A Computing Utility

Must support

– information processing
– information storage
– information communication
A Computing Utility

Must support

– information processing
  ✓ processor multiplexing
  ✓ memory multiplexing

– information storage
  • devices
  • abstractions
    » files
    » databases

– information communication
Permanent Storage Devices

• Magnetic disks
  • Storage that rarely becomes corrupted
  • Large capacity at low cost
  • Block level random access
    • Slow performance for random access
    • Better performance for streaming access

• Flash memory
  • Storage that rarely becomes corrupted
  • Capacity at intermediate cost (50x disk)
  • Block level random access
    • Good performance for reads;
    • Worse for random writes
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 1 TB

# RAM (Memory) vs. HDD (Disk), 2018

<table>
<thead>
<tr>
<th></th>
<th>RAM</th>
<th>HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Typical Size</strong></td>
<td>8 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>$10 per GB</td>
<td>$0.05 per GB</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>3 W</td>
<td>2.5 W</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>15 ns</td>
<td>15 ms</td>
</tr>
<tr>
<td><strong>Throughput (Sequential)</strong></td>
<td>8000 MB/s</td>
<td>175 MB/s</td>
</tr>
<tr>
<td><strong>Read/Write Granularity</strong></td>
<td>word</td>
<td>sector</td>
</tr>
<tr>
<td><strong>Power Reliance</strong></td>
<td>volatile</td>
<td>non-volatile</td>
</tr>
</tbody>
</table>

[C. Tan, buildcomputers.net, codecapsule.com, crucial.com, wikipedia]
Disk operations

Must specify:
• cylinder #  (distance from spindle)
• head #
• sector #
• transfer size
• memory address

Operations:
• seek
• read
• write
Disk Tracks

~ 1 micron wide (1000 nm)
  • Wavelength of light is ~ 0.5 micron
  • Resolution of human eye: 50 microns
  • 100K tracks on a typical 2.5” disk

Track length varies across disk
  • Outside:
    – More sectors per track
    – Higher bandwidth
  • Most of disk area in outer regions

*not to scale: head is actually much bigger than a track
Disk Operation Overhead

*Disk Latency* = **Seek Time** + **Rotation Time** + **Transfer Time**

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

Allows sequential transfer to continue after seek.

Figure 37.4: Three Tracks: Track Skew Of 2

Sectors are often skewed like this because when switching from one track to another, the disk needs time to reposition the head (event on eight-boring tracks). Without such skew, the head would be moved to the next track but the desired next block would have already rotated underneath the head, and thus the drive would have to wait almost the entire rotational delay to access the next block.

Another reality is that outer tracks tend to have more sectors than inner tracks, which is a result of geometry; there is simply more room out there. These tracks are often referred to as multi-zoned disk drives, where the disk is organized into multiple zones, and where a zone is consecutive set of tracks on a surface. Each zone has the same number of sectors per track, and outer zones have more sectors than inner zones.

Finally, an important part of any modern disk drive is its cache, for historical reasons sometimes called a track buffer. This cache is just some small amount of memory (usually around 8 or 16 MB) which the drive can use to hold data read from or written to the disk. For example, when reading a sector from the disk, the drive might decide to read in all of the sectors on that track and cache them in its memory; doing so allows the drive to quickly respond to any subsequent requests to the same track.

On writes, the drive has a choice: should it acknowledge the write as completed when it has put the data in its memory, or after the writes actually been written to disk? The former is called write back caching (or sometimes immediate reporting), and the latter write through. Write back caching sometimes makes the drive appear “faster”, but can be dangerous; if the file system or applications require that data be written to disk in a certain order for correctness, write-back caching can lead to problems (read the chapter on file-system journaling for details).
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?**

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
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
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)
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

Redundant Array of Inexpensive Disks
- small, slower disks are cheaper
- parallelism is free.

Benefits of RAID
- cost
- capacity
- reliability
RAID-0: Simple Striping

Chunk size: number of consecutive blocks on a disk.
RAID-0: Simple Striping

Chunk size: number of consecutive blocks on a disk.

- **disk 0**: block 0, block 4, block 8, block 12, block 16, block 20, block 24, block 28
- **disk 1**: block 1, block 5, block 9, block 13, block 17, block 21, block 25, block 29
- **disk 2**: block 2, block 6, block 10, block 14, block 18, block 22, block 26, block 30
- **disk 3**: block 3, block 7, block 11, block 15, block 19, block 23, block 27, block 31
RAID-0: Simple Striping

Chunk size: 2

disk 0
- block 0
- block 1
- block 8
- block 9
- block 16
- block 17
- block 24
- block 25

disk 1
- block 2
- block 3
- block 10
- block 11
- block 18
- block 19
- block 26
- block 27

disk 2
- block 4
- block 5
- block 12
- block 13
- block 20
- block 21
- block 28
- block 29

disk 3
- block 6
- block 7
- block 14
- block 15
- block 22
- block 23
- block 30
- block 31
Striping and Reliability

Striping reduces reliability

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

How to improve Disk Reliability?
Each block is stored on 2 separate disks. Read either copy; write both copies (in parallel)
RAID-4: Parity for Errors

Parity block for each stripe – saves space. Read block; write full stripe (including parity)
How to Compute Parity

Parity $P( Bi, Bj, Bk)$: $\text{XOR}( Bi, Bj, Bk)$

... keeps an even number of 1’s in each stripe

$\text{XOR}(0,0)=0 \quad \text{XOR}(0,1)=1 \quad \text{XOR}(1,0)=1 \quad \text{XOR}(1,1)=0$

Thm: $\text{XOR}( Bj, Bk, P( Bi, Bj, Bk )) = Bi$
How to Update Parity

Two approaches:

1. Read all blocks in stripe and recompute

2. Use subtraction
   - Given data blocks: Bold, Bnew
   - Given parity block: Pold

Thm: \( P_{\text{new}} := \text{XOR}(\text{Bold}, \text{Bnew}, \text{Pold}) \)

Note: Parity disk becomes bottleneck.
Parity Block by Subtraction

Thm: \( P_{\text{new}} := \text{XOR}( \text{Bold}, \text{B}_{\text{new}}, \text{P}_{\text{old}}) \)

\[
\text{XOR}(\text{Bold}, \text{B}_{\text{new}}, \text{P}_{\text{old}})
= [\text{defn of } \text{P}_{\text{old}}]
\]

\[
\text{XOR}(\text{Bold}, \text{B}_{\text{new}}, \text{B}_1,\text{B}_2, \ldots \text{ Bold}, \ldots, \text{B}_n)
= [\text{XOR is commutative}]
\]

\[
\text{XOR}(\text{B}_{\text{new}}, \text{Bold}, \text{Bold}, \text{B}_1,\text{B}_2,\ldots, \text{B}_n)
= [\text{XOR(A,A)=0}]
\]

\[
\text{XOR}(\text{B}_{\text{new}},0,\text{B}_1,\text{B}_2,\ldots \text{ B}_n)
= [\text{XOR(A,0)=A, XOR is associative}]
\]

\[
\text{XOR}(\text{B}_{\text{new}},\text{B}_1,\text{B}_2,\ldots \text{ B}_n)
= [\text{XOR is commutative}]
\]

\[
\text{XOR}(\text{B}_1,\text{B}_2,\ldots, \text{B}_{\text{new}}, \ldots \text{B}_n)
= [\text{defn of } \text{P}_{\text{new}}]
\]

\( P_{\text{new}} \)
RAID-5: Rotating Parity

Parity block for each stripe – saves space.
Read block; write full stripe (including parity)
RAID-2 and RAID-3

RAID-2:
- Bit level striping
- Multiple ECC disks (instead of parity)

RAID-3:
- Byte level striping
- Dedicated parity disk

RAID-2 and RAID-3 are not used in practice
Flash-Based SSD’s

Flash-based Solid-State Storage Device

• Value stored by transistor
  – SLC (Single-level cell): 1 bit
  – MLC (Multi-level cell): 2 bits
  – TLC (triple-level cell): 3 bits
<table>
<thead>
<tr>
<th></th>
<th>RAM</th>
<th>HDD</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical Size</td>
<td>8 GB</td>
<td>1 TB</td>
<td>256 GB</td>
</tr>
<tr>
<td>Cost</td>
<td>$10 per GB</td>
<td>$0.05 per GB</td>
<td>$0.32 per GB</td>
</tr>
<tr>
<td>Power</td>
<td>3 W</td>
<td>2.5 W</td>
<td>1.5 W</td>
</tr>
<tr>
<td>Read Latency</td>
<td>15 ns</td>
<td>15 ms</td>
<td>30 µs</td>
</tr>
<tr>
<td>Read Speed (Seq.)</td>
<td>8000 MB/s</td>
<td>175 MB/s</td>
<td>550 MB/s</td>
</tr>
<tr>
<td>Read/Write Granularity</td>
<td>word</td>
<td>sector</td>
<td>page*</td>
</tr>
<tr>
<td>Power Reliance</td>
<td>volatile</td>
<td>non-volatile</td>
<td>non-volatile</td>
</tr>
<tr>
<td>Write Endurance</td>
<td>*</td>
<td>**</td>
<td>100 TB</td>
</tr>
</tbody>
</table>

[C. Tan, buildcomputers.net, codecapsule.com, crucial.com, wikipedia]
Solid State Drives (Flash)

Most SSDs based on NAND-flash

- retains its state for months to 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/
NAND Flash

Charge is stored in Floating Gate (can have Single and Multi-Level Cells)

Floating Gate MOSFET (FGMOS)

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

A **block** comprises a set of **pages**.

- **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
  - time: 10s of milliseconds
- **Read page**:
  - read granularity = 1 page = 2-4KBytes
  - time: 10s of microseconds
Flash Limitations

• can’t write 1 byte/word (must write whole pages)
• 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 deleted at same time (e.g., keep blocks of a file together)
- Wear-levelling: only write each physical page a limited number of times
- Remap pages that no longer work (sector sparing)

Transparent to the device user