Disks and RAID

CS 4410
Operating Systems

[R. Agarwal, L. Alvisi, A. Bracy, F. Schneider, E. Sirer, R. Van Renesse]
Storage Devices

• Magnetic disks
  • Large capacity at low cost
  • Block level random access
  • Slow performance for random access
  • Good performance for streaming access

• Flash memory
  • Capacity at intermediate cost
  • Block level random access
  • Medium performance for random writes
  • Good performance otherwise
Magnetic Disks are 60 years old!

**THAT WAS THEN**
- 13th September 1956
- The IBM RAMAC 350
- Total Storage = 5 million characters (just under 5 MB)

**THIS IS NOW**
- 2.5-3.5” hard drive
- Example: 500GB Western Digital Scorpio Blue hard drive
- easily up to a few TB

Reading from disk

Must specify:
• cylinder #
  (distance from spindle)
• head #
• sector #
• transfer size
• memory address
Disk Latency = \textbf{Seek Time} + \textbf{Rotation Time} + Transfer Time

- **Seek**: to get to the track (5-15 milliseconds)
- **Rotational Latency**: to get to the sector (4-8 milliseconds)
  (on average, only need to wait half a rotation)
- **Transfer**: get bits off the disk (25-50 microseconds)
Disk Scheduling

Objective: minimize seek time

Context: a queue of cylinder numbers (#0-199)

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67

Metric: how many cylinders traversed?
Disk Scheduling: **FIFO**

- Schedule disk operations in order they arrive
- Downsides?

**FIFO Schedule?**
**Total head movement?**

<table>
<thead>
<tr>
<th>Head pointer @ 53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue: 98, 183, 37, 122, 14, 124, 65, 67</td>
</tr>
</tbody>
</table>
Disk Scheduling: **FIFO**

- Schedule disk operations in order they arrive
- Downsides?

**FIFO Schedule?**

**Total head movement?**

---

Head pointer @ 53

Queue: 98, 183, 37, 122, 14, 124, 65, 67

640 cylinders
Disk Scheduling: **Shortest Seek Time First**

- Select request with minimum seek time from current head position
- A form of Shortest Job First (SJF) scheduling
- Not optimal: suppose cluster of requests at far end of disk $\rightarrow$ starvation!

**SSTF Schedule?**
**Total head movement?**

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67
Disk Scheduling: **Shortest Seek Time First**

- Select request with minimum seek time from current head position
- A form of Shortest Job First (SJF) scheduling
- Not optimal: suppose cluster of requests at far end of disk → starvation!

**SSTF Schedule?**
**Total head movement?**

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67

236 cylinders
Disk Scheduling: SCAN

Elevator Algorithm:
- arm starts at one end of disk
- moves to other end, servicing requests
- movement reversed @ end of disk
- repeat

SCAN Schedule?
Total head movement?

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67
Disk Scheduling: SCAN

Elevator Algorithm:
- arm starts at one end of disk
- moves to other end, servicing requests
- movement reversed @ end of disk
- repeat

SCAN Schedule?
Total head movement?

208 cylinders

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67
Disk Scheduling: C-SCAN

Circular list treatment:
- head moves from one end to other
- servicing requests as it goes
- reaches the end, returns to beginning
- no requests serviced on return trip

More uniform wait time than SCAN

C-SCAN Schedule?
Total Head movement?

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67
Disk Scheduling: C-SCAN

Circular list treatment:
• head moves from one end to other
• servicing requests as it goes
• reaches the end, returns to beginning
• no requests serviced on return trip
+ More uniform wait time than SCAN

C-SCAN Schedule?
Total Head movement?

Head pointer @ 53
Queue: 98, 183, 37, 122, 14, 124, 65, 67
Disk Failure Cases

(1) Isolated Disk Sectors (1+ sectors down, rest OK)

**Permanent:** physical malfunction (magnetic coating, scratches, contaminants)

**Transient:** data corrupted but new data can be successfully written to / read from sector

(2) Entire Device Failure

- Damage to disk head, electronic failure, wear out
- Detected by device driver, accesses return error codes
- Annual failure rates or Mean Time To Failure (MTTF)
Solid State Drives (Flash)

Most SSDs based on NAND-flash

- retains its state for years without power

Metal Oxide Semiconductor Field Effect Transistor (MOSFET)

Floating Gate MOSFET (FGMOS)

https://flashdba.com/2015/01/09/understanding-flash-floating-gates-and-wear/
Flash Operations

• **Erase block:** sets each cell to “1”
  • erase granularity = “erasure block” = 128-512 KB
  • time: several ms

• **Write page:** can only write erased pages
  • write granularity = 1 page = 2-4KBytes

• **Read page:**
  • read granularity = 1 page = 2-4KBytes
Flash Limitations

• can’t overwrite individual pages (must write blocks)
• limited # of erase cycles per block (memory wear)
  • $10^3$-10$^6$ erases and the cell wears out
  • reads can “disturb” nearby words and overwrite them with garbage

• **Lots of techniques to compensate:**
  • error correcting codes
  • bad page/erasure block management
  • wear leveling: trying to distribute erasures across the entire driver
Flash Translation Layer

Flash device firmware maps logical page # to a physical location

- Garbage collect erasure block by copying live pages to new location, then erase
  - More efficient if blocks stored at same time are kept together
- Wear-leveling: only write each physical page a limited number of times
- Remap pages that no longer work (sector sparing)

Transparent to the device user
## RAM (Memory) vs. HDD (Disk) vs. SSD, 2020

<table>
<thead>
<tr>
<th></th>
<th>RAM</th>
<th>HDD</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical Size</td>
<td>100 GB</td>
<td>1 TB</td>
<td>1TB</td>
</tr>
<tr>
<td>Cost</td>
<td>$5-10 per GB</td>
<td>$0.05 per GB</td>
<td>$0.10 per GB</td>
</tr>
<tr>
<td>Latency</td>
<td>15 ns</td>
<td>15 ms</td>
<td>1ms</td>
</tr>
<tr>
<td>Throughput (Sequential)</td>
<td>8000 MB/s</td>
<td>175 MB/s</td>
<td>500 MB/s</td>
</tr>
<tr>
<td>Power Reliance</td>
<td>volatile</td>
<td>non-volatile</td>
<td>non-volatile</td>
</tr>
</tbody>
</table>
What do we want from storage?

- **Fast:** data is there when you want it
- **Reliable:** data fetched is what you stored
- **Affordable:** won’t break the bank

Enter: **Redundant Array of Inexpensive Disks (RAID)**

- In industry, “I” is for “Independent”
- The alternative is SLED, single large expensive disk
- RAID + RAID controller looks just like SLED to computer (*yay, abstraction!*).
RAID-0

Files striped across disks

+ Fast
  latency?
  throughput?

+ Cheap
  capacity?

– Unreliable
  max #failures?
  MTTF?

Disk 0

- stripe 0
- stripe 2
- stripe 4
- stripe 6
- stripe 8
- stripe 10
- stripe 12
- stripe 14

Disk 1

- stripe 1
- stripe 3
- stripe 5
- stripe 7
- stripe 9
- stripe 11
- stripe 13
- stripe 15
Striping and Reliability

Striping reduces reliability

• More disks → higher probability of some disk failing
• $N$ disks: $1/N^{th}$ mean time between failures of 1 disk

What can we do to improve Disk Reliability?
RAID-1

Disks Mirrored:
data written in 2 places

+ Reliable
  deals well with disk loss
  but not corruption

+ Fast
  latency?
  throughput?

– Expensive
RAID-4 (rarely used)

*block*-level striping + parity disk

+ Cheap

– Slow Writes

– Unreliable

parity disk is write bottleneck and wears out faster
Using a parity disk

- $D_N = D_1 \oplus D_2 \oplus \ldots \oplus D_{N-1}$
- $\oplus = \text{XOR operation}$
- If one of $D_1 \ldots D_{N-1}$ fails, we can reconstruct its data by XOR-ing all the remaining drives
Updating a block in RAID-4

• Suppose block lives on disk $D_1$
• Method 1:
  • read corresponding blocks on $D_2 \ldots D_{N-1}$
  • XOR all with new content of block
  • write disk $D_1$ and $D_N$ in parallel
• Method 2:
  • read $D_1$ (old content) and $D_N$
  • XOR both with new content of block
  • write disk $D_1$ and $D_N$ in parallel
• Note that in both write cases $D_N$ must always be updated
  $\Rightarrow D_N$ is a write performance bottleneck
• Either way:
  • throughput: $\frac{1}{2}$ of single disk
  • latency: double of single disk
Streaming update in RAID-4

- Save up updates to stripe across $D_1 \ldots D_{N-1}$
- Compute $D_N = D_1 \oplus D_2 \oplus \ldots \oplus D_{N-1}$
- Write $D_1 \ldots D_N$ in parallel
- $(N-1) \times$ seq. throughput of single disk
RAID 5: Rotating Parity w/Striping

+ Reliable
  
you can lose one disk

+ Fast
  
$(N-1)x$ seq. throughput of single disk

$N/4x$ random write throughput

+ Affordable

![Disk Diagram]

- Disk 0:
  - parity 0-3
  - data 4
  - data 8
  - data 12
  - data 16

- Disk 1:
  - data 0
  - parity 4-7
  - data 9
  - data 13
  - data 17

- Disk 2:
  - data 1
  - data 5
  - parity 8-11
  - data 14
  - data 18

- Disk 3:
  - data 2
  - data 6
  - data 10
  - parity 12-15
  - data 19

- Disk 4:
  - data 3
  - data 7
  - data 11
  - data 15
  - parity 16-19