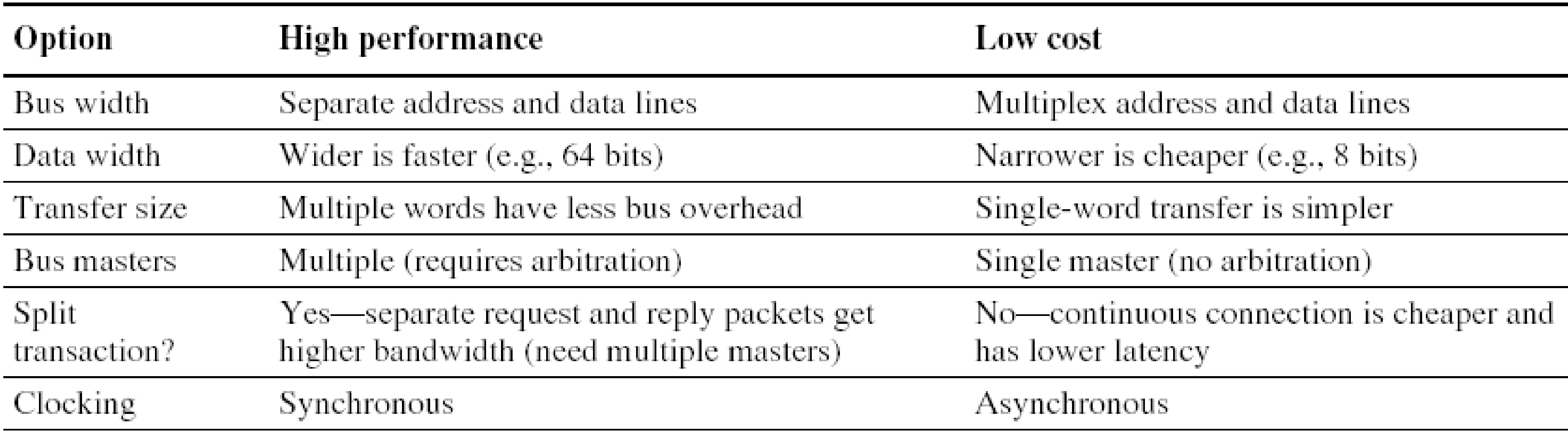
Bus Design Decisions

goals

- decisions depend on cost and performance
- higher performance at more cost.

The first three options in the figure are clear

- separate address and data lines,
- wider data lines, and multiple
- word transfers



Option	High performance	Low cost
Bus width	Separate address and data lines	Multiplex address and data lines
Data width	Wider is faster (e.g., 64 bits)	Narrower is cheaper (e.g., 8 bits)
Transfer size	Multiple words have less bus overhead	Single-word transfer is simpler
Bus masters	Multiple (requires arbitration)	Single master (no arbitration)
Split transaction?	Yes—separate request and reply packets get higher bandwidth (need multiple masters)	No—continuous connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

Structure, Width, and Transfer Length

Separate vs. Multiplexed Address/Data

- Multiplexed: save wires
- Separate: more performance

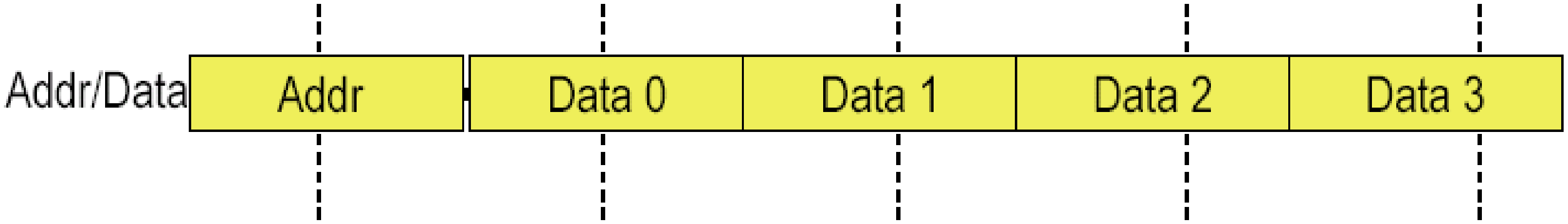
Wide words: higher throughput, less control per transfer

- On-chip cache to CPU busses: 256 bits wide
- Serial Busses

Data Transfer Length

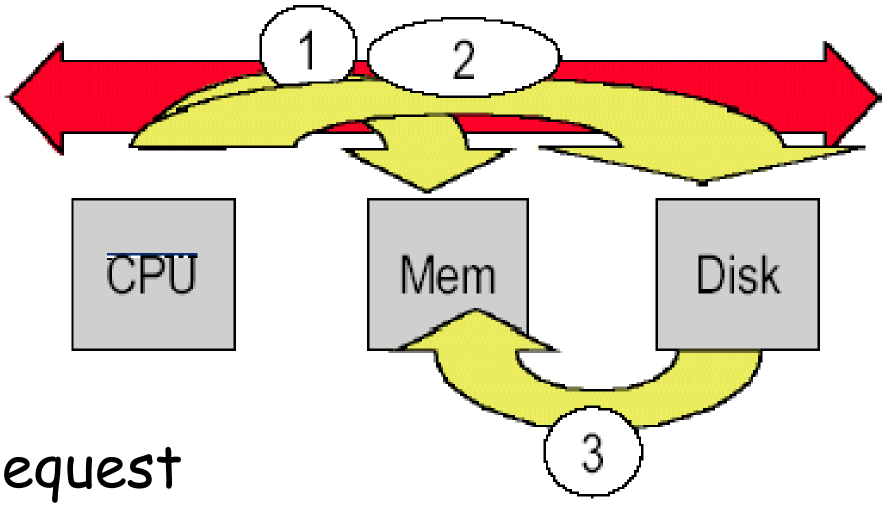
- More data per address/control transfer

Example: Multiplexed Addr/Data with Data transfer of 4



Bus Mastering

Bus Master: a device that can initiate a bus transfer



1. CPU makes memory request
2. Page Fault in VM requires disk access to load page
3. Mover data from disk to memory

If the CPU is master, does it have to check to see if the disk is ready to transfer?

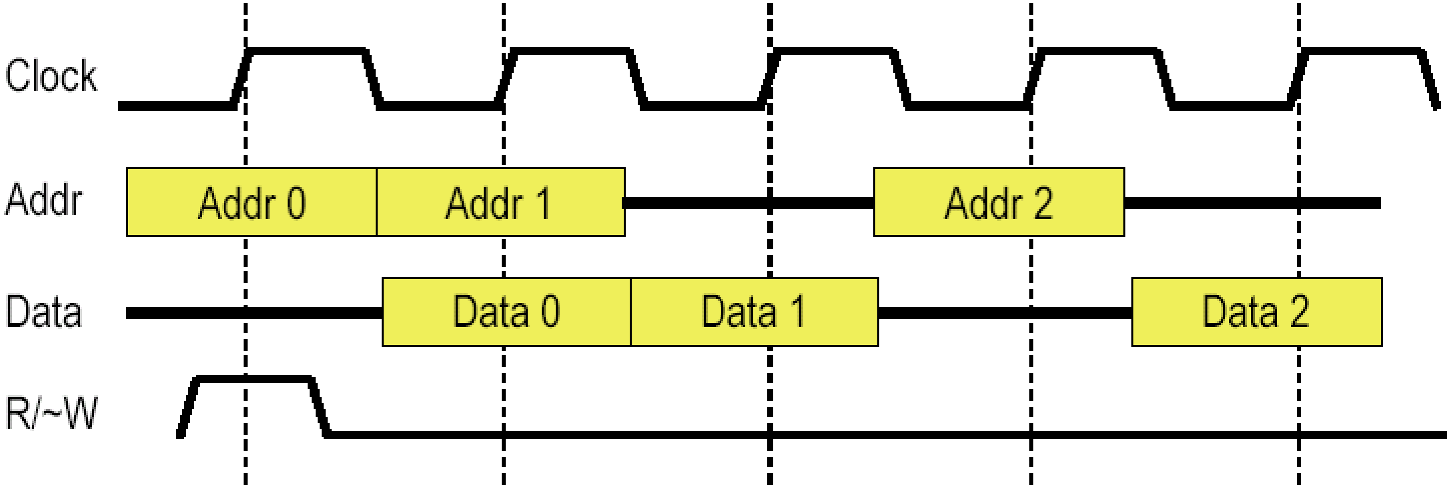
Multiple Bus Masters

- What if multiple devices could initiate transfers?
  - Update might take place in background while CPU operates
- Multiple CPUs on shared memory systems
- Challenge: *Arbitration*
  - If two or more masters want the bus at the same time, who gets it?

Bus Clocking: Synchronous

Synchronous

Sample the control signals at edge of clock



Pro: Fast and High Performance

Con:

- Can’t be long (skew) or fast at same time
- All bus members must run at the right speed

Bus Clocking: Asynchronous

Asynchronous

- Edge of control signals determines communication
- "Handshake Protocol"

Pros:

- No clock
- Slow and fast components on the same bus

Con:

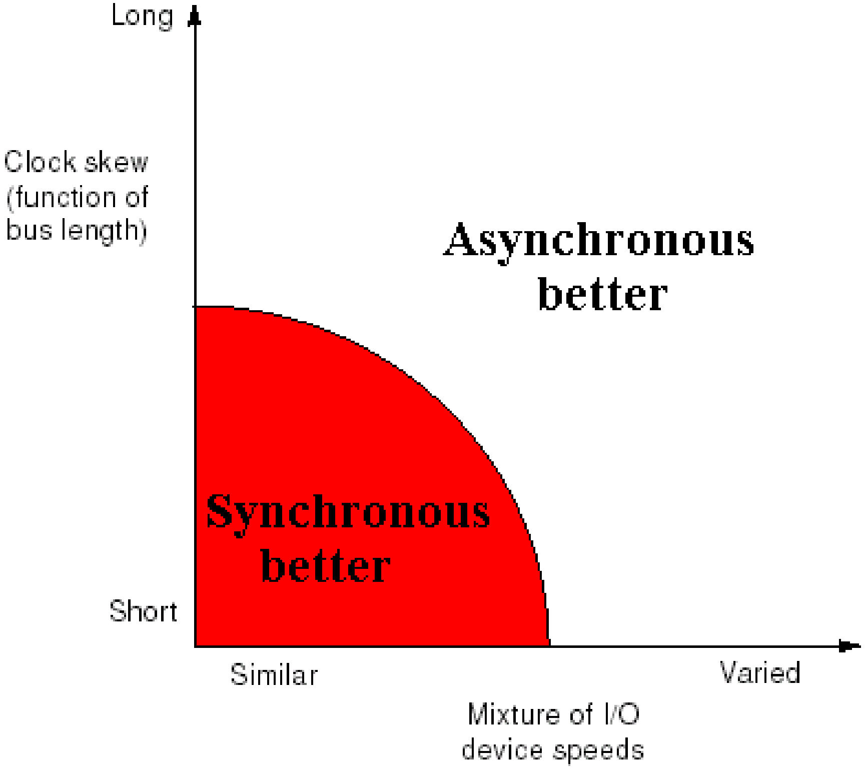
- Inefficient: two round trips

Like somebody who always repeats what was said to them

1. *Request* (with actual transaction)
2. *Acknowledge* causes de-assert of *Request*
3. De-assert of *Request* causes de-assert of *Ack*
4. De-assert of *Ack* allows re-assertion of *Request*

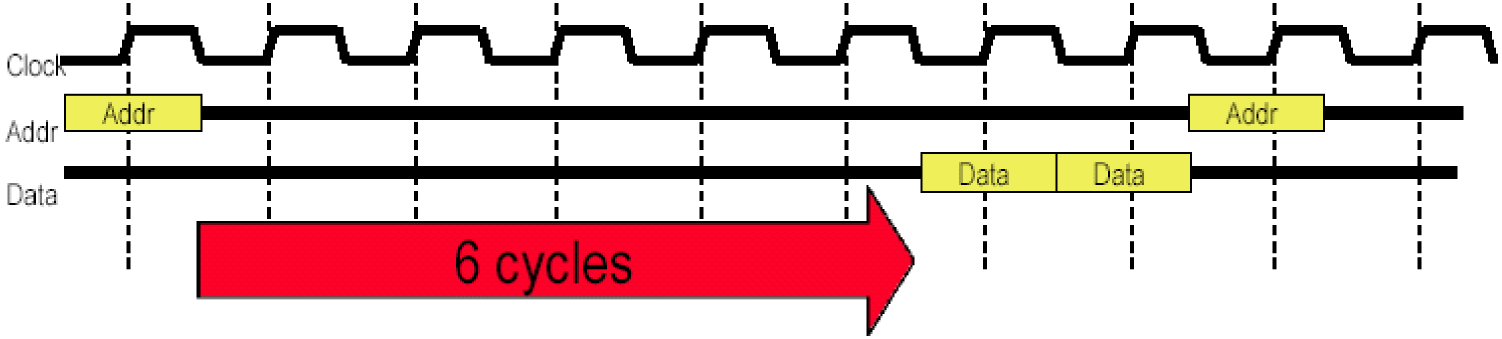
Synchronous vs Asynchronous

- Preferred bus type as a function of length/clock skew and variation in I/O device speed.
- **Synchronous is best when the distance is short and the I/O devices on the bus all transfer at similar speeds.**

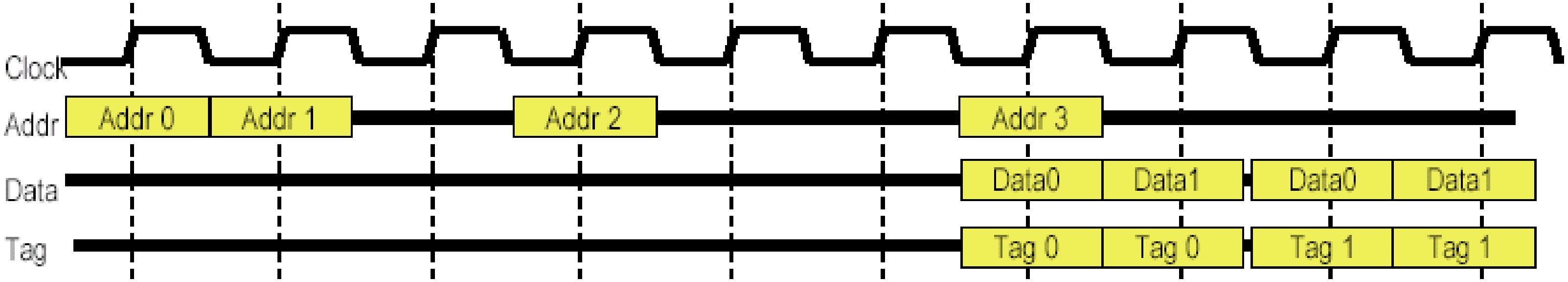

Long
Clock skew (function of bus length)
Asynchronous better
Synchronous better
Short
Similar
Varied
Mixture of I/O device speeds

Split Transactions

- Problem: Long wait times



- Solution: Split Transaction Bus



Bus Standards

- I/O bus --- interface---devices

Standards

  - let the computer designer and I/O-device designer work independently play a large role in buses.
  - Any I/O device can connect to any computer.

- Document
  - Defines how to connect devices to computers
- De facto standards
  - Machines sometimes grow to be so popular that their I/O buses become de facto standards
  - PDP-11 Unibus、IBM PC-AT

Examples of Buses

- Buses in common use
  - – Common desktop I/O buses,
  - – I/O buses found in embedded devices,
  - – CPU-memory interconnects found in servers

Summary of parallel I/O buses.

Summary of serial I/O buses (Embedded computers)

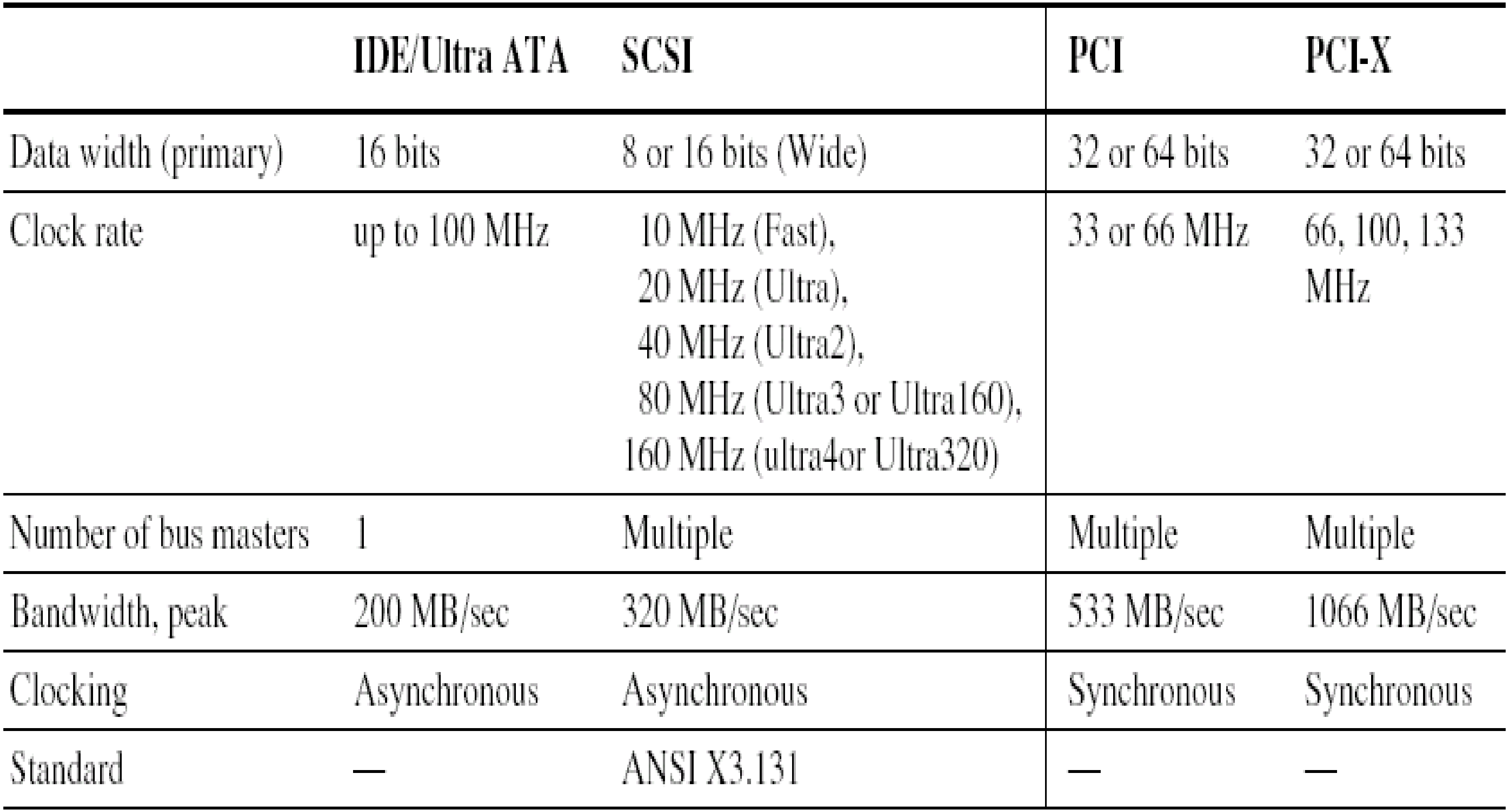
Summary of CPU-memory interconnects found in 2000 servers.

Summary of parallel I/O buses

- IDE---Integrated Drive Electronics
  - Early disk standard that connects two disks to a PC.
  - It has been extended by *AT-bus Attachment* (*ATA*), to be both wider and faster.
- SCSI---Small Computer System Interconnect
  - connects up to 7 devices for 8-bit busses and up to 15 devices for 16-bit busses.
  - They can even be different speeds, but they run at the rate of the slowest device.
  - The peak bandwidth of a SCIS bus is the width (1 or 2 bytes) times the clock rate (10 to 160 MHz). Most SCSI buses today are 16-bits.
- PCI---Peripheral Component Interconnect

  PCI-X ,PCI Extended
  - Connect main memory to peripheral devices

Summary of parallel I/O buses



	IDE/Ultra ATA	SCSI	PCI	PCI-X
Data width (primary)	16 bits	8 or 16 bits (Wide)	32 or 64 bits	32 or 64 bits
Clock rate	up to 100 MHz	10 MHz (Fast), 20 MHz (Ultra), 40 MHz (Ultra2), 80 MHz (Ultra3 or Ultra160), 160 MHz (ultra4or Ultra320)	33 or 66 MHz	66, 100, 133 MHz
Number of bus masters	1	Multiple	Multiple	Multiple
Bandwidth, peak	200 MB/sec	320 MB/sec	533 MB/sec	1066 MB/sec
Clocking	Asynchronous	Asynchronous	Synchronous	Synchronous
Standard	—	ANSI X3.131	—	—

Summary of serial I/O buses

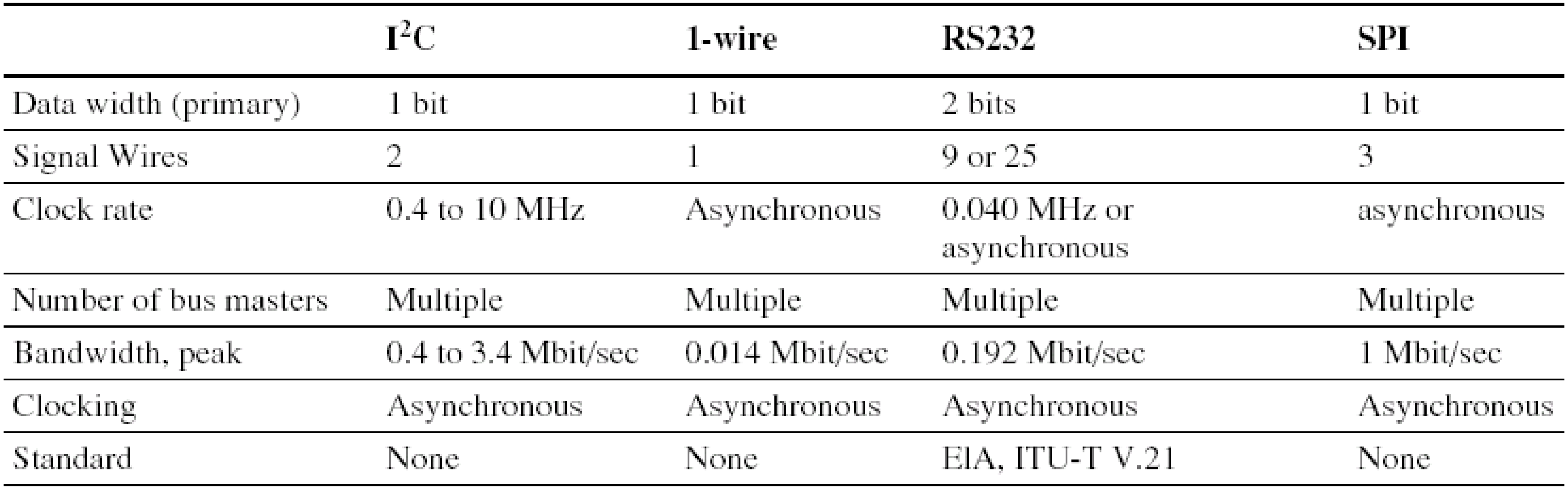
Often used in embedded computers.

I²C ----- invented by Phillips in the early 1980s.

1-wire -- developed by Dallas Semiconductor.

RS-232 -introduced in 1962.

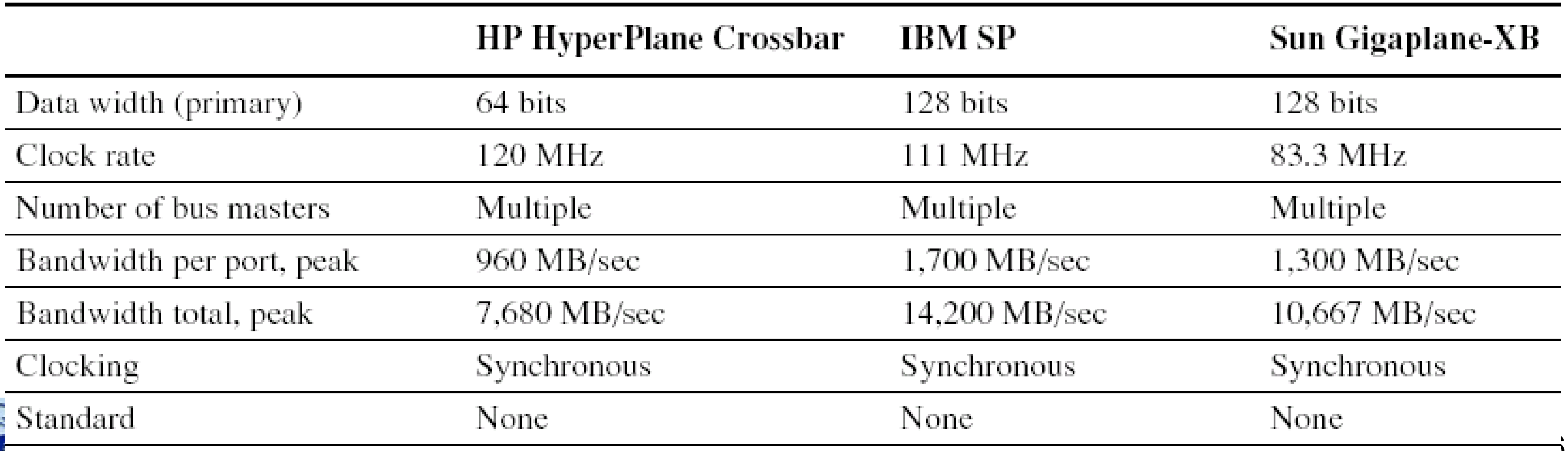
SPI ----- created by Motorola in the early 1980s.



	I²C	1-wire	RS232	SPI
Data width (primary)	1 bit	1 bit	2 bits	1 bit
Signal Wires	2	1	9 or 25	3
Clock rate	0.4 to 10 MHz	Asynchronous	0.040 MHz or asynchronous	asynchronous
Number of bus masters	Multiple	Multiple	Multiple	Multiple
Bandwidth, peak	0.4 to 3.4 Mbit/sec	0.014 Mbit/sec	0.192 Mbit/sec	1 Mbit/sec
Clocking	Asynchronous	Asynchronous	Asynchronous	Asynchronous
Standard	None	None	EIA, ITU-T V.21	None

Summary of CPU-memory interconnects found in 2000 servers

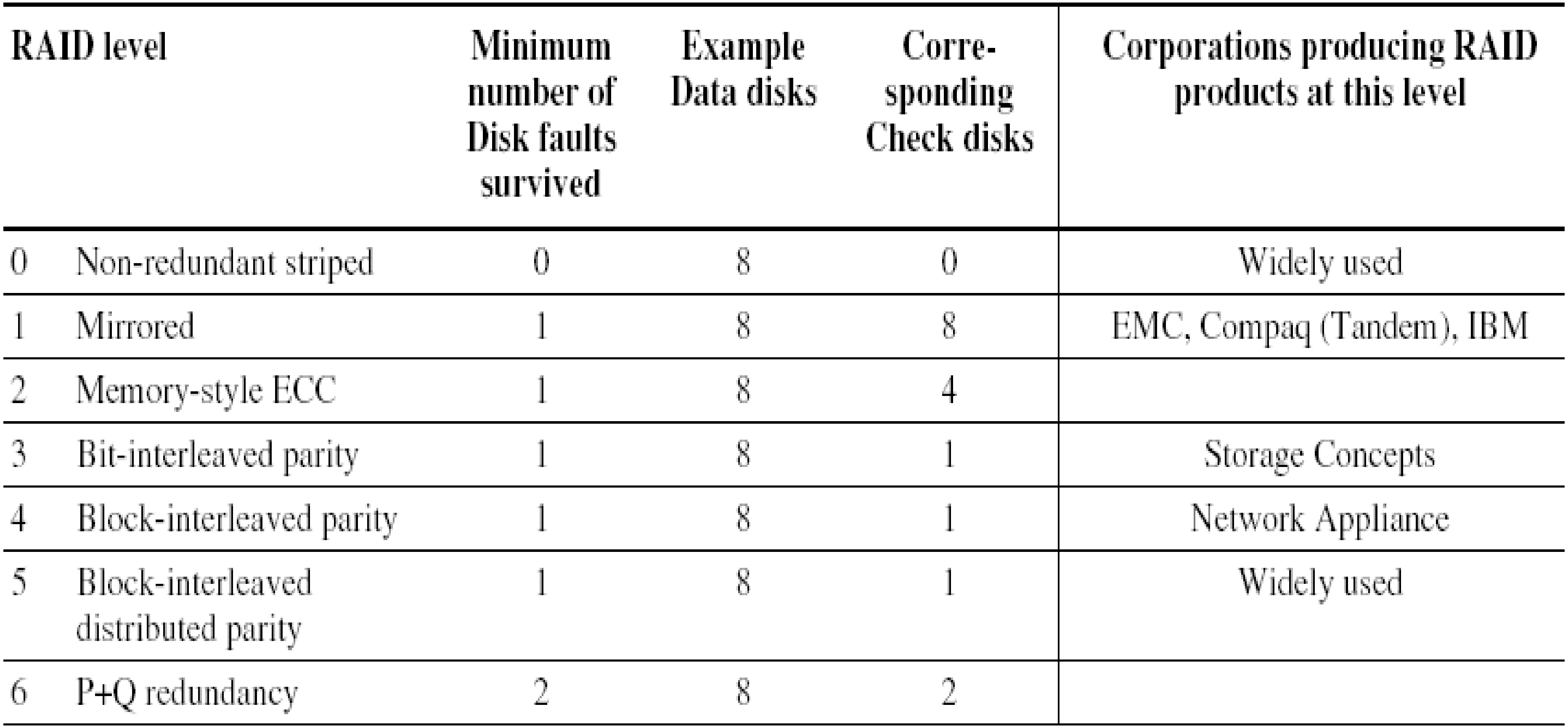
- Shared bus
- crossbars switches
  - Each bus connects up to four processors and memory controllers, and then the crossbar connects the busses together.
  - The number of slots in the crossbar is 16, 8, and 16, respectively.
  - These servers use crossbars switches to connect nodes processors together instead of a shared bus interconnect.



	HP HyperPlane Crossbar	IBM SP	Sun Gigaplane-XB
Data width (primary)	64 bits	128 bits	128 bits
Clock rate	120 MHz	111 MHz	83.3 MHz
Number of bus masters	Multiple	Multiple	Multiple
Bandwidth per port, peak	960 MB/sec	1,700 MB/sec	1,300 MB/sec
Bandwidth total, peak	7,680 MB/sec	14,200 MB/sec	10,667 MB/sec
Clocking	Synchronous	Synchronous	Synchronous
Standard	None	None	None

7.5 RAID: Redundant Arrays of Inexpensive Disks

A disk arrays replace larger disk



RAID level		Minimum number of Disk faults survived	Example Data disks	Corre-sponding Check disks	Corporations producing RAID products at this level
0	Non-redundant striped	0	8	0	Widely used
1	Mirrored	1	8	8	EMC, Compaq (Tandem), IBM
2	Memory-style ECC	1	8	4	
3	Bit-interleaved parity	1	8	1	Storage Concepts
4	Block-interleaved parity	1	8	1	Network Appliance
5	Block-interleaved distributed parity	1	8	1	Widely used
6	P+Q redundancy	2	8	2	

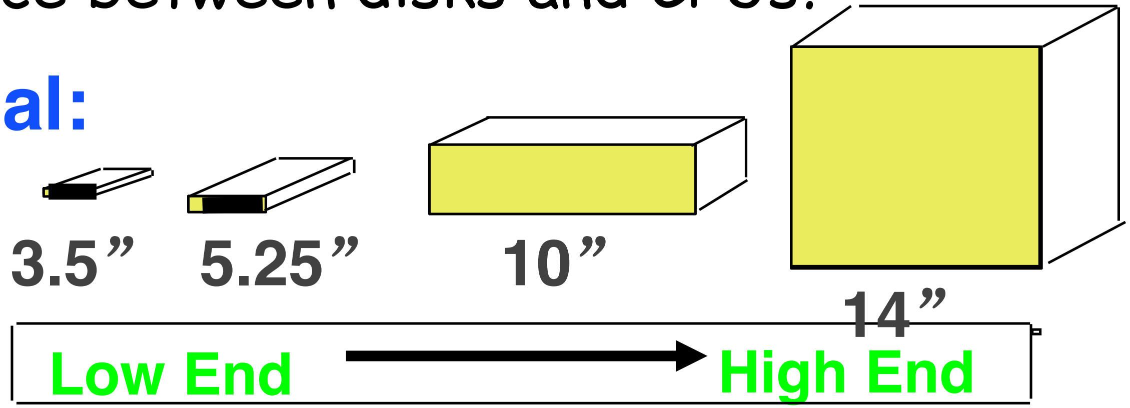
- David patterson, Garth Gibson, and Randy Katz, A Case for Redundant Arrays of Inexpensive Disks (RAID), *ACM SIGMOD conference,* 1988

Use Arrays of Small Disks?

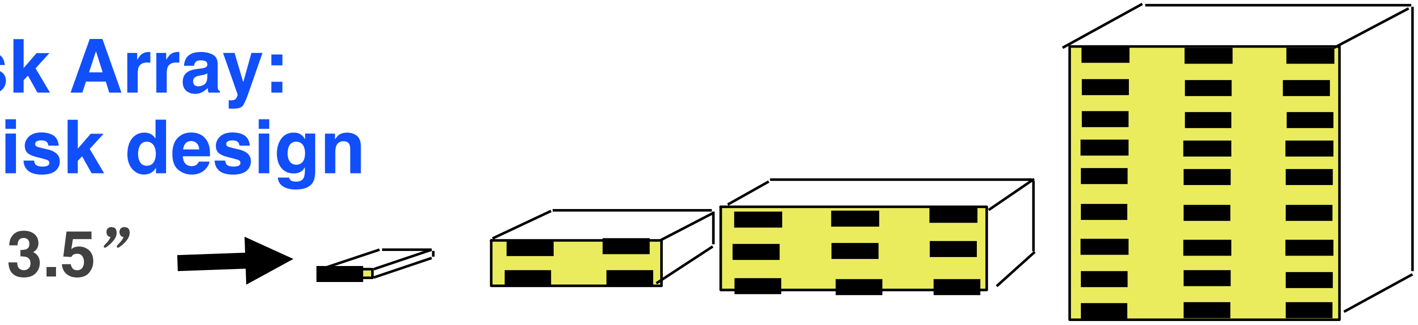
- Katz and Patterson asked in 1987:

  Can smaller disks be used to close gap in performance between disks and CPUs?

Conventional:
4 disk designs



Disk Array:
1 disk design



Replace Small Number of Large Disks with Large Number of Small Disks! (1988 Disks)

	IBM 3390K	IBM 3.5" 0061	x70	
Capacity	20 GBytes	320 MBytes	23 GBytes	
Volume	97 cu. ft.	0.1 cu. ft.	11 cu. ft.	9X
Power	3 KW	11 W	1 KW	3X
Data Rate	15 MB/s	1.5 MB/s	120 MB/s	8X
I/O Rate	600 I/Os/s	55 I/Os/s	3900 IOs/s	6X
MTTF	250 KHrs	50 KHrs	??? Hrs	
Cost	$250K	$2K	$150K	

Disk Arrays have potential for large data and I/O rates, high MB per cu. ft., high MB per KW, but what about reliability?

Array Reliability

Reliability of N disks = Reliability of 1 Disk ÷ N

50,000 Hours ÷ 70 disks = 700 hours

Disk system MTTF: Drops from 6 years to 1 month!

- Arrays (without redundancy) too unreliable to be useful!

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

Redundant Arrays of (Inexpensive) Disks

- Files are "striped" across multiple disks
- Redundancy yields high data availability
  - Availability: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array

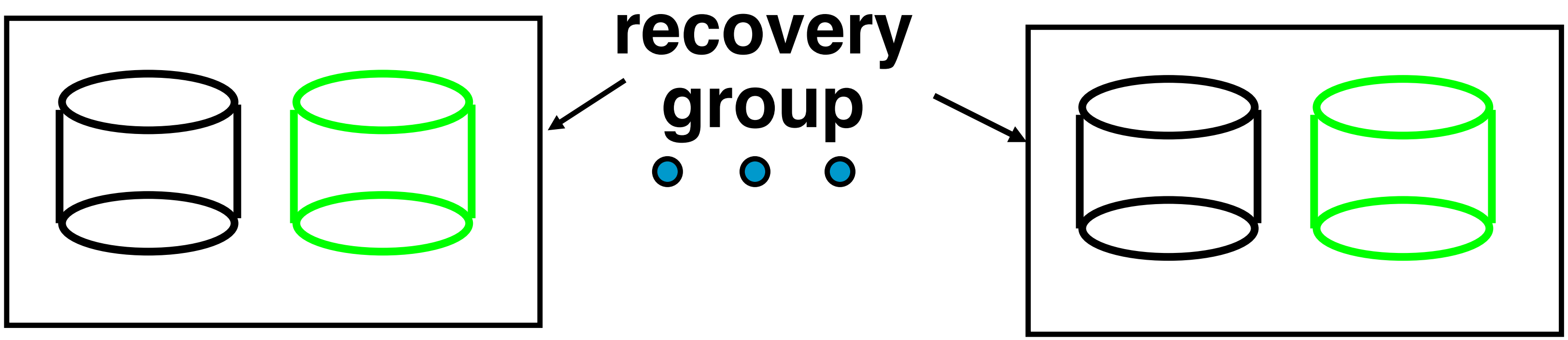
  Capacity penalty to store redundant info

  Bandwidth penalty to update redundant info

RAID 0: No Redundancy

- Data is striped across a disk array but there is no redundancy to tolerate disk failure
- It also improves performance for large accesses, since many diskscan operate at once.
- RAID 0 something of a misnomer as there is no redundancy,

RAID 1: Disk Mirroring/Shadowing



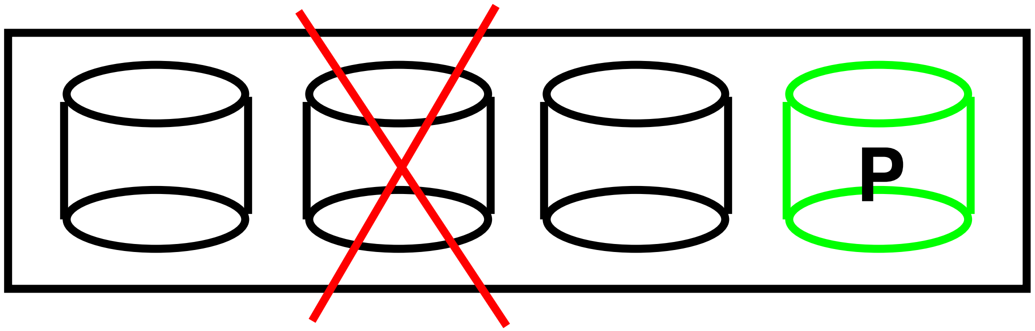
- Each disk is fully duplicated onto its "mirror"
  Very high availability can be achieved
- Bandwidth sacrifice on write:
  Logical write = two physical writes
  - Reads may be optimized
- Most expensive solution: 100% capacity overhead

- (RAID 2 not interesting, so skip)

RAID 3: Bit-Interleaved Parity Disk

10010011
11001101
10010011
. . .

logical record



Striped physical records

1	1	1	1
0	1	0	1
1	0	1	0
0	0	0	0
0	1	0	1
0	1	0	1
1	0	1	0
1	1	1	1

P contains sum of other disks per stripe mod 2 ("parity")

If disk fails, subtract P from sum of other disks to find missing information

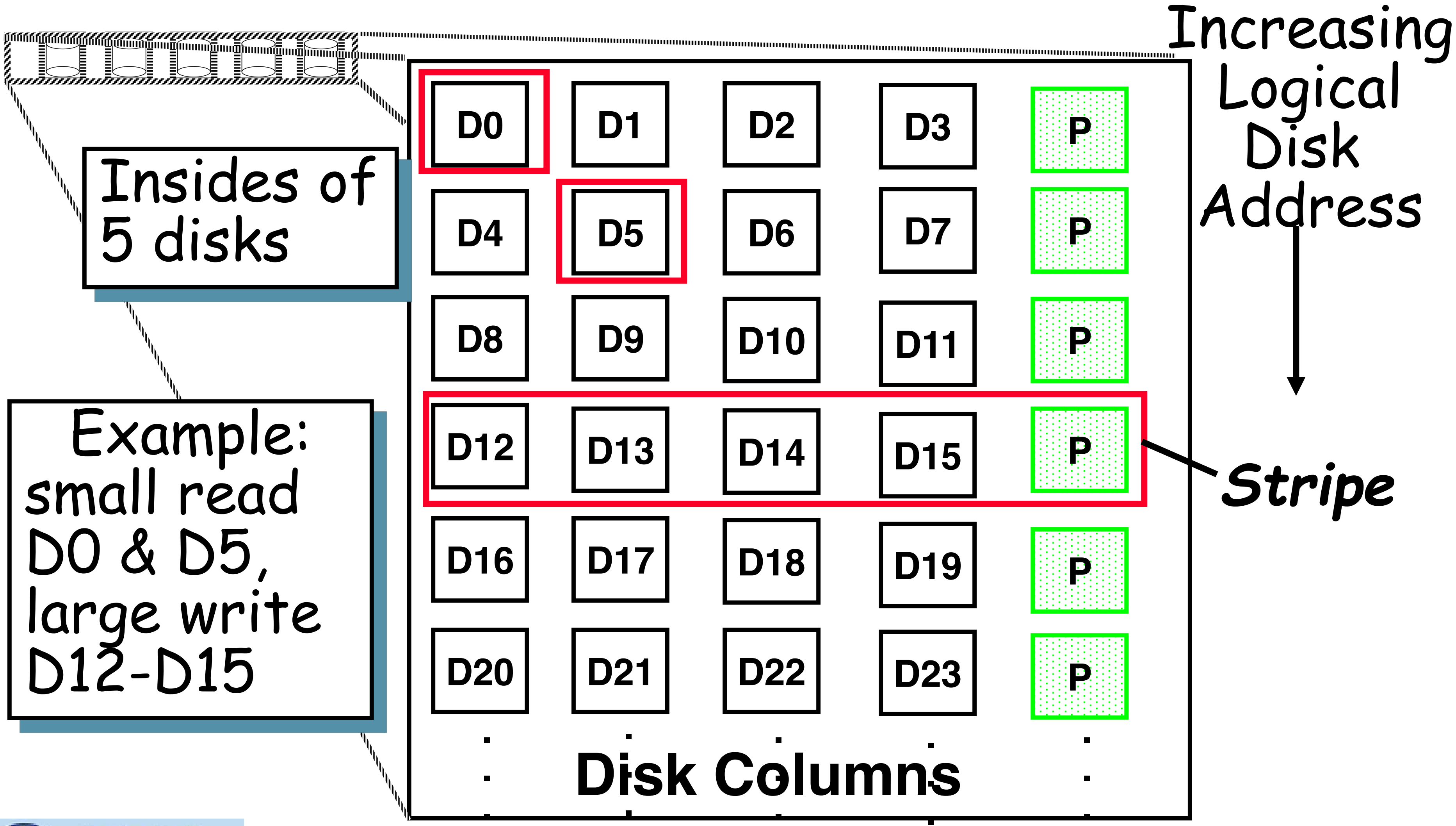
RAID 3

- Sum computed across recovery group to protect against hard disk failures, stored in P disk
- Logically, a single high capacity, high transfer rate disk: good for large transfers
- Wider arrays reduce capacity costs, but decreases availability
- 33% capacity cost for parity in this configuration

Inspiration for RAID 4

- RAID 3 relies on parity disk to discover errors on Read
- But every sector has an error detection field
- Rely on error detection field to catch errors on read, not on the parity disk
- Allows independent reads to different disks simultaneously

RAID 4: High I/O Rate Parity



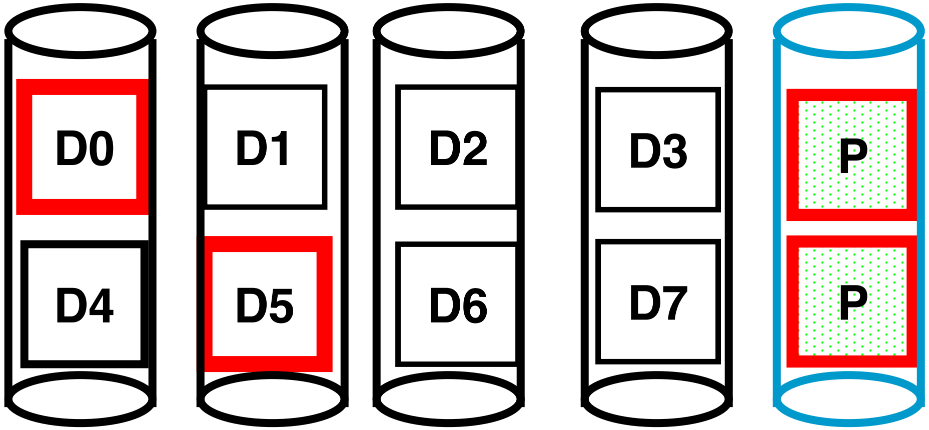
Insides of 5 disks

Example: small read D0 & D5, large write D12-D15

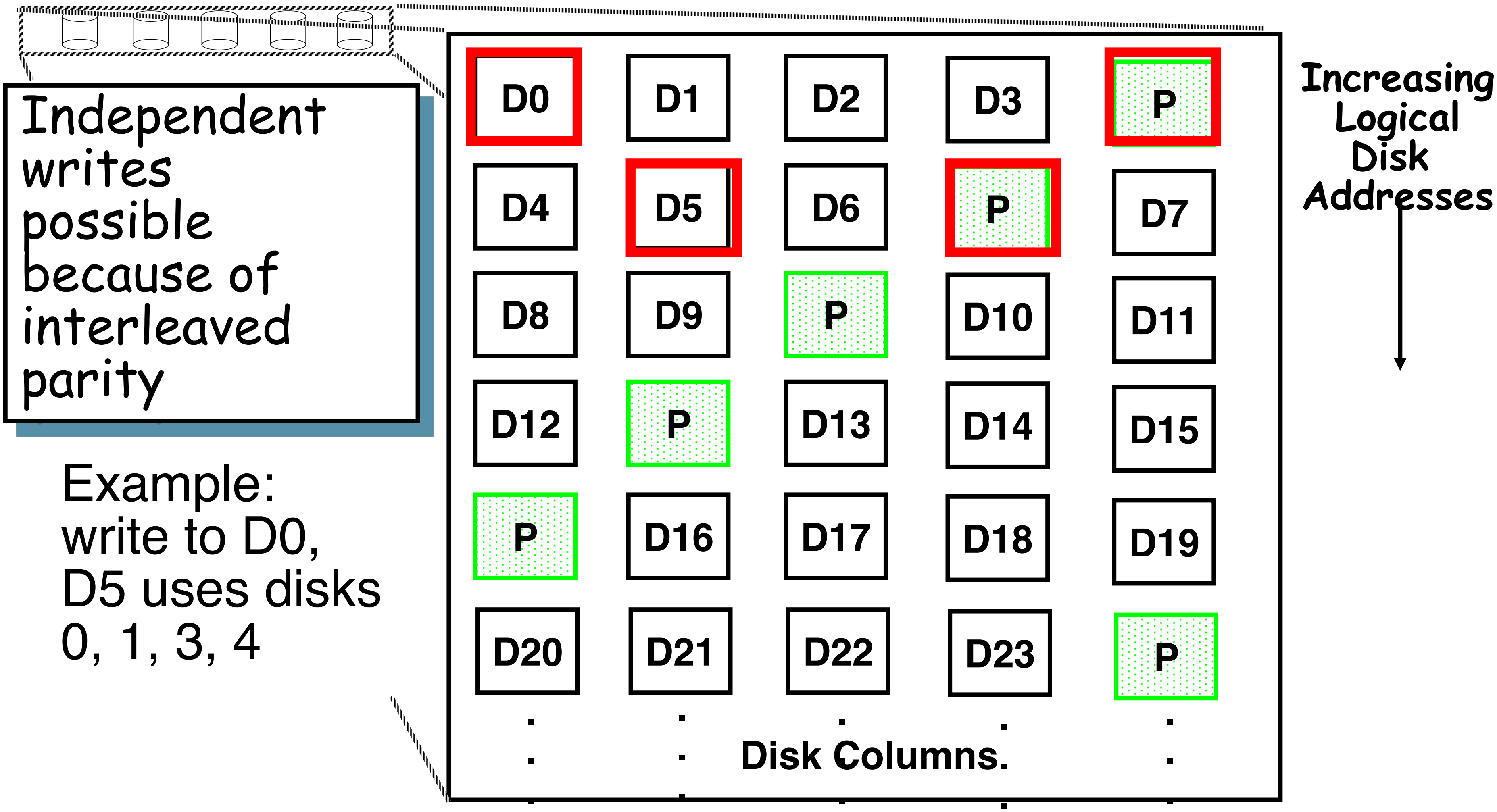
D0	D1	D2	D3	P
D4	D5	D6	D7	P
D8	D9	D10	D11	P
D12	D13	D14	D15	P
D16	D17	D18	D19	P
D20	D21	D22	D23	P
.	.	.	.	.

Inspiration for RAID 5

- RAID 4 works well for small reads
- Small writes (write to one disk):
  - Option 1: read other data disks, create new sum and write to Parity Disk
  - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk



RAID 5: High I/O Rate Interleaved Parity



D0	D1	D2	D3	P
D4	D5	D6	P	D7
D8	D9	P	D10	D11
D12	P	D13	D14	D15
P	D16	D17	D18	D19
D20	D21	D22	D23	P
.	.	.	.	.
.	.	.	.	.

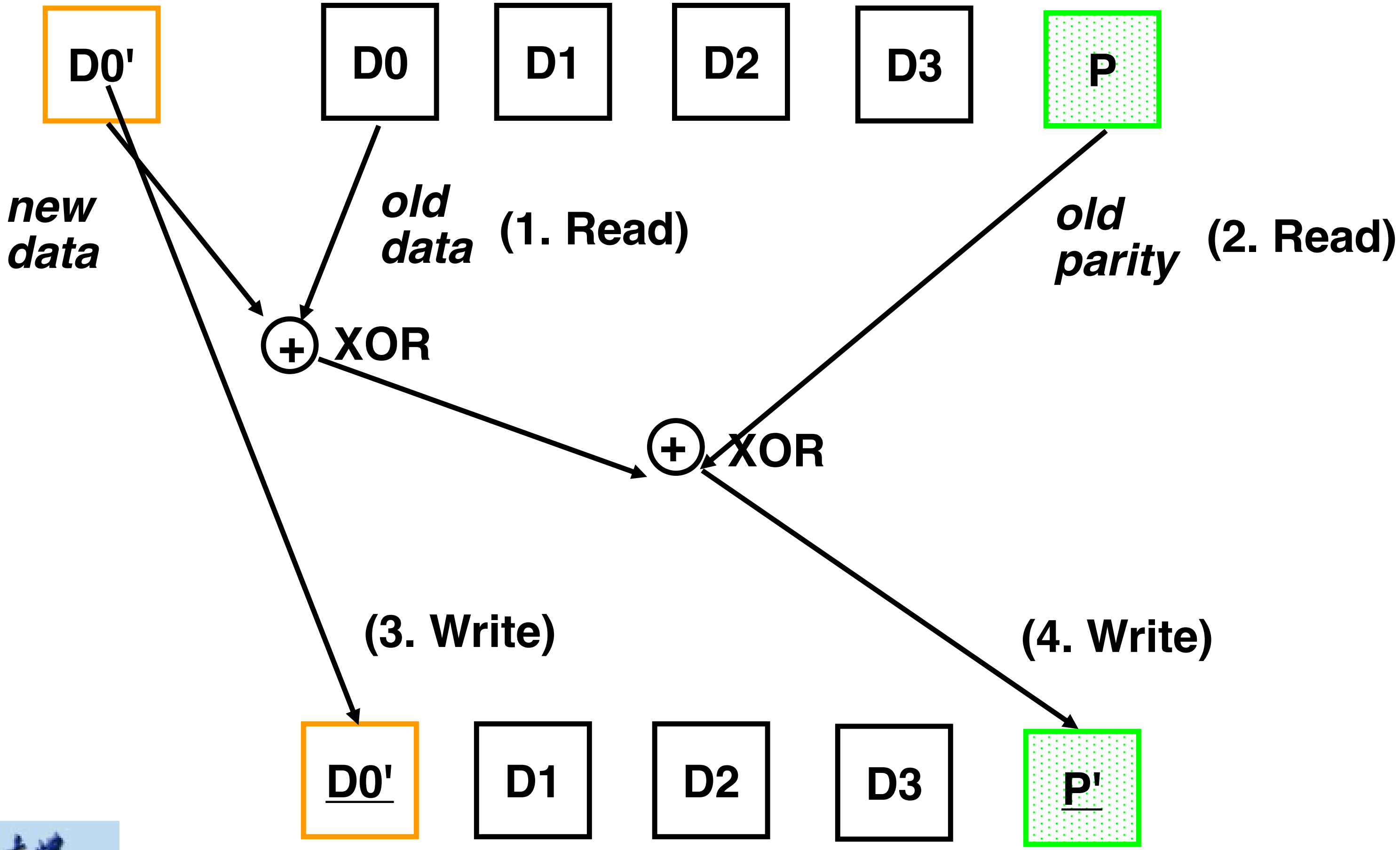
Independent writes possible because of interleaved parity

Example: write to D0, D5 uses disks 0, 1, 3, 4

Problems of Disk Arrays: Small Writes

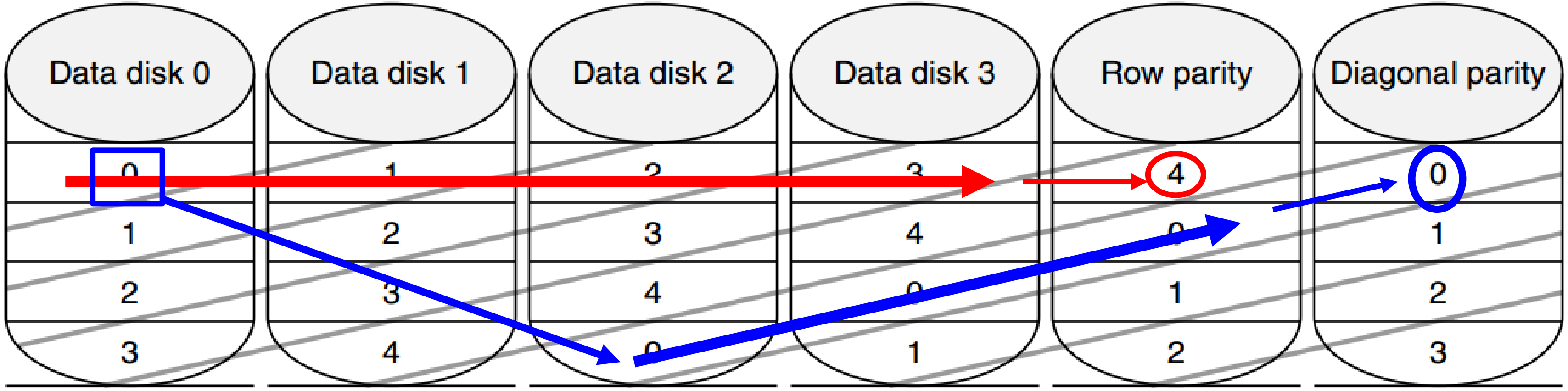
RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



RAID-DP: P365 in 4th Edition

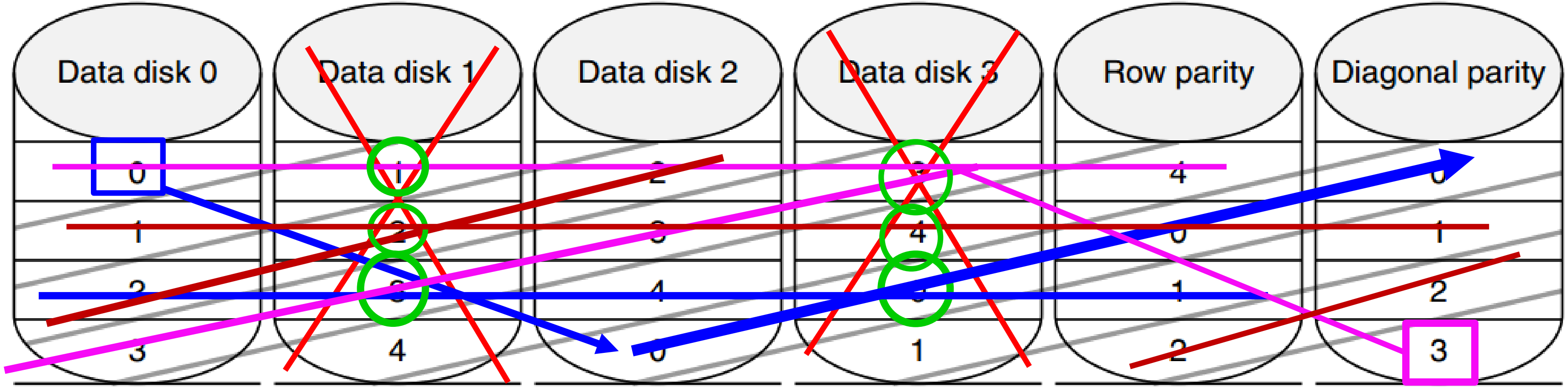
- Protect against double failure
  - Row parity: one parity for per-stripe in red
  - Diagonal parity: showed in blue.
  - Row-Diagonal for p=5
    - P-1 dada disk, 1 row parity, 1 diagonal parity, total P+1 disk



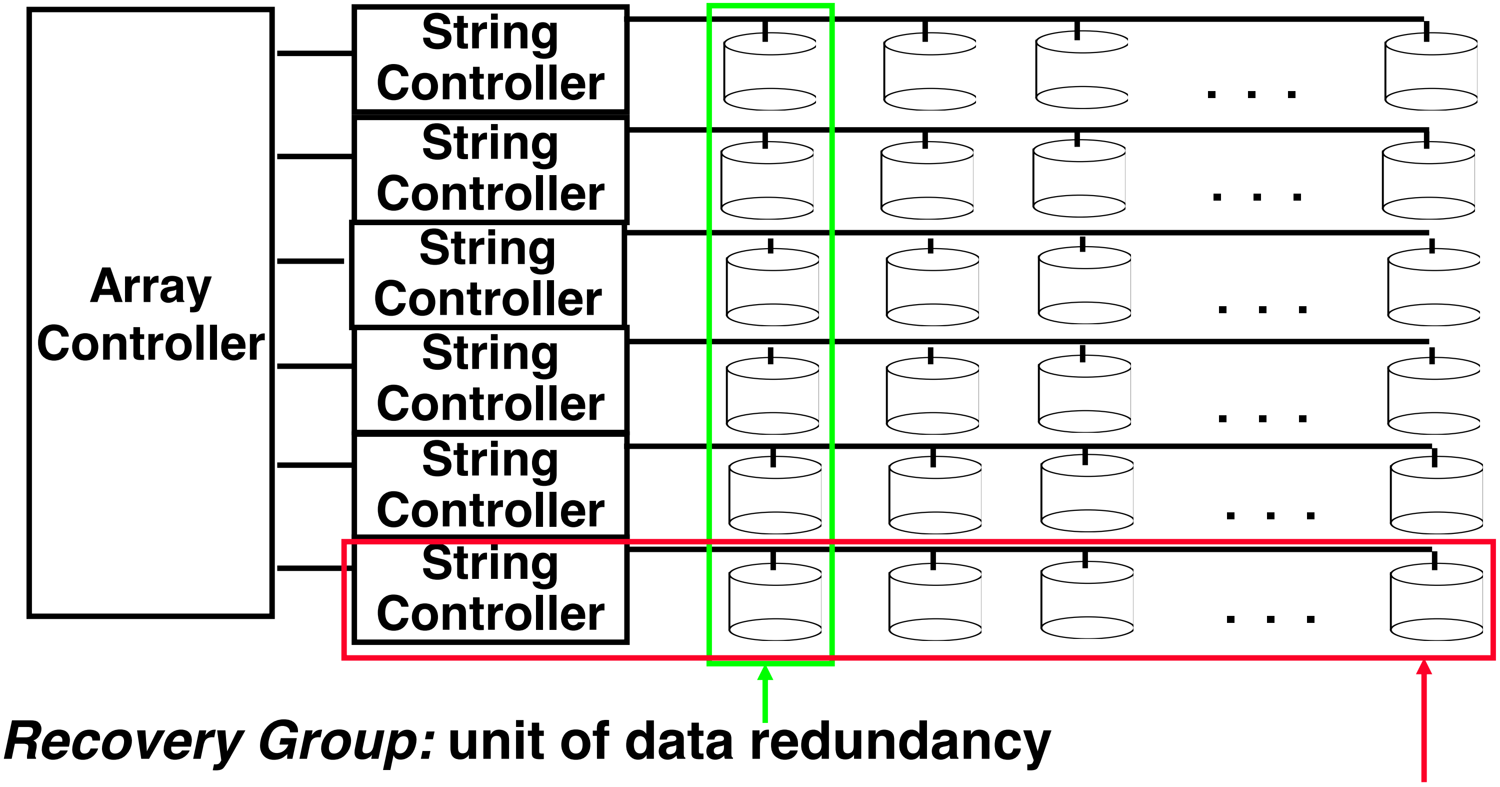
Data disk 0	Data disk 1	Data disk 2	Data disk 3	Row parity	Diagonal parity
0	1	2	3	4	0
1	2	3	4	0	1
2	3	4	0	1	2
3	4	0	1	2	3

Case: Recovery of double failure

- Recovery solution 1:
  - → D3-0(from Diagonal parity)
  - → D1-3(from Row parity)
  - → D3-3(Diagonal Parity)
  - → D1-1(Row parity)
  - → D2-2(Diagonal Parity)
  - → D3-4(Row Parity)
  - → D3-1 (Diagonal Parity)
  - → D1-4 (Row Parity)



System Availability: Orthogonal RAIDs

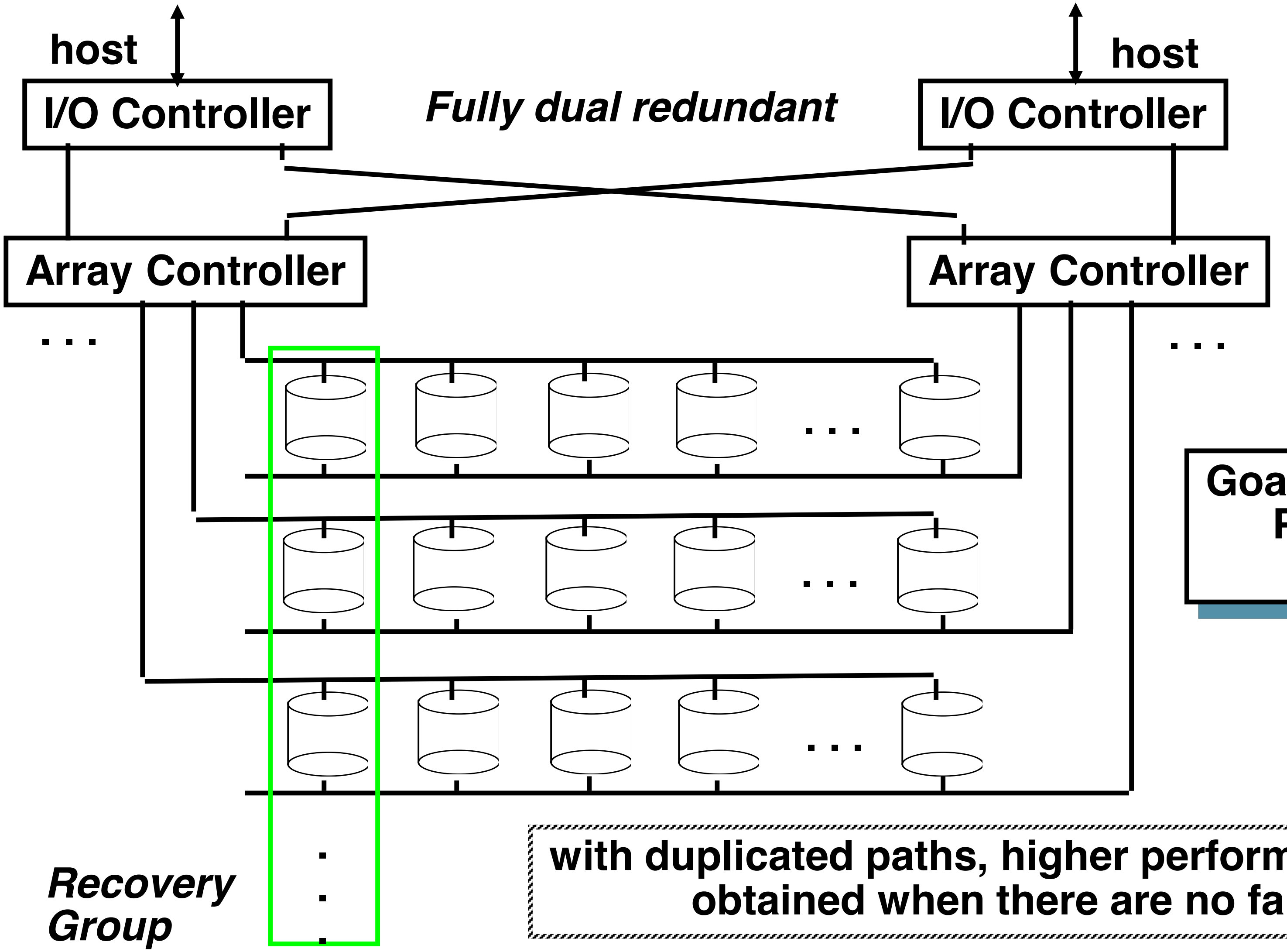


Data Recovery Group: **unit of data redundancy**

Redundant Support Components: **fans, power supplies, controller, cables**

End to End Data Integrity: **internal parity protected data paths**

System-Level Availability



Goal: No Single Points of Failure

with duplicated paths, higher performance can be obtained when there are no failures

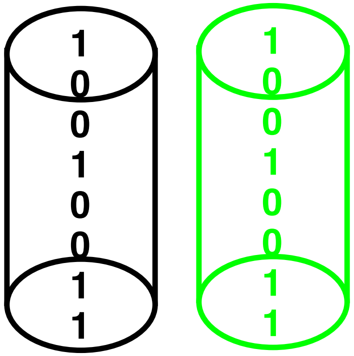
Summary: RAID Techniques: Goal was performance, popularity due to reliability of storage

- ***Disk Mirroring, Shadowing (RAID 1)***

  Each disk is fully duplicated onto its "shadow"

  Logical write = two physical writes

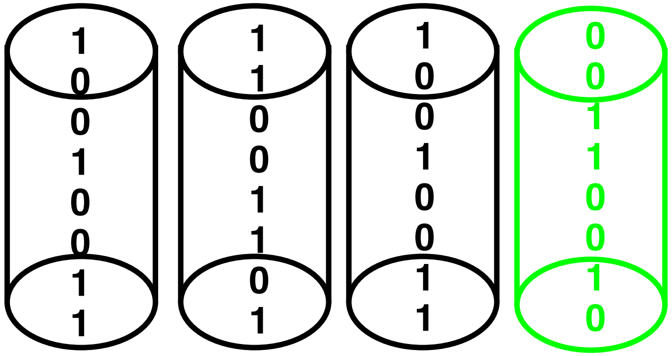
  100% capacity overhead



- ***Parity Data Bandwidth Array (RAID 3)***

  Parity computed horizontally

  Logically a single high data bw disk

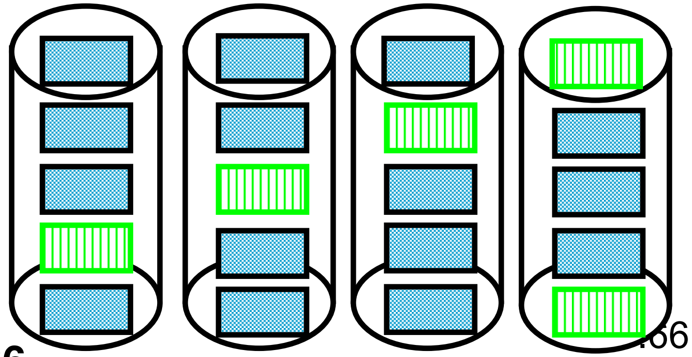


- ***High I/O Rate Parity Array (RAID 5)***

  Interleaved parity blocks

  Independent reads and writes

  Logical write = 2 reads + 2 writes



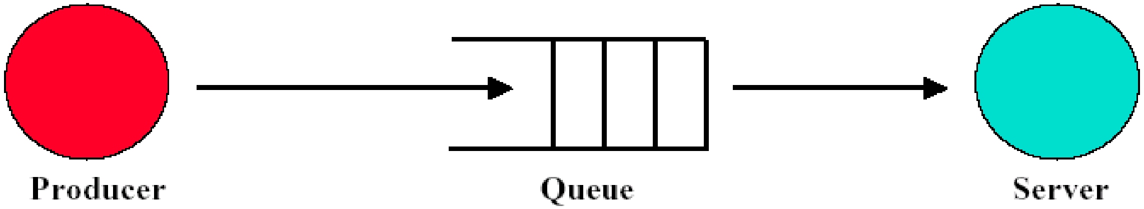
7.7 I/O Performance Measures

I/O System performance depends on many aspects of the system ('limited by weakest link in the chain')

- The CPU
- The memory system:
  - Internal and external caches
  - Main Memory
- The underlying interconnection (buses)
- The I/O controller
- The I/O device
- The speed of the I/O software (Operating System)
- The efficiency of the software's use of the I/O devices

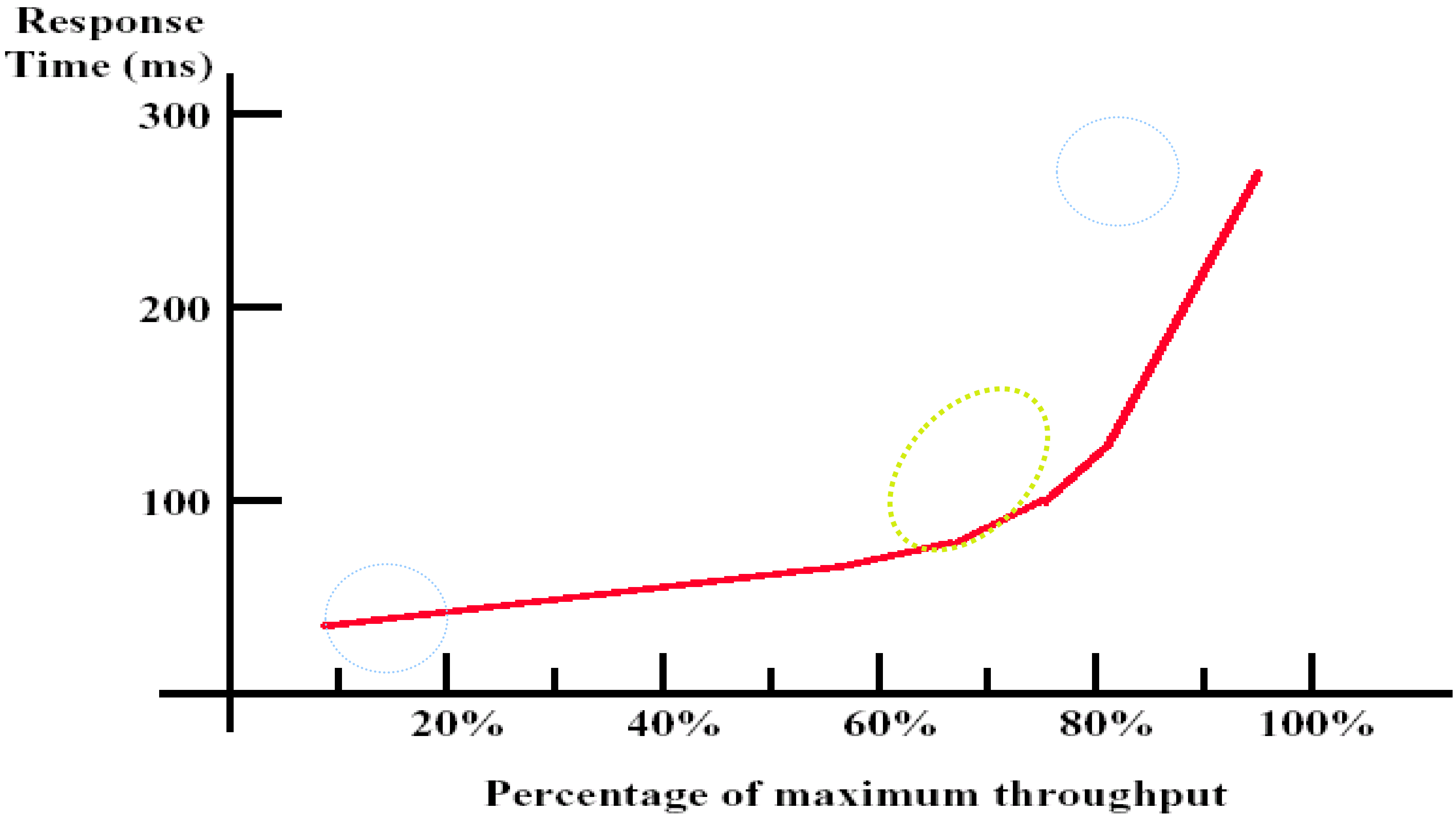
- Two common performance metrics:
  - Throughput: I/O bandwidth
  - Response time: Latency

Simple Producer-Server Model



- Throughput
  - The number of tasks completed by the server in unit time
  - In order to get the highest possible throughput:
    - The server should never be idle
    - The queue should **never be empty**
- Response time
  - Begins when a task is placed in the queue
  - Ends when it is completed by the server
  - In order to minimize the response time:
    - The queue should be **empty**
    - The server will be idle

Throughput versus Respond Time

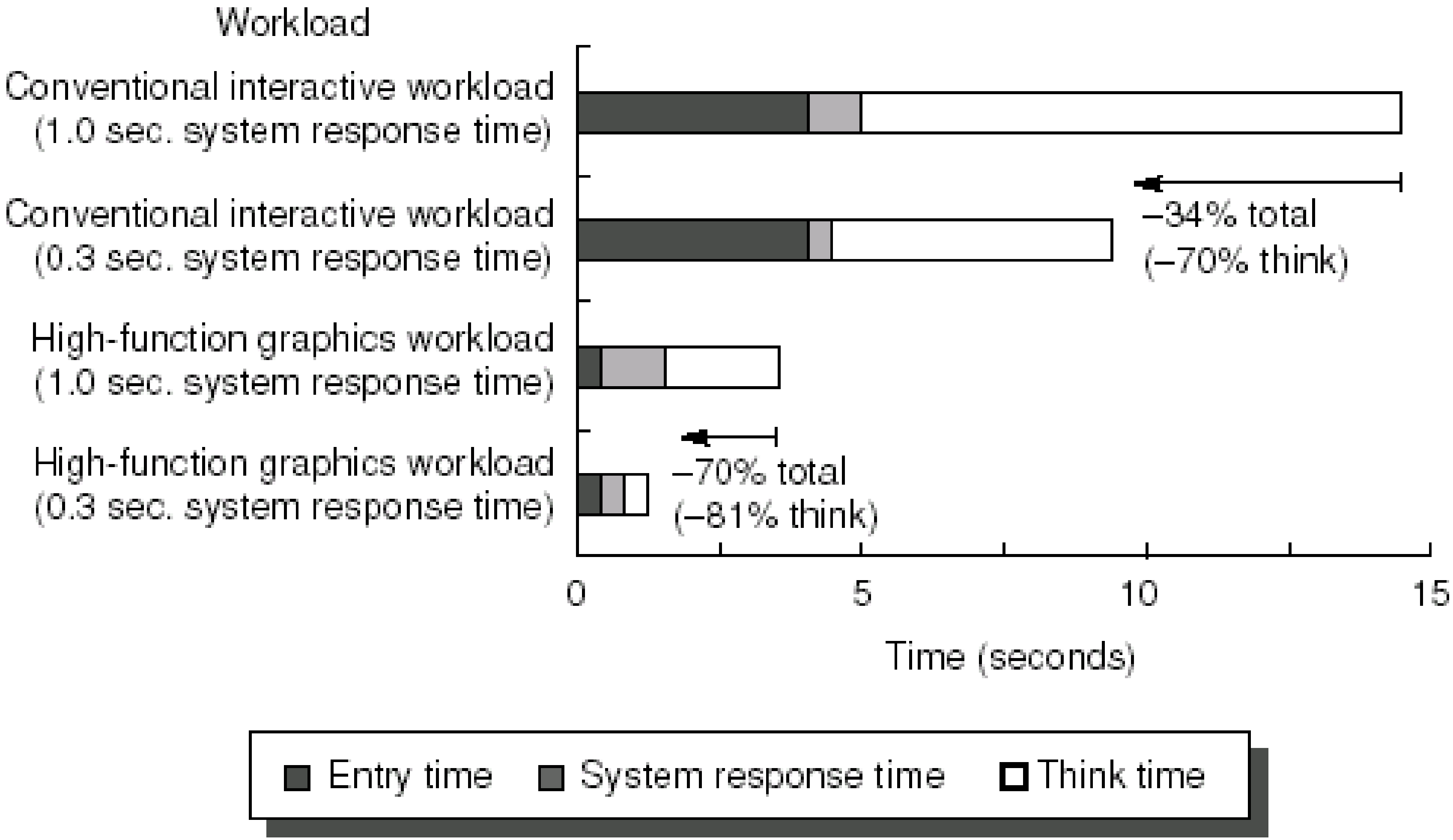


Response time relate to Interaction

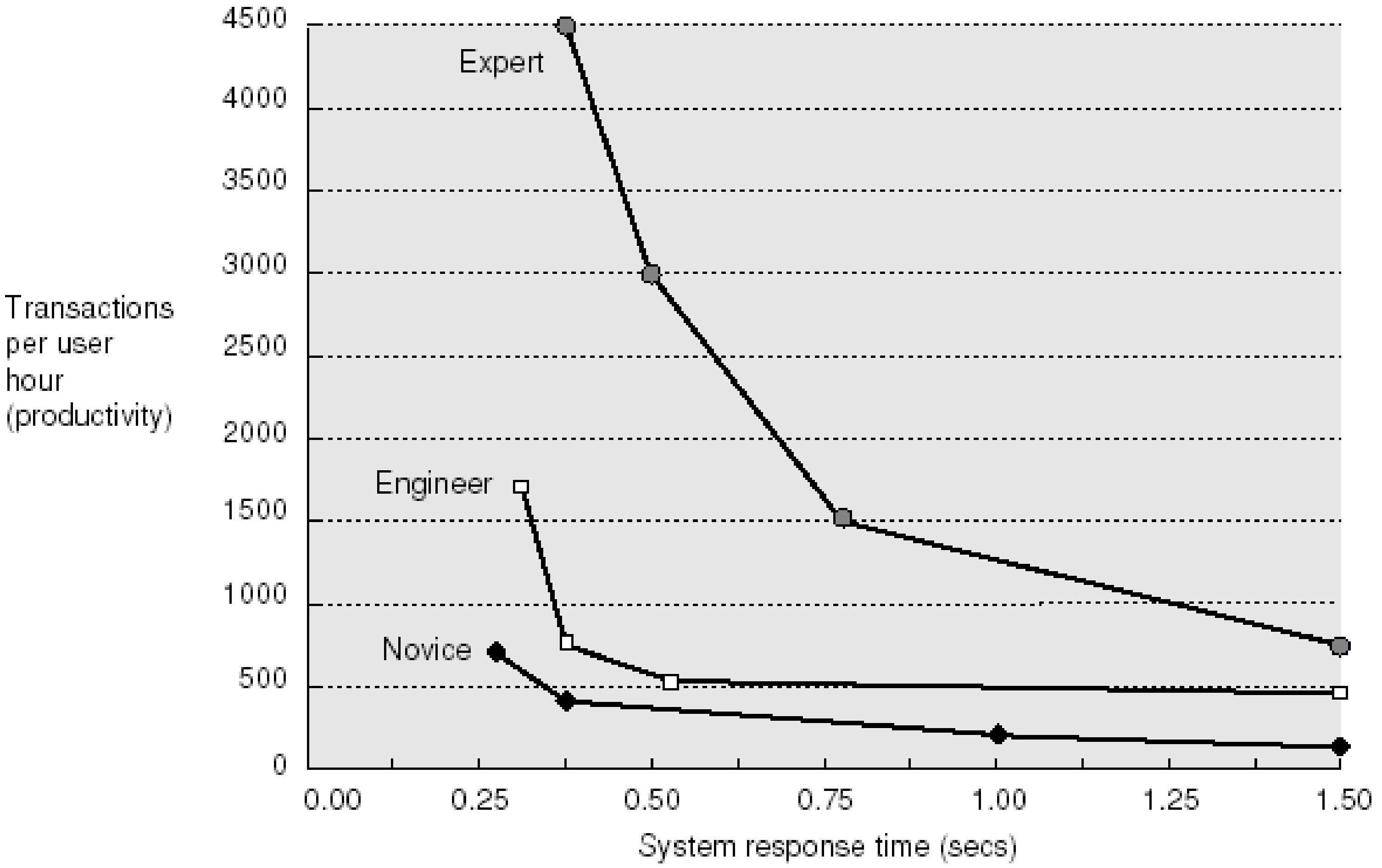
- An interaction, or *transaction*, with a computer is divided into three parts:

1. Entry time---The time for the user to enter the command.
   - The graphics system required 0.25 seconds on average to enter a command versus 4.0 seconds for the keyboard system.
2. System response time---The time between when the user enters the command and the complete response is displayed.
3. Think time---The time from the reception of the response until the user begins to enter the next command.

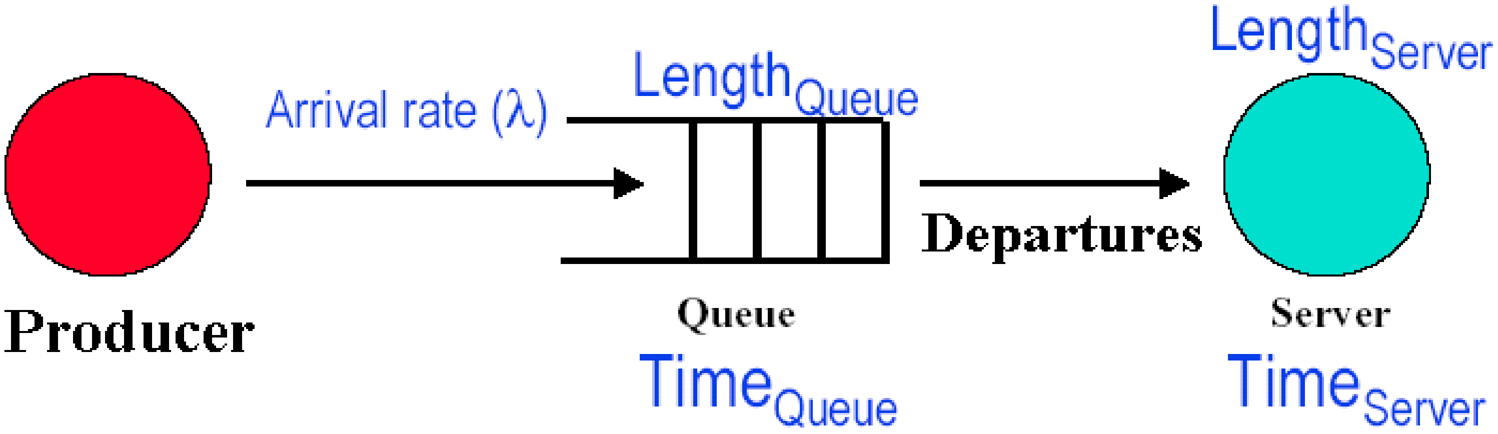
Response time relate to Interaction



Response time vs Manipulator



7.8 A Little Queuing Theory



Assumption: steady state characteristics, FIFO

Little's Law:

- Length$_{System}$ = Arrival rate x Time$_{System}$
- (Length$_{Queue}$ + Length$_{Server}$) = l x (Time$_{Queue}$ + Time$_{Server}$)

How busy a system is !

$$\text{Server Utilization} = \frac{\text{Arrival rate}}{\text{Service rate}}$$

- Server Utilization (U)

  $U = l \text{ x } \text{Time}_{\text{Server}}$

  (Arrival Rate < Service Rate)
- Example:
  - Single disk (server) gets 10 requests per second
  - Avg time to service a request: 50 ms
- What is the utilization?
  - Arrival rate: 10 IOPS
  - Service rate: 1/50ms = 20 IOPS

$$\text{Server Utilization} = \frac{10}{20} = 50\%$$

  - How many requests at the disk on average?
- Length$_{\text{Server}}$ = Arrival rate x Time$_{\text{Server}}$ = 10 IOPS x 0.05s

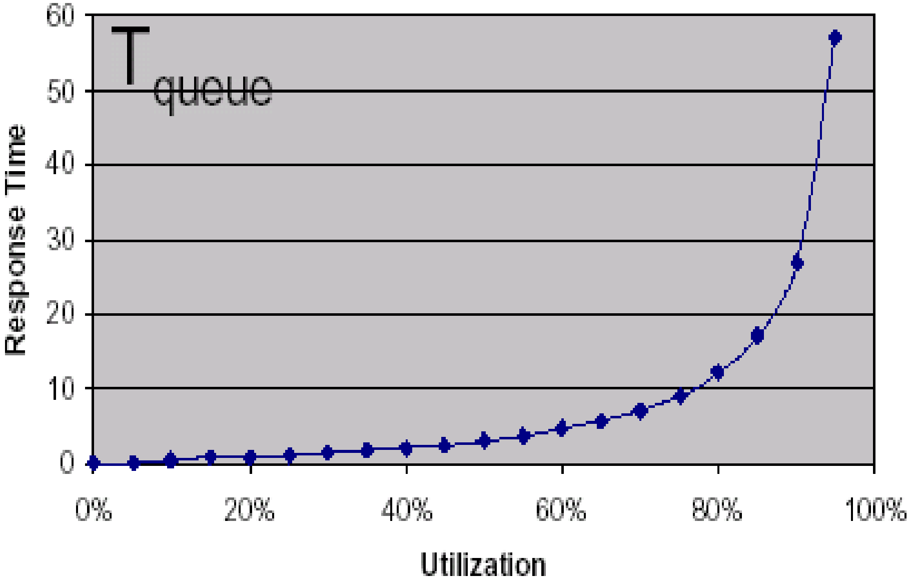
  = 0.5 in disk at any one time

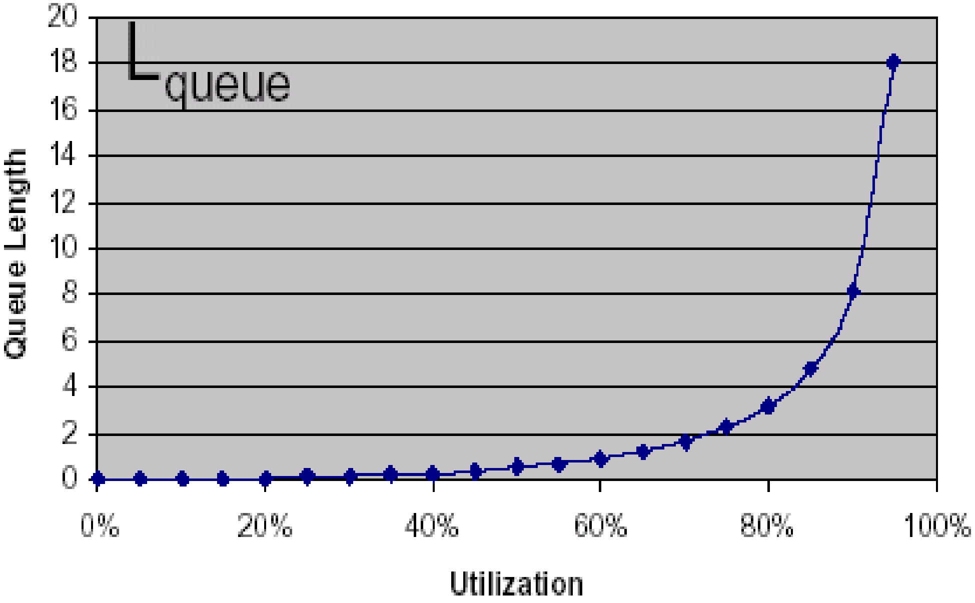
For I/O systems

(making some assumptions + doing a little algebra)

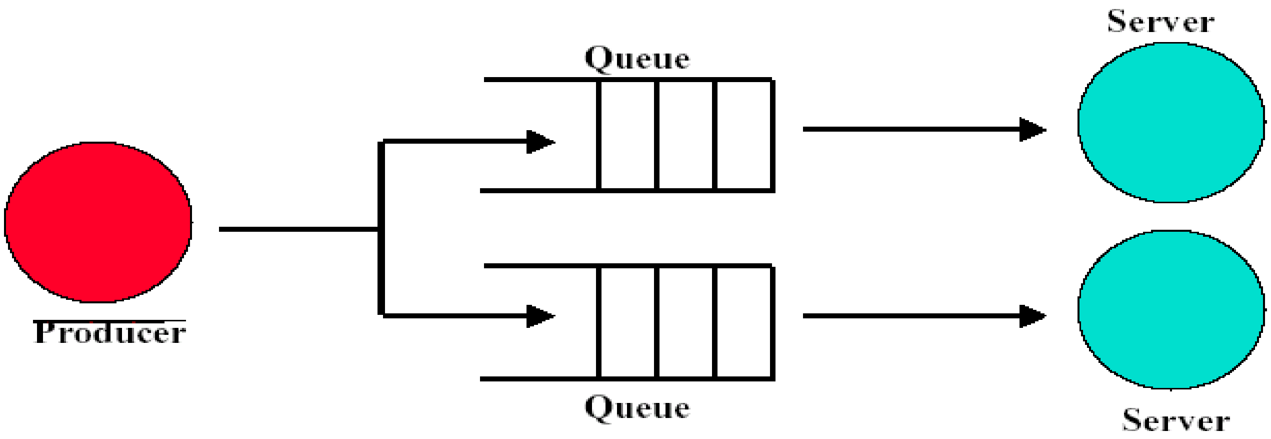
$$T_{queue} = T_{server} \times U / (1-U)$$

$$L_{queue} = U^2 / (1-U)$$



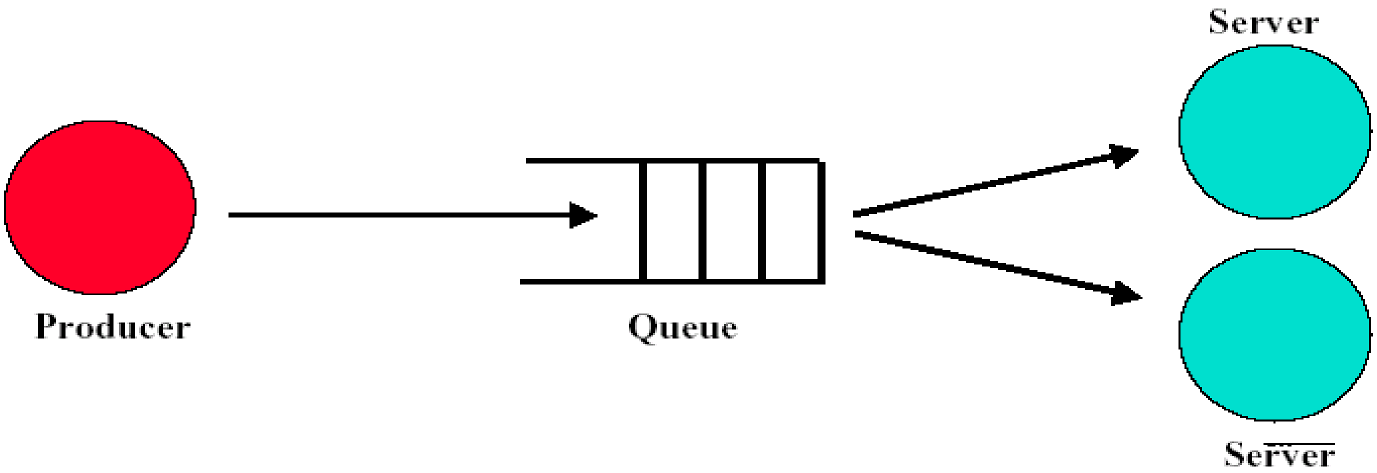


Throughput Enhancement-1



- In general throughput can be improved by:
  - Throwing more hardware at the problem
- Parallel Queues
  - Increases system throughput
- Problem: One queue is full while other is empty

Throughput Enhancement-2



- Little's Law still holds
- Server utilization could be greater than 1.0
- Response time is much harder to reduce
  - Minimum: $1/T_{server}$