Concepts Introduced

- I/O introduction
- magnetic disks
- flash memory
- communication with I/O devices
- controlling devices

I/O Cannot Be Ignored

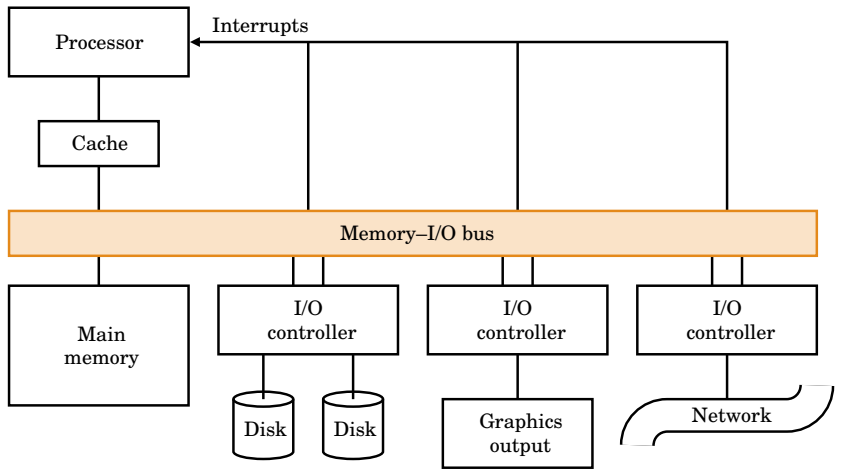
- Assume a program requires 100 seconds, 90 seconds for accessing main memory and 10 seconds for I/O.
- Assume main memory improves by 10% a year and I/O remains unchanged.

After n years	DRAM time	I/O time	Elapsed time	% I/O time	After n years	DRAM time	I/O time	Elapsed time	% I/O time
0	90	10	100	10%	11	28	10	38	26%
1	81	10	91	11%	12	25	10	35	29%
2	73	10	83	12%	13	23	10	33	30%
3	66	10	76	13%	14	21	10	31	32%
4	59	10	69	14%	15	19	10	29	34%
5	53	10	63	16%	16	17	10	27	37%
6	48	10	58	17%	17	15	10	25	40%
7	43	10	53	19%	18	14	10	24	42%
8	39	10	49	20%	19	12	10	22	45%
9	35	10	45	22%	20	11	10	21	48%
10	31	10	41	24%	21	10	10	20	50%

I/O Issues

- performance
  - response time (latency)
  - throughput (bandwidth)
- expandability
- standard interfaces
- fault tolerance

Typical Collection of I/O Devices



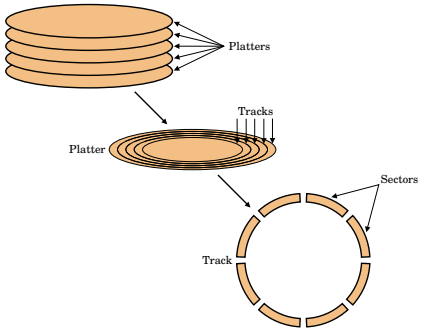
Diversity of I/O Devices

- There is a diversity of I/O devices. The ones that directly interact with a human are the slowest.

Device	Behavior	Partner	Data rate (KB/sec)
keyboard	input	human	0.01
mouse	input	human	0.02
laser printer	output	human	200.00
scanner	input	human	400.00
magnetic disk	storage	machine	200,000.00
network/LAN	input or output	machine	1,000,000.00

Magnetic Hard Disk Organization

- Consists of a set of platters, each having two recording surfaces, that rotate on a spindle.
- Each disk surface has a set of tracks (tens of thousands today).
- Each track is divided into sectors (thousands per track today), where each sector is 512 to 4096 bytes in size.



Magnetic Hard Disk Organization (cont.)

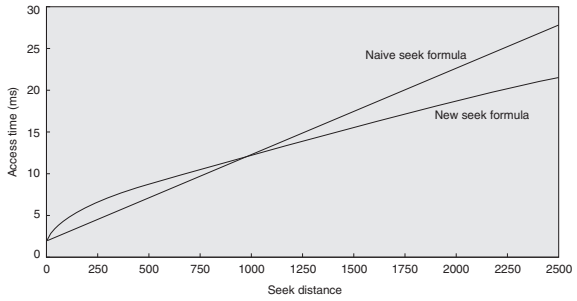
- Each sector contains a sector number, a gap, data for that sector, a gap.
- The disk heads for each surface are connected together so that every head is over the same track of every surface.
- A cylinder refers to all the tracks under the heads on all the surfaces of the disk.
- The diameter of today's disks is 2.5 to 3.5 inches and there are typically one or two platters per drive.

Disk Access

- seek time - move the read/write head to the desired track
- rotational delay - time for the requested sector to rotate under the head
- transfer time - time it takes to transfer data under the read-write head
- controller time - overhead the controller imposes in performing an I/O access

Seek Time

- Average seek time is reported as the average of all possible seeks.
- The average seek time will be less due to locality of disk references, which is due to files stored in the same cylinder (spatial locality) and repeated accesses to the same file (temporal locality).
- The seek time between tracks is not linear as moving the disk arm requires a lift off, acceleration, maximum traveling speed, deceleration, and settle time (stop vibration) portions.



Rotational Delay

- The rotational delay depends on two factors.
  - distance on the track from the current read-write head position to the desired sector
  - speed at which the disk is spinning
- Given a random position of the read-write head, the average time would be the time required for 1/2 a rotation.

Transfer Time and Controller Time

- The transfer time is how long it takes for a sector of data to be transferred and depends on:
  - the speed that the disk spins
  - the length of the sector, which depends the density of the data
- The controller overhead is typically a constant value.

Average Access Time Example

- What is the average time to access a 4096 byte sector for a disk rotating at 5400 RPM? Assume the average seek time is 8ms, the transfer rate is 10MB per second, and the controller overhead is 1ms.

$$avg\_access\_time = avg\_seek\_time + avg\_rot\_latency + transfer\_time + contr\_time$$
$$= 8ms + avg\_rot\_latency + transfer\_time + 1ms$$

$$avg\_rot\_latency = \frac{0.5\ rotation}{5400\ RPM} = \frac{0.5 rotation}{90\ RPS} \approx 0.0056sec \approx 5.6ms$$

$$transfer\_time = \frac{4KB}{10MB\ per\ sec} = \frac{2^{12}B}{10 * 2^{20}B\ per\ sec}$$
$$= \frac{1}{10 * 2^{8}B\ per\ sec} = 0.000390625sec \approx 0.4ms$$

$$avg\_access\_time = 8ms + 5.6ms + 0.4ms + 1ms = 15.0ms$$

Disk Density

- Disk capacity is measured in bits per square inch or *areal density*.

$$\frac{tracks}{inch} \text{ on a disk surface } * \frac{bits}{inch} \text{ on a track}$$

- Through 1988 the areal density improved about 29% per year. From 1989 to 1996 it improved about 60% per year. From 1997 to 2003 it improved about 100% per year. From 2004 it dropped to about 30% per year. In 2011 the highest disk areal density was about 400 billion bits per square inch.
- Improved areal density decreases both access time and cost. The cost per GiB has dropped by almost a factor of 1,000,000 between 1983 to 2011.

Two 3.5 Inch Disk Drives in 2011

- The SATA disk was designed for high capacity and low cost per GiB. Thus, it seeks more slowly, spins at 5900 RPM, and holds 2000 GB.
- The SAS disk was designed for performance. Thus, it seeks faster, spins at 15,000 RPM, and holds 600 GB.

	Capacity (GB)	Price	Platters	RPM	Diameter (inches)	Average seek (ms)	Power (watts)	I/O/sec	Disk BW (MB/sec)	Buffer BW (MB/sec)	Buffer size (MB)	MTTF (hrs)
SATA	2000	$85	4	5900	3.7	16	12	47	45–95	300	32	0.6M
SAS	600	$400	4	15,000	2.6	3–4	16	285	122–204	750	16	1.6M

Recent Trends in Using Magnetic Disks

- file caches
- disk buffers
- smarter disk controllers
- asychronous I/O becoming more common
- redundant array of inexpensive disks (RAID)

File Caches

- File caches also exploit the principle of locality to avoid disk accesses.
- A portion of main memory is used like a cache to hold recently accessed files to avoid disk traffic.
- File caches are so effective that they are found in every Unix system and often use the majority of DRAM available on the system.

Disk Buffers

- A disk buffer is comprised of DRAM on the disk and serves as a cache for recently accessed sectors.
- When reading one sector, the subsequent sectors are typically read and stored in the buffer as there is a high likelihood of them being accessed in the near future.
- Sectors can also be written to the disk buffer and stored at a later time so that disk sector reads can have priority over disk sector writes.

Smarter Disk Controllers

- Current disk controllers can schedule disk requests out-of-order to reduce seek time and rotational latency.
- Controllers may reorder sectors so that logically sequential sectors are not on placed in the same order.

Redundant Array of Inexpensive Disks (RAID)

- In the late 1980's, the high performance storage option was large and expensive disks.
- Accelerating I/O throughput was the original motivation of using arrays of disks by using more read/write heads.
  - Supports striping that spreads data over multiple disks that can be accessed in parallel, which increases bandwidth.
  - Supports independent accesses.
- Requires less energy since smaller platters, fewer platters, and slower rotation all require less power.
- However, the mean time to failure (MTTF) would be increased since each individual disk could fail.

Redundant Array of Inexpensive Disks (RAID) (cont.)

- The solution to reliability was to add redundancy so a disk system could deal with disk failures without losing data.
- There are currently 7 publicized levels of redundancy, which are referred to as RAID 0-6.
- Improved reliability and availability became the key reason for the widespread popularity of RAID.

RAID 0

- All RAID levels have an array of disks, which can be used to improve performance.
  - May allow striping of logically sequential blocks over multiple disks to improve throughput.
  - May allow support for independent accesses.
- No redundancy in RAID 0.

RAID 1: Mirroring

- RAID 1 is referred to as *mirroring*.
- Uses twice as many disks as RAID 0.
- Most expensive option since it requires $2n$ disks, where n is the number of original disks containing data.
- Whenever data is written to one disk, that data is also written to a redundant disk, so that there are always two copies of each block.
- When a disk fails, the disk system just uses the mirror disk.
- So data is always available unless both disks fail.

RAID 2: ECC Parity

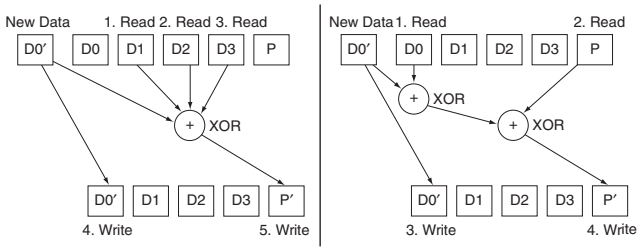
- RAID 2 is modeled after the error detection and correction codes that are most often used for memories.
- This approach is currently not being manufactured.

RAID 3: Bit-Interleaved Parity

- RAID 3 is referred to as bit-interleaved parity.
- Data is organized into stripes and all reads and writes access all n disks to improve throughput.
- Rather than have a complete copy of the original data for each disk, have 1 redundant disk that holds the parity for each corresponding bit from the n data disks.
- The parity is simply the sum of the corresponding bits modulo 2, which is also the exclusive OR of all the bits.
- When a disk fails, you simply replace the disk by calculating the parity of all the other disks to restore the bits from the failed disk to the new disk.
- The disadvantage of this approach over RAID 1 is that many disks must be read to restore the failed disk, taking longer to recover from the failure.
- The advantage over RAID 1 is there is less cost for the redundant storage.

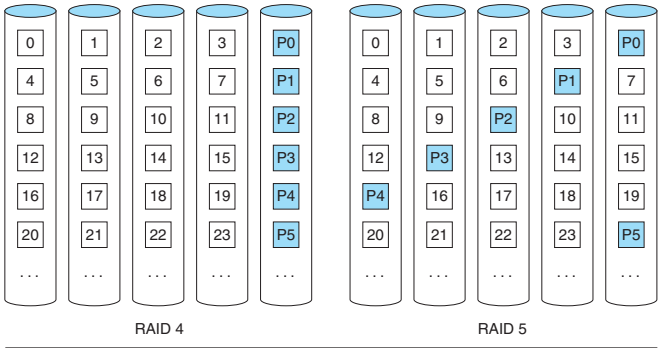
RAID 4: Block-Interleaved Parity

- RAID 4 is referred to as block-interleaved parity.
- Data are organized into blocks, allowing independent accesses to occur in parallel (true also for RAID 5-7).
- It would seem that a write would require reading *n* corresponding blocks and writing the block on the disk to be updated and the parity disk.
- Instead we can (1) read the old block of data that is about to change, (2) compare it to the new block of data that is about to be written, (3) read the old parity, (4) flip the parity bits for the bits in the new block that have changed, and (5) write the new data and new parity.



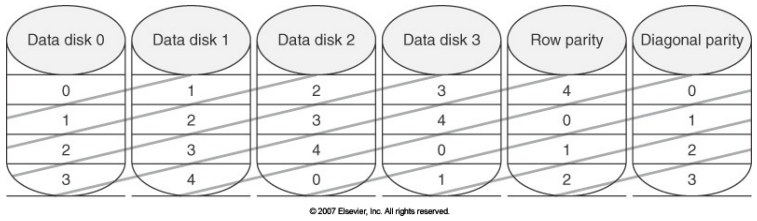
RAID 5: Distributed Block-Interleaved Parity

- RAID 5 is referred to as distributed block-interleaved parity.
- The parity disk can become a performance bottleneck since it has to be updated on every write and independent writes will have to wait to access the parity disk.
- This approach distributes the parity across multiple disks so not all independent writes will access the same disk to update parity.



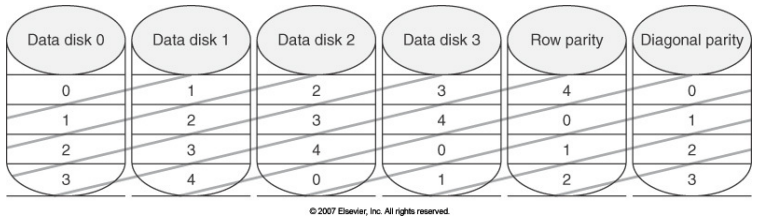
RAID 6: Row-Diagonal Parity

- RAID 6 supports recovery from 2 disk failures.
- The parity can be stored twice on two different disks using an approach called *row-diagonal* parity.
- The row parity is similar RAID 5, but it does not cover the diagonal parity block.
- The diagonal parity contains the parity for each diagonal, where each diagonal parity does not cover the corresponding block for one disk.

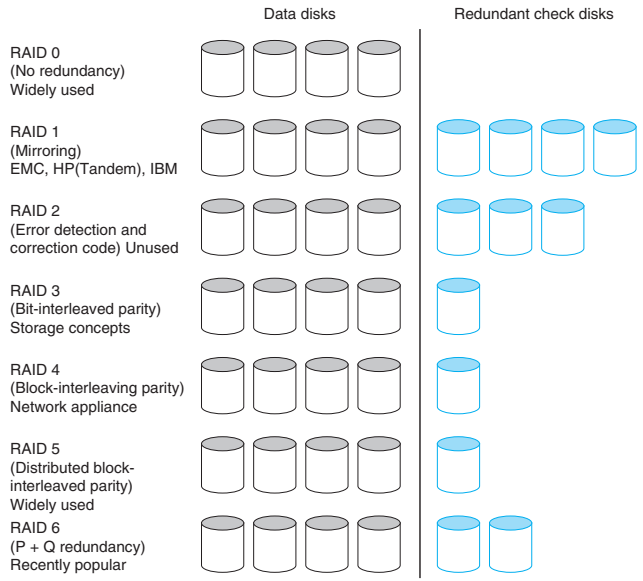


Example of Using Row-Diagonal Parity

- Assume both disks 1 and 3 fail. Their restoration occurs as follows.
  - Block 2 on disk 1 and block 0 on disk 3 are the only blocks within their diagonals that are unavailable, so we can use diagonal parity to restore these blocks.
  - At this point block 3 on disk 1 and block 4 on disk 3 are the only blocks within their rows that are unavailable, so we can use row parity to restore these blocks.
  - Now block 4 on disk 1 and block 3 on disk 3 are the only blocks in their diagonals are unavailable, so we can use diagonal parity to restore them.
  - Finally, block 1 on disk 1 and block 1 on disk 3 are the only blocks in their row that are unavailable and they are restored using row parity.



Extra Disks for Each RAID Level

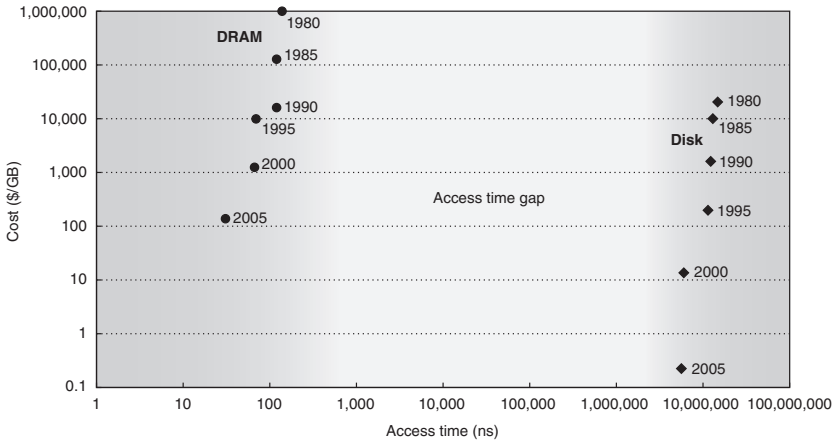


RAID Levels

RAID level		Disk failures tolerated, check space overhead for 8 data disks	Pros	Cons	Company products
0	Nonredundant striped	0 failures, 0 check disks	No space overhead	No protection	Widely used
1	Mirrored	1 failure, 8 check disks	No parity calculation; fast recovery; small writes faster than higher RAIDs; fast reads	Highest check storage overhead	EMC, HP (Tandem), IBM
2	Memory-style ECC	1 failure, 4 check disks	Doesn't rely on failed disk to self-diagnose	~ Log 2 check storage overhead	Not used
3	Bit-interleaved parity	1 failure, 1 check disk	Low check overhead; high bandwidth for large reads or writes	No support for small, random reads or writes	Storage Concepts
4	Block-interleaved parity	1 failure, 1 check disk	Low check overhead; more bandwidth for small reads	Parity disk is small write bottleneck	Network Appliance
5	Block-interleaved distributed parity	1 failure, 1 check disk	Low check overhead; more bandwidth for small reads and writes	Small writes → 4 disk accesses	Widely used
6	Row-diagonal parity, EVEN-ODD	2 failures, 2 check disks	Protects against 2 disk failures	Small writes → 6 disk accesses; 2× check overhead	Network Appliance

Cost and Access Time Gaps between DRAM and Disk

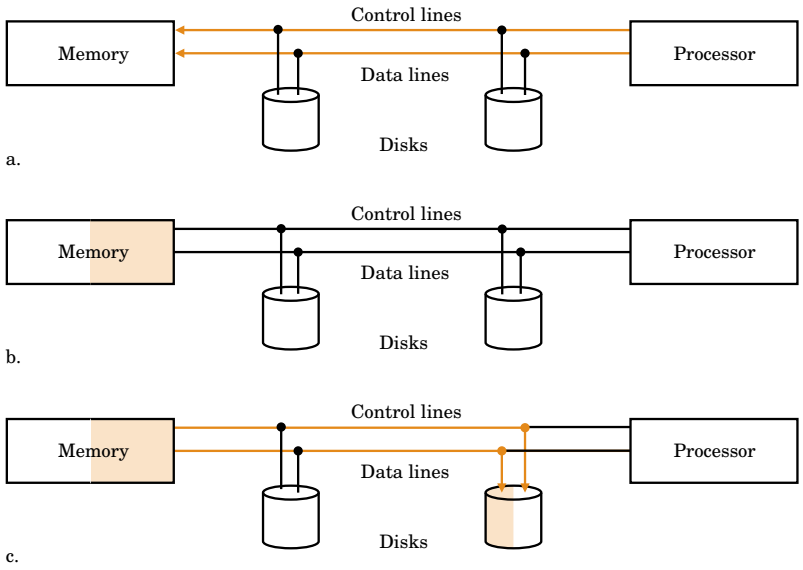
- There is a two order of magntitude gap in cost and a five order of magnitude gap in access times between DRAM and disk.



Flash Memory

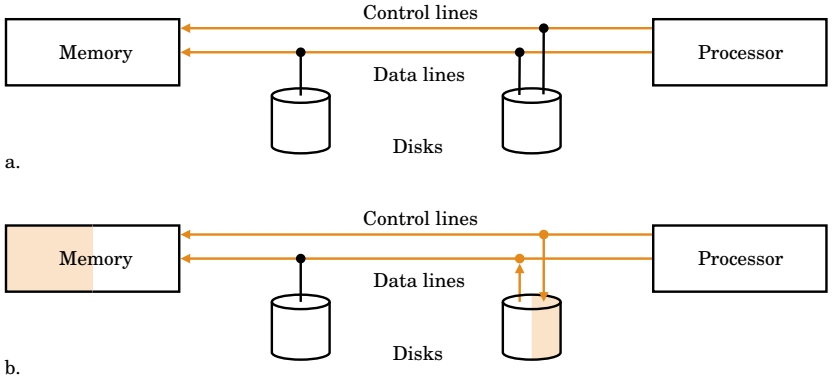
- Is a nonvolatile semiconductor memory and is the standard secondary memory for personal mobile devices.
- 15-20 times cheaper per GiB than DRAM.
- Similar bandwidth to disks, but latency is 100-1000 times faster.
- Cost per GiB is about 15-25 times higher than disk.
- More power efficient than disks since it is a solid state device.
- Only a limited number of writes can be made to each block.
- Includes a controller to spread the writes by remapping blocks that have been written more frequently, which is a technique called wear leveling.

The Three Steps of an Output Operation



The Two Steps of an Input Operation

- An input operation takes less active time because the device does not need to wait for memory to access data.



Specifying I/O Operations

- dedicated CPU I/O instructions - Is not commonly used as it requires instruction opcodes and I/O operations are very infrequent.
- memory mapped I/O
  - Portions of the address space are reserved for I/O devices.
  - Loads and stores to these addresses are ignored by the memory system, but are intercepted on the I/O bus and sent to the appropriate device to cause data to be transferred.

Managing I/O Operations

- polling - Periodically check status bits.
- interrupt-driven I/O - An interrupt is generated after each I/O event.
- direct memory access (DMA) - A special controller transfers data between an I/O device and memory independent of the processor. A DMA transfer is accomplished in the following steps.
  - The process specifies the device ID, I/O operation (read or write), memory address, and the number of bytes to transfer.
  - The DMA performs the transfer.
  - The DMA interrupts the processor.

DMA Addressing

- Problems with DMA using physically mapped I/O:
  - Transfers larger than a physical page are not likely to be contiguous in memory.
  - Would have to ensure that the OS cannot update pages while the DMA operation occurs.
- Using virtual DMA addressing requires a translation of each address before the data is transferred.
  - The buffer would be contiguous in virtual memory.
  - Either the address table of the DMA could be updated if a page is moved or specific pages could be locked into physical memory until the DMA operation is complete.

Asynchronous I/O

- The straightforward approach is synchronous (or blocking) I/O. The OS performs an I/O request for data, OS switches to another process until the desired data arrives, and then switches back to the requesting process.
- In asynchronous I/O, the process is not blocked after making the I/O request. Thus, many I/O requests can be operating simultaneously, which is similar to a nonblocking data cache.
- Asynchronous I/O will require software support in the OS (to allow switching to another process and knowing when the I/O operation is complete) and in the application (do useful work in the process while the I/O operation occurs).

Stale Data

- Copies of the data can be in:
  - a data cache
  - main memory
  - disk (virtual memory)
- *Stale data* occurs when these copies are not consistent.
- problems with stale data:
  - The CPU may not read the most recently input data since the cache may not be updated after an I/O input operation.
  - An I/O operation may not output the correct data because memory is not up-to-date.

Dealing with Stale Data

- Route all I/O through the cache instead of memory would interfere with the CPU and displace data that may be needed soon.
- I/O reads
  - Invalidate entire cache or selective cache entries after each read operation.
  - Perform read to a buffer that is marked as noncacheable.
- I/O Writes
  - Use write-through.
  - Force write backs to occur (flush cache) before each write operation.

Fallacies and Pitfalls

- Fallacy: Operating systems are the best place to schedule disk accesses.
- Fallacy: The time of an average seek of a disk in a computer system is the time for a seek across one third of the number of tracks.