Disks: Structure and Scheduling

References: Operating Systems Concepts (9e), Linux Kernel Development, previous W4118s

Copyright notice: care has been taken to use only those web images deemed by the instructor to be in the public domain. If you see a copyrighted image on any slide and are the copyright owner, please contact the instructor. It will be removed.
Disk Structure

- Range from 30GB to 3TB per drive
- Aluminum platters with magnetic coating
- Commonly, 2-5 platters per drive
- Common platter sizes: 3.5”, 2.5”, and 1.8”
- Magnetic heads
The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second
Disk Interface

• From FS perspective: disk is addressed as a one dimension array of logical sectors

• Disk controller maps logical sector to physical sector identified by track #, surface #, and sector #

• Note: Old drives allowed direct C/H/S (cylinder/head/sector) addressing by OS. Modern drives export LBA (logical block address) and do the mapping to C/H/S internally.
Disk Latencies

• **Rotational delay**: rotate disk to get to the right sector

• **Seek time**: move disk arm to get to the right track

• **Transfer time**: get bits off the disk
Seek Time

- Must move arm to the right track
- Can take a while (e.g., 5–10ms)
  - Acceleration, coasting, settling (can be significant, e.g., 2ms)
Transfer Time

- Transfer bits out of disk
- Actually pretty fast (e.g., 125MB/s)
I/O Time (T) and Rate (R)

• T = Rotational delay + seek time + txfer time

• R = Size of transfer / T

• Workload 1: large sequential accesses?

• Workload 2: small random accesses?
## Example: I/O Time and Rate

<table>
<thead>
<tr>
<th></th>
<th>Barracuda</th>
<th>Cheetah 15K.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>1TB</td>
<td>300GB</td>
</tr>
<tr>
<td>Rotational speed</td>
<td>7200 RPM</td>
<td>15000 RPM</td>
</tr>
<tr>
<td>Rotational latency (ms)</td>
<td>4.2</td>
<td>2.0</td>
</tr>
<tr>
<td>Avg seek (ms)</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Max Transfer</td>
<td>105 MB/s</td>
<td>125 MB/s</td>
</tr>
<tr>
<td>Platters</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Connects via</td>
<td>SATA</td>
<td>SCSI</td>
</tr>
</tbody>
</table>

- **Random 4KB read**
  - **Barracuda**: $T = 13.2\text{ms}, R = 0.31\text{MB/s}$
  - **Cheetah**: $T = 6\text{ms}, R = 0.66\text{MB/s}$

- **Sequential 100 MB read**
  - **Barracuda**: $T = 950\text{ms}, R = 105 \text{ MB/s}$
  - **Cheetah**: $T = 800\text{ms}, R = 125 \text{ MB/s}$
Design tip: Use Disks Sequentially!

- Disk performance differs by a factor of 200 or 300 for random v.s. sequential accesses

- When possible, access disks sequentially
Mapping of logical sectors to physical

- Logical sector 0: the first sector of the first (outermost) track of the first surface
- Logical sector address incremented within track, then tracks within cylinder, then across cylinders, from outermost to innermost
- Track skew
Parallel Reading from Heads

• All heads should point to same place on track
  – Why not read from all heads in parallel?

• Need perfectly aligned heads
  – Hard to do in practice
  – Heads not perfectly aligned because of
    • Mechanical vibrations
    • Thermal gradients
    • Mechanical imperfections
  – High density makes problem worse
  – Needs high throughput read/write circuitry

• Consequence: most drives have a single active head at a time
Pros and cons of default mapping

• Pros
  – Simple to program
  – Default mapping reduces seek time for sequential access

• Cons
  – FS can’t precisely see mapping
  – Reverse-engineer mapping in OS is difficult
    • # of sectors per track changes
    • Disk silently remaps bad sectors
Disk cache

- Internal memory (8MB-32MB) used as cache

- Read-ahead: “track buffer”
  - Read contents of entire track into memory during rotational delay

- Write caching with volatile memory
  - Write back or immediate reporting: claim written to disk when not
    - Faster, but data could be lost on power failure
  - Write through: ack after data written to platter
Disk scheduling

• Goal: minimize positioning time
  – Performed by both OS and disk itself
  – Why?

• OS can control:
  – Sequence of workload requests

• Disk knows:
  – Geometry, accurate positioning times
Schedule requests in order received (FCFS)

**Advantage:** fair

**Disadvantage:** high seek cost and rotation

Total head movement of 640 cylinders
SSTF: Shortest Seek Time First

- Shortest seek time first (SSTF):
  - Form of Shortest Job First (SJF) scheduling
  - Handle nearest cylinder next
  - Advantage: reduces arm movement (seek time)
  - Disadvantage: unfair, can \textit{starve} some requests

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

Total head movement of 236 cylinders
Elevator (aka SCAN or C-SCAN)

• Disk arm **sweeps** across disk
• If request comes for a block already serviced in this sweep, queue it for next sweep
SCAN (Elevator) Disk Scheduling

Make up and down passes across all cylinders

Pros: efficient, simple

Cons: Unfair. Oldest requests (furthest away) also wait longest.

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

Total head movement of 208 cylinders
C-SCAN

• Provides a more uniform wait time than SCAN

• The head moves from one end of the disk to the other, servicing requests as it goes
  – When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip

• Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

• Total number of cylinders?

C-SCAN (Elevator) Scheduling

Head reads in one direction only
Wrap around without reading when end is reached (like circular linked list)
Provides a more uniform wait time than SCAN

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
C-LOOK Scheduling

- In practice, don’t need to scan to ends of disk
- Wrap around when no more requests
- Wrap to earliest outstanding request

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
Modern disk scheduling issues

- Elevator (or SSTF) ignores rotation!
- **Shortest positioning time first (SPTF)**
- OS + disk work together to implement
I/O Scheduling in Practice

• Simple: Linus Elevator scheduler
  – Default until 2.4
  – Variant of C-LOOK algorithm
  – Merge new request with existing where possible
  – Otherwise, insert in sorted order between existing requests
  – If no suitable location found, insert at queue tail

• In practice situation is more complicated due to
  – Interactions with filesystem (data layout)
  – Interactions with caches (write-back vs. write-through)
  – Write and read requests have different priority
  – Delay sensitive applications such as multimedia
  – Additional algorithms: Deadline, Completely Fair Queuing (CFQ)

• Will look in a bit more depth later
Disk technology trends

• Data ➔ more dense
  – More bits per square inch
  – Disk head closer to surface
  – Create smaller disk with same capacity

• Disk geometry ➔ smaller
  – Spin faster ➔ Increase b/w, reduce rotational delay
  – Faster seek
  – Lighter weight

• Disk price ➔ cheaper

• Density improving more than speed (mechanical limitations)
New mass storage technologies

- New memory-based mass storage technologies avoid seek time and rotational delay
  - No moving parts means more reliable, shock resistant
  - NAND Flash: ubiquitous in mobile devices
  - Battery-backed DRAM (NVRAM)

- Disadvantages
  - Price: more expensive than same capacity disk
  - Reliability: more likely to lose data
  - Other significant quirk: can’t rewrite easily

- Open research question: how to effectively use flash in commercial storage systems

- Will look in more depth later