Persistence and I/O

The structure of a typical system is given as this:

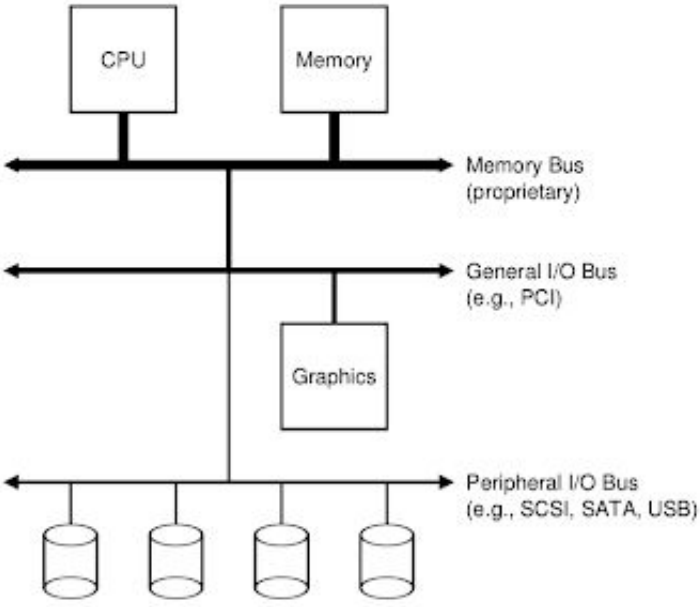

Figure 36.1: **Prototypical System Architecture**

A hierarchical structure like this is followed because the faster a bus is, the shorter it must be, and high performance buses are very costly. So components that desire high performance have the faster buses.

A Canonical Device

A device has two main components : the interface it presents to the system and the internal structure of the device.

A Canonical Protocol

The device interface consists of three registers

1. Status
2. Command
3. Data

To make the device do something the OS does the following

1. Wait until the device is ready by repeatedly reading the status register (polling)
2. Send data to the data register via the main CPU (programmed I/O)
3. Write a command to perform in the command register
4. Wait for device to finish by polling

Here, polling continuously is very inefficient and wastes a lot of CPU time.

Lowering CPU Overhead with Interrupts

Instead of polling the device repeatedly, the OS issues a request, puts the calling process to sleep and switches to another task. The device, once done, raises an interrupt causing the CPU to jump into the OS at a predetermined interrupt service routine (ISR) or an interrupt handler. And the handler finishes the request and wakes up the process and schedules it again.

This allows the overlap of computation and I/O, which improves utilisation.

Using interrupts is not always good, because some devices may service I/O requests very quickly. Here we use polling in the beginning, then migrate to an interrupt mechanism to best utilize the CPU.

Another reason for not using interrupts are livelocks, where an OS finds itself only servicing interrupts rather than doing useful work.

An additional interrupt based optimization is coalescing, where a device which needs to raise an interrupt waits for a bit, to service other requests that also may complete, reducing overhead.

The Direct Memory Access

The canonical protocol is still bound by a very trivial task of transferring data to slow devices. A DMA engine takes this process and the DMA does the transfer on behalf of the CPU, and raises an interrupt once done.

Methods of Device Interaction

1. I/O instructions
    a. Specify a way for the OS to send data to specific device registers
    b. Privileged instructions, only OS is allowed to directly communicate
2. Memory mapped I/O
    a. Hardware makes device registers available as if they were memory locations

Device Drivers

A device driver abstracts the working of the device from the OS by hiding the details of the device interactions from the OS subsystems. This provides a generic interface to every device, which may not always be good, as devices can have very rich error handling and reporting, while the underlying OS does not.

Hard Disk Drives

Hard disks are the main form of persistent storage available today. A drive consists of a large number of sectors, numbered from 0 to n-1, on a disk with n sectors. Thus a disk is viewed as an array of sectors, called its address space.

Multi-sector operations are possible, and a single 512 byte write is atomic on the drive. The key assumptions made about these drives are

1. Accessing two blocks near to each other is faster than blocks that are far apart.
2. Accessing blocks is sequential and faster than a random pattern

Basic Geometry

A platter is a circular hard surface on which data is stored persistently by inducing magnetic changes in it. Each side of a platter is called a surface. All platters are bound together on a spindle that rotates and spins the platters around, and the speed is measured in rotations per minute. Data is encoded on each surface in concentric circles, called tracks. The disk head is used per surface to read data from a track, and the disk arm moves the head around to read different tracks.

Rotational Delay

In a simple disk, the time taken for the desired sector to rotate under the disk head is called the rotational delay.

Multiple Tracks : Seek Time

Seeking on a multi-track disk has many phases

1. Acceleration : To get the disk arm moving
2. Coasting : Arm moving at full speed
3. Deceleration : To stop the disk arm
4. Settling : head positioning on the correct track

The process of moving the head to the correct track is called a seek.

Minor Details

Track skew is implemented because when switching from one track to another occurs, the disk needs time to reposition the head. Without skew, the head will be moved to the next track, but the desired block may have rotated. The skew accounts for this delay.

Outer tracks tend to have more sectors than the inner tracks, known as multi-zoned tracks, as these tracks are divided into zones of consecutive tracks.

The cache or the track buffer is used to hold data read or written from the disk. On writes, the disk can choose to implement write-back caching, where it acknowledges the write when it is put in memory, or write-through caching, where it acknowledges the write when the data is actually written to disk.

I/O time

I/O time is represented as the sum of T_seek, T_rotation and T_transfer. The rate of I/O R is the Size of transfer / I/O time.

We consider two workloads, random and sequential workloads.

The disk drive market trades off on performance and cost, being inversely proportional to each other. So the I/O time for these two workloads changes depending on the drive.

Average seek time is calculated as the sum of all seek distances for all tracks for N sectors as

$$\int_{y=0}^{x} (x-y)\,dy + \int_{y=x}^{N} (y-x)\,dy = \frac{N^3}{3}$$

Disk Scheduling

The length of each job is known beforehand with disk scheduling, so it usually uses the principle of shortest job first.

SSTF: Shortest Seek Time First

SSTF orders the queue of I/O requests by track, picking the requests on the nearest track to complete first. This has drawbacks as

1. Drive geometry is not available to the host
    a. Use nearest block first instead
2. Starvation

Elevator or SCAN or C-SCAN

This approach simply moves across the disk servicing requests in order across the tracks. A single pass is called a sweep. This avoids starvation.

A variant called F-SCAN freezes the request queue during a sweep, avoiding starvation of far away requests.

Circular SCAN or C-SCAN sweeps from outer to inner, then inner to outer. This is referred as the elevator algorithm.

The problem with these is that it does not approximate SJF well.

Shortest Positioning Time First (SPTF)

This is used when seek time is faster than rotation, or roughly equivalent. It is generally difficult to implement in an OS that does not know its track boundaries perfectly, though.

Other Scheduling Operations

I/O merging is done when many requests to a drive are batched together to be performed in a single I/O. Work conservation is kept in mind.

Sometimes, it is better to have a non-work-conserving approach, where anticipatory disk scheduling is used, where the disk waits for a better or more heavy request to come and batches it together. The waiting time must be carefully defined in this case.