Lecture 14: Input/Output & Storage Management

Adam Hawley

March 13, 2019

Contents

1 I/O Management

1.1 Introduction

I/O subsystem is responsible for controlling devices connected to a computer. It must provide processes with a sufficiently simple interface and also take device characteristics into account to maximise performance and efficiency.

There is a large variety of I/O devices:

- Storage (e.g disk drives, non-volatile memory)
- Communications (e.g. Ethernet, WiFi, Bluetooth, USB)
- User Interface (e.g. mouse, touch, keyboard, display, sound)

1.2 Device Drivers

Device drivers are low-level sofware that interacts directly with device hardware, they hide the hardware details to the higher levels of the OS and user applications and are often developed by the hardware vendor. They track the status of the device and enforce access/allocation policies. Types of drivers:

Dedicated Each device is allocated to a single process

Shared Each device is shared between multiple processes

Virtual Hides sharing from processes

1.3 Devices

Devices usually have registers where the device driver places commands, addresses and data to write or read data from registers after command execution. A minimum setup usually consists of the following:

- Data-In Register
- Data-Out Register
- Status Register
- Control Register

Where each register is typically 1-4 bytes and they may be contained in a FIFO buffer.

Devices themselves also have addresses used by direct I/O instructions or memory-mapped I/O.

1.4 I/O Management

The I/O subsystem provides interfaces to access devices via device drivers (or access to specific devices in a family of devices hidden by the device driver). There are three main device communication mechanisms:

- Polling & Interrupts
- Direct Memory Access (DMA)
- Buffering

1.5 Polling

Polling is about checking if a device is ready for communication so for each byte of I/O:

1. Read busy bit from status register until 0
2. Host sets read or write but and if write copies data into data-out register.
3. Host sets command-ready bit
4. Controller sets busy bit, executes transfer
5. Controller clears busy bit, error bit, command-ready bit when transfer done.

Step 1 is a busy-wait cycle to wait for I/O from device. This is reasonable if the device is fast but inefficient if it is slow. The CPU could switch to other tasks, but if miss a cycle data could be overwritten/lost.

1.6 Interrrupts

Polling can happen in 3 instruction cycles:

1. Read status
2. Extract status bit
3. Branch if not zero

How to be more efficient if devices are seldom ready?

CPU Interrupt-Request line triggered by I/O device (checked by processor after each instruction). The interrupt handler receives interrupts (maskable to ignore or delay some interrupts). Interrupt vector to dispatch interrupt to correct handler. This has a context switch at the start and at the end. We get **interrrupt chaining** if more than one device at same interrrupt number.

1.7 Direct Memory Access (DMA)

This is used to avoid programmed I/O (one byte at a time) for large data movement. This requires a DMA controller and bypasses CPU to transfer data directly between I/O device and memory.

The OS writes DMA command block into memory.

- Source and destination addresses
- Reade or write mode
- Count of bytes
- Writes location of command block to DMA controller.

2 Storage Devices

2.1 Introduction

The hierarchy of storage devices is driven by performance and volatility of data. **Data access time** includes:

Ready time Time to prepare set up storage media to read/write data at the appropriate location (e.g. wind/rewind tape, rotate disk, charge memory row)

Transfer time Time to read/write data from media

Different devices may impose access latencies at different orders of magnituse and hence, the OS should manage each of them appropriatly and mediate transfers (e.g. buffering).

2.2 Tertiary Storage

Tertiary storage is usually used for backups, storage of infrequently used data and transfer between systems. The two main forms of tertiary storage are:

- Magnetic tapes:
  - GB to TB capacity
  - Very slow access time (must wind and rewind to position tape under read-write head but once in place, reasonable transfer rates >140 MB/s)
- Optical discs:
  - MB to GB capacity
  - Read-only or read-write using high intensity laser beams

2.3 Secondary Storage

Secondary storage is mainly used for non-volatile storage, high-capacity storage supporting swapping/paging.

2.3.1 Magnetic Disks (HDDs)

- Made of n disks (2/n/ sides), each side is divided into tracks (circular), and each track into sectors.
- **Cyclinder**: Set of tracks at the same position on all sides
- **Access Time**: Seek time (disk head movement) + Search time (rotational delay) + Transfer time
- Typical Avg Values:
  - Seek = 25ms
  - Search = 4ms
  - Transfer = 0.00094ms/MB
  - Rotation speed = 7200rpm (120rps)

2.3.2 Non-volatile memory (NVMs, SSDs)

- Made of no mechanical components
- More reliable than HDDs (no moving parts), can be faster (no seek time or latency), consumes less power.
- More expensive per MB, lower capacity, may have shorter lifespan (writes wear it out).

2.3.3 Redundant Arrays of Independent (Inexpensive) Disks (RAIDs)

- Set of physical disks viewed as a single logic unit by the OS.
- Simultaneous access to multiple drives
  - Increased I/O performance
  - Improved data recovery in case of failure
- Data is divided into segments called stripes, which are distributed across the disks in the array.
- RAIDs can be classified as level 0,1, ..., 6 (different levels denote different approaches to data redundancy and error correction methods).

1. RAID 0
   - Block-level striping: data is divided into segments that are stored across the disks in the array.
   - Minimum disks: 2
   - Read speed-up is roughly proportional to the number of disks in the array, because distinct data can be read from different disks simultaneously.
   - Write speed-up is roughly proportional to the number of disks in the array, because distinct data can be written to different disks simultaneously.
   - No redundancy, therefore no fault tolerance.
   - As reliability is inversely proportional to the number of disks, a RAID 0 will be more vulnerable to faults than a single hard disk.

Where $MTTF$ is Mean Time To Failure:

$$MTTF_{group} \approx \frac{MTTF_{disk}}{number} \tag{1}$$

It is possible to use disks of different capacities, but the storage space added to the array by each disk is limited to the size of the smallest disk.

- For example, if a 120GB disk is striped together with a 100GB disk, the capacity of the array will be 200GB.

2. RAID 1
   - Full redundancy: mirroring (data is copied in all disks of the array)
   - Minimum disks: 2
   - Tolerates faults on up to $N - 1$ disks
   - Read speed-up is roughly proportional to the number of disks in the array because distinct daya can be read from different disks simultaneously.
   - No write speed-up, as writes have to be done on all disks.
   - Very low space efficiency: $1/N$
3. RAID 2,3,4 **SKIPPED IN LECTURE AND NOT ASSESSED**: SEE LECTURES/SILBERSCHATZ FOR DETAILS
4. RAID 5
   - Block-level striping, distributed parity. (Think of parity bits from ICAR)
   - Minimum 3 disks, tolerates fault in one disk
   - Writes are costly operations
   - Widely used
5. RAID 6
   - Block-level striping, double distributed parity. (Think of parity bits from ICAR)
   - Minimum disks: 4, tolerates faults in two disks.
   - Writes are costly
   - Widely used

3 Storage Management

Multiple requests have to be handled concurrently, several programs may have requested storage operations. There are a number of policies for servicing disk requests.

The I/O scheduler is similar to the process scheduler. It choses which request should be chosen next and often depends on some criteria which usually aim to reduce average response times. There may also be prioritised requests e.g. from OS components. Specific devices may require dedicated scheduling policies (e.g. to minimise seek time in magnetic disks and avoid redundant writes in non-volatile memory).

4 Disk Management

4.1 Disk Formatting

- Low-level (physical) disk formatting is when a disk is divided into sectors (built-in error correction codes, also called checksums).
- Logical formatting is to do with creating a file-system - directory trees, maps of free and allocated space.

4.2 Partitions

Partitions are when there are multiple logical disks on a single physical disk.

4.3 Blocks

Boot Block Initial bootable code at a known sector (useful as it helps reduce power-up complexity in hardware).

Bad Blocks These are blocks which have been permanently corrupted and file systems have to be able to mark them.

4.4 Defragmentation

Defragmentation is when sectors which are used by files are rearranged to be contiguous.