CS 111: Operating System Principles

Lecture 17

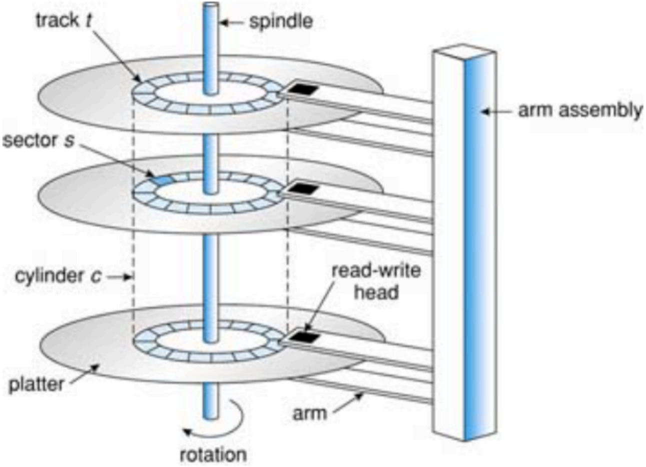
Disks

1.0.0

Jon Eyolfson

May 18, 2021

The Structure of a Hard Disk Drive (aka HDD)



Access Speed Depends on Locality

Sectors on same track can be read continuously

Switching tracks needs repositioning of the arm

Repositioning the arm is expensive)

You Physically Address a HDD Using Cylinder-Head-Sector (CHS)

Data has the following Coordinates (multi-dimensional polar coodinates):

- Platter: which revolving platter (addressed as head) [z Axis]
- Track: which track lane on platter (historically cylinder) [$||r||$]
- Sector: which sector on track [Θ]

The historical CHS has an approximate 8 GB limit of addressable space

(512 bytes/sector)×(63 sectors/track)×(255 heads (tracks/cylinder))
×(1024 cylinders)

LBA (Logical Block Addressing) uses one number to address
any block and is not limited to 8 GB

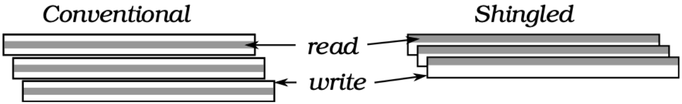
Shingled Magnet Recording (SMR)

The write head only writes in the center of a track, and has unused padding

You can’t write to this padding without destroying neighboring tracks

SMR however, allows you to write over the padding, if you do it sequentially

Drive performance may suffer, but it’s easier to increase capacity



HDDs Have Latencies Dependent on the Distance Travelled

Rotational delay: physically rotate the disk to get to the correct sector

- Typically 4-8 ms (average delay is half of a full rotation)

Seek time: moving the disk arm to get to the correct track

- Typically 0.5-2 ms

Transfer time: how long it takes to read bytes from the disk

- Typically the maximum transfer speed is 125 MB/s

Calculating Transfer Rate

The total time, T, is equal to

rotational delay + seek time + transfer time

The transfer rate, R, is equal to

Size of the transfer / T

What is the transfer rate of

Large sequential accesses?

Small random accesses?

We Should Use HDDs Sequentially Whenever Possible

	HDD 1	HDD 2
Rotational speed	7,200 RPM	15,000 RPM
Rotational latency (ms)	4.2	2.0
Average seek (ms)	9	4
Max transfer	105 MB/s	125 MB/s
Platters	4	4
Interface	SATA	SCSI

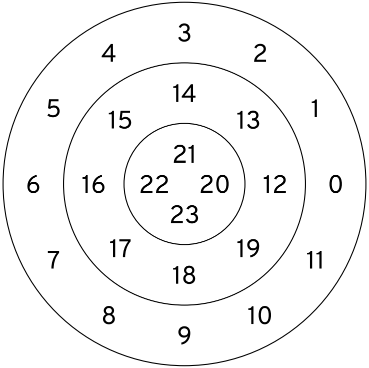
Sequential 100 MB read:

- HDD 1, T = 950 ms, R = 105 MB/s
- HDD 2, T = 800 ms, R = 125 MB/s

Random 4 KB read:

- HDD 1, T = 13.2 ms, R = 0.31 MB/s
- HDD 2, T = 6 ms, R = 0.66 MB/s

Logical Mapping Could Place All Sectors Next to Each Other



You may want to offset the sectors in different tracks so the head has time to settle

Track skew allows the disk to be efficient with minimal head movement

You May Want More Flexibility Than the Default Mapping

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

- Cons
    - Filesystem can’t inspect or try to optimize the mapping
    - Trying to reverse the mapping is difficult
        - Number of sectors per track changes
        - Disk silently remaps bad sectors

A Cache Can Significantly Speed Up Disk Transfers

Disks have some internal memory (WD Red - 64 MB) for caching

Implement a read-ahead “track buffer”

- Read the entire contents of the track into memory during the rotational delay

Write caching with volatile memory

- Write back: claim data is written to disk
  - Fast, but there’s data loss if there’s a power failure
  - Write through: acknowledge after data is physically written

We Can Schedule Disk Accesses

We want to minimize the time the disk moves without reading or writing data

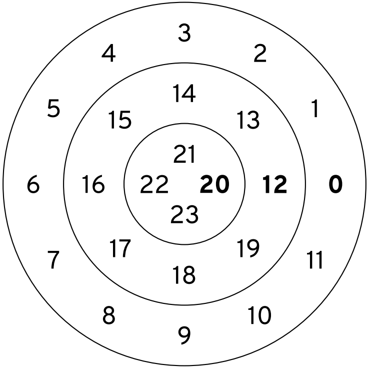
FCFS: schedule requests in the order received

- Fair, but it has a high seek and rotation cost

SSTF: shortest seek time first

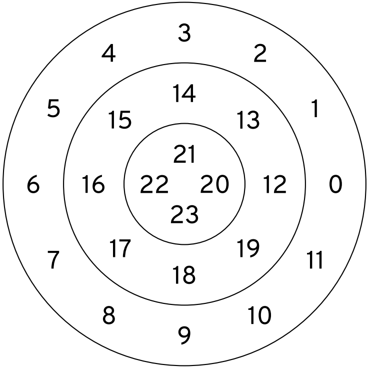
- Handle the nearest cylinder/sector next
    - Pro: reduces arm movement (seek time)
    - Con: unfair, can starve some requests

Elevator (aka SCAN or C-SCAN) Sweeps Across the Disk



If a request comes in for a track already serviced this sweep, queue it for the next

Elevator (or SSTF) Ignores Rotation



Shortest positioning time first (SPTF) is often the best strategy

The OS and disk need to work together to implement this

Solid State Drives (SSD) Are More Modern

Use transistors (like RAM) to store data rather than magnetic disks

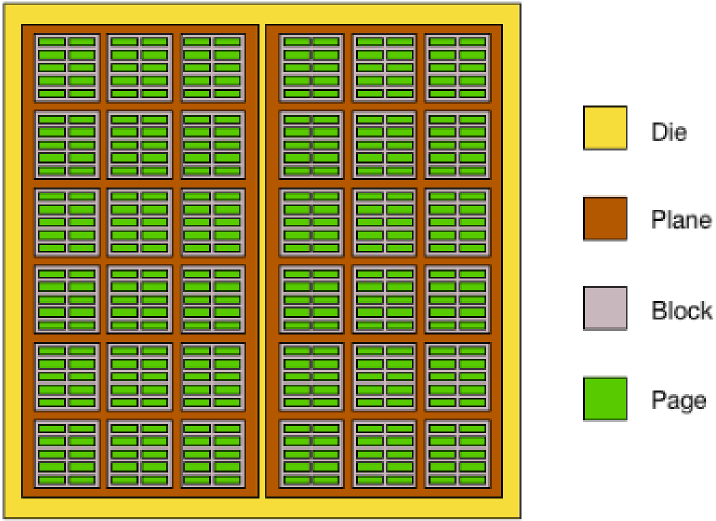
Pros

- No moving parts or physical limitations
- Higher throughput, and good random access
- More energy efficient
- Better space density

Cons

- More expensive
- Lower endurance (number of writes)
- More complicated to write drivers for

A SSD Contains Pages



SSDs Using NAND Flash Are Much Faster Than HHDs

Pages are typically 4 KiB

Reading a page: 10 µs
Writing a page: 100 µs
Erasing a block: 1 ms

NAND Flash Programming Uses Pages and Blocks

You can only read complete pages and write to freshly erased pages

Erasing is done per block (a block has 128 or 256 pages)

- An entire block needs to be erased before writing

Writing is slow (may need to create a new block)

The OS Can Help Speed Up SSDs

SSDs need to garbage collect blocks

- Move any pages that are still alive to a new block (may be overhead)

The disk controller doesn’t know what blocks are still alive

- SSD may think the disk is full, when a file could be deleted (not erased)

The OS can use the *TRIM* command to inform the SSD a block is unused

- The SSD can freely erase the block without moving overhead

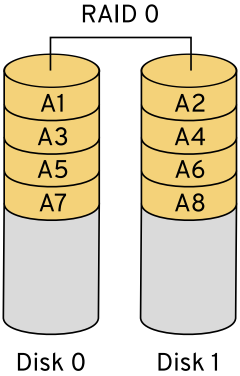
So Far We’ve Been Talking About Single Devices

- Sometimes called Single Large Expensive Disk (SLED)
  - Just one large disk for data
    - Single point of failure

- There’s also Redundant Array of Independent Disks (RAID)
  - Data distributed on multiple disks
    - Use redundancy to prevent data loss
    - Use redundancy to increase throughput

RAID 0 is Called a Striped Volume

Data stripes (128KB and 256KB) are distributed over disks



RAID 0 is For Performance Only

The data is stripped across all disks in the array (you can have more than 2)

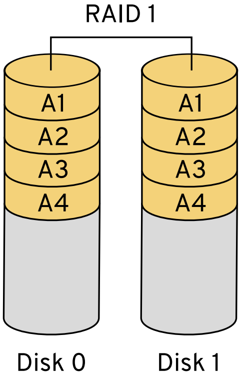
Pro

- Faster parallel access, roughly N times speed

Con

- Any disk failure results in a data loss (more points of failure)

RAID 1 Mirrors All Data Across All Disks



by Cburnett licensed under CC BY-SA 3.0

RAID 1 is Simple, But Wasteful

Every disk in the array has a mirrored copy of all the data

Pro

- Good reliability, as long as one disk remains, no data loss
- Good read performance

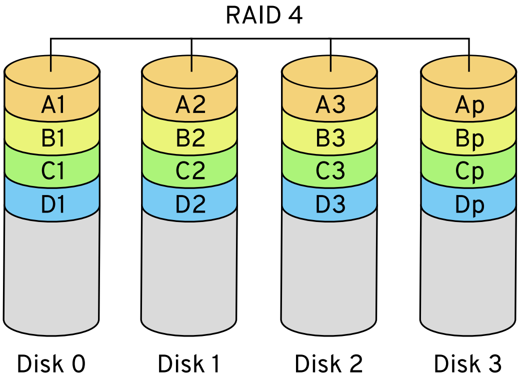
Con

- High cost for redundancy (we can do better)
- Write performance is the same as a single disk

RAID 4 Introduces Parity

Data stripes distributed over disks with a dedicated parity disk (p = parity)

Parity stores xor ⊕ of copies 1-3, any one copy can be reconstructed



by Cburnett licensed under CC BY-SA 3.0

RAID 4 Can Use the Parity Drive to Recover

With parity, we can use $1 - \frac{1}{N}$ of the available space

- Requires at least 3 drives

Pro

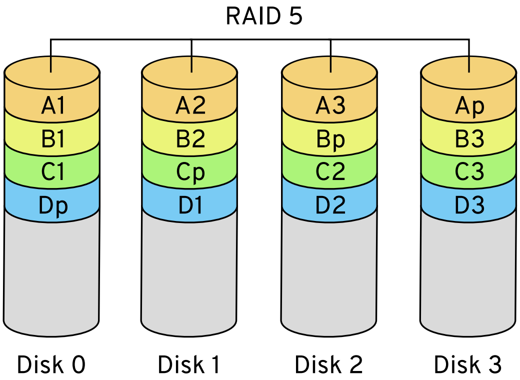
- We get $(N - 1)$ times performance (removing parity disk)
- We can replace a failed disk and rebuild

Con

- Write performance can suffer, every write must write to parity disk

RAID 5 Distributes Parity Across All Disks

Data stripes distributed over disks and each disk takes turns with parity blocks



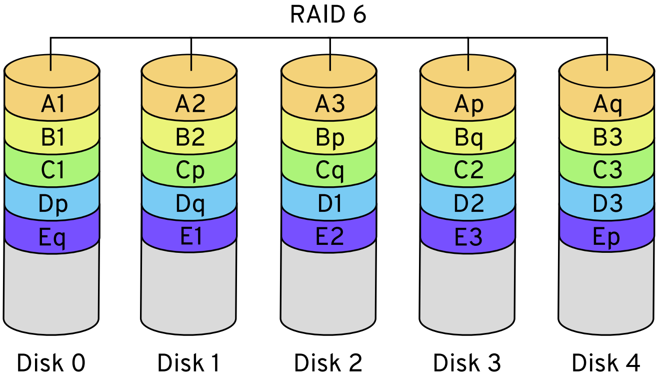
by Cburnett licensed under CC BY-SA 3.0

RAID 5 is an Improved Raid 4

It has all the same pros as RAID 4

Write performance is improved, no longer a bottleneck on a single parity drive

RAID 6 Adds Another Parity Block Per Stripe



by Cburnett licensed under CC BY-SA 3.0

RAID 6 Can Recover from 2 Simultaneous Drive Failures

Due to the extra parity, we can use $1 - \frac{2}{N}$ of the available space

- Requires at least 4 drives

Write performance is slightly less than RAID 5, due to another parity calculation

Disks Enable Persistence

We explored two kinds of disks: SSDs and HDDs

- Magnetic disks have poor random access (need to be scheduled)
- Shortest Positioning Time First (SPTF) is the best scheduling for throughput
- SSDs are more like RAM except accessed in pages and blocks
- SSDs also need to work with the OS for best performance (TRIM)
- Use RAID to tolerate failures and improve performance using multiple disks