Lecture 18: Storage Devices

(Storage Methods, Abstraction, Sharing)

Xin Liu

Florida State University
xl24j@fsu.edu

COP 4610 Operating Systems
https://xinliulab.github.io/cop4610.html
November 5, 2024

Overview

Review

- Operating System
  - Manager of machine states
  - Objects + API
  - Abstraction of I/O Devices
    - Device Layer: I/O devices (registers)
    - Driver Layer: readable/writable/controllable objects

Questions Answered in This Class

- **Q1:** How is the current state of the machine stored?
- **Q2:** How are more persistent states stored?
- **Q3:** How do applications share access to storage devices?

Main Topics for This Class

- 1-bit storage methods
- Volatile/non-volatile storage
- Storage device abstraction
- Storage device sharing

Storage Methods

How to Store 1-Bit Data?

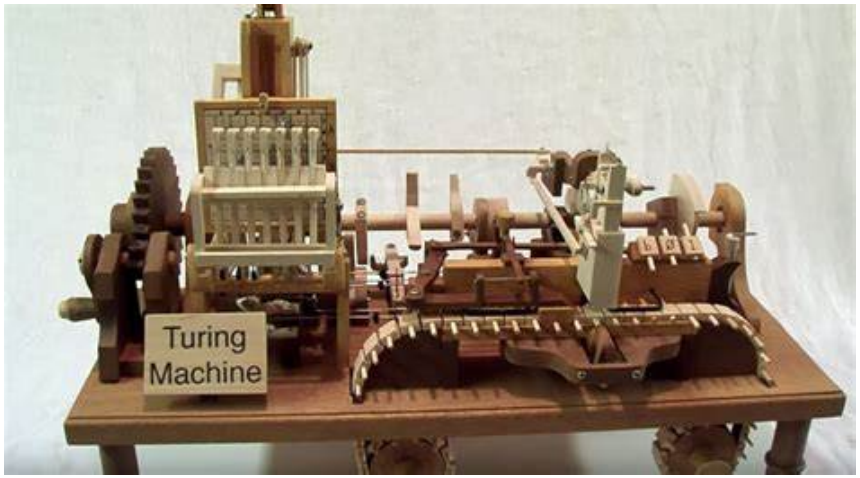
Computers Need to Store the "Current State"

Instruction Set Architecture only has "two" states

- Registers: rax, rbx, ..., cr3, ...
- Physical memory

Requirements for Storing the "Current State"

- Must be addressable
  - Read and write data based on encoding
- Access speed should be as fast as possible
  - Even at the cost of losing state after power-off
  - This is why we have a memory hierarchy

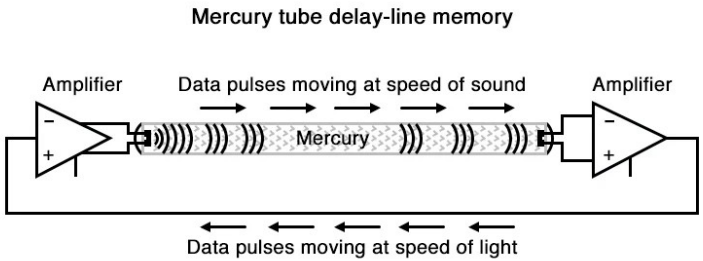


The mechanical Turing machine does not lose state after power-off.

Storage of "Current State"

Delay Line: Acoustic Delay Line

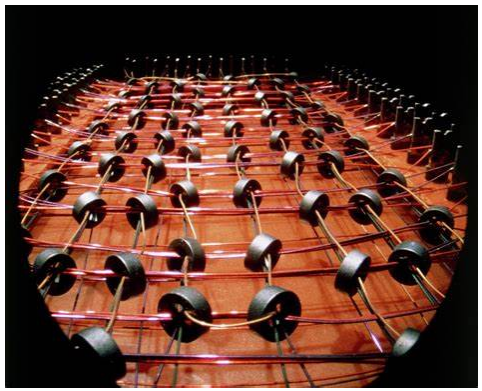
- Due to signal decay, requires continuous amplification



An acoustic delay line used to store data as sound waves. Data pulses travel along the line, and amplifiers boost the signal to counteract decay.

Magnetic Core Memory

- The origin of "Segmentation fault (core dumped)"
- Non-volatile memory!



Magnetic core memory, storing data by the magnetization direction of tiny ferrite cores. Each core represents a single bit, retaining data even when powered off.

SRAM/DRAM: Flip-Flop and Capacitors

- Today's implementation approach

The Journey Towards Persistence

Persistence

"A firm or obstinate continuance in a course of action in spite of difficulty or opposition."

Beyond just the "current state," we want larger and more data to be "retained" (and managed efficiently by the operating system).

The Nature of Persistent Storage

- The foundation of all file structures
  - Conceptually a bit/byte array
  - Managed in blocks, allowing us to read and write in "chunks" according to locality
- Evaluation criteria: cost, capacity, speed, reliability
- Witnessing yet another highlight of human civilization!

Storage Medium: Magnetic

”Persistence” May Not Be As Difficult As Imagined

Magnetic drawing board where the magnetic particles change direction to display drawings. Erased by resetting particles’ direction.

Further Step: Representing 1-Bit Information Using the ”Magnetization Direction” of Iron Particles

- **Read**: Amplify the induced current
- **Write**: Magnetize the iron particles with a magnetic needle

Magnetic Tape (1928)

1D Storage Device

- Rolls up bits
  - Iron magnetic particles evenly coated on the tape
- Requires only one mechanical component (rotation) for positioning

Fritz Pfleumer with the first magnetic tape recording device, 1928

Magnetic Tape: Analysis as a Storage Device

Analysis

- Price
  - Very Low - Made from inexpensive materials
- Capacity
  - Very High
- Read/Write Speed
  - Sequential Access: Strong - Requires waiting for positioning
  - Random Access: Nearly impossible
- Reliability
  - Contains mechanical components, requires stable storage environment

Current Applications

- Storage and backup of cold data

PowerVault LTO-9

$6,839.00

Price Match Guarantee

Financing Offers

Learn More | Pre-Qualify Now

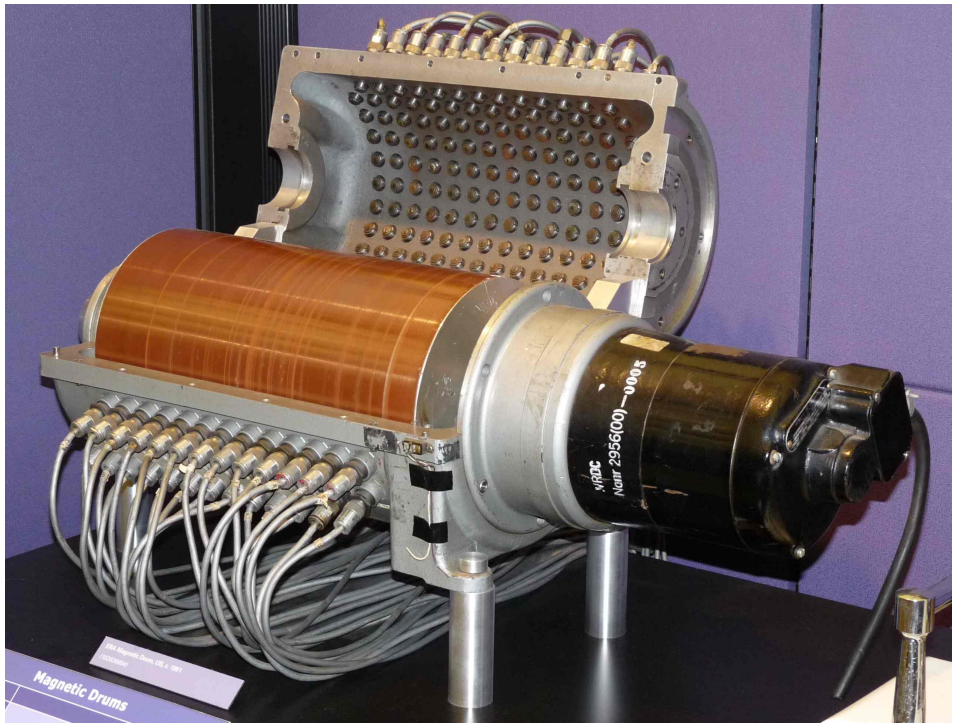
Add to Cart

Example: DELL LTO-9 Magnetic Tape, 18TB Capacity

Magnetic Drum (1932)

1D $\rightarrow$ 1.5D (1D $\times$ n)

- Stores data on a rotating 2D surface
  - No inner roll, limited capacity
- Read/write delay does not exceed the rotation period
  - Significant improvement in random access speed

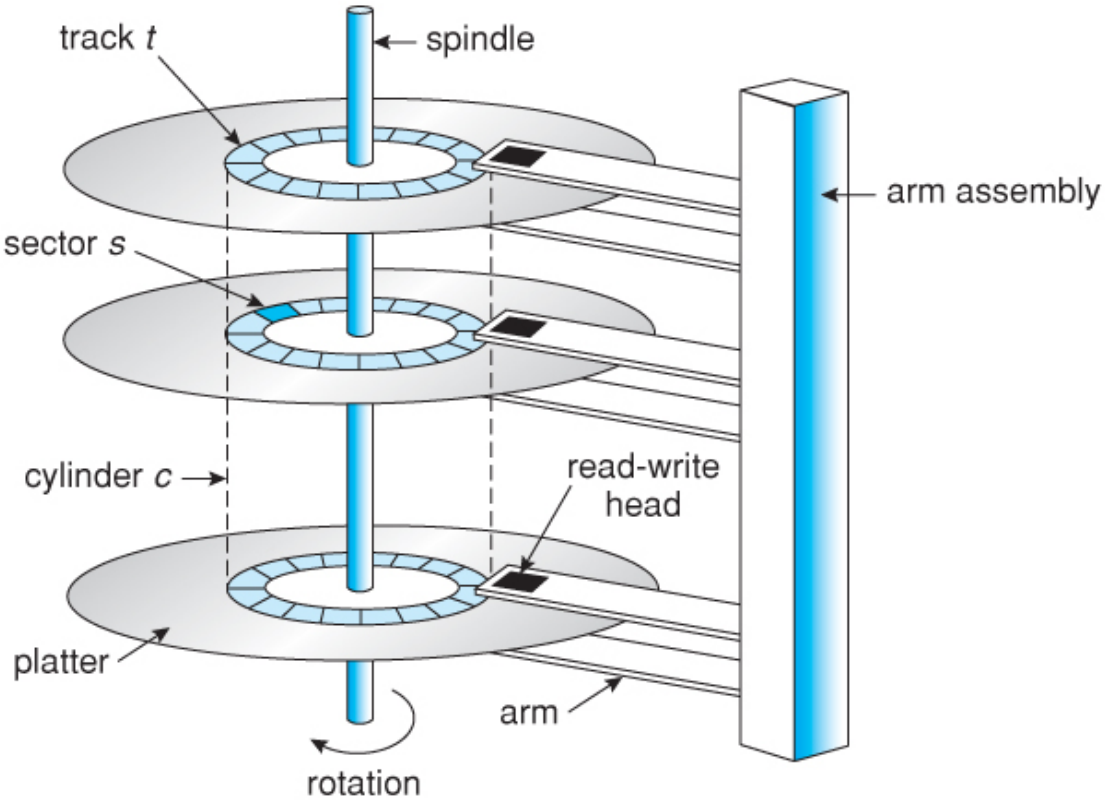


Magnetic Drum, an early form of storage using magnetic coatings on a rotating drum to store data.

Hard Disk (1956)

1D $\rightarrow$ 2.5D (2D $\times$ n)

- Multiple magnetic platters on a two-dimensional plane



Hard Disk structure with multiple platters and a read-write head for each platter, allowing 2D access within each platter and across multiple platters.

Hard Disk: Analysis as a Storage Device

Analysis

- Price
  - Low - Higher density, lower cost
- Capacity
  - High (2.5D) - Tens of thousands of tracks on each platter
- Read/Write Speed
  - Sequential Access: Relatively High
  - Random Access: Strong
- Reliability
  - Mechanical parts present; head crash can damage platters and lead to data loss

Current Applications

- Primary data storage for computer systems (large data volume; low cost is key)

Hard Disk: Performance Optimization

To read/write a sector:

1. The read/write head must reach the correct track
   - 7200 rpm $\rightarrow$ 120 rps $\rightarrow$ "Seek" time of 8.3 ms
2. The spindle rotates the platter to align the sector with the read/write head
   - Head movement time also typically takes several ms

Optimizations through caching/scheduling:

- Examples include the well-known "elevator" scheduling algorithm
- Modern HDDs have advanced firmware to manage disk I/O scheduling
  - `/sys/block/[dev]/queue`
  - `[mq-deadline]` none (priority to reads; writes do not starve)

Note: Modern operating systems no longer handle disk operations directly; everything is managed by the disk controller. Therefore, traditional disk scheduling algorithms are outdated!

Floppy Disk (1971)

Separating the read/write head and the disk enables data movement:

- Floppy disk drive (Break Time!) on computers + removable floppy disk
  - Sizes: 8” (1971), 5.25” (1975), 3.5” (1981)
    - The earliest floppies were very cheap, essentially just a piece of paper
    - The 3.5-inch floppy was made ”hard” to improve reliability

Various sizes of floppy disks, including 8”, 5.25”, and 3.5”.

Floppy Disk: Analysis as a Storage Device

Analysis

- Price
  - Low - Made of plastic, diskette, and some small materials
- Capacity
  - Low (Exposed storage medium, limited density)
- Read/Write Speed
  - Sequential/Random Access: Low
- Reliability
  - Low (Exposed storage medium)

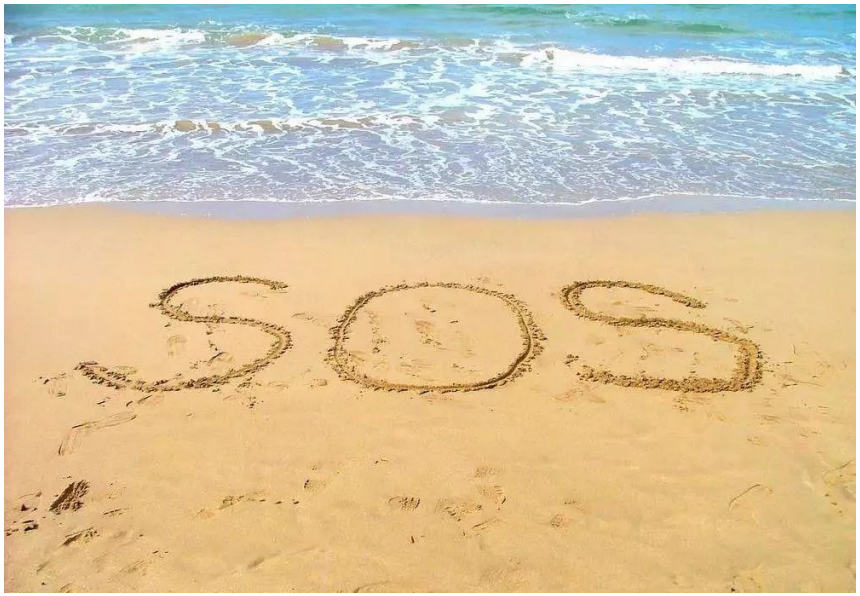
Current Applications

- Displayed in museums for public viewing
- Completely replaced by USB Flash Drives

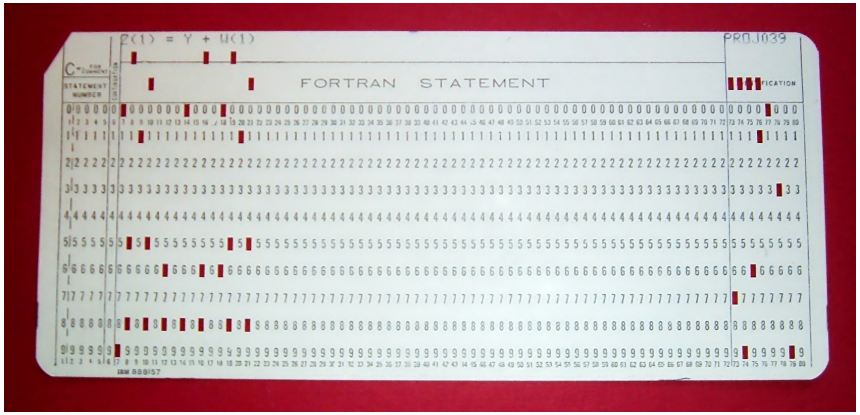
Storage Medium: Pit

Holes: Naturally Readable Data Storage

- Example of data storage that is intuitively easy to ”read”



An ”SOS” message written in the sand, easily readable by humans.



A punch card used for data storage, with holes representing coded information.

Compact Disk (CD, 1980)

- Pits (0) are etched on a reflective surface (1) to store data
- A laser scans the surface to read information from the pits
  - Invented by Philips and Sony (for digital audio)
  - Capacity of 700 MiB, which was very large at the time

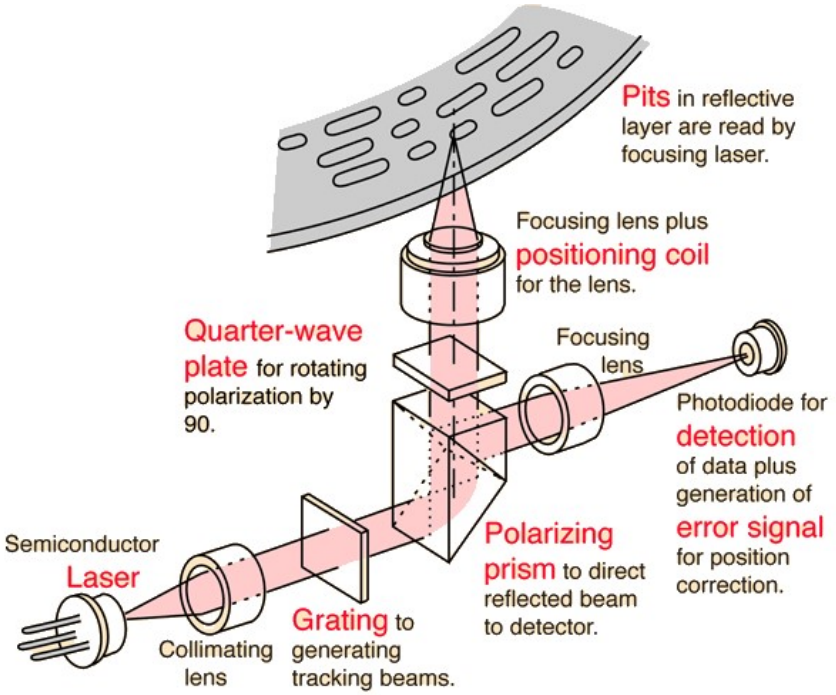


Diagram of how a CD works, with a laser reading pits on the reflective layer.

Close-up view of a CD surface, showing the pattern of pits and lands.

CD-RW

Can the limitation of read-only be overcome?

- Method 1
  - Etch a pit using a laser (”write once”)
  - Use an append-only data structure
- Method 2: Alter the reflective properties of the material
  - PCM (Phase-change Material)
  - How do rewritable CDs work?

Advancements in ”Pit” Technology

- **CD (740 MB)**
  - 780 nm Infrared Laser
- **DVD (4.7 GB)**
  - 635 nm Red Laser
- **Blu-ray (100 GB)**
  - 405 nm Blue-violet Laser

PS5 Drive

Optical Disk: Analysis as a Storage Device

Analysis

- Price
  - Very low (and easy to replicate with ”pressing” technology)
- Capacity
  - High
- Read/Write Speed
  - High sequential read speed; strong random read performance
  - Low write speed (pits are hard to form and fill)
- Reliability
  - High

Current Applications

- Distribution of digital media (increasingly replaced by internet-based ”on-demand” distribution)

Storage Medium: Electrical

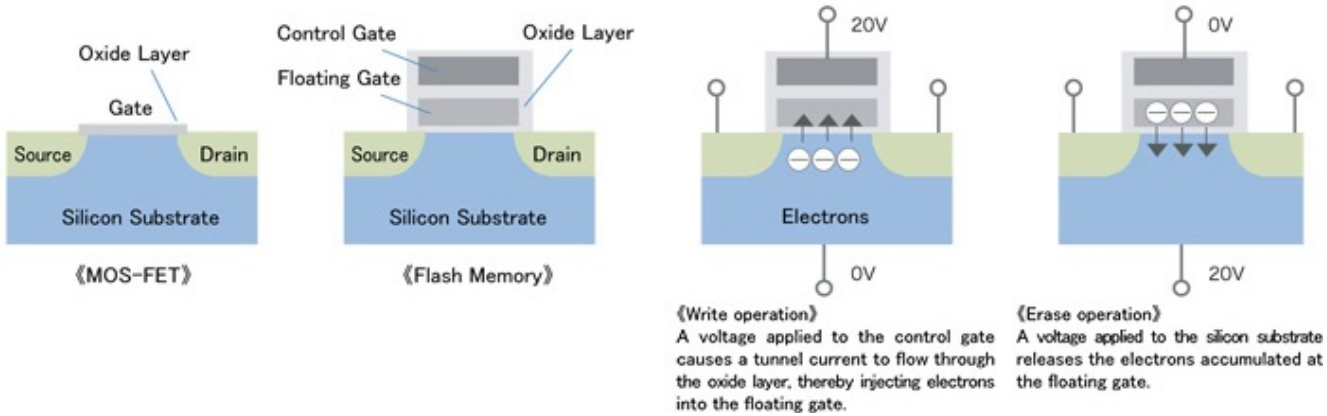
Solid State Drive (1991)

Previous persistent storage mediums had critical drawbacks:

- Magnetic storage: Mechanical components cause delays in the millisecond range
- Optical storage (CDs): Once a pit is created, it is difficult to modify (CDs are read-only)

Eventually, electricity (circuits) provided a solution:

- Flash Memory
  - Floating gate technology stores 1-bit information by charging/discharging



USB Flash Disk (1999)

USB drives have large capacity, fast speed, and are relatively affordable.

- Quickly replaced floppy disks and became a common storage medium
  - Compact Flash (CF, 1994)
  - USB Flash Disk (1999, developed by M-Systems)

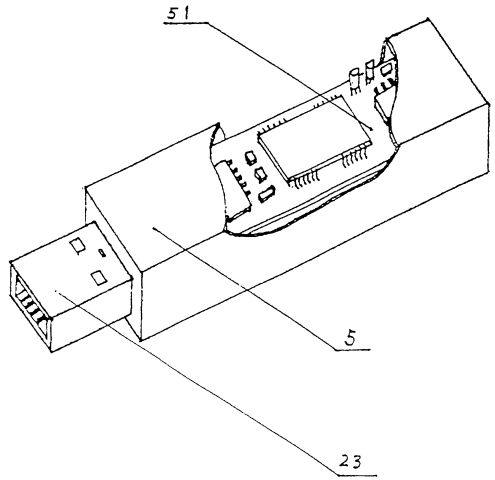


Fig.2

Image from Patent US6829672

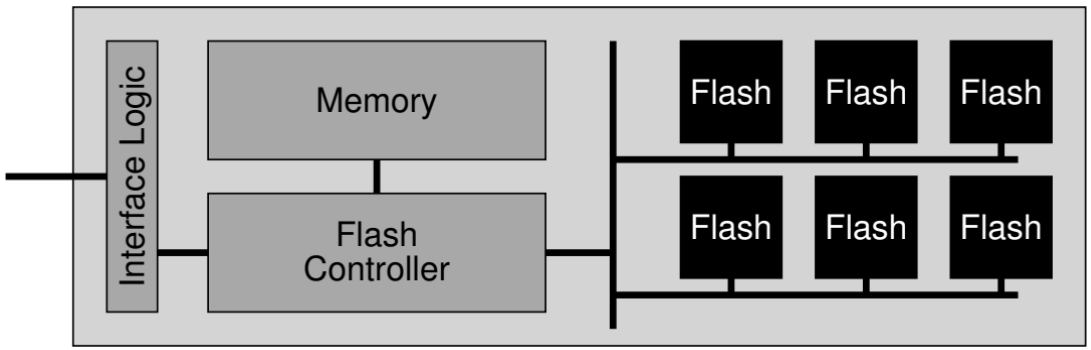
Challenges with Flash Memory:

- Erasing (discharging) cannot completely clear cells.
  - After thousands/millions of erase cycles, cells act ”charged.”
  - Leads to dead cells and ”wear out” issues.
- This issue must be resolved for SSDs to be viable.

Solution to NAND Wear-Out

Software-Defined Storage: Every SSD contains a complete computing system.

- FTL: Flash Translation Layer
  - **Wear Leveling**: Software manages blocks that may become problematic
  - Similar to a managed runtime (with garbage collection)



SSD internal structure with Flash Translation Layer (FTL)

Interfaces of Storage Devices

Common Storage Devices

- **Hard Disk Drive (HDD)**
  - **ATA (PATA):** Primarily used in older HDDs and optical drives; provides parallel data transfer.
  - **SATA:** Replaced PATA for faster serial data transfer; still used in HDDs and some SSDs.
- **Solid State Drive (SSD)**
  - **Interface:** SATA, NVMe (Non-Volatile Memory Express)
  - SATA SSDs use the same interface as HDDs but are faster due to no moving parts.
  - NVMe SSDs, utilizing PCIe (Peripheral Component Interconnect Express), are optimized for high-speed data access.
- **Optical Drives (CD/DVD/Blu-ray)**
  - **Interface:** ATA, SATA for internal drives
  - Usually used for media playback and data storage in a sequential access manner.

Interfaces of Storage Devices (Cont.)

- **USB Flash Drives**
  - **Interface:** USB (Universal Serial Bus)
  - Plug-and-play, portable, used widely for data transfer and temporary storage.
- **Network-attached Storage (NAS)**
  - **Interface:** Ethernet, Wi-Fi
  - Allows data access over a network, suitable for shared storage in a networked environment.

Storage Device Abstraction

Disk Controller and ATA Interface

ATA (Advanced Technology Attachment)

- **IDE (Integrated Drive Electronics) Interface for Disk**
  - **Primary** Channel: `0x1f0 - 0x1f7`
  - **Secondary** Channel: `0x170 - 0x177`
- **Modern OS no longer manages disk scheduling:**
  - The OS only requests specific data blocks through the ATA interface.
  - The internal controller of the disk performs scheduling and optimization independently (e.g., seek optimization), without OS intervention.
- **Role of Device Driver:**
  - Sends commands via the ATA interface, specifying the location of the data block.
  - Does not involve specific scheduling algorithms; only responsible for data request transmission.

```
void readsect(void *dst, int sect) {
   waitdisk();
   out_byte(0x1f2, 1); // sector count (1)
   out_byte(0x1f3, sect); // sector
   out_byte(0x1f4, sect >> 8); // cylinder (low)
   out_byte(0x1f5, sect >> 16); // cylinder (high)
   out_byte(0x1f6, (sect >> 24) | 0xe0); // drive
   out_byte(0x1f7, 0x20); // command (read)
   waitdisk();
   for (int i = 0; i < SECTSIZE / 4; i++)
      ((uint32_t*)dst)[i] = in_long(0x1f0); // data
}
```

Abstraction of Storage Devices

Characteristics of Disk (Storage Device) Access

1. **Access in Data Blocks**
   - Data is accessed in **blocks**, not individual bytes.
   - Transfer has a ”minimum unit size” and does not support arbitrary access.
   - Optimal transfer mode depends on the type of storage (HDD vs. SSD).
2. **High Throughput**
   - Data transfer is managed using **Direct Memory Access (DMA)** for efficient handling.
3. **No Direct Access by Applications**
   - Access is typically managed through the **file system**, which maintains data structures on the disk.
   - Concurrency is handled as multiple processes access the file system simultaneously.

Comparison with Other Devices (e.g., Terminal, GPU)

- **Terminal:** Small data volumes with direct streaming data transfer.
- **GPU:** Large data volumes transferred using DMA, optimized for high throughput.

Linux Block I/O Layer

Interface between the File System and Disk Devices

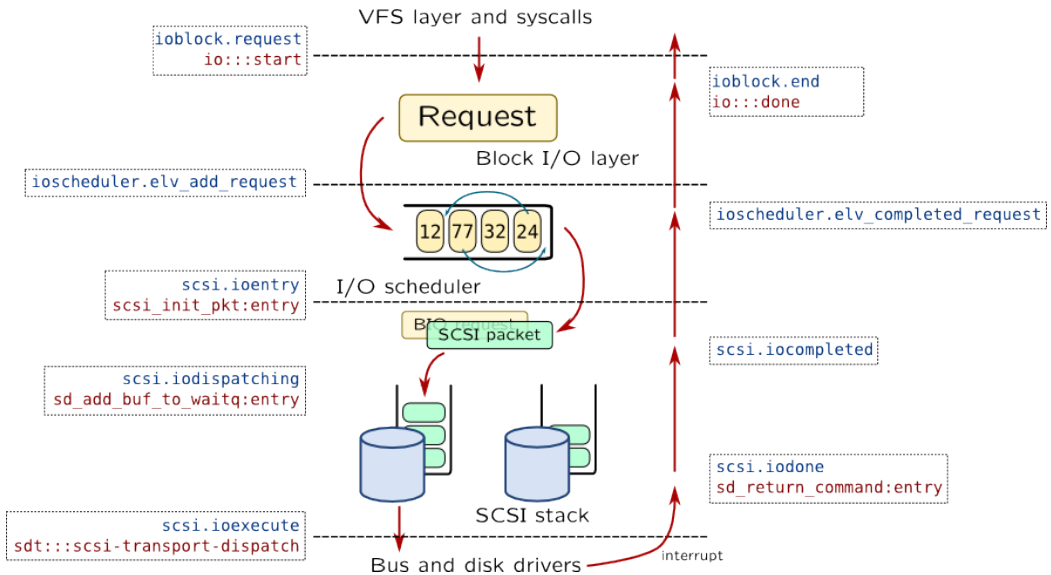


Diagram showing the Linux Block I/O layer, including the I/O scheduler and SCSI stack.

- **Request:** The file system sends a block request to the I/O layer.
- **I/O Scheduler:** Orders requests for optimal performance, previously using methods like the elevator algorithm.
- **SCSI Stack:** Converts requests into SCSI commands and handles communication with disk drivers.
- **Bus and Disk Drivers:** Final layer handling the actual data transfer between the system and the storage device.

Block I/O: The Foundation of Persistence

File System

- Persistent data structures built on top of the Block I/O API

File System Implementation

- `bread`: Read blocks from disk
- `bwrite`: Write blocks to disk
- `bflush`: Flush block cache to disk
- Supports file and directory operations

Why do we need file system?

Sharing Devices Among Applications

Terminal

- Multiple processes printing concurrently – how to ensure no mix-up? (see `man 2 printf`)
- Multiple processes reading concurrently can lead to conflicts.
  - Race condition – who wins, who loses (sometimes acceptable).
  - Background processes may receive `SIGTTIN` when trying to read from the terminal (RTFM).

GPU (CUDA)

- Each CUDA application is a series of CUDA API calls.
  - `cudaMemcpy`, kernel calls, etc.
- All scheduling and isolation are handled by the device driver.
  - Kernels must wait for available thread warps to execute, and control is returned upon completion.

Sharing Devices Among Applications (Cont.)

Persistent Data on Disk

- **Program Data**
  - Executable files and dynamic libraries
  - Application data (high-resolution images, cutscenes, 3D models, etc.)
- **User Data**
  - Documents, downloads, screenshots, replay files, etc.
- **System Data**
  - Manpages
  - System configuration files

A byte sequence is not an ideal abstraction for disks.

- Should all applications share a disk? A single program bug could compromise the entire operating system.

File System: Virtual Disk

File System: Design Goals

1. Provide a reasonable API for multiple applications to share data
2. Offer some level of isolation to ensure that malicious or erroneous programs cannot cause widespread harm

Virtualization of "Storage Device (Byte Sequence)"

- Disk (I/O device) = a readable/writable byte sequence
- Virtual Disk (file) = a dynamically readable/writable byte sequence
  - **Name Management**
    - Naming, indexing, and traversal of virtual disks
  - **Data Management**
    - `std::vector<char>` (random access/write/resize)

Takeaways

Questions Answered in This Lecture

- **Q**: How is the ”current state” of a state machine and persistent state stored?

Take-away Messages

- **1-Bit Information Storage**
  - Magnetic media (tape, disk), pits (optical disk), and electrical (Flash SSD)
  - Different types of storage devices with varied characteristics
- **Rethinking ”Storage of State”**
  - With NVM (Non-Volatile Memory), the state of main memory remains intact without power loss
    - Note: Cache/buffer storage is still volatile