Logistics

• Lab 1: Unix utilities, system calls, & C
  • Due Thursday
  • Anyone stuck?
  • Questions about C/system calls?
    • `man 2 read`
Last Class

• I/O Devices
  • Physical Interfaces
  • Device Drivers vs. Firmware
  • Polling vs. Interrupts
• Big Picture: Memory Hierarchy / Layers

This Class

• HDD Guarantees
  • Performance & correctness
• Physical components and Geometry
  • Breaking down an I/O
• The role of caching
• Scheduling requests
Hard Disk Drives (HDDs)

- High capacity, low cost
- Predictable performance

- “Unwritten contract”: Tracks (LBAs) near each other are more efficient to access than tracks (LBAs) that are far away
HDDs

- Disk Head (seeks in/out)
- Platters (rotate)
- Tracks (concentric circles)

HDDs

- Disk Head (seeks in/out)
- Sector (unit of transfer)
- Platters (rotate)
- Tracks (concentric circles)
“Unwind” The Tracks

Seeking through the Linear Address Space
HDDs

- Disks are addressed by **LBA**: \([0, \text{MAX}_{\text{LBA}}]\)

- Transfer data in fixed-size units: “disk block”

  - “block interface” used for both reads and writes
Breaking Down an I/O

• Two costs to every operation:
  • **Setup**: Moving the disk head, rotating the platters
  • **Transfer**: Reading/writing while the disk rotates

  Ex: `data <- read(10024, 10048)`

Performance Observations

• **Setup** (placing the disk head) is expensive \(O(10 \text{ ms})\)
  • seeking to target track
  • Up to a full rotational delay to locate sector

• Once the disk head is in place, data **transfer** is quite fast \(O(100 \text{ MiB/s})\)
Why Does This Matter?

To maximize performance, minimize seeks and maximize the ratio of time spent transferring.
**Simplified Storage Stack**

Data flow:
- Application (user space)
- File System (OS kernel)
- HDD

Data operations:
- \(data = \text{read}(LBA)\)
- \(\text{write}(data, LBA)\)

**Good Cases**

**Sequential I/O**
- Write a large file to an empty file system.
- Read an existing file in order

*higher is better*
Good Cases

- Write a large file to an empty file system.

Good Cases

- Read an existing file in order
Bad Cases

Random I/O

- Randomly update an existing file
- Randomly reading an existing file
- Reading data from many independent files

Bad Cases

- Randomly update an existing file
**Takeaway:**  
*Locality* Matters

**Disk Geometry**

- High level idea gets us most of the way, but disk geometry adds complications (*opportunities?*)
  - Multi-zoned disks
  - Track Skew
**Takeaway:**
Sometimes it pays to “open the black box”. Abstractions are important, but they hide important details.

**Scheduling**

- **High Level Question:** You are given a series of requests that must be completed (LBAs), what order do you perform the work?
- Obstacles?
- Who does the scheduling?
Scheduling

- Greedy: **Shortest job first**
  - Shortest-seek-time-first (SSTF)
  - Nearest-block-first (NBF)
- Problems?
  - **Starvation**: one (or more) requests never receive access to the resources they need to complete

Scheduling

- Elevator!
Any Questions?

HDD Handout
(15-20 minutes)
Activity: HDD Modeling
https://github.com/williams-cs/cs333-class