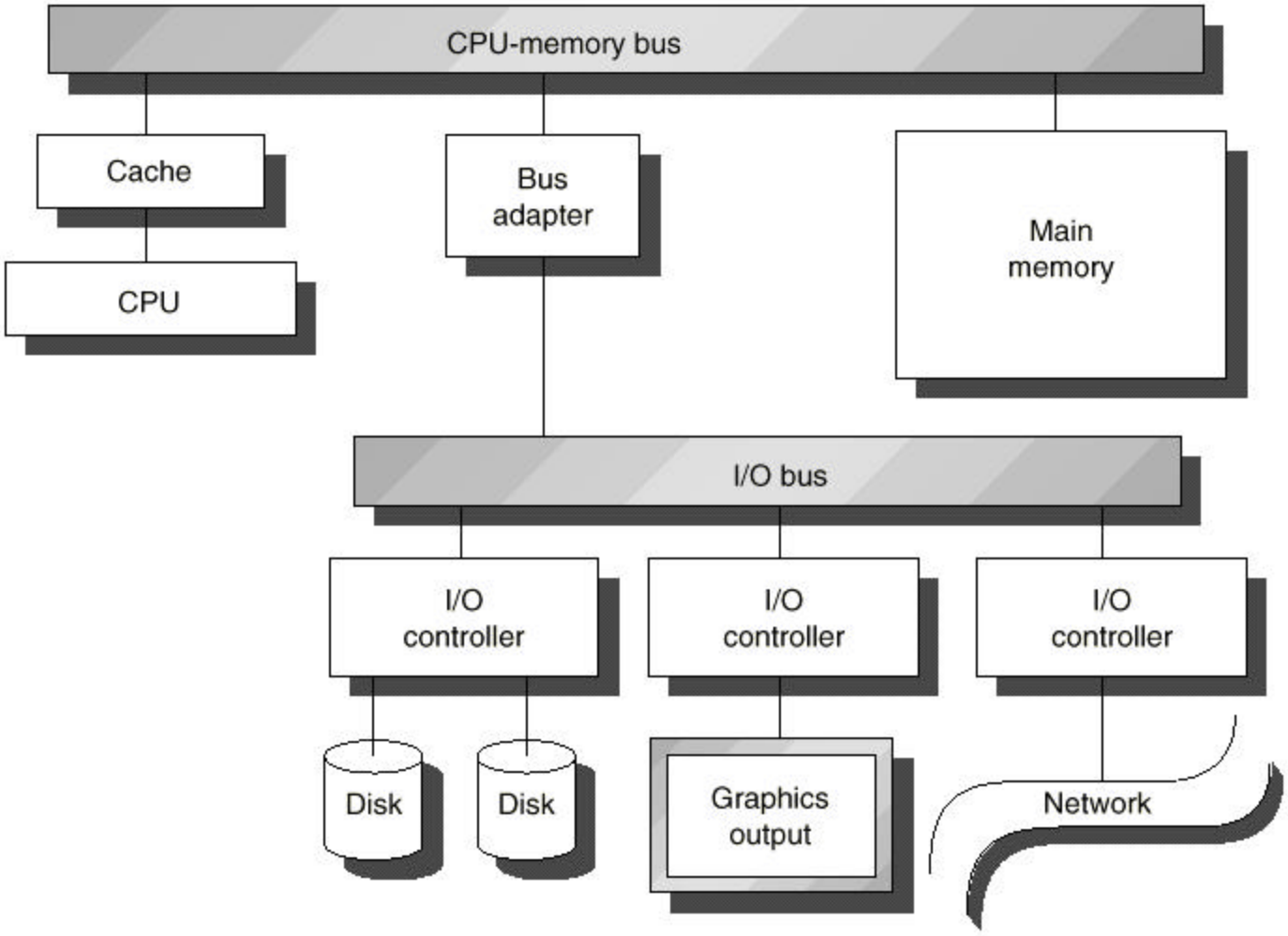
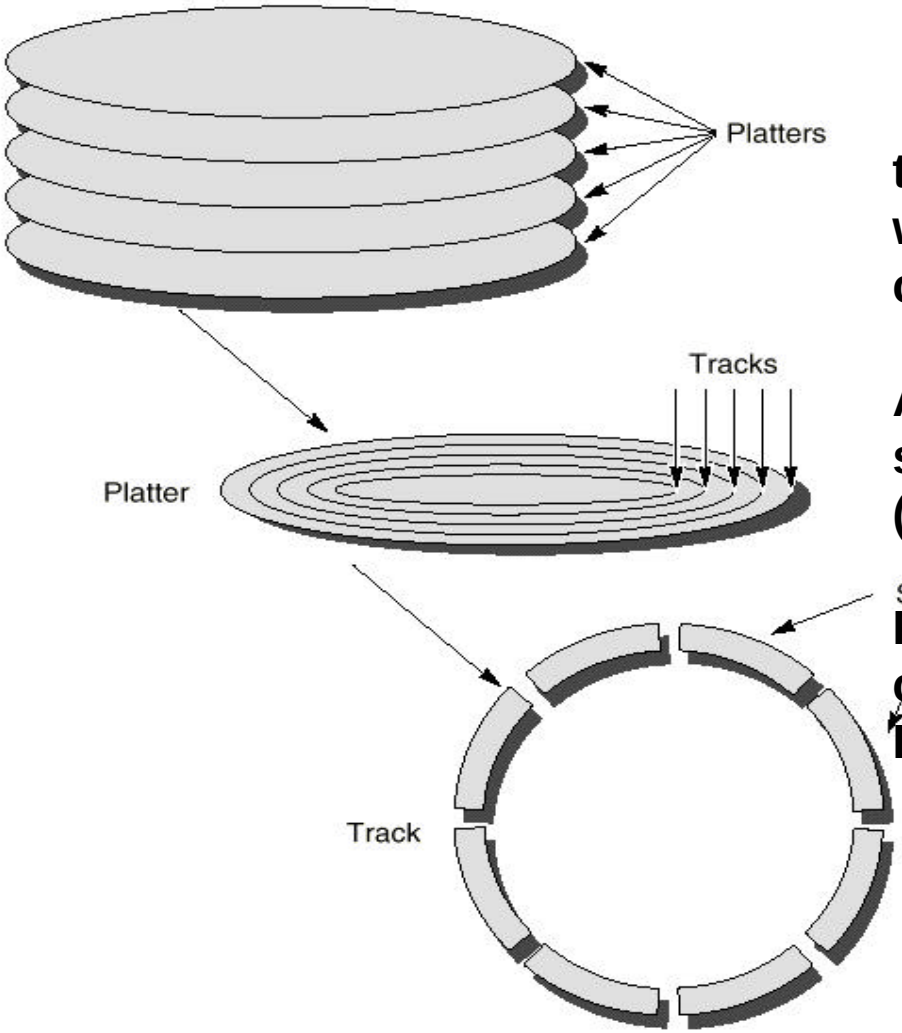
Typical Interface of I/O Devices



Magnetic Disks



tracks on all platters with the same distance to center form a cylinder.

A drive configuration can be specified by C/H/S numbers (Cyclinder/Head/Sector)

For example, a 4195MB disk contains (531/255/63) numbers Each sector has 512 bytes.

Characteristics of a 1993 magnetic disk

Characteristics	Segate ST31401N Elite-2 SCSI Drive	other choices
Disk Diameter	5.25”	3.5”, notebook 2.5”
Formated data capacilty	2.8	
Cylinders	2627	
Tracks per cylinder	21	
Sectors per track	~99	
Bytes per sector	512	
Rotation speed (RPM)	5400	7200, 10000, 4200
Average seek in ms (Random cylinder to cylinder)	11.0	8-15
minimum seek in ms	1.7	
maximum seek in ms	22.5	
Data transfer rate in MB/sec	~4.6	UWS40, IDE33.6

Disk Performance

Average Disk Access Time= average rotation time+
average seek time+
data transfer time+
controller overhead time.

For a disk with 7200RPM, the average rotation time= 0.5 rotations/7200RPM
= 0.00415 sec

What is the average time to read/write a 512 byte sector for a typical disk:

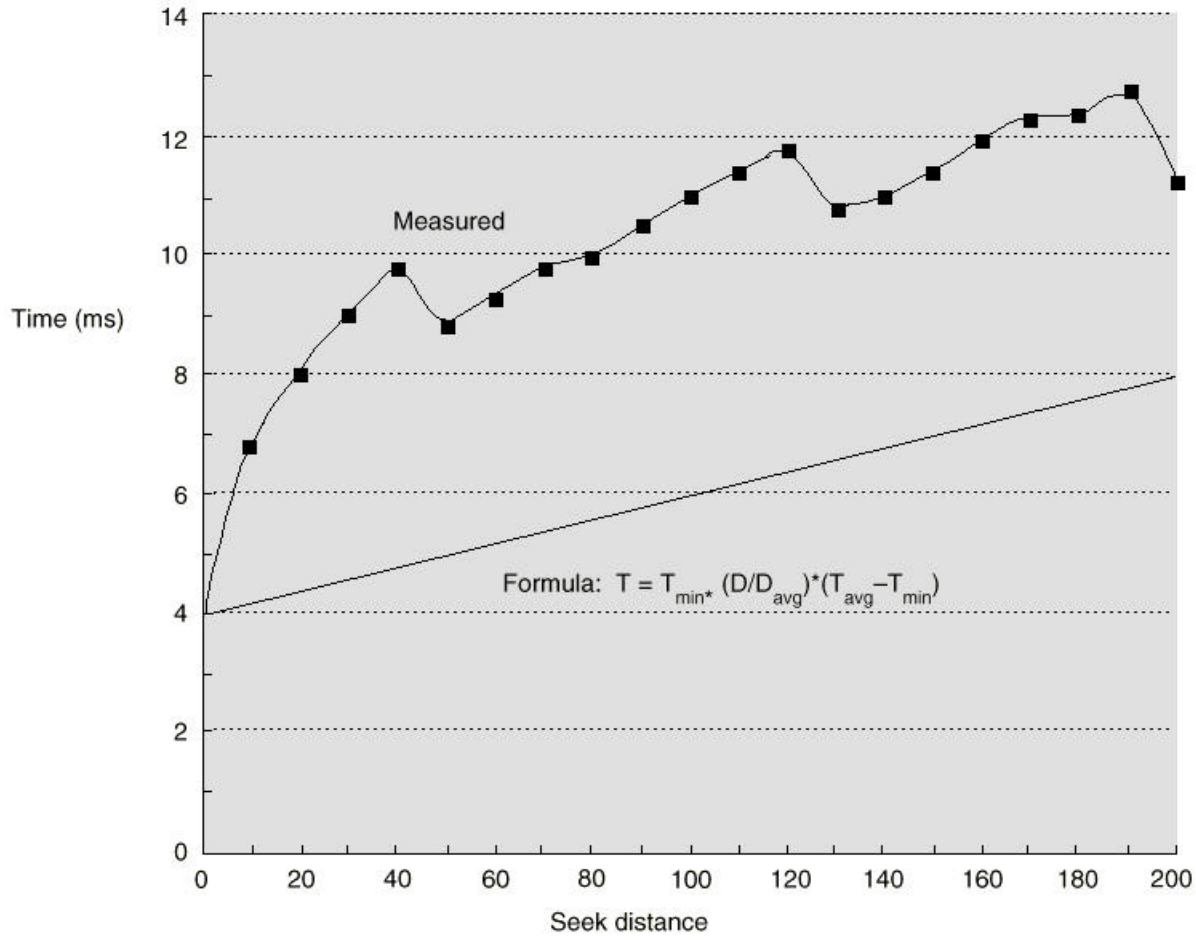
average seeek time=9ms, data transfer rate is 4MB/s, RPM=7200, controller overhead=1 ms. Assume no queueing delay.

Ans: 9ms+(0.5/(7200/60))*1000 ms+0.5KB/4.0MB/s+1ms=9+4.15+0.125+1=14.3 ms.

Fallacy:

Average Seek Time = Time to seek 1/3 of cylinders

Real Measurement:



CD-ROM, Erasable Optical Disks, Tape

CD-ROM: 640MBytes

Transfer time: basic speed (150KBps), 2x(300KBps), quad (4x, 540KBps), 6x

now we have 32x (typical 24x)

Seek time : avg 75-85ms,

but access time also important in deciding the response time.

CD-R: writable CD-ROM system 4x read, 2x write common.

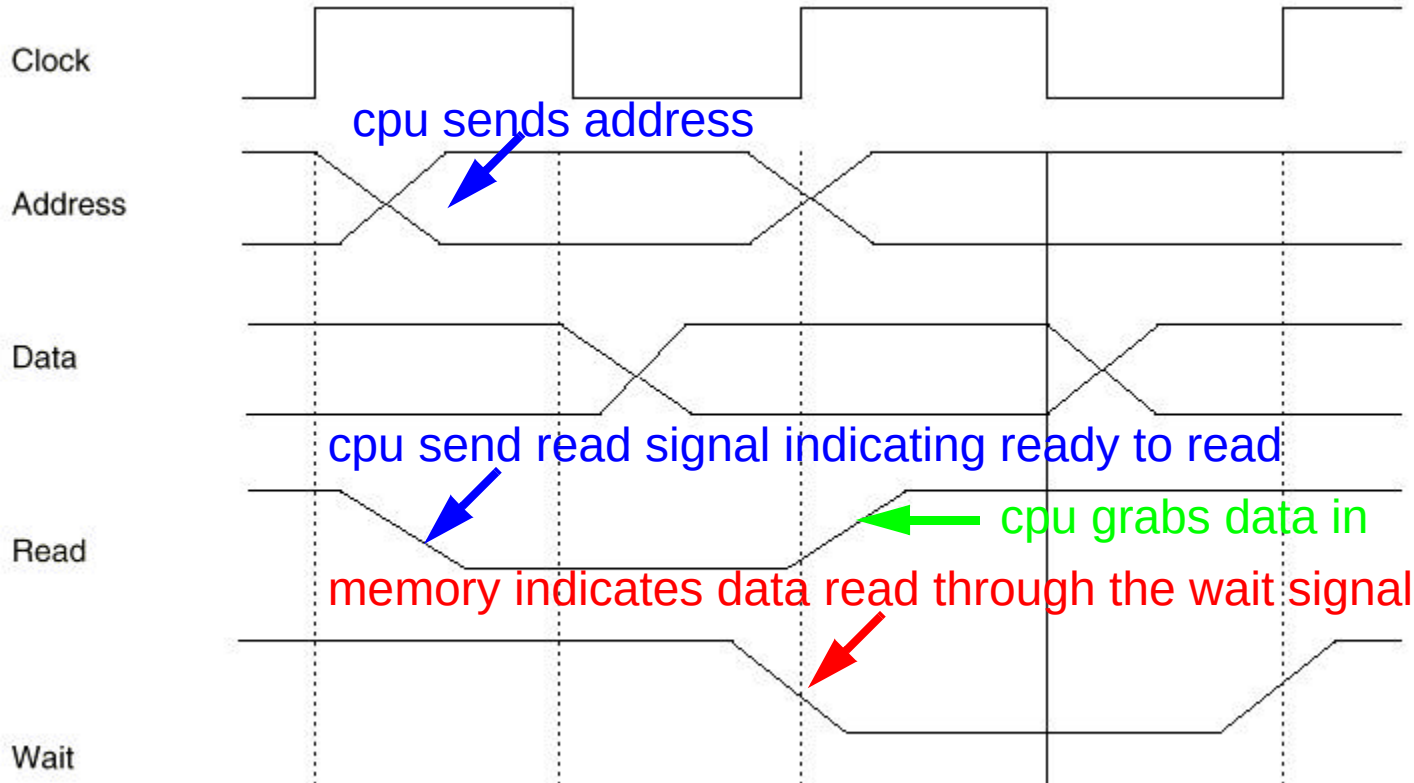
DVD: 2x (can read 24x? 20MB/s), now we have 4.8x (4.8MB/s, 32x)

average seem time: 135ms. 17GB data.

Eraseable optical disk: 120, 680MBytes

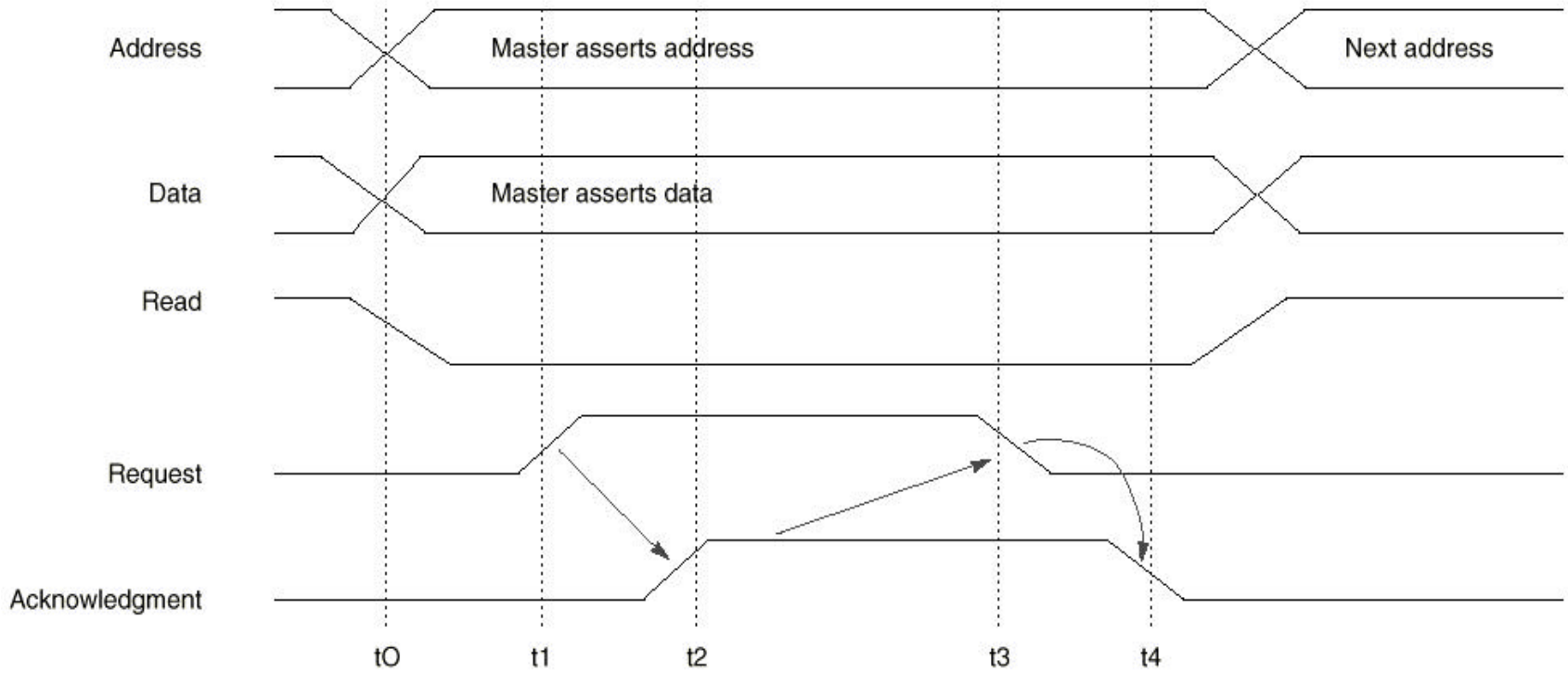
Synchronous Bus Read Transaction

All the operations synchronized with the clock signal line on the bus.



Separate address and data for fast speed; wider bus e.g., 64bits;
multiple word transfer size; multiple bus master (require arbitratoin);

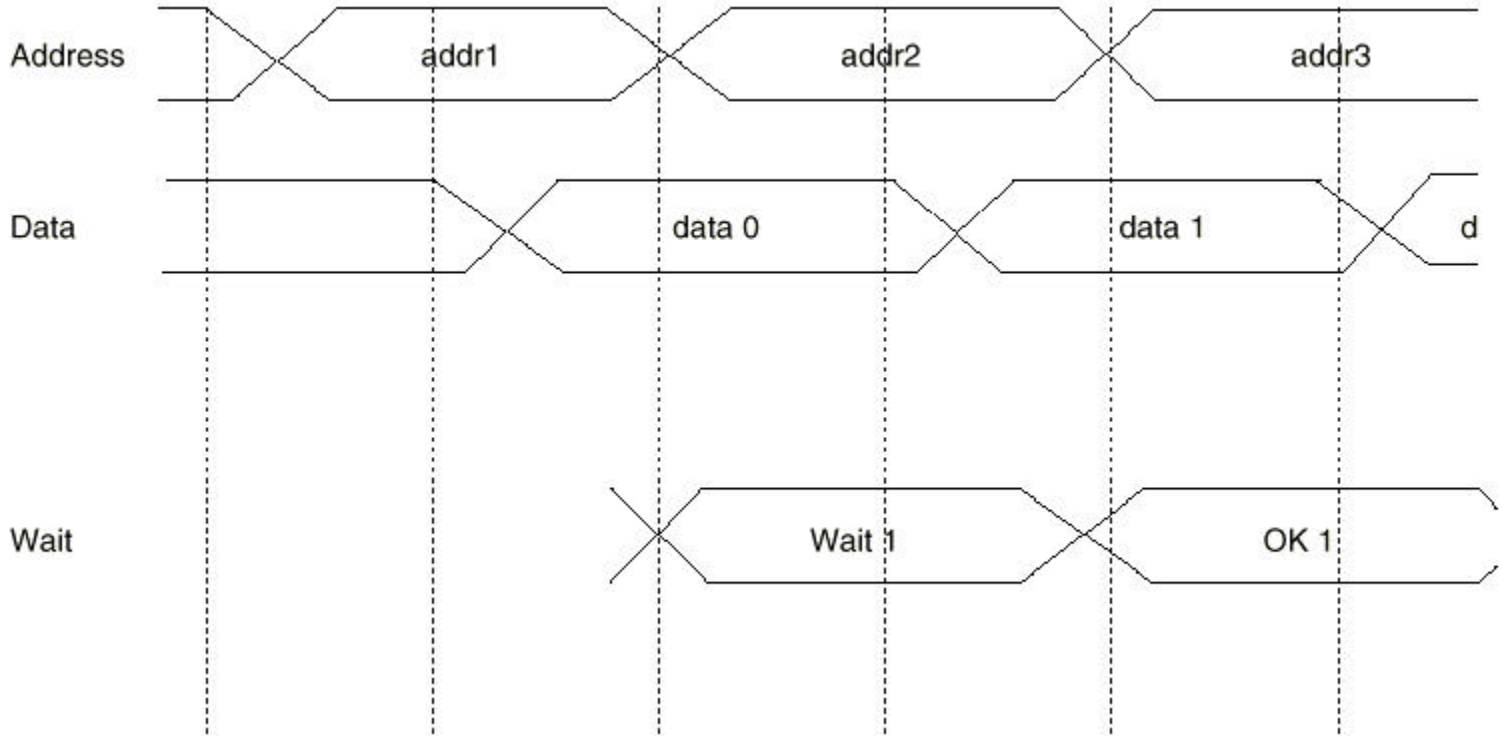
Asynchronous Bus Operation



multiplexing address and data lines; narrow bus e.g., 8bits; single word transfer, single master (no arbitration); allow more flexibility in terms of match wide speed differences.

Split-Transaction Bus

For multiple masters, with address in the signals (address/data) we can interleave the request/response without holding the bus until a transaction is done.



Example of I/O Buses

	S Bus	MicroChannel	PCI	IPI	SCSI
Data Width	32 bits	32 bits	32 to 64 bits	16 bits	8-16 bits
Clock rate	16 to 25MHz	Asynch.	33MHz	Asynch.	10MHz. or Asynch.
# of bus masters	Multiple	M.	M.	Single	Multiple
Bandwidth, 32-bit reads	33MB/s	20MB/s	33MB/s	25MB/s	20MB/s or 6MB/s
Bandwidth, peak	89MB/s	79MB/s	111MB/s	25MB/s	20MB/s or 6MB/s
standard	none	-	-	ANSI X3.129	ANSI X3.131

Example of CPU-Memory Buses

	HP Summit	SGI Challenge	Sun XDBus
Data width(Primary)	128 bits	256 bits	144 bits
Clock rate	60MHz	48MHz	66 MHz
# of bus masters	Multiple	Multiple	Multiple
Bandwidth, Peak	960 MB/s	1200MB/s	1056MB/s
Standard	none	none	none

Interface Storage Devices to CPU
Q1. Where should the I/O Bus be connected?

1. Connecting to memory bus: (more usual case)
2. Connecting to cache:

In low cost system, I/O bus is the memory bus.

Q2. How does CPU address an I/O device for read/write?

1. Memory-mapped I/O (most common used).
   Portion of the address space are assigned to I/O devices.
   Some portion of the I/O address space are used for device control.
2. Dedicated I/O opcode.
   CPU send a signal indicates the address is for I/O devices, e.g., x86, 370

Q3. When to start the next I/O operations?

Polling: CPU periodically check status bits of I/O device.

Interrupt-driven: Device can interrupt CPU processing. CPU after sending out an I/O request can work on other process. This is the key to multitasking OS. But there is an OS overhead. Not good for real-time systems.

Direct Memory Access (DMA)

Interrupt driven I/O relieves the CPU for busy waiting for every I/O event.

But it still spends a lot of CPU cycles in transferring data.

To transfer a block of 2048 word, requires 2048 load/store CPU instructions execution.

Solution: Use DMA hardware to allow transfer between memory and I/O devices without the intervention of CPU. DMA must act as a master on the bus.

DMA Operation:

- CPU set up DMA registers, including memory address and # of bytes to be transferred. DMA is often part of the controller for an I/O device.
- DMA issues the I/O requests to the I/O device.
- When data arrives from the I/O device.
- DMA grabs the memory bus and start transfer data to memory.
- When data transfer is complete, DMA interrupts CPU.

I/O processors (or I/O controllers, channel controllers)

DMA devices with enhanced intelligence:

OS can down load programs to I/O processor.

Operations:

- OS sets up a queue of I/O control blocks which contains the data location (source and destination) and data size.
- I/O processor takes items from the queue, perform everything requested.
- I/O processor sends a single interrupt when task complete.

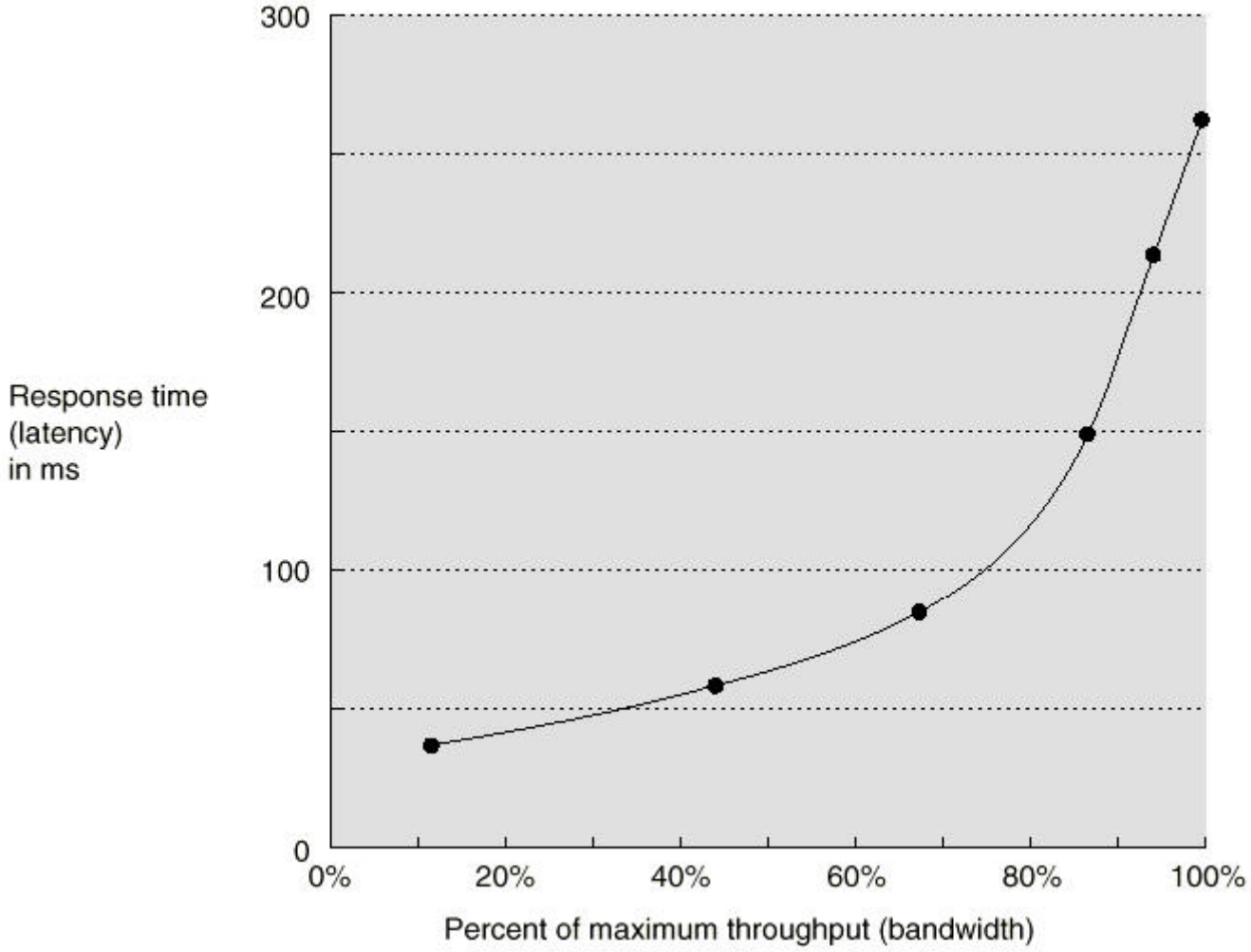
I/O performance Measures

Response Time (latency): begin when a task is placed in the buffer and end when it is completed by the server.

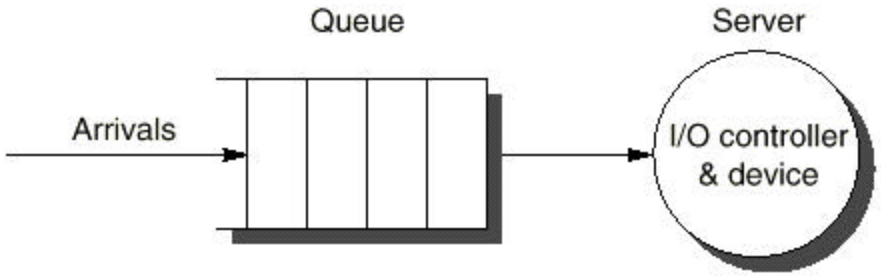
Throughput (I/O bandwidth): the number of tasks completed by the server in unit time.

The following is the traditional producer-server model.

Throughput vs. Response Time of a typical I/O system



Apply Simple Queueing Theory to Evaluate I/O Systems



Assumption: system in equilibrium (input rate = output rate)

Lengthqueue - Average number of tasks in the queue

Lengthserver - Average number of tasks in the server.

Lengthsystem - Average number of tasks in the system (both in queue and server)

ArrivalRate - Average number of arriving tasks/seconds.

ServiceRate - Averate number of tasks completed by the server.

Timequeue - Average time a task spent in the queue (waiting to be served)

Timeserver - Average time a task spent in the server (being served)

Timesystem - Average time a task spent in the system=Timequeue+Timeserver

Little's law: Lengthsystem = ArrivalRate x Timesystem

ServiceUtilization - a measure representation how busy a system, 1 >= value>= 0

ServiceUtilization = ArrivalRate/ServiceRate

M/M/1 Queue

In this type of Queue, both the interarriaval time and the service time are exponentially distributed and there is only one server.

With M/M/1 queue, $Time_{queue} = Time_{server} \times \frac{ServeUtilization}{1 - ServerUtilization}$

Reliability vs. Availability

Reliability - Referred to “Is anything broken?”

R: a measure for evaluating reliability,
it represents the fraction of time the component is working

R = 1 - MTTR/MTBF
where MTTR is mean time to repair and MTBF is mean time between failures
e.g., SyQuest EZ135 Drive with MTBF: 200,000 hours.

Availability - Referred to “Is the system still available to the user?”

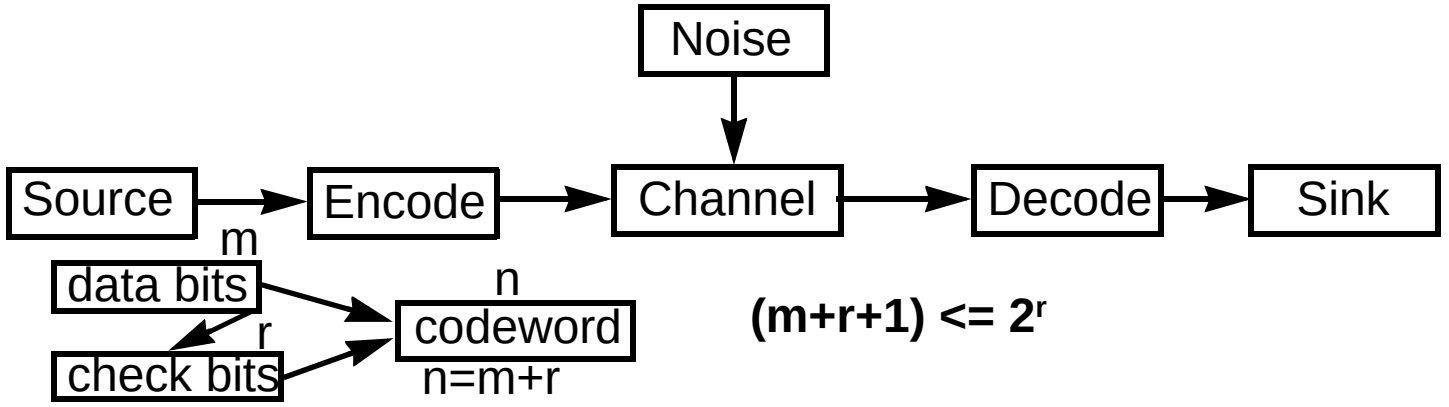
Adding hardware can improve availability but not reliability. (e.g., use Error Correcting Code on Memory)

Reliability can only be improved by

- better operating conditions
- more reliable components
- building with fewer component

(Hamming's Single) Error Correcting Code (ECC)

	1	2	3	4	5	6	7	8	9	10	11	Positions								
	C_o	C_1	D_1	C_2	D_2	D_3	D_4	C_3	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	
H			**1**		0	0	**1**		0	0	0	original data								$C_3C_2C_1C_0$
	1	1										D_1 is 1 in position 3⇒contribute								0 0 1 1
	1	1		1								D_4 is 1in position 7⇒contribute								0 1 1 1
	0	0	1	1	0	0	1	0	0	0	0	Encoded data using even parity								
							X					bit 7 Error								
	0	0	1	1	0	0	0	0	0	0	0	Retrieved data								
	1	1										only D_1 is 1 in position 3⇒contribute								0 0 1 1
	1	1		0				0				Regenerate Check bits								
	X	X		X								Errors in the check bits								
	1+2+			4=7			X					position of error is bit 7.								
	0	0	1	1	0	0	1	0	0	0	0	Corrected data								



ECC on Memory

For 8 bit data: it requires 4 check bits. The encoded data has 12 bits.

One possible design: put 12 bits in 12 different memory chips.

Store operation:

- receive 8 bit data from CPU,
- send though ECC circuit, generate 12 bits encoded data,
- save in 12 chips.

Load operation:

- retrieve 12 bits from 12 chips,
- go through ECC decoding and correct any single bit error,
- send 8 bit data to CPU.

ECC on Disks (RAID Level 2)

Store operation:

- receive 8 bit data from CPU,
- send though ECC circuit, generate 12 bits encoded data,
- save 12 bits in 12 different disks.

Load operation:

- retrieve 12 bits from 12 different disks,
- go through ECC decoding and correct any single bit error,
- send 8 bit data to CPU.

Redundant Array of Inexpensive Disks (RAID)

It improves both the **availability** and the **performance** of storage system.

By spreading data over multiple disks, called **Striping**, RAID increases the throughput.

RAID improves throughput but not necessary the latency. Why?

Reliability Drops with more devices; $R = R_{disk}^{n} \cdot R_{coordinationCircuit}$

RAID Level	Failures survived	Data disks	Check Disks
0 Nonredudant	**0**	**8**	**0**
1 Mirrored	**1**	**8**	**8**
2 memory-style ECC	**1**	**8**	**4**
3 Bit-interleaved parity	**1**	**8**	**1**
4 Block-interleaved parity	**1**	**8**	**1**
5 Block-interleaved distributed parity	**1**	**8**	**1**
6 P+Q redundancy	**2**	**8**	**2**

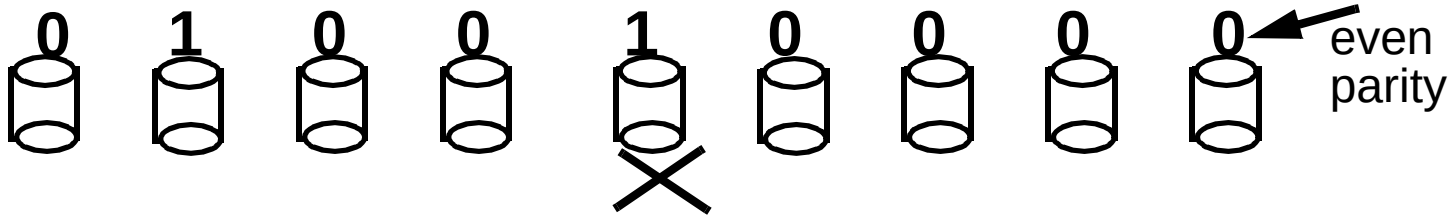
Different RAID Levels

Mirroring or Shadowing (RAID 1)

- Use one redundant disk. Write to both disk. A traditional fault-tolerant computing techniques.
- When the normal disk fails, the system goes to the “mirror” disk for data.
- The most expensive solution, twice as many disks is needed.

Bit-Interleaved Parity (RAID 3)

- One disk contains the parity bits.



Through parity check, we know the bit in the failed disk must be 1.

- The failed disk can be identified through other tests or signals.
- The failed disk content can be regenerated (painstakingly slow).

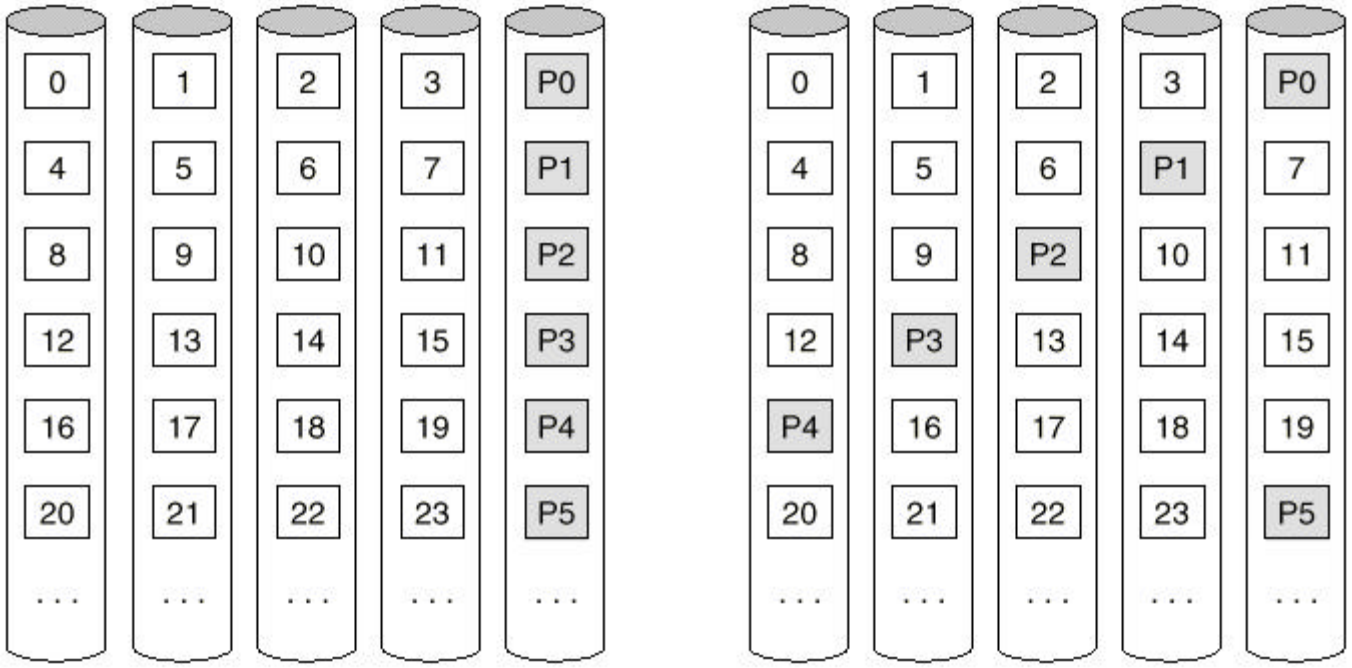
Block-interleaved parity (RAID 4)

- Each disk with a block of data, the parity disk has the block of parity bits generated by computing from the same nth bits of the blocks in other 8 disks.

Block-interleaved Distributed Parity (RAID 5)

- If minimum access is one sector, we can utilize the error detection information in each sector. Read does not have to access all disks. This is called “small read.”
- “Small write” - only involve 4 disk accesses instead of 17? disk accesses.
  1. read old data block from a disk,
  2. compare with new data block for parity bits that need to be changed,
  3. read old parity block from the parity disk,
  4. change the parity bits, write new parity block back to the parity disk,
  5. write new data block to the disk.
- To allow parallel write accesses to data, the order of block sets in the disk can be rearranged so that the parity blocks are distributed evenly to all disks.

RAID 4 vs. RAID 5



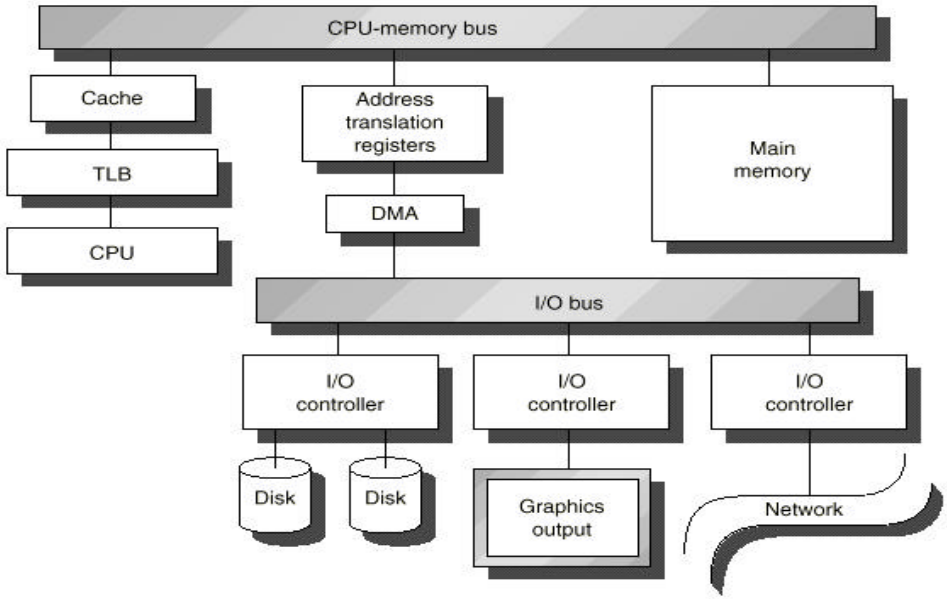
Virtual DMA

Problems with DMA using physical addresses in a virtual memory system:

- A I/O buffer with multiple page size are not usually mapped to sequential pages in physical memory.
- When DMA is ongoing between memory and a frame buffer, the OS removes/relocates some of the pages.

Virtual DMA - use virtual addresses and address translation registers.

- OS needs to lock “pages” involved with DMA until the transfer is complete.



Six steps to design a I/O system

1. List the different types of I/O devices to be connected to the machine, or a list of standard buses to be supported.
2. List the physical requirements for each I/O devices. (volume, power, connectors, bus slots, expansion cabinets, etc.)
3. List the cost of each I/O device, including portion of cost of controller needed.
4. Record the CPU resource demands of each I/O device, including
   - Clock cycles for instructions used to initiate, support, and complete an I/O.
   - CPU stalls due to waiting for I/O to finish using the memory, bus, or cache.
   - CPU clock cycles to recover from I/O activity, such as cache flush.
5. List the memory and I/O bus resource demands of each I/O device.
6. The final step is establishing performance of the different ways to organize these I/O devices. (Use simulation or Queuing Theory.) Then select the best organization.

I/O System Design Example (page 530)

Given CPU 500MIPS, $30k; 16-byte-wide memory, 100 ns cycle time;
200 MB/s I/O Bus with rooms for 20 SCSI-2 buses/controller (also called strings); SCSI-2 buses can transfer 20 MB/s, 15 disks/bus;
A SCSI-2 bus controller cost $1500 and adds 1 ms overhead to Disk I/O;
OS uses 10K CPU instructions for a disk I/O.

You have choices of large disk — 8 GB; small disk — 2 GB; ($0.25/MB).
Disks rotate at 7200 RPM with 8 ms average seek time & 6 MB/s transfer time.

The I/O system requires 200 GB storage capacity and average I/O size is 16 KB.

Evaluate the cost per I/O per second (IOPS) of using small and large drives.
Assume that every disk I/O requires an average seek and rotational delay; all devices are used in 100% capacity; and work load is evenly divided among all disks.

Answer: First look at the limitation on each components in the system.

max. IOPS for CPU = 500 MIPS/ 10 k inst per I/O = 50k IOPS

max. IOPS for mem. system = (1/100 ns) * 16 / 16 KB per I/O = 10k IOPS

max. IOPS for I/O Bus = 200 MB/s / 16 KB per I/O = 12.5 k IOPS

SCSI-2 bus transfer time = 16 KB per I/O / 20 MB/s = 0.8 ms per I/O

max. IOPS for SCSI-2 controller = 1/ (1+0.8) ms per I/O = 556 IOPS

average disk I/O time per I/O= 8 ms (seek time) + 0.5 rotation/7200RPM (rotation delay) + 16 KB / 6 MB/s (transfer time) = 14.9 ms.

max. IOPS for Disk = 1 /14.9ms = 67 IOPS

200 GB $\Rightarrow$ 25 8-GB disks or 100 2-GB disks.

max. IOPS for 25 8-GB disk =25*67 =1675 IOPS

max. IOPS for 100 2-GB disk =100*67 =6700 IOPS

minimum no. of SCSI-2 strings for 25 8-GB disks = $\left\lceil \frac{25}{15} \right\rceil = 2$

minimum no. of SCSI-2 strings for 100 2-GB disks = $\left\lceil \frac{100}{15} \right\rceil = 7$

max. IOPS for 2 SCSI-2 strings = 2 * 556 =1112 IOPS

max. IOPS for 7 SCSI-2 strings = 7 * 556 =3892 IOPS

max. IOPS for system use (Min(CPU, mem, I/OBus, Disk, String))

8GB disks; 2 strings= Min(50K, 10K, 12.5K, 1675, **1112**) =1112 IOPS

8GB disks; 4 strings= Min(50K, 10K, 12.5K, **1675**, 2224) =1675 IOPS

additional strings let disks perform at their full potential.

2GB disks; 7 strings= Min(50K, 10K, 12.5K, 6700, **3892**) =3892 IOPS

2GB disks; 13 strings= Min(50K, 10K, 12.5K, **6700**, 7228) =6700 IOPS

Cost of

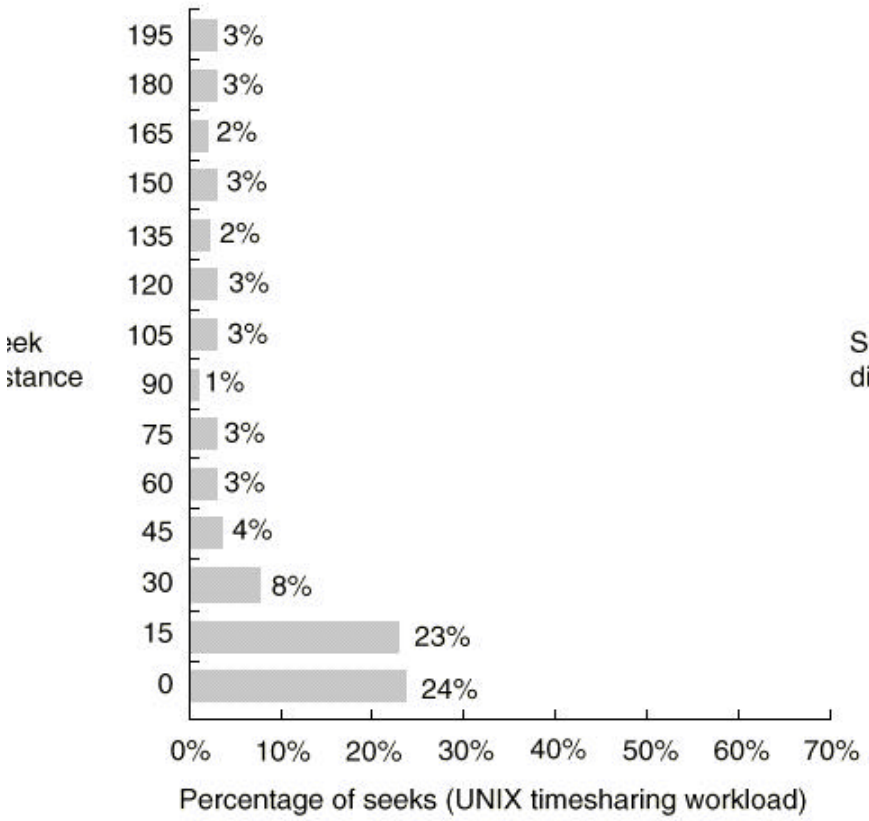
8GB disks, 2 strings = $30K+2*$1500+25*(8192*$0.25)	=$84,200
8GB disks, 4 strings = $30K+4*$1500+25*(8192*$0.25)	=$87,200
2GB disks, 7 strings = $30K+7*$1500+100*(2048*$0.25)	=$91,700
2GB disks, 13 strings = $30K+13*$1500+100*(2048*$0.25)	=$100,700

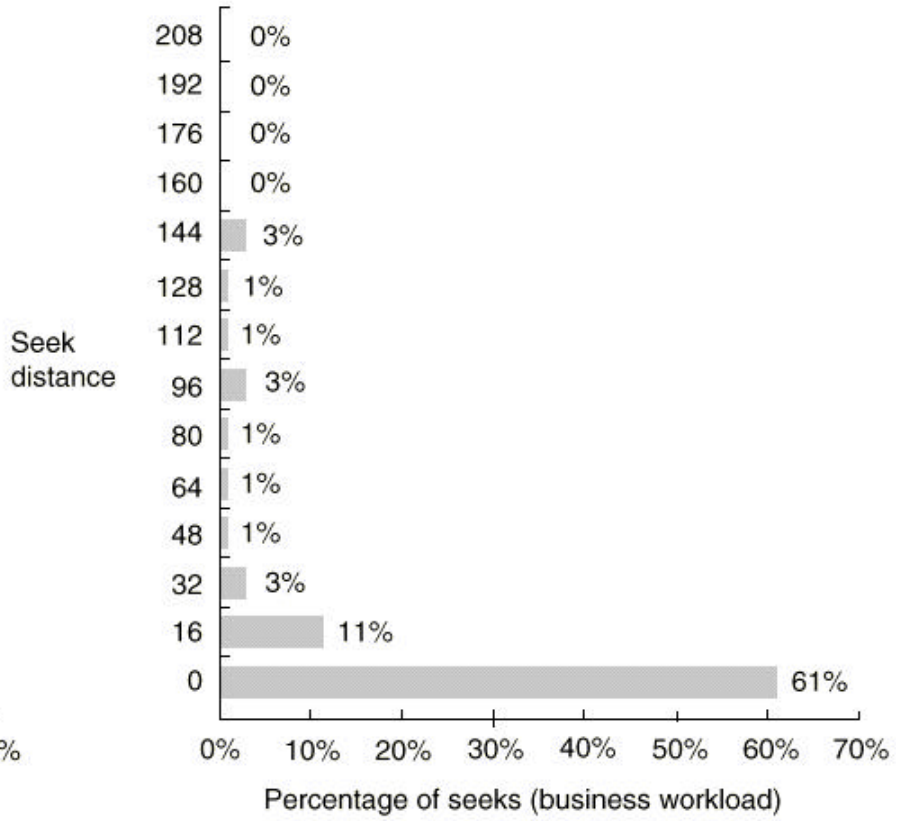
Cost per IOPS of

8GB disks, 2 strings = 84200/1112	=$76
8GB disks, 4 strings = 87200/1675	=$52
2GB disks, 7 strings = 91700/3892	=$24
2GB disks, 13 strings = 100700/6700	=$15

Small disks have 3.5 times better cost/performance than large disks.

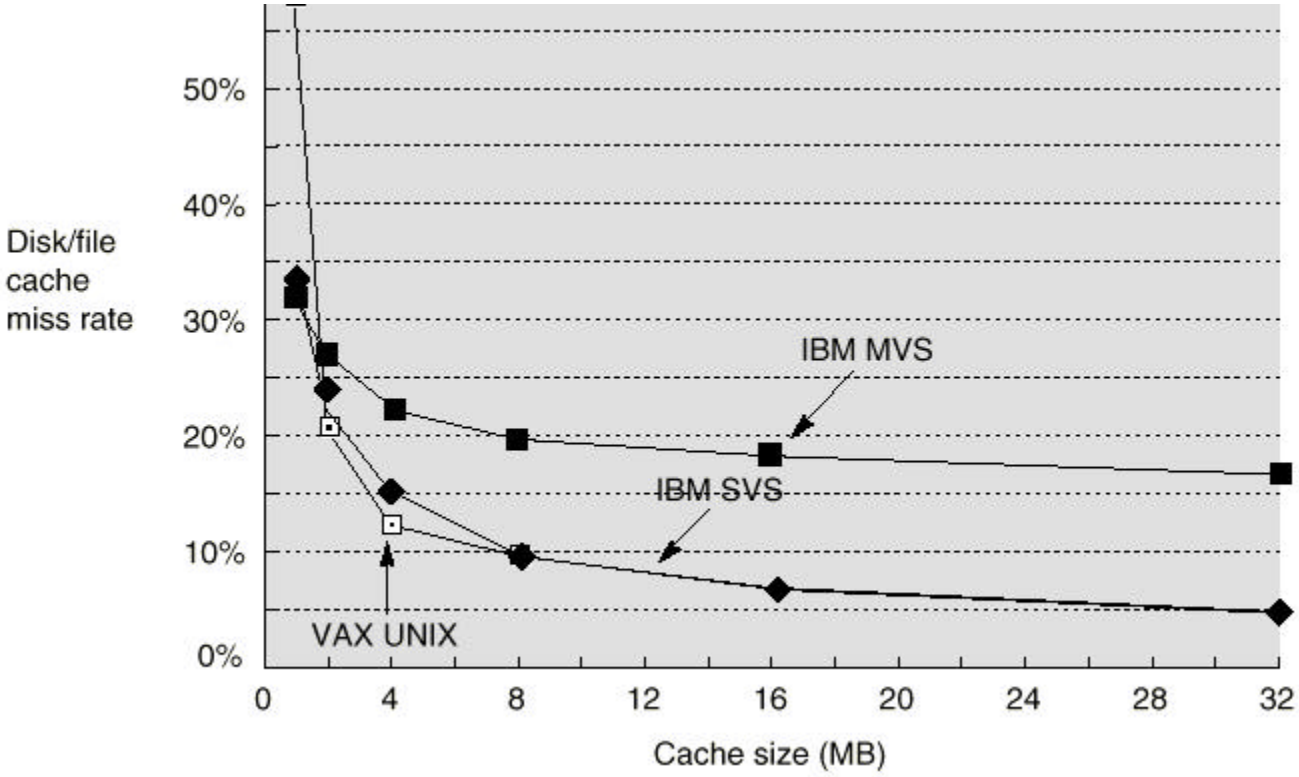
Disk Performance





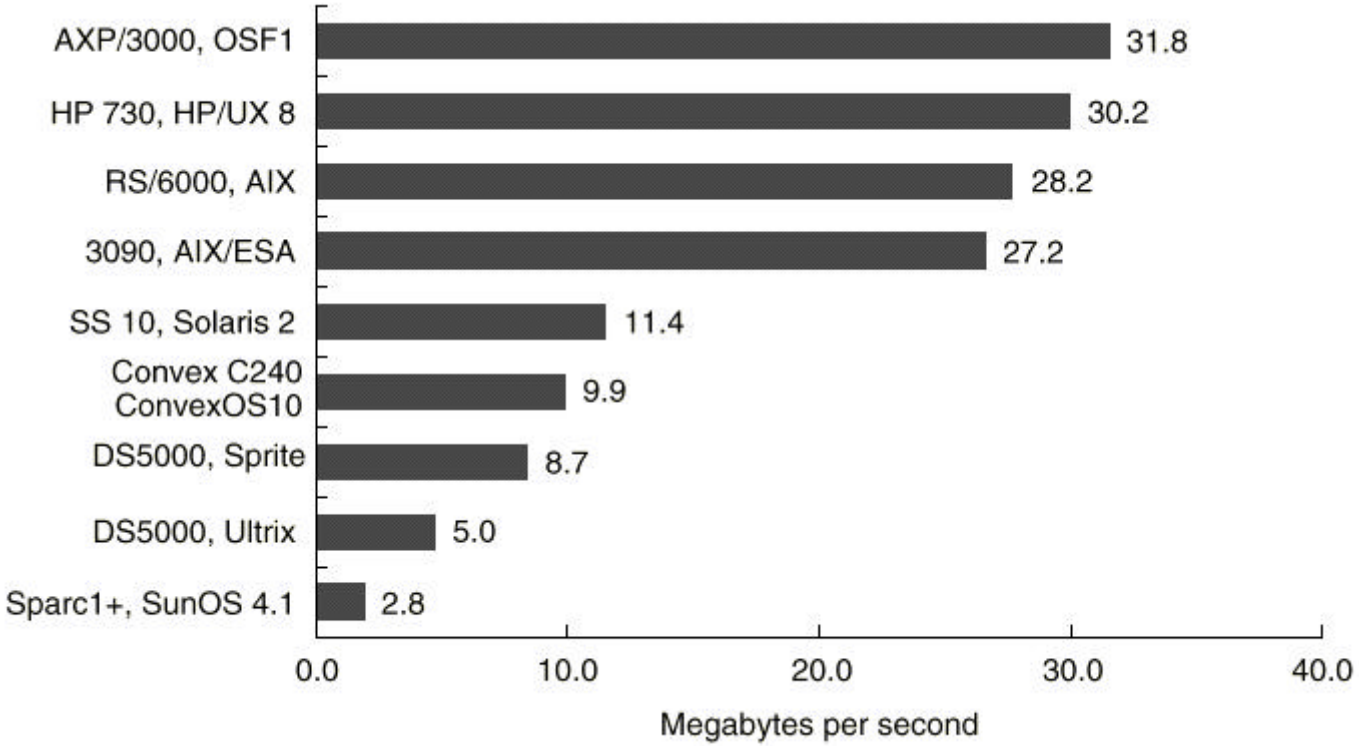
File/Disk Caches

- Use main memory as a cache (buffer) for file or disk accesses
- Reduce 50~70% of disk accesses.
- File cache—use logical block no.
- Disk cache—use the physical block address: the track and sector numbers.



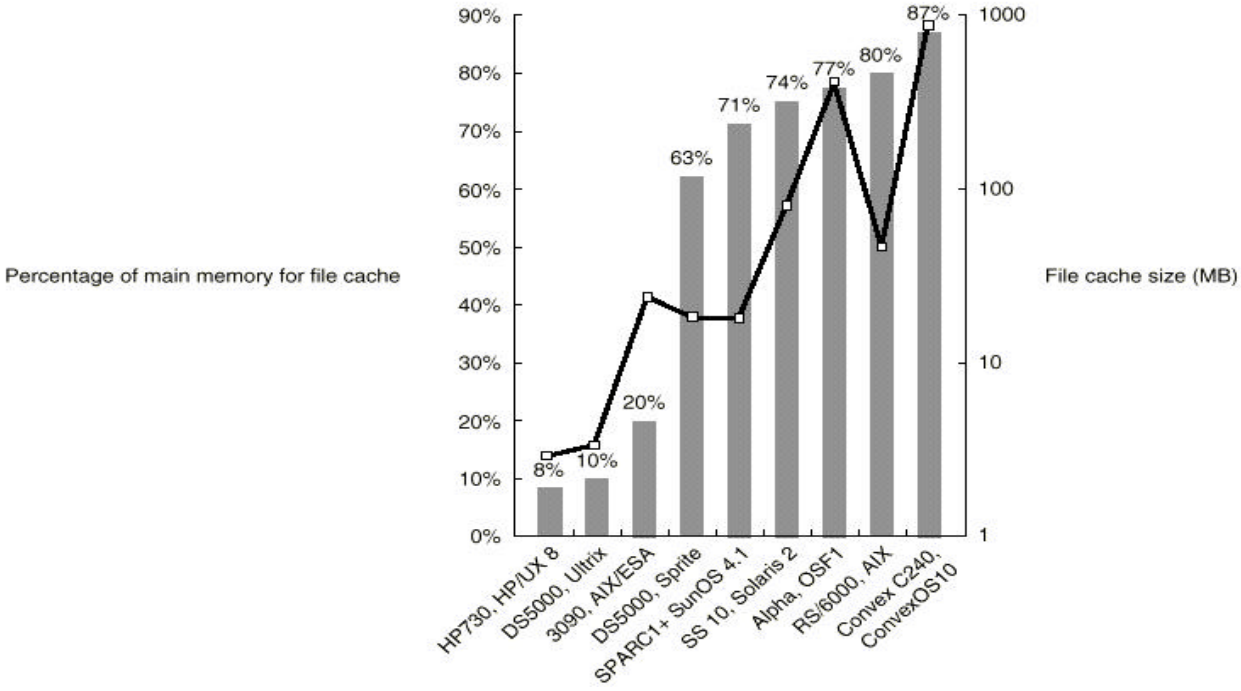
File Cache Performance (32KB read)

Workstations perform better than mainframe and mini-supercomputers.
OS impact: compare Sprite with Ultrix on DEC5000.
Almost four times difference between SS10 (AXP3000) and SS1 (DEC5000)

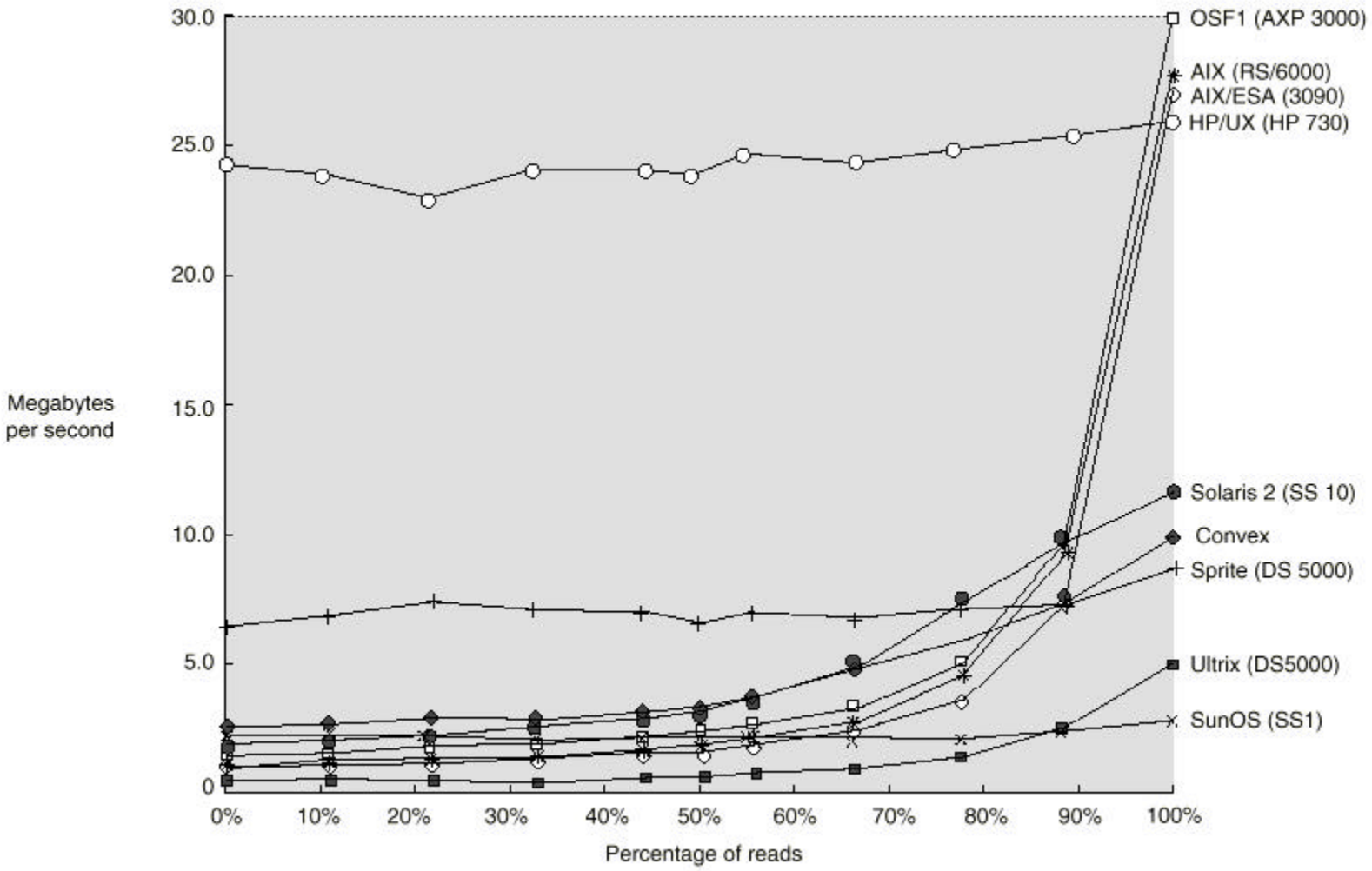


File Cache Size

Early UNIX used fixed percentage of main memory for file cache (10%).
New systems allow to adjust file cache size according to workload.
File servers may grow file cache to virtually full size of main memory.
Figure 6.39 show Sprite has six times the cache size used by Ultrix.
Sparc1 has the file cache as large as IBM 3090.



Write Policy of File Cache

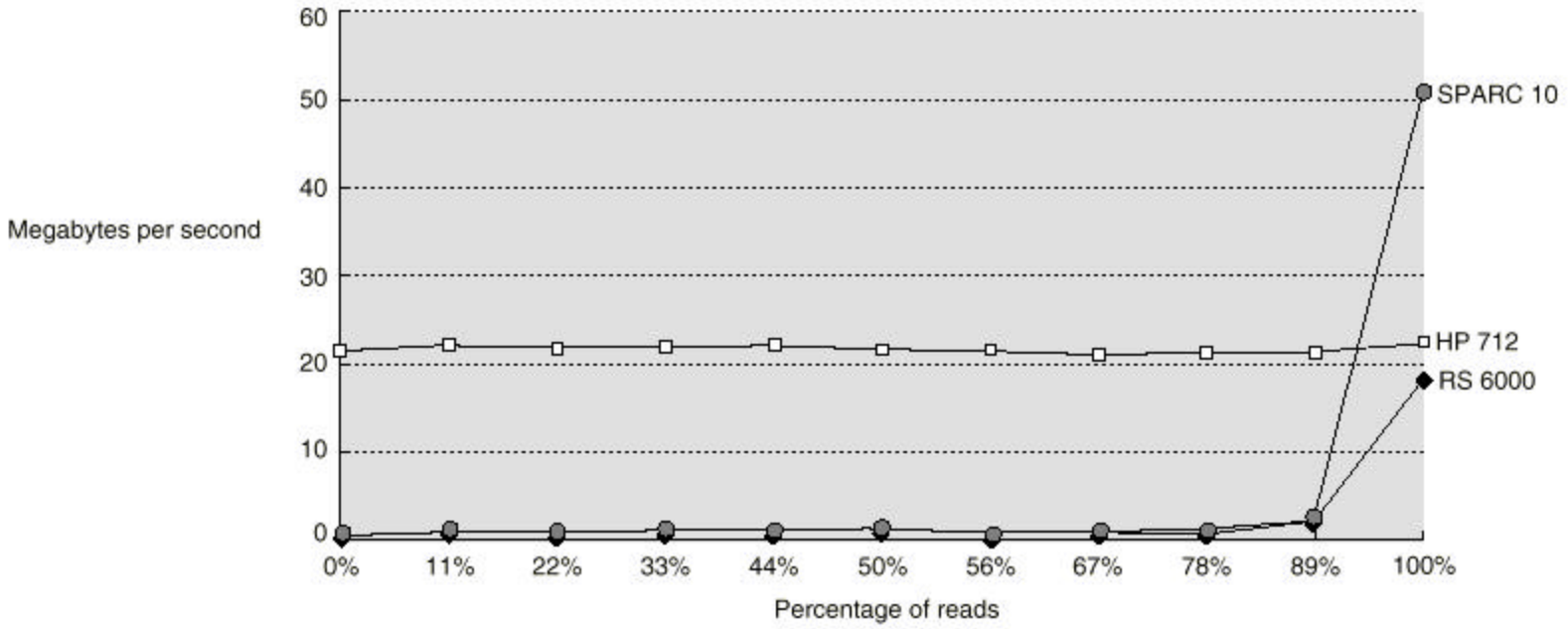


Write Policy for Client/Server Computing

For write back file cache, server needs to take appropriate action when the file is shared and some one is writing it.

HP/UX9 use DUX protocol to allows client-level caching of writes.

SUN NFS does write through.



Important Observation on File System Performance

- File cache policy determines the performance of most I/O event. It is the place to start improving I/O performance.
- File cache performance on workstations improve rapidly, better than some of the old mainframes and mini-supercomputers.
- RAID systems can deliver higher disk performance
  but cannot overcome weaknesses in file cache policy, why?
  Anyway to get around that.

  An potential semester project is to examine the above statement.

File System Evaluation Project

1. Come up with a program similar to that in Exercise 5.2 for finding the file cache size of the different computer systems, (DEC3000, SGI, DEC5000, DEC3100, SparcClassic), in our lab.
2. Evaluate the file system performance of the above computer systems.

Exercise #4:

Design an I/O system with the best cost performance given the limited budget.

Performance is measured as MB/s using average disk access sizes.

Ignore the cost of CPU and assume that main memory and processor have enough bandwidth to keep up with whatever I/O system we design.

Also the OS in the file server has no file cache because all the main memory is needed for the application.

Here are the design ground rules:

- **The disk storage capacity must be at least 10 GB.**
- **Output to the disk can not be buffered, i.e., the data must be written on the disk before the write is considered to be complete. That is with a write-through disk cache your write hit time is the disk access time (not the disk cache access time), only the read hit time is equal to the disk cache access time.**
- **To maintain a reasonable response time, no resource can be used at more than 80% of its rated max. bandwidth.**
- **The max. number of disks connected over a string to a disk controller is 10.**
- **The ratio of disk reads to disk write is 2:1.**
- **The average size of a disk access is 4KB.**
- **The seek distribution is like the UNIX timesharing workload in Figure 9.40 on page 559.**
- **Assume that every disk I/O requires an average rotational delay.**

Here are the cost and performance of the components:

Component	Cost	Max. Performance
Main Memory	$500/MB	250-ns for 8 bytes
I/O bus+rack	$500	Up to 15 disk controllers
Disk controller+string	$2000	Extra overhead of 1 ms per I/O event and string bandwidth is 10 MB/sec
3.5” disk	$1000	1 GB storage, 3600 RPM, seek distance >= 16 tracks take 10 ms, seeks of 1 to 15 tracks take 5 ms, no seek time for the same track. Max. transfer rate is 2 MB/s.
2.5” disk	$600	0.5 GB, 5400 RPM, seek distance >= 16 tracks take 8 ms, seeks of 1 to 15 tracks take 4 ms, no seek time for the same track. Max. transfer rate is1.75 MB/s.
1.8” disk	$300	0.25 GB, 7200 RPM, seek distance >= 16 tracks take 4 ms, seeks of 1 to 15 tracks take 2 ms, no seek time for the same track. Max. transfer rate is1.4 MB/s.

Your maximum budget is $22,000.

Hint: Compare different combinations of type of disks and the disk cache size. Select the design with the biggest MB per second per $1000 number.

Step 1. Find out the raw characteristics of a disk access per disk size.

Disk Diameter	Size (GB)	RPM	Seek Time 1-15	Seek Time 16-	Xfer rate MB/s	Cost	Avg Rotate (ms)	Avg Seek (ms)	Xfer Time (ms)	Disk Access time (ms)	I/O /sec /disk	MB /sec /disk
3.5	1	3600	5	10	2	$1,000	8.33	6.45	2.00	17.78	56.24	0.225
2.5	0.5	5400	4	8	1.75	$600	5.56	5.16	2.29	14.00	71.42	0.286
1.8	0.25	7200	2	4	1.4	$300						

For 3.5”disk, avg rotation time = (60/3600)*0.5=8.33ms.

To compute the average seek time you have to consider the seek distribution on UNIX time sharing workload. There are 24% accesses with seek distance of 0 track and 23% of accesses with seek distance of 1-15 tracks. 0.24*0+0.23*5+(1-0.24-0.23)*10=6.45ms

Transfer time = 4KB/2MB/s=2 ms.

Disk access time has four contributing factors. 8.33+6.45+2.00+1 (disk controller overhead)=17.78

I/O per second =1/(17.78*10^{-3})=56.24. MB/sec/disk=56.24*4KB=0.225MB.

Assume the 80% rule applies only to the average bandwidth case, since the average MB/s traffic generated by the 10 disks of any types are about 2.25~3.50 MB/s and not exceed 80% of the 10 MB/s bandwidth provided by the disk controller and the string. Therefore we assume it is OK to connect 10 disks to a string. The following calculation will be based on the above assumption.

If we assume 80% rule applies to peak rate (over any transient period) then your answer will be different.

Step 2. Calculate the average MB per second for each type of disks with increasing disk cache size, and the cost of basic disk design options.

Use the disk cache miss rate from Figure 9.30, page 538..

Disk Cache (MB) on each controller	Disk Cache Miss Rate	Time per disk cache access	MB/s for 100% disk cache access	MB/s for 10 3.5" disk+ cache	MB/s for 10 2.5" disk+ cache	MB/s for 10 1.8" disk+ cache	Total cost using 10 3.5" disks+ cache	Total cost using 20 2.5" disks+ cache	Total cost using 40 1.8" disks+ cache
0	100%	1.125	3.56	2.25	2.86		$12,500	$16,500	
1	58%	1.125	3.56	2.62	3.05		$13,000	$17,500	
2	21%	1.125	3.56	2.94	3.22		$13,500	$18,500	
3	16%	1.125	3.56	2.98	3.25		$14,000	$19,500	
4	12%	1.125	3.56	3.02	3.27		$14,500	$20,500	
8	10%	1.125	3.56	3.03	3.28		$16,500	$24,500	
16	8%	1.125	3.56	3.05	3.29		$20,500	$32,500	

MB/s for 100%disk cache access=4KB/1.125sec=3.56MB/sec.

MB/s for 10 3.5" disk+1MB cache=3.56*0.42*read hit ratio+2.25*(.58+0.42*write hit ratio)=2.62

10 3.5"disks+1 controller+1bus&rack=$12,500. 20 2.5"disks+2controllers+1bus&rack=$16,500.

For 2.5" disk system with 16MB disk cache on each controller, 16,500+16*2*500=32,500.

Step. 3 Calculate the MB per second and MB per second per $1000 for each option.

Disk Cache Size (MB)	MB/s 3.5”	MB/s/$1000 3.5”	MB/s 2.5”	MB/s/$1000 2.5”	MB/s 1.8”	MB/s/$1000 1.8”
0	2.25	0.18	5.72	0.3467		
1	2.62	0.201	6.11	**0.3493**		
2	2.94	0.218	6.46	0.3490		
3	2.98	0.213	6.50	0.3335		
4	3.02	0.208	6.54	0.3191		
8	3.04	0.184	6.56	0.2678		
16	3.05	0.149	6.58	0.2024		

2.25/($12500/$1000)=0.18. 2.94/(($12500+2*$500)/$1000)=0.218,

$16500 buys 20 2.5” disks with aggregate 5.72 MB/s performance. 5.72/($16500/$1000)=0.3467.

a) Which configuration gives the best MB/s/$1000 number?

For 3.5”, it is the configuration with 2 MB cache on each disk controller. 0.218

For 2.5”, it is the configuration with 1 MB cache on each disk controller. 0.3493

b) With the availability of 3.5” and 2.5” disks only, given $20500, what is your suggested configuration?
We will suggest the configuation with 20 1.8” disks with 4 MB cache for each disk controller.

Homework #4

Repeat Exercise 4 by considering the new 1.8” disk.

a) Fill the vacant table entries in the above three tables for 1.8” disks.

b) If we use 1,8” disk, which configuration gives the best MB/s/$1000 number?

c) Consider all three types of disks and given $20500, what is your suggested configuration?