Why is I/O so important?

Remember Amdahl's Law:

Speeding up part of the problem while largely ignoring the rest leads to diminishing returns.

This means that focusing on the CPU will result in larger fractions of time spent on I/O.

Response time versus **throughput**

It can be argued that I/O speed does not matter in a multiprogrammed environment.

If a process waits for a peripheral, run another task.

This is an argument that performance is measured as throughput and response time does not matter.

This is hardly the case today with desktop computers running interactive software.

Response time is directly related to productivity.

Storage devices

Storage devices are **mechanical**, so they are subject to real physical limitations.

For example, it is not possible to spin a disk at 100,000 RPM.

A disk made of any material that we know of today would fly apart.

These mechanical delays are far more difficult to eliminate than electronic delays.

Reducing the size of mechanical devices is more difficult than reducing the size of electronic circuits.

We will focus on storage devices of highest capacity:

- Magnetic disks.
- Magnetic tapes.
- CD-ROMS.
- Automated tape libraries.

Magnetic disks

Magnetic disks have dominated *secondary storage* since 1965.

Magnetic disks are used for two main purposes.

- They provide the lower level of the virtual memory system.
- They provide long-term, nonvolatile storage for files.

Disks are usually the highest level of the storage pyramid that is non-volatile.

Important terms associated with disks:

- *Platter*

  Disks are built from several metal platters (1 to 20) covered with metal oxides or other magnetic recording media.

  Platters are 1 - 8 inches wide.

- *Surface*

  Each platter has two surfaces, both of which may be used for recording.

Magnetic disks

Important terms associated with disks:

- *Track*

  A surface is divided into tracks, which are concentric circles containing data.

  There are 500 - 2500+ tracks per surface.

  The set of tracks at corresponding locations on all of the surfaces is called a cylinder.

- *Sector*

  A track is divided into sectors each of which holds a fixed amount of data, usually 512 - 4096 bytes.

  Older disks have a constant number of sectors per track.

  Newer disks record more sectors on the outer tracks using a **constant bit density**.

Magnetic disks

Important terms associated with disks:

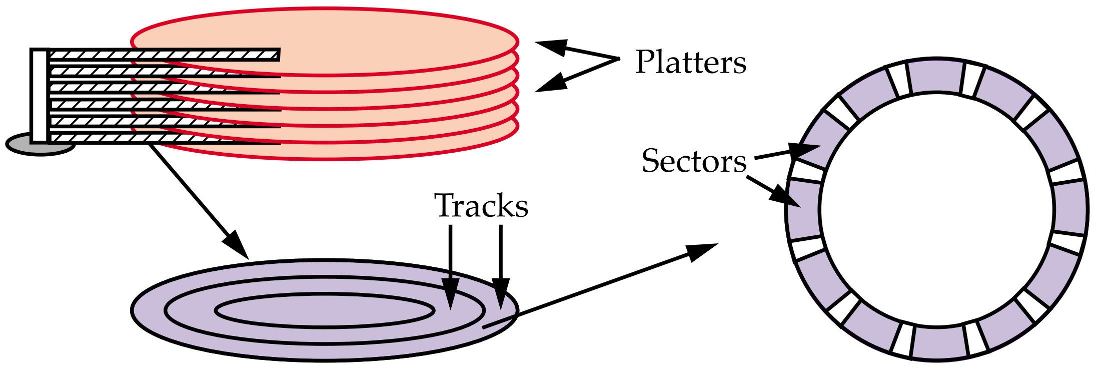
- *Disk arm/head*

Each surface is read and written by its own head.

All of the heads in a drive are connected together and move as a single unit.

Most disks have just one such unit.

However, more expensive drives may have two or more assemblies to increase the I/O rate.



Magnetic disks

Performance

There are three main components of each disk access:

- Seek time.
- Rotational delay.
- Transfer time.

Seek time

The seek time is the time necessary to move the arm from its previous location to the correct track for the current I/O request.

The industry standard is to use **average seek time**.

This is computed by averaging the time necessary for all possible seeks.

This tends to **overestimate** actual average seek time because of locality of disk references.

Advertised average seek times of 8 ms to 12 ms may be smaller by 25% to 33%.

Magnetic disks

Rotational delay

The time necessary for the requested sector to rotate under the head.

Most disks rotate at 3600 - 7200 RPM (note that disks rotated at 3600 RPM in 1975!).

On average, the disk must rotate *halfway* around to get to a desired sector.

On a disk rotating at 6000 RPM, this delay is 0.5 / (6000 RPM / 60,000 ms/min) = 5 ms.

Therefore, the two mechanical components, moving the disk arm and waiting for the data to rotate under the head, add to the latency.

Some disks can **read data out of order** into a buffer, reducing rotational delay for a large transfer.

A full track can be transferred in one rotation regardless of where the I/O actually starts.

Magnetic disks

Transfer time

The transfer time is the time it takes to read or write a sector.

For the disk, this is approximately equal to the time it takes for the sector to *pass fully* under the read/write head.

It is a function of the block size, rotation speed, recording density and the speed of the electronics.

Typical rates in 1995 were 2 to 8 MB/sec.

A disk rotating at 6000 RPM with 64 sectors per track and 512 bytes per sector can read a **sector** in 10 ms/64 = 0.156 ms.

Since 0.5 KB are transferred in this time, bandwidth is 3.2 KB/ms, or 3.2 MB/s.

Note that the transfer rate is higher for sectors on outside tracks (on disks that have variable number of sectors per track.)

Magnetic disks

Other components to latency

Note, too, that other parts of the system can add delays.

Controllers and I/O buses can add their own delays, possibly decreasing bandwidth and increasing latency.

For example:

Suppose we have a new disk and want to see how quickly we can read a single sector.

Each sector is 1 KB long, tracks have 32 sectors each, average seek time is 8 ms, and the disk rotates at 7200 RPM.

The controller adds 2 ms overhead to each request.

$$\text{Total time} = 8ms + \frac{0.5}{120RPS} + \frac{1}{120RPS \times 32} + 2ms$$

$$\text{Total time} = 8ms + 4.17ms + 0.26ms + 2ms = 14.43ms$$

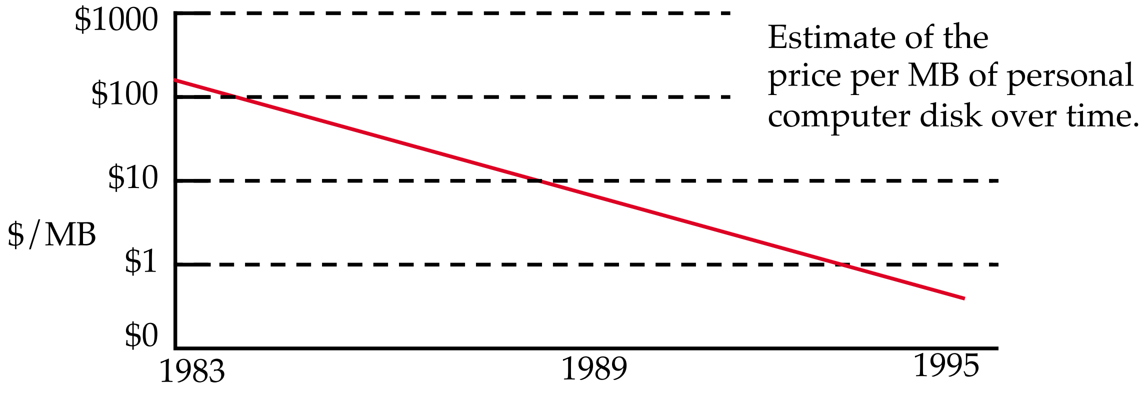
The Future of Disks

- *Density*

Disk capacity is measured in **areal density** (the number of bits per square inch).

This is the product of tracks per inch on a surface and bits per inch on a track.

Density has recently improved at a rate of 60% per year, matching density increases of DRAM.



The Future of Disks

- *Transfer rate*

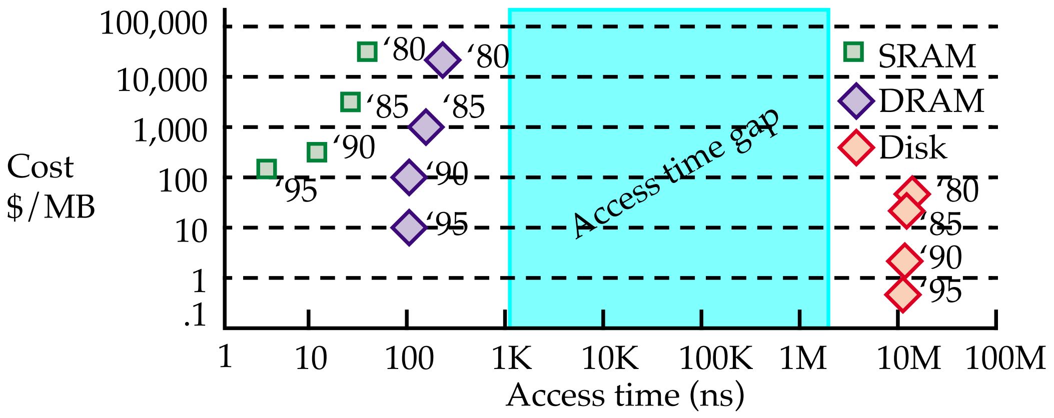
  Since transfer rate is proportional to bits per track, improvements in **density** usually lead to higher bandwidth.

  This increase is usually the **square root** of overall density increase.

  Seek time is reduced by smaller disks and lighter heads.

  However, seek time is still limited by mechanical considerations.

The Access Time Gap



The Future of Disks

In 1995, the price/megabyte of Disk is about 100 times cheaper than the price/megabyte of DRAM.

However, DRAM is about 100,000 time faster.

Many have tried to fill the **Access Time Gap** with a new technology.

So far, nobody has succeeded.

Technologies such as bubble memory and flash RAM have been proposed, but usually end up being "taken over" by either memory or disk.

Other Storage Devices

A challenger to magnetic disks.

- *Optical disks (CDs)*

  Optical disks contain data stored optically rather than magnetically.

  Optical storage is removable and inexpensive to manufacture.
  For example, it costs \$1 to \$2 per 500MB CD-ROM.

  CDs are a popular medium for the distribution of software.

  Optical disks are usually write-once.
  They are also called **WORM** (Write Once, Read Many) storage.

  Agreement on standards for CDs have slowed their time-to-market, allowing magnetic disks to stay ahead.

  However, writable optical disks may have the potential to compete with new tape technologies for archival storage.

Other Storage Devices

- *Magnetic tapes*

  Magnetic tapes use the same basic kind of recording technology as magnetic disks.

  The major difference:
  - For tapes, the read/write heads actually touch magnetic tapes.
  - For disks, they *float* above the disk surfaces (the Bernoulli effect).

  Tapes are cheaper than disks for two reasons:
  - They pack more medium into a fixed size by recording on long strips.
  - They are removable (allowing one reader to access multiple tapes).

  This means that a 9GB cartridge can cost less than \$20.
  \$2/GB is considerably less than the \$30/GB for magnetic disk.

Other Storage Devices

- *Magnetic tapes*
  - **Longitudinal scan (linear scan)**

    This is similar to audio cassette tapes.

    Data is written in a track that runs the length of the tape.

    As with audio tape, a single tape may have more than one track, and the tracks may run in opposite directions.

  - **Helical scan**

    These tapes put their heads on a rapidly spinning cylinder, which spin at on a diagonal to the tape.

    By increasing the speed of the heads relative to the tape, they allow much denser storage (20 to 50 times).

    The problem is that they cannot search as quickly, and they wear out more quickly (100's of passes versus >1000's of passes for longitudinal.)

Other Storage Devices

- *Robots (tape or optical disk)*

  Robots can be used automatically move media (tapes or optical disks) between drives and shelves.

  This allows **terabytes** of data to be accessible within ten's of seconds.

  Since reader cost is fixed, tape robots cost less per unit of storage as more media share the same number of readers

  The robot cost is also amortized.

  Of course, more tapes per drive means more contention for drives and lower bandwidth per tape.

  A large tape robot can cost as little as $40 per GB for a full system.

Other Storage Devices

- *Holographic storage (future?)*

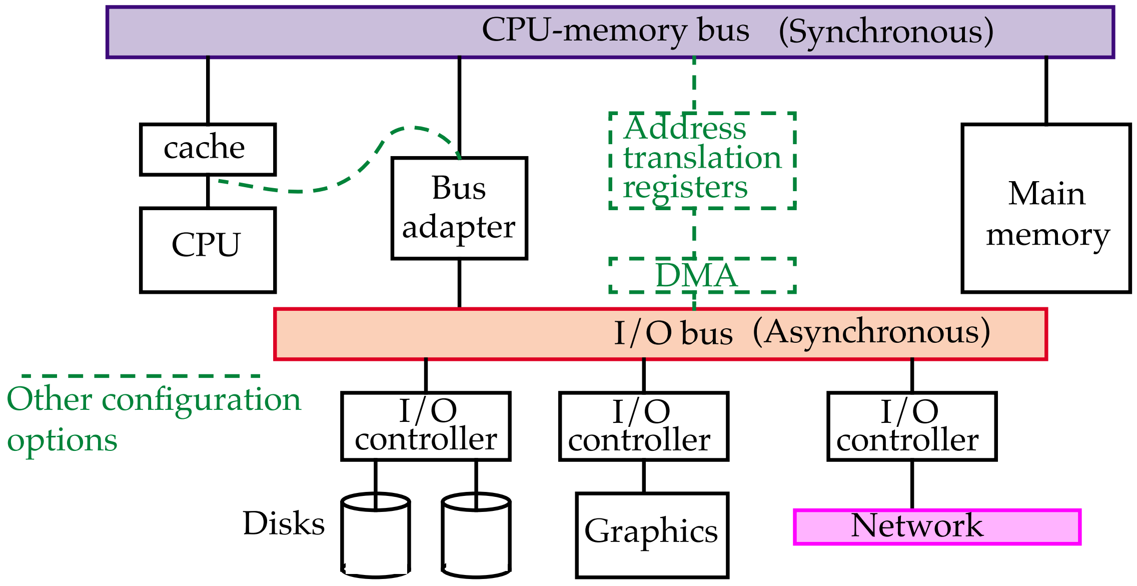
  There is lots of research on using cubes of *light-sensitive* material to store information holographically.

  A device would read or write an entire "slice" at once using lasers.
  This would provide very high bandwidth and high storage density.

  There are currently several problems researchers are working to solve.
  In particular, making this medium read/write is difficult.

Buses

The bus serves as a shared communication link between the subsystems of a computer, e.g. the CPU, memory and the I/O devices.



Buses

Bus advantages:

- They are *low cost.*
  A single set of wires is used to connect multiple components.
- They are *versatile.*
  New devices can be easily added or removed.

Disadvantage:

- They create a *communication bottleneck*, possibly limiting I/O throughput.

Two types:

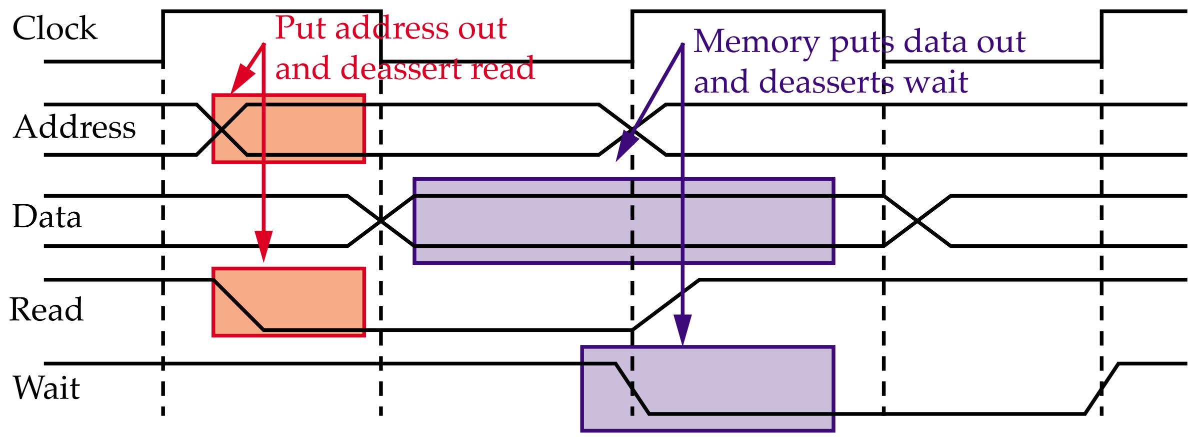
- *CPU-memory buses*
  Short, generally high speed and matched to maximize memory-CPU bandwidth.
- *I/O buses*
  Lengthy with many types of devices connected to them, each of which may have different data bandwidths.
  They usually follow a standard.

Buses

Basic bus transactions:

- If the transaction is a write (to memory), the data may be sent immediately after (or perhaps in parallel with) the address.
- For reads (from memory), there is usually a delay between the time the address is sent and the time the data is ready.
  This delay occurs because of the memory latency.

Typical bus read transaction (on a synchronous bus):



Bus Design Decisions

Cost versus performance trade-offs:

Option	Better performance	Lower cost
Bus width	Separate address & data lines (adv for writes)	Multiplex address & data lines
Number of bus lines	Wider is faster	Narrower is cheaper
Transfer size	Multiple words have less bus overhead	Single word transfer has simpler logic
Bus masters	Multiple, arbitrated	Single, no arbitration
Split transaction	Yes. Separate request and reply packets get higher bandwidth	No. Continuous connection is cheaper and has lower latency
Clocking	Synchronous	Asynchronous

The choice of using any of the options listed in the first three rows is straightforward.

All give higher performance at more cost.

Bus Options

Bus masters

A bus master is a device on the bus that has the power to initiate a transaction.

The CPU is always a bus master.

I/O devices can also be bus masters.

With multiple masters, an arbitration scheme is required to determine who gets the bus next.

Split transactions

For many buses, a transaction must be performed atomically.

This means the bus is busy from the time the data is request until the time the request is complete.

This may work well for memory, but it can be intolerable for I/O where data may not be ready for 10ms after the request.

Bus Options

Split transactions

In systems that have **multiple bus masters**, split transactions provide an improvement in bandwidth.

In a split transaction, the read request is sent and the bus is *released* while the data is prepared.

Once memory has retrieved the data, it arbitrates for the bus and 'tags' the data reply for the CPU.

This makes the bus available for other bus masters while memory reads the words from the requested address.

Split transactions can be even more beneficial for I/O buses.

For example, a SCSI bus can have **several** long disk transactions in progress at the same time.

However, a split transaction may have *higher latency* than a bus that is held during the complete transaction (due to arbitration.)

Bus Options

Synchronous vs. asynchronous

Synchronous buses require a fixed protocol for addresses and data relative to a global clock that all devices agree upon.

Signals are only considered valid at certain points on the clock waveform (usually the rising or falling edge).

If all devices are equally "fast," synchronous works well and it is inexpensive and fast.

Two disadvantages:

- Cheap devices, such as keyboards, must have logic that is able to run under the same fast clock rate used for disks.
- Clock-skew problems limit the length of synchronous buses.

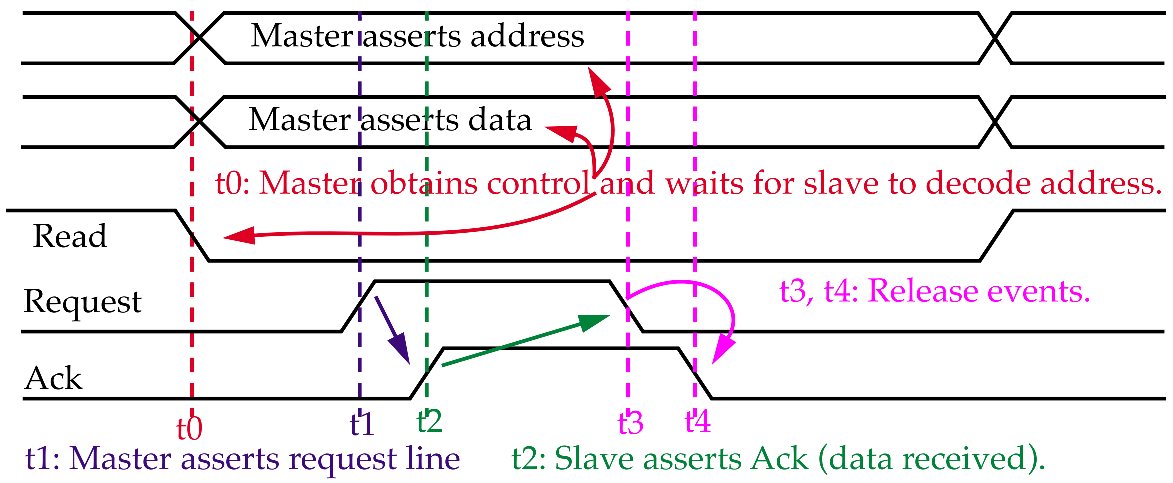
CPU-memory buses are typically synchronous.

Bus Options

Synchronous vs. asynchronous

The alternative is *asynchronous logic*, which uses hardware handshaking to exchange information on the bus (usually an **I/O bus**).

Typical write transaction on a asynchronous bus:


Master asserts address
Master asserts data
t0: Master obtains control and waits for slave to decode address.
Read
Request
t3, t4: Release events.
Ack
t0
t1
t2
t3
t4
t1: Master asserts request line
t2: Slave asserts Ack (data received).

Asynchronous buses address previous disadvantages are usually slower because of the overhead associated with the handshaking protocol.

Performance Metrics

- *Throughput*

  For I/O devices, throughput is usually called *I/O bandwidth*.

  This measures the amount of data a particular I/O device can transfer in a given period of time.

- *Response time*

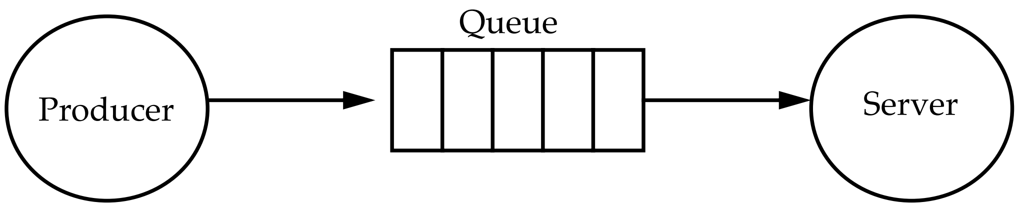
  This is called *I/O latency*.

  This measures the time taken from when a request is placed into a device queue until the time that service is completed.

  Note that this may include time spent waiting in a queue as well as the usual service time.

Performance Metrics

Trading off throughput & response time.



For minimum **response time**, queues should be kept empty so an incoming request can be serviced as soon as possible.

However, maximum **bandwidth** makes the opposite true.

Try and keep the queue full so the device always has something to do.

Throughput can be improved in many ways.

For example, a designer could add additional disks to a system allowing more requests to be serviced at a given time.

Requests would likely spend less time waiting in queues, which **may** improve response time as well.

Human-computer interaction

In general, improving performance does not always improve both response time and throughput.

Given that this is true, which is more important ?

In a multiprogramming environment, one might say *throughput*.

However, user productivity is **non-linearly** related to *response time* as shown below.

Interactions between humans and computers are divided into three "types" of time:

- *Entry time*

  This is the time to enter a command into the computer.
- *System response time*

  The delay between when the command is completed and the full answer is returned.
- *Think time*

  The time from the reception of the response until the user begins to enter the next command.

Human-computer interaction

The sum of these times is the **transaction time**.

One study shows that user productivity is inversely proportional to transaction time.

In particular, people get more productive as the computer reacts more quickly.

In other words, reducing response time actually decreases transaction time by more than just the reduction in response time.

In general, people need less time to think when given a faster response time.