Operating Systems

Mass-Storage Structure
Based on Ch. 10-11 of
OS Concepts by SGG

Mass Storage Devices

- Disks
  - A r/w head “flies” above the spinning disk
  - Each disk is divided into tracks and each track in sectors
  - A set of tracks on different platters is a cylinder
  - Rotational velocity is important. Most drives are at 7200, 10,000 or 15,000 rpm
  - Positioning time is seek time plus rotational latency
- SSDs
  - No moving parts, but finite number of r/w ops
  - Smaller, faster (recently), cooler, quieter than HDDs
  - Zero positioning time, huge transfer rates
  - Can be written, can be read, can be erased (a whole block at a time), but cannot be overwritten.
    - Tricky to erase a single page within a block.
    - Hard to keep the number of erasings to a minimum.

Volatile Disks

- Mapped to memory
- Known as ram disks or ram drives
- Useful for mapping /tmp or root filesystem for booting.

Busses

- ATA: Antedeluvian Technology Attachment
- SATA: Serial ATA (6Gb/s, ~600MB/s)
- M.2: several times faster than SATA depending on configuration
- PCIe: up to 32Gb/s
- eSATA: external SATA
- USB: 10Gb/s (for 3.1)

Disk Structure

- Disk space is split into blocks (512byte is common)
- Blocks are mapped onto sectors
- Mapping is *almost* one-to-one
  - We have to skip defective sectors
  - We have to arrange for replacement sectors
  - We have to keep the info about replacement sectors
- Mapping gets trickier for disks that do not have the same number of sectors per cylinder

Disk Attachment

- Host-Attached Storage
  - Disks, RAIDs, DVD-ROMs, etc
  - All in the same box
- Network Attached Storage
  - Disks are somewhere on a (local area) network
  - Shared by many computers
  - Used with NFS, SSHFS, SMB (CIFS)
- Storage Area Network
  - Uses a special purpose network just for file serving
  - Several disk arrays and file servers connected to it
  - Usually uses Fiber Channel.

Disk Scheduling

- Seek time and rotational latency are important factors in the overall latency on disks
- We can minimize them by reordering the outstanding requests
  - Avoid having the disk head go in and out like mad
  - Not different than an elevator.

FCFS

- Fair but inefficient
- Can even lead to premature failure if the first and last cylinder are addressed alternatively

Shortest Seek Time First

- Selects the next destination to be the “closest”
- Performs very well, but it is not optimal
- Can suffer from starvation

SCAN, C-SCAN

- With SCAN we sweep from the first cylinder to the last and back servicing requests on the way
- C-SCAN (Circular), like scan but service requests only in the forward direction (more uniform waiting time)
- Linux *deadline* scheduler. Separate read and write queues (two of each one FCFS another LBA (log. Block Address) i.e. SCAN)

Selecting Disk Scheduling Algorithm

- None is optimal
- For very low traffic, all are more or less the same
- For medium traffic SSTF has an advantage
- For heavy traffic SCAN et al exhibit no starvation.

Disk Formatting

- Low level formatting
  - Usually at the factory
  - Involves setting up headers/trailers, with ECC
  - Identification of bad blocks
- High level formatting
  - Partitioning the disk
  - Setting up filesystems on partitions
    - FS like vfat, ext4, btrfs, raiserfs
    - Raw or swap space

MBR

- Master Boot Record
- Microsoft term
- Contains the partition table, boot code
- Identifies the boot partition

GPT

- Master Boot Record successor
- Can be used on bigger disks
- Can have many more partitions
- Keeps multiple copies of itself
- GPT = GUID Partition Table
  - GUID = Globally Unique Identifier
- Meant to be used with UEFI (successor of BIOS)
  - UEFI = Unified Extended Firmware Interface

Boot Sequence

- The computer when powered up starts executing from its ROM
- The ROM is traditionally tiny (not any more)
- The code mainly identifies the disks and other devices and selects where to boot from
  - The device must have at least a valid MBR (or GPT)
- Part of the boot is POST
  - POST = Power On Self Test

Bad Blocks

- Almost every disk has a few bad blocks
  - We use ECC to find out which.
- These are identified and replacements (spares) are used.
- Two techniques are used:
  - Replace the bad block with a spare
    - Neat but results in blocks being out of order
  - Shift all blocks between the bad and the spare by one position
    - Preserves the order but need shifting (copying)

Swap space

- Every frame in the memory can be mapped to the disk in two ways
  - If it is a memory-map from a (named) file (eg from a .so file) it is mapped to the original (named) file
  - All other frames (the anonymous memory) are mapped to the swap space
- The swap space can be of two kinds
  - A partition on the disk
  - A large file created to act as swap space (swapon command)

RAID

- Redundant Array of Independent Disks
- A medium system with 100 disks, if each disk has MTBF 100,000 hours, will fail every 1000 hours (about a month and a half)
- A home system with 2 disks, if each disk has MTBF 100,000 hours, will fail every about 5 years.

RAID for Safety

- Mirroring is the simplest approach
- The system will fail if the second disk fails while we recover the data of the first
  - If MTBF is 100,000h
  - And MTTR is 10h
  - We get one recoverable fail every 50,000h
  - And a fatal fail every 500,000,000h (50 centuries)

RAID for Performance

- We can use stripping
  - If we have 8 disks we store one bit of every byte on different disks. This is bit level stripping.
  - We can increase the (read) transfer rate 8 times
    - The positioning time goes up, since the positioning time of the RAID is the positioning time of the slowest
  - When we write we involve all disks
- We can also have byte level and block level stripping.

RAID Levels

- RAID 0: just stripping
- RAID 1: just mirroring
- RAID 2: ECC redundancy
- RAID 3: bit interleaved parity
- RAID 4: block interleaved parity
- RAID 5: block interleaved distributed parity
- RAID 6: like 5 but can handle 2 errors
- RAID 01 or 10: stripping and mirroring

Implementing RAID

- Hardware (Host Bus Adapter), motherboard, or special disk array
- Kernel based
- SAN

Problems with RAID

- Most disk problems happen around power failures, during boot, natural disasters, etc
  - Ie failures are not statistically independent.
- Disks from the same batch often fail together
- No protection against corruption by faulty software, malware, etc.
- In simple mirrored systems, after a fault we still have to find which is the correct version.

Solaris ZFS

- Uses checksums to ascertain the correctness of each block
- The checksum is stored separately (in the i-node)
- The checksum of the i-node is stored in the directory that contains the file. And so on...
- ZFS also does volume management (allows N filesystems to share M RAID clusters)

Operating Systems

I/O Systems
Based on Ch. 11 of
OS Concepts by SGG

I/O Hardware

- Bus: a set of wires and a communication protocol (electric characteristics, handshake sequence, transmission parameters/encoding, etc)
- A bus could be two wire (half duplex serial bus), or have many wires
- Can be simple and cheap (like I2C for simple projects)
- Can be expensive (like PCIe x32)
- Buses (may) have controllers (special purpose chips) to coordinate the devices connected to them.

Typical buses

- PCIe
  - Can have 1, 2, 4, 8, 12, 16 or 32 lanes (two pairs of wires)
- SAS (Serial Attached SCSI)
- SATA (daughter fo PATA, Parallel AT Attachment)
- USB bus

Memory mapped I/O

- Happens at the hardware level
- Every controller has registers that the CPU can read and write
- The CPU can have physical ports with dedicated port lines (old or simple systems)
- Or the registers could be mapped to addresses in the physical memory

Memory mapped I/O

- Typically there are four registers
  - Data in
  - Data out
  - Status: for the device to send feedback to CPU
  - Control (or command): for the CPU to send commands to the device
- The registers could be several bytes long and have FIFOs attached to them

Polling

- The CPU checks periodically the status of all devices (until device is not busy)
- Then places command on the control register
- Eventually the device will see that the command is available and execute it
- Execution may involve further communication through the data registers.
- In the end the controller indicates to the CPU that the command succeeded/failed.

Interrupts

- The other way to communicate is through interrupts.
- Typical senario:
  - Device controller raises an interrupt
  - CPU catches the interrupts and dispatches it to ISR (or handler) to service it
  - After that the interrupt is cleared and regular CPU things resume

Interrupts

- Interrupts can be deferred or masked (may be not all of them)
- Can have an address with them
- Can have interrupts with varying priorities
- Can have software traps

DMA

- For the exchange of small chunks of data direct communication between CPU and device is OK.
- For large chunks we use DMA
  - CPU writes a DMA control bloc to memory
  - Informs the device through the control register
  - The device informs the CPU through the status register/interrupt.

Operating Systems

File System Interface
Based on Ch. 12 of
OS Concepts by SGG

File Attributes

- Name
- File ID
- Type
- Location
- Size
- Protection
- Time, date, etc
- User

File Attributes on Linux

- A directory contains a list of file-names and their associated inode (or i-node) numbers.
- The inode number is a pointer (integer) to the i-node table.
- Every entry to the table contains information like:
    - Device ID, i-node number, mode and type, number of hardlinks, user/group ID, special file info, size, number of blocks allocated, block size, times,

Extended File Attributes

- Application that created the file
- Icon
- Character encoding
- Checksums
- Security attributes

File Operations

- Read
- Write
- Create
- Delete
- Truncate
- Reposition
- Lock (shared/exclussive, advisory/mandatory)
- To make these more efficient we also have open/close

File types

- These used to be many
- Linux keeps it to a minimum
  - Regular, directory, symbolic link, device, pipe, etc
- Windows has more

Directories

- The filesystem is a reliable name-space
- It is a graph
  - Tree
  - DAG for the adventurous
  - General graph for the suicidal
- Can have hard links to create cycles

Mounting

- We attach disks with filesystems on our namespace
- Windows is simple: C: D: etc
- Linux is very sophisticated
- Can attach on filesystems
  - Kernel structures
  - Remote systems
  - etc

Protection

- In Linux we use *owner*, *group*, *other* to group users
- Read, Write Execute to group actions
- Simple but try to use it to implement Moodle.

FAT

- Stands for “File Allocation Table”
- Developed in 1977
- Flavours: FAT8, FAT12, FAT16, FAT32
- Contains:
  - Boot sector (with partition table)
  - File Allocation table (2 copies)
  - Root directory
  - Data segment (other directories and all files)

FAT

- Every entry of the FAT contains the next block (cluster) of the file.
- Some entries have special meaning (FAT16):
  - 0x0000 available
  - 0xFFF7 damaged block
  - 0xFFF8 end of file.

FAT Sizes

- FAT12:
  - 12 bit/entry, 4096 entries, 4k bytes/cluster-> about 15MB
- FAT16:
  - 16 bit/entry, 64K entries, 2K bytes/cluster -> about 127MB
- FAT16:
  - 16 bit/entry, 64K entries, 8K bytes/cluster -> about 511MB

FAT example

2
4
Avail
EOF
EOF
Avail
Avail
BAD

Myfile->1
Yourfile->5
<empty>
<empty>
<empty>

VFAT

- Successor to FAT/FAT32
- Introduced by Windows 95
- Allows bigger file names, spaces, etc
  - Does this through aliasing to achieve backwards compatibility

EXT2

- The grandfather of ext4 filessytem
  - Ext4 adds journaling
- The inode under ext2 contains:
  - The fstat info
  - Pointers to 12 blocks with the file data
  - A pointer to a block of 128 pointers to data blocks (indirect block)
  - A pointer to a block of 128 pointers to a block of 128 pointers to data blocks (doubly indirect block)
  - A pointer to a .... (trebly indirect block)
- Total 15 pointers
- Number of entries on indirect blocks depends on block size.
- The block size cannot be bigger than the page size.

EXT2

- Directories are special files
- Contain a table of directory entries
- Each entry is a name and an inode number
  - Ext3 allows also a data structure more efficient than a simple table.
- A directory always has these two entries:
  - . and ..
- Uses a data allocation bitmap and tries to allocate blocks in nearby areas of the disk

EXT2

- Simple an efficient. Lack of journaling means fewer writes (good for flash memories)
- Today ext4 (its grandchild) is used
- With 512byte block, 128 pointers per pointer block can have up to about 1GB files.

EXT2 Details

- Blocks belong to block groups
  - Blocks in a group are located nearby (mimize seeks)
  - Blocks in the same group share many pointer bits
  - Makes pointers smaller
- Block groups are described in the superblock
- Every block group contains a copy of the superblock, block group descriptor table, block allocation bitmap, inode table, inode bitmap and data blocks.

EXT2 Sizes

- Depend on number of blocks, size of blocks, number of inodes
- Block size cannot be larger that the page size
- Hard upper limits
  - Max volume size 2-32TB
  - Max file size 16GB-2TB
  - Max filename length 255 char

EXT4

- Currently the default Linux FS
- Successor to ext3
- Handles huge storage volumes, huge files
- Uses extends (ranges of contiguous blocks)
- High res timestamps (nsec)