Disks and RAID
50 Years Old!

- 13th September 1956
- The IBM RAMAC 350
• 80000 times more data on the 8GB 1-inch drive in his right hand than on the 24-inch RAMAC one in his left…
What does the disk look like?
Some parameters

- 2-30 heads (platters * 2)
  - diameter 14” to 2.5”
- 700-20480 tracks per surface
- 16-1600 sectors per track
- sector size:
  - 64-8k bytes
  - 512 for most PCs
  - note: inter-sector gaps
- capacity: 20M-100G

- main adjectives: BIG, slow
Disk overheads

• To read from disk, we must specify:
  – cylinder #, surface #, sector #, transfer size, memory address

• Transfer time includes:
  – Seek time: to get to the track
  – Latency time: to get to the sector and
  – Transfer time: get bits off the disk
# Modern disks

<table>
<thead>
<tr>
<th></th>
<th>Barracuda 180</th>
<th>Cheetah X15 36LP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>181GB</td>
<td>36.7GB</td>
</tr>
<tr>
<td><strong>Disk/Heads</strong></td>
<td>12/24</td>
<td>4/8</td>
</tr>
<tr>
<td><strong>Cylinders</strong></td>
<td>24,247</td>
<td>18,479</td>
</tr>
<tr>
<td><strong>Sectors/track</strong></td>
<td>~609</td>
<td>~485</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>7200RPM</td>
<td>15000RPM</td>
</tr>
<tr>
<td><strong>Latency (ms)</strong></td>
<td>4.17</td>
<td>2.0</td>
</tr>
<tr>
<td><strong>Avg seek (ms)</strong></td>
<td>7.4/8.2</td>
<td>3.6/4.2</td>
</tr>
<tr>
<td><strong>Track-2-track (ms)</strong></td>
<td>0.8/1.1</td>
<td>0.3/0.4</td>
</tr>
</tbody>
</table>
Disks vs. Memory

- Smallest write: sector
- Atomic write = sector
- Random access: 5ms
  - not on a good curve
- Sequential access: 200MB/s
- Cost $.002MB
- Crash: doesn’t matter (“non-volatile”)

- (usually) bytes
- byte, word
- 50 ns
  - faster all the time
- 200-1000MB/s
- $.10MB
- contents gone (“volatile”)
Disk Structure

- Disk drives addressed as 1-dim arrays of *logical blocks*
  - the logical block is the smallest unit of transfer
- This array mapped sequentially onto disk sectors
  - Address 0 is 1st sector of 1st track of the outermost cylinder
  - Addresses incremented within track, then within tracks of the cylinder, then across cylinders, from innermost to outermost
- Translation is theoretically possible, but usually difficult
  - Some sectors might be defective
  - Number of sectors per track is not a constant
Non-uniform #sectors / track

- Reduce bit density per track for outer layers (Constant Linear Velocity, typically HDDs)
- Have more sectors per track on the outer layers, and increase rotational speed when reading from outer tracks (Constant Angular Velocity, typically CDs, DVDs)
Disk Scheduling

• The operating system tries to use hardware efficiently
  – for disk drives ⇒ having fast access time, disk bandwidth

• Access time has two major components
  – *Seek time* is time to move the heads to the cylinder containing the desired sector
  – *Rotational latency* is additional time waiting to rotate the desired sector to the disk head.

• Minimize seek time

• Seek time ≈ seek distance

• Disk bandwidth is total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
Disk Scheduling (Cont.)

- Several scheduling algos exist service disk I/O requests.
- We illustrate them with a request queue (0-199).

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

Head pointer 53
FCFS

Illustration shows total head movement of 640 cylinders.

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

- Selects request with minimum seek time from current head position
- SSTF scheduling is a form of SJF scheduling
  - may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.
SSTF (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
The disk arm starts at one end of the disk,
  - moves toward the other end, servicing requests
  - head movement is reversed when it gets to the other end of disk
  - servicing continues.

Sometimes called the *elevator algorithm*.

Illustration shows total head movement of 208 cylinders.
queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
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 it immediately returns to beginning of the disk
    • No requests serviced on the return trip.
• Treats the cylinders as a circular list
  – that wraps around from the last cylinder to the first one.
C-SCAN (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53
C-LOOK

- Version of C-SCAN
- Arm only goes as far as last request in each direction,
  - then reverses direction immediately,
  - without first going all the way to the end of the disk.
C-LOOK (Cont.)

queue  98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
Selecting a Good Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better under heavy load
- Performance depends on number and types of requests
- Requests for disk service can be influenced by the file-allocation method.
- Disk-scheduling algorithm should be a separate OS module
  - allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable default algorithm
Disk Formatting

- After manufacturing disk has no information
  - Is stack of platters coated with magnetizable metal oxide
- Before use, each platter receives low-level format
  - Format has series of concentric tracks
  - Each track contains some sectors
  - There is a short gap between sectors

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Data</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Also contains cylinder and sector numbers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data is usually 512 bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ECC field used to detect and recover from read errors</td>
<td></td>
</tr>
</tbody>
</table>
Cylinder Skew

• Why cylinder skew?

• How much skew?
• Example, if
  – 10000 rpm
    • Drive rotates in 6 ms
  – Track has 300 sectors
    • New sector every 20 µs
  – If track seek time 800 µs
    ⇒ 40 sectors pass on seek

Cylinder skew: 40 sectors
Formatting and Performance

• If 10K rpm, 300 sectors of 512 bytes per track
  – 153600 bytes every 6 ms \( \Rightarrow \) 24.4 MB/sec transfer rate

• If disk controller buffer can store only one sector
  – For 2 consecutive reads, \( 2^{nd} \) sector flies past during memory transfer of \( 1^{st} \) track
  – Idea: Use single/double interleaving
Disk Partitioning

- Each partition is like a separate disk
- Sector 0 is MBR
  - Contains boot code + partition table
  - Partition table has starting sector and size of each partition
- High-level formatting
  - Done for each partition
  - Specifies boot block, free list, root directory, empty file system
- What happens on boot?
  - BIOS loads MBR, boot program checks to see active partition
  - Reads boot sector from that partition that then loads OS kernel, etc.
Handling Errors

- A disk track with a bad sector
- Solutions:
  - Substitute a spare for the bad sector (sector sparing)
  - Shift all sectors to bypass bad one (sector forwarding)
RAID Motivation

- Disks are improving, but not as fast as CPUs
  - 1970s seek time: 50-100 ms.
  - 2000s seek time: <5 ms.
  - Factor of 20 improvement in 3 decades

- We can use multiple disks for improving performance
  - By Striping files across multiple disks (placing parts of each file on a different disk), parallel I/O can improve access time

- Striping reduces reliability
  - 100 disks have 1/100th mean time between failures of one disk

- So, we need Striping for performance, but we need something to help with reliability / availability

- To improve reliability, we can add redundant data to the disks, in addition to Striping
RAID

• A RAID is a Redundant Array of Inexpensive Disks
  – In industry, “I” is for “Independent”
  – The alternative is SLED, single large expensive disk
• Disks are small and cheap, so it’s easy to put lots of disks (10s to 100s) in one box for increased storage, performance, and availability
• The RAID box with a RAID controller looks just like a SLED to the computer
• Data plus some redundant information is Striped across the disks in some way
• How that Striping is done is key to performance and reliability.
Some Raid Issues

• **Granularity**
  – fine-grained: Stripe each file over all disks. This gives high throughput for the file, but limits to transfer of 1 file at a time
  – coarse-grained: Stripe each file over only a few disks. This limits throughput for 1 file but allows more parallel file access

• **Redundancy**
  – uniformly distribute redundancy info on disks: avoids load-balancing problems
  – concentrate redundancy info on a small number of disks: partition the set into data disks and redundant disks
Raid Level 0

- Level 0 is **nonredundant** disk array
- Files are Striped across disks, no redundant info
- High read throughput
- Best write throughput (no redundant info to write)
- Any disk failure results in data loss
  - Reliability worse than SLED
Raid Level 1

- Mirrored Disks
- Data is written to two places
  - On failure, just use surviving disk
- On read, choose fastest to read
  - Write performance is same as single drive, read performance is 2x better

Expensive
Parity and Hamming Codes

• What do you need to do in order to detect and correct a one-bit error?
  – Suppose you have a binary number, represented as a collection of bits: \( <b_3, b_2, b_1, b_0> \), e.g. 0110

• Detection is easy

• Parity:
  – Count the number of bits that are on, see if it’s odd or even
    • EVEN parity is 0 if the number of 1 bits is even
  – \( \text{Parity}(<b_3, b_2, b_1, b_0>) = P_0 = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \)
  – \( \text{Parity}(<b_3, b_2, b_1, b_0, p_0>) = 0 \) if all bits are intact
  – \( \text{Parity}(0110) = 0, \text{Parity}(01100) = 0 \)
  – \( \text{Parity}(11100) = 1 \) => ERROR!
  – Parity can detect a single error, but can’t tell you which of the bits got flipped
Parity and Hamming Code

- Detection and correction require more work
- Hamming codes can detect double bit errors and detect & correct single bit errors
- 7/4 Hamming Code
  - $h_0 = b_0 \oplus b_1 \oplus b_3$
  - $h_1 = b_0 \oplus b_2 \oplus b_3$
  - $h_2 = b_1 \oplus b_2 \oplus b_3$
  - $H_0(<1101>) = 0$
  - $H_1(<1101>) = 1$
  - $H_2(<1101>) = 0$
  - Hamming($<1101>$) = $<b_3, b_2, b_1, h_2, b_0, h_1, h_0> = <110110>$
  - If a bit is flipped, e.g. $<1110110>$
  - Hamming($<1111>$) = $<h_2, h_1, h_0> = <111>$ compared to $<010>$, $<101>$ are in error. Error occurred in bit 5.
Raid Level 2

- Bit-level Striping with Hamming (ECC) codes for error correction
- All 7 disk arms are synchronized and move in unison
- Complicated controller
- Single access at a time
- Tolerates only one error, but with no performance degradation
Raid Level 3

- Use a parity disk
  - Each bit on the parity disk is a parity function of the corresponding bits on all the other disks
- A read accesses all the data disks
- A write accesses all data disks plus the parity disk
- On disk failure, read remaining disks plus parity disk to compute the missing data

Single parity disk can be used to detect and correct errors
Raid Level 4

- Combines Level 0 and 3 – block-level parity with Stripes
- A read accesses all the data disks
- A write accesses all data disks plus the parity disk
- Heavy load on the parity disk
Raid Level 5

- Block Interleaved Distributed Parity
- Like parity scheme, but distribute the parity info over all disks (as well as data over all disks)
- Better read performance, large write performance
  - Reads can outperform SLEDs and RAID-0
Raid Level 6

- Level 5 with an extra parity bit
- Can tolerate two failures
  - What are the odds of having two concurrent failures?
- May outperform Level-5 on reads, slower on writes
RAID 0+1 and 1+0

a) RAID 0 + 1 with a single disk failure.

b) RAID 1 + 0 with a single disk failure.
Stable Storage

• Handling disk write errors:
  – Write lays down bad data
  – Crash during a write corrupts original data

• What we want to achieve? Stable Storage
  – When a write is issued, the disk either correctly writes data, or it does nothing, leaving existing data intact

• Model:
  – An incorrect disk write can be detected by looking at the ECC
  – It is very rare that same sector goes bad on multiple disks
  – CPU is fail-stop
Approach

- Use 2 identical disks
  - corresponding blocks on both drives are the same
- 3 operations:
  - Stable write: retry on 1\textsuperscript{st} until successful, then try 2\textsuperscript{nd} disk
  - Stable read: read from 1\textsuperscript{st}. If ECC error, then try 2\textsuperscript{nd}
  - Crash recovery: scan corresponding blocks on both disks
    - If one block is bad, replace with good one
    - If both are good, replace block in 2\textsuperscript{nd} with the one in 1\textsuperscript{st}
CD-ROMs

Spiral makes 22,188 revolutions around disk (approx 600/mm). Will be 5.6 km long. Rotation rate: 530 rpm to 200 rpm
CD-ROMs

Symbols of 14 bits each

42 Symbols make 1 frame

Frames of 588 bits, each containing 24 data bytes

Preamble

98 Frames make 1 sector

Mode 1 sector (2352 bytes)

Logical data layout on a CD-ROM