

CS3.301 Operating Systems and Networks

Persistence: RAIDs

Karthik Vaidhyanathan

<https://karthikvaidhyanathan.com>

1



Acknowledgement

The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidyanathan

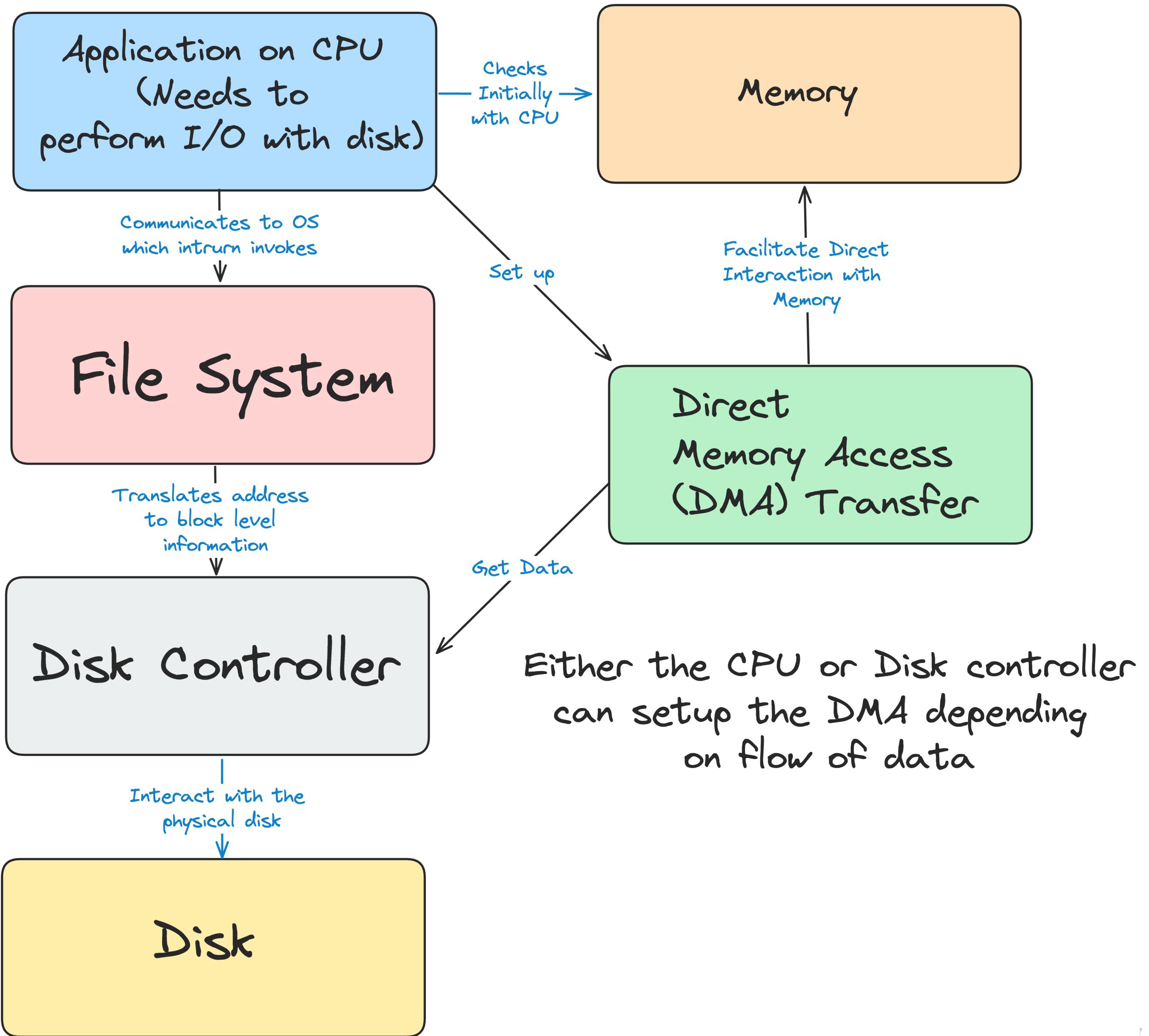
Sources:

- Operating Systems in Three Easy Pieces by Remzi et al.



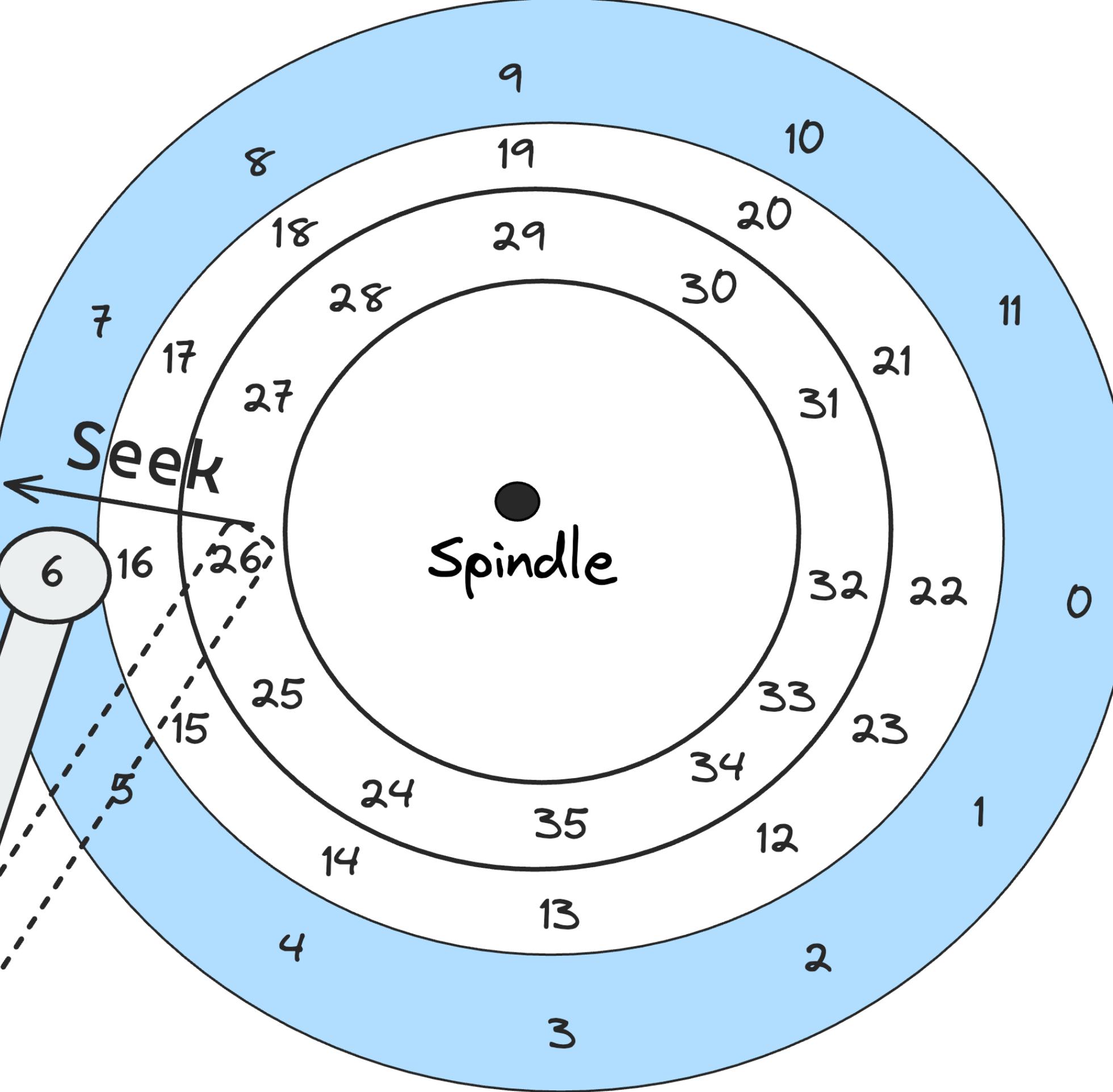
The flow of access

- Application performs read or write to a file
- CPU communicates to OS which invokes the File System (FS)
- The OS may check in its cache if its already there
- FS prepares block level information to disk controller
- A Direct Memory Access (DMA) is set up
- Disk controller performs the physical read or write based on commands from DMA and file system
- If its read, Disk -> DMA, for writes, DMA -> Disk



Quick Overview

Rotates this way



- Disk rotates on a spindle
 - The arm can move across (seek) or stay as the disk rotates
 - The head is used to read/write
- Data is arranged in tracks as blocks/sectors
- There are 100s of tracks on a single disk
- **Seek, rotate and transfer** - three key phases

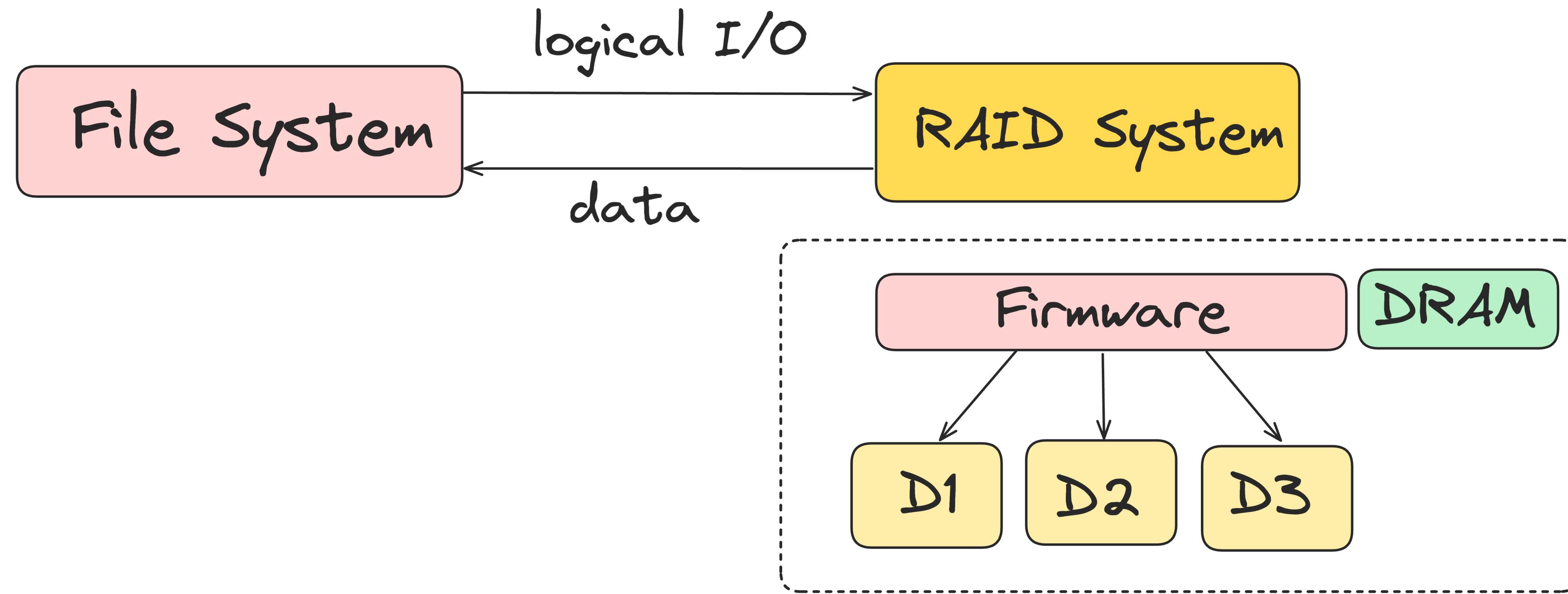
Redundant Arrays of Inexpensive Disks (RAID)

Redundant Arrays of Independent Disks!

- Techniques to use multiple disks in concert to build **faster, bigger and more reliable** disk system
- Term introduced in late 90's by a group of researchers in UC Berkley
- Externally RAIDs look just like group of blocks one can read or write
 - Internally RAID is very complex
 - Consisting of **multiple disks**
 - **Its own memory** - DRAM
 - **One or more processor** to manage the system



RAIDs: Simple Illustration



- As far as **File System** (the subcomponent inside OS) is concerned
 - RAID is just like a disk
 - Linear array of blocks each of which can be read or written

RAID in Action



L

ThinkSystem ST250 V2 Tower Server

Specs

Features

Models

Starting at: ₹125,990

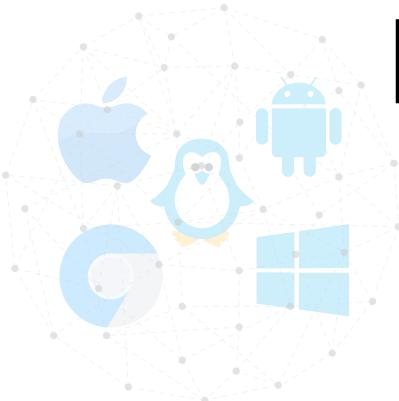
Shop Now

Expansion Slots	Slot 3: PCIe x4 slot with PCIe Gen3 x4 lanes Slot 4: PCIe x8 slot with PCIe Gen3 x4 lanes
Network Interface	2x GbE on-board ports (Broadcom BCM5720); 1x GbE port dedicated for XCC management
Ports	Front: 1x USB 3.2 G2 (10Gb) port, 1x USB 2.0 port for local management using the XCC Mobile app Rear: 4x USB 3.2 G2 (10Gb) ports, 2x RJ45 Gigabit Ethernet ports, 1x 1GbE dedicated XCC port for remote management, 1x Serial port and 1x VGA port
HBA/RAID Support	Intel® VROC Software RAID support with both simple-swap and hot-swap configuration; multiple hardware RAID configurations supported



RAIDs

- At a high level, RAIDs are like a computer system
 - RAID is like a box with standard (SCSI or SATA) to a host
 - Provides a consistent interface to the OS
- Internally RAIDs are very complex
 - Consists of a **microcontroller** that runs a firmware
 - **Volatile memory** such as DRAM to buffer data blocks as they are read and written
 - **Non-volatile memory** to buffer writes safely and for parity calculation as well
- Instead of running application RAID, runs specialised software designed to operate RAID



Evaluating RAIDs

- Many approaches are there to build a RAID system
 - Each has different characteristics
- Three axes can be used for evaluation
 - **Capacity**
 - **Reliability**
 - **Performance**

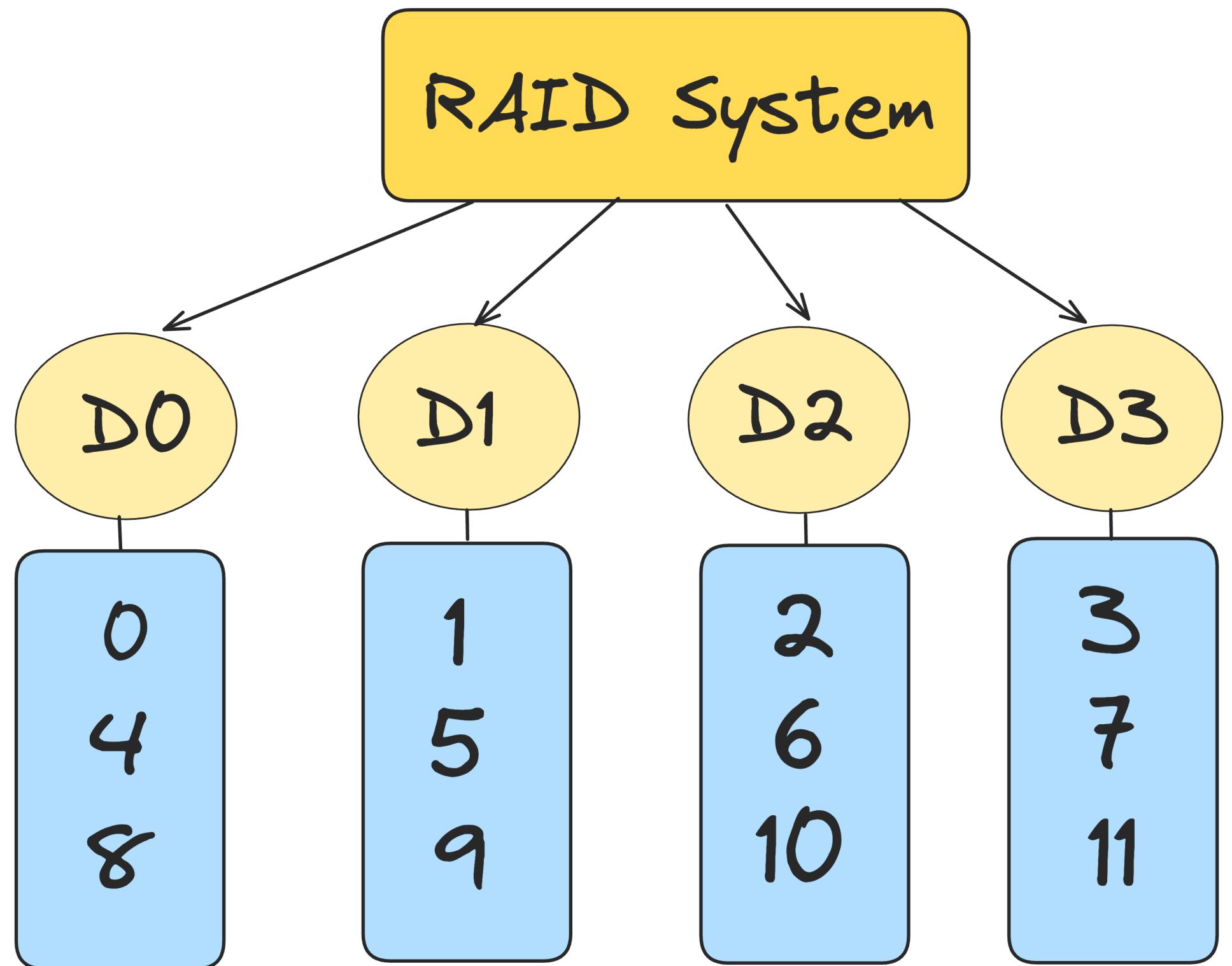


Evaluating RAIDs

- **Capacity**
 - Given a set of N disks each of size B blocks. How much capacity is available for usage?
 - Some redundancy may be required => $N/2$ when each is replicated
- **Performance**
 - What's the impact of different workload on the latency of I/O?
 - What's the throughput? Rate of transfer -Transfers/second!
- **Reliability**
 - How many failures/faults can the RAID system tolerate?
 - The fault model considered: A fault => total disk has failed!



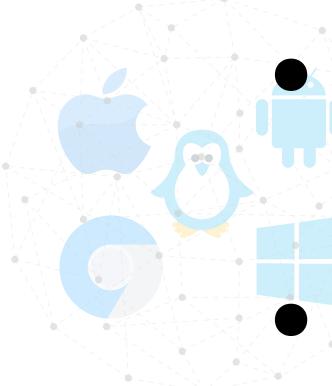
RAID level 0: Striping



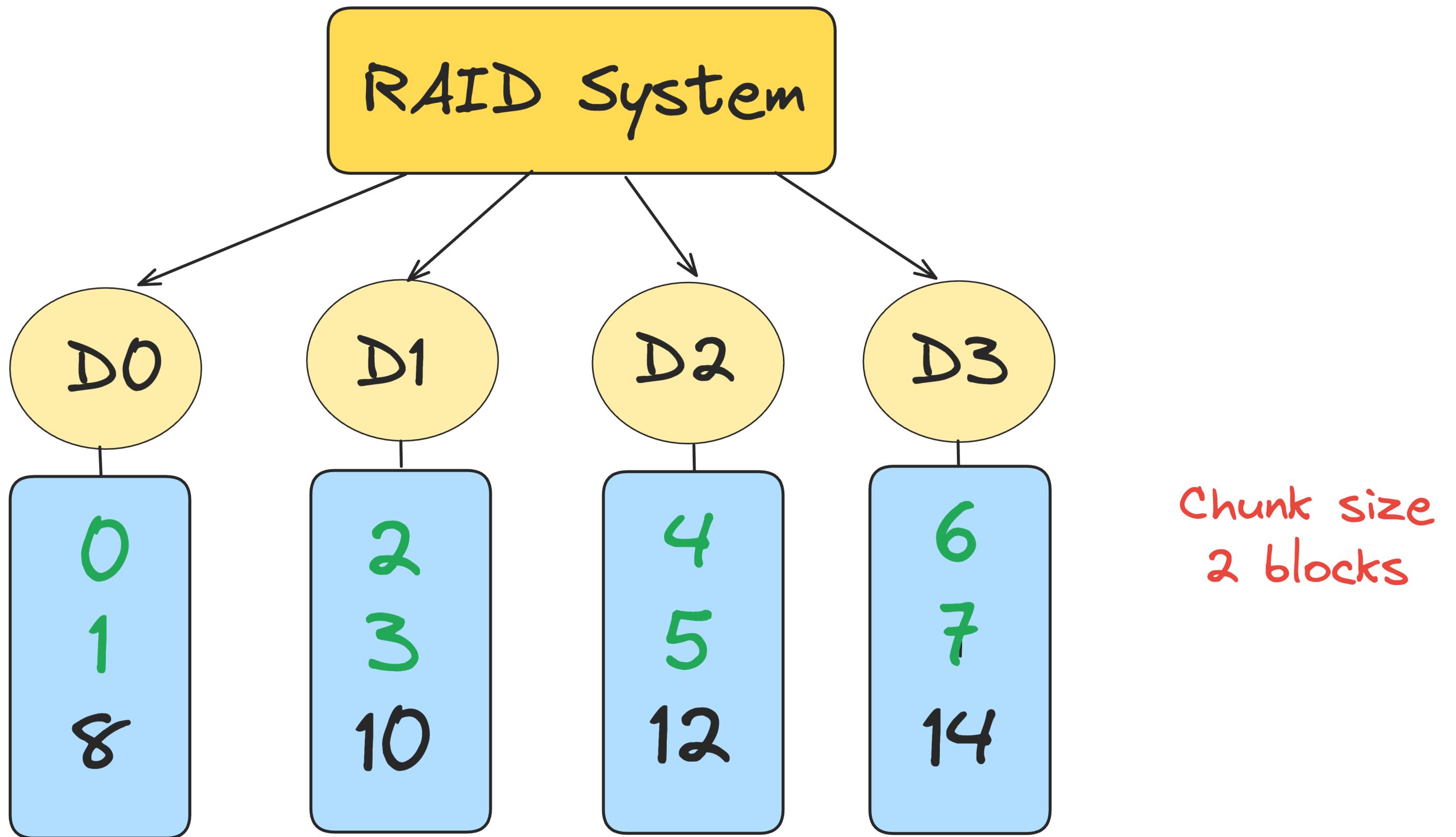
Here blocks 0, 1, 2 and 3 are in same stripe

Each block is of size 4 KB

- **Simple form:** Spread the blocks across the disks in a round robin fashion
 - Blocks in the same row - **Stripe**
 - No redundancy



RAID level 0: Striping

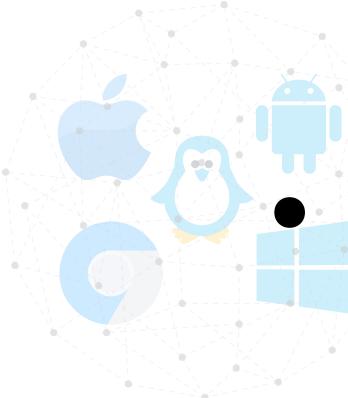


- Two 4 KB blocks are placed in one disk before moving to another
- Chunk size is 8 KB and a stripe consists of 4 chunks -> 32 KB of data
- Chunk size do have an impact on the performance! - **How?**



RAID Level 0: Impact of Chunk Size

- **Small chunk size**
 - Many files will get striped across disks
 - Increases parallelisms of reads and writes
 - Positioning time to access blocks across disks increases
- **Big chunk size**
 - Reduces intra-file parallelism, relies on multiple concurrent request to achieve high throughput
 - Large chunk size reduces positioning time (One file in one disk) same as using one disk
- **Best chunk size is hard to get - Depends on the workload!**



RAID Level 0: Performance Analysis

- Two main things to evaluate:
 - **Single-request latency:** latency of single I/O request to RAID
 - **Steady-state throughput:** Total bandwidth of concurrent requests
- Two main workloads:
 - **Sequential:** Request to disk arrive in large contiguous chunks
 - **Random:** Each request is small to a random location on disk
- Assume disk transfers at **S MB/s** under sequential and **R MB/s** under random



RAID Level 0: Performance Analysis

- Consider the following disk characteristics
 - Sequential transfer of size 10 MB on average
 - Random transfer of size 10 KB on average
 - Average seek time 7 ms
 - Average rotational delay 3 ms
 - Transfer rate of disk 50 MB/s
- How to calculate **S** and **R**?

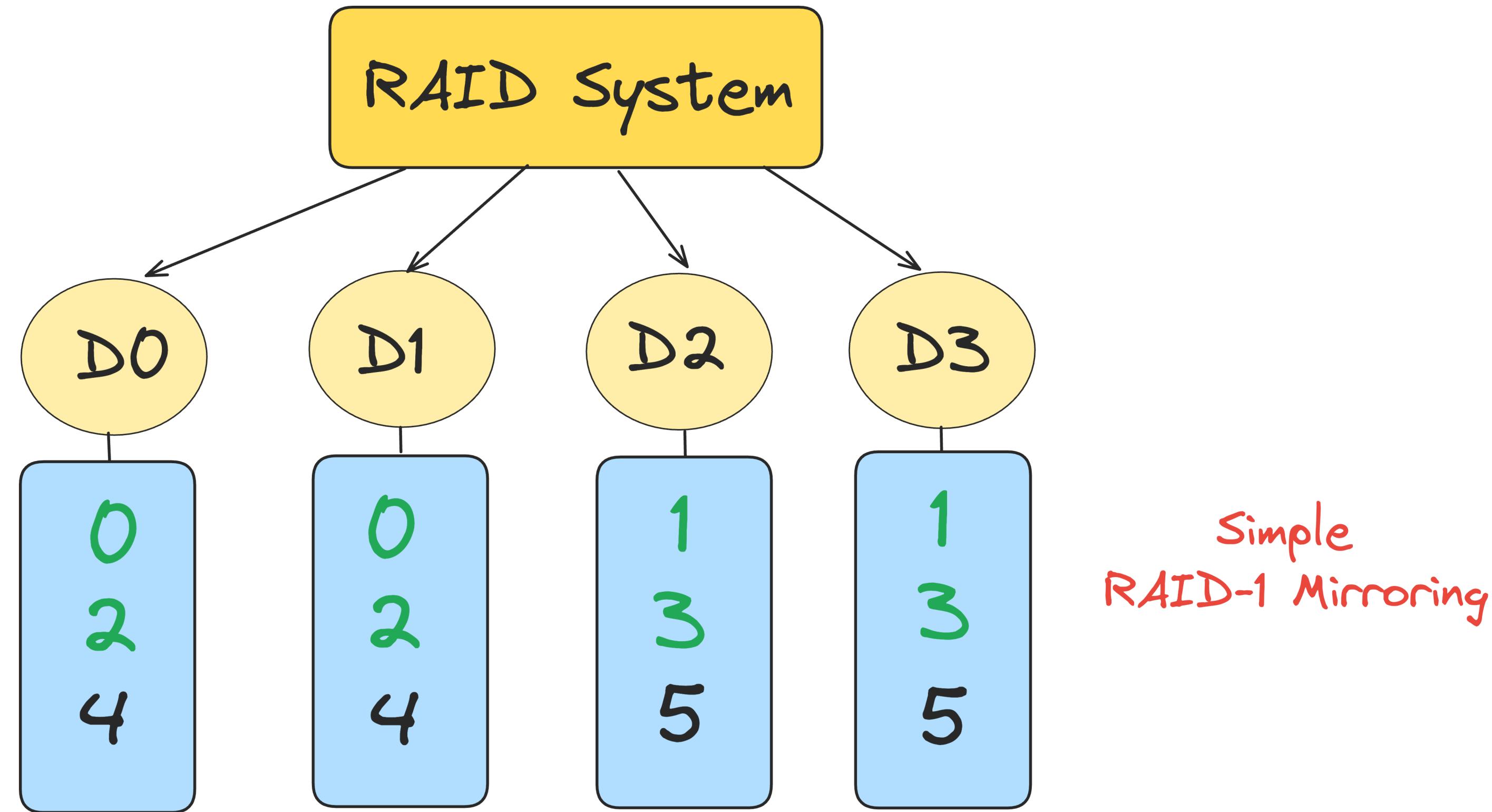


RAID Level 0: Analysis

- 7 ms spend seeking and 3 ms spend in rotation => total: 10 ms
- 10 MB @ 50MB/s => 200 ms for transfer => total: $200 + 10 = 210$ ms
- $S = 10 \text{ MB} / 210\text{ms} = 47.62 \text{ MB/s}$
- For R, 10 KB @ 50 MB/s => 0.195 ms => total: $10 + 0.195 = 10.195$ ms
- $R = 10 \text{ KB} / 10.195 \text{ ms} = 0.981 \text{ MB/s}$
- Steady-state throughput equals $N*S$ MB/s or $N*R$ MB/s depending on workload
 - RAID 0 is more like an upper bound



RAID level 1: Mirroring



- Copies are made, each copy is placed in a different disk - Handle failures!
- Data is stripped across mirrored pairs



RAID Level 1: Mirroring

- **Read**
 - When reading from a block, RAID has a choice!
 - Assume a read comes to 0, the system can either use Disk 0 or 1
- **Write**
 - No choice exists, the write needs to happen in both copies of data
 - This promotes reliability, writes can happen in parallel



RAID 1: Analysis

- **Capacity**, with all replicated, achieved capacity: $N/2$
- **Reliability**, RAID 1 can tolerate failure of 1 disk
- **Performance**
 - For single read request, RAID-1 just needs to redirect to one of the copies
 - Write is little different: Two writes needs to happen and it will happen in parallel => time will be almost equal to single write
 - But, due to worst case rotational of two requests, it will be higher than write to a single disk

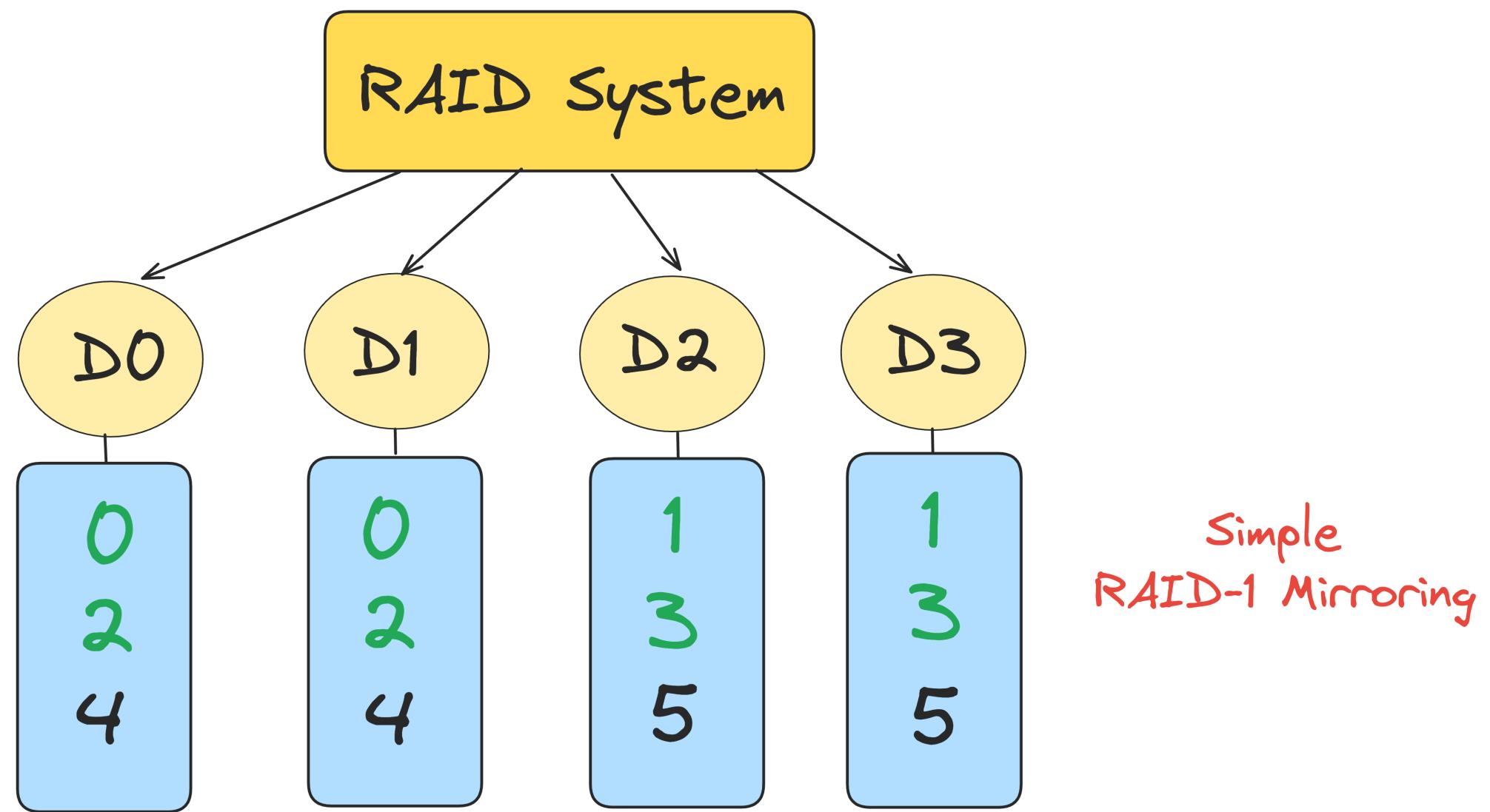


RAID 1: Analysis

- Steady state throughput
 - Bandwidth during sequential write is $(N/2) * S$ MB/s or half the peak
 - Each write involves writing in two different locations
 - Sequential reads also has a similar bandwidth:
 - Consider reads that needs to be done on blocks: 0,1, 2, 3, 4, 5, 6, 7
 - What will be the bandwidth or steady state throughput in this case?



RAID 1: Analysis

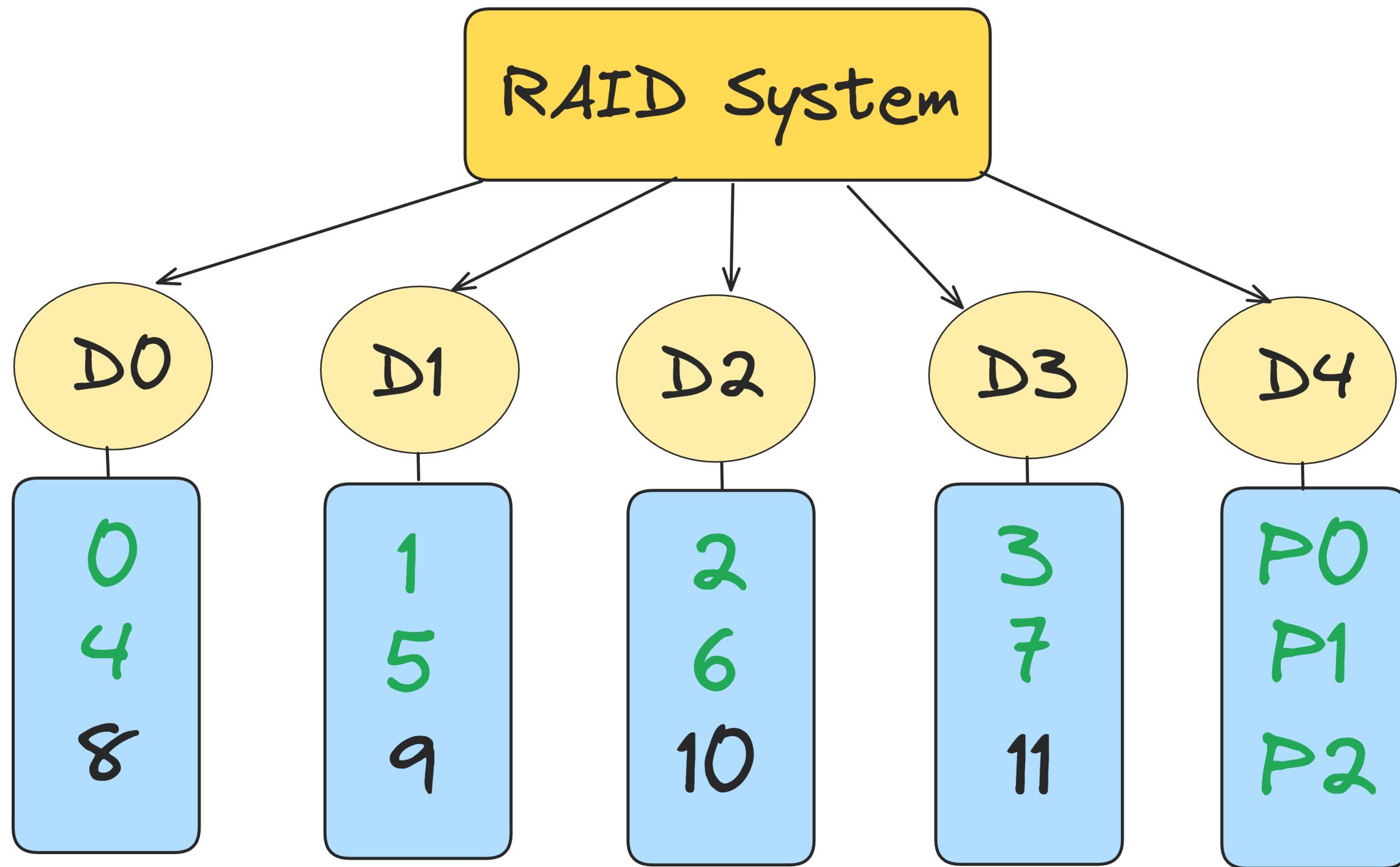


- 0 is send to D0, 1 to D2, 2 to D1, 3 to D3....
- 0 comes to D0 then next one is 4, 2 is skipped (since it goes to D1)
- Simply keeps rotating without doing useful transfer (as D1 is taken care)
- Each disk will only deliver half the peak bandwidth, $(N/2) * S$ MB/s for **Sequential reads**
- **Random reads** $N*R$ and write $(N/2) * R$ MB/s

Redundancy is good but can we do better?



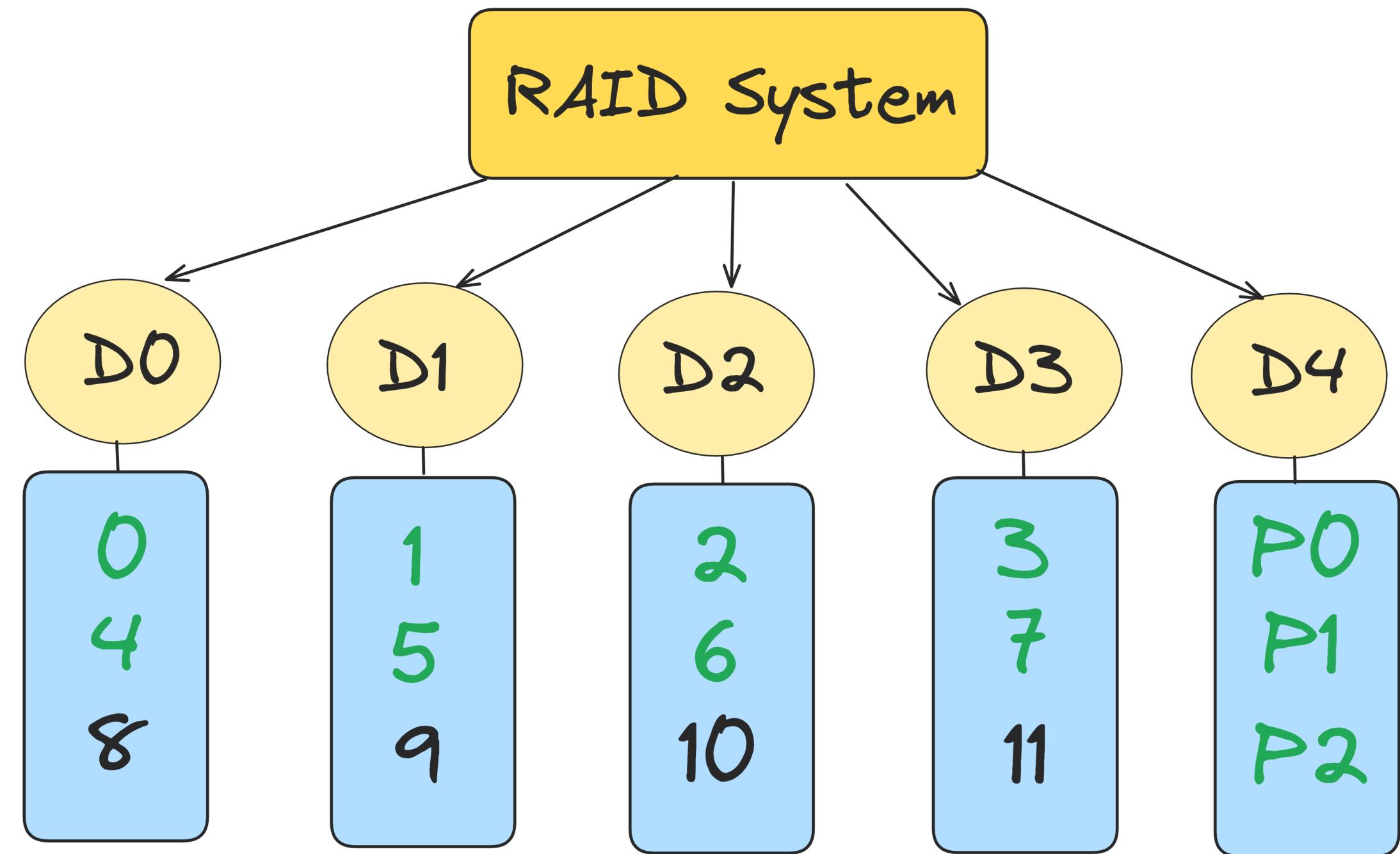
RAID Level 4: Introducing Parity



- Another method for better managing redundancy: **Parity**
- They aim to use less capacity and overcome space issues at cost of performance
- For each stripe of data above, a parity block is added that stores the redundant information for that block



RAID Level 4



- P1 has redundant information that it is calculated from blocks 4, 5, 6 and 7
- To compute parity **XOR** function is used
 - XOR returns 1 if there are odd no of 1's
 - XOR returns 0 if there are even no of 1's
 - This allows to identify if there were some faults in any of the disks - **how?**



RAID Level 4

C0	C1	C2	C3	P
0	0	1	1	XOR (0,0,1,1) = 0
0	1	0	0	XOR (0,1,0,0) = 1

- The parity information can be used to recover from failure
- Assume **data in first row of C2 is lost (it was 1)**
 - Read all the other values in the row and reconstruct the answer
 - Without value of C2 (1), $\text{XOR } (0,0,0,1) = 0$; Hence we can find that C2 needs to be 1



RAID Level 4

Block 0	Block 1	Block 2	Block 3	Parity
00	10	11	10	11
10	01	00	1	10

- In the larger context perform bitwise XOR of all the bits
- Perform Bitwise XOR across each bit of data blocks
 - Put the result of each bit in the corresponding bit slot in parity block
 - Assume that Block 2 fails
 - $\text{Block 2} = (00) \text{ XOR } (10) \text{ XOR } (10) \text{ XOR } (11) = 11$



RAID Level 4: Analysis

- **Capacity:** 1 disk is for parity hence $(N-1)*B$
- **Reliability:** Tolerates 1 disk failure, if more than 1 is lost, no way to recover
- **Performance, Steady-state-throughput:**
 - **Sequential reads:** $(N-1)*S$ MB/s
 - **Sequential writes:** $(N-1)*S$ MB/s (write also parity in parallel, full-stripe write)
 - **Note:** writing to parity at same time is not performance gain for client! Hence N-1
 - **Random read:** $(N-1)*R$ MB/s
 - **Random writes?**



RAID Level 4: Analysis

- Main operations involved in write, especially random write:
 - Update a block + update of parity
- Method 1: **Additive Parity**
 - Read in all of the other blocks in that stripe
 - XOR those blocks with the new block
- **Problem:** As number of blocks increase, this can be challenging, reading of all blocks to perform XOR



RAID Level 4: Analysis

- Method 2: Subtractive Parity

C0	C1	C2	C3	P
0	0	1	1	XOR (0,0,1,1) = 0

- Update C2(old) -> C2 (new)
- Read old data in C2 ($C2(\text{old})=1$) and old data in parity ($P(\text{old}) = 0$)
- Calculate $P(\text{new}) = (C2(\text{old}) \text{ XOR } C2(\text{new})) \text{ XOR } P(\text{old})$
 - If $C2(\text{new}) == C2 (\text{old}) \rightarrow P(\text{new}) = P(\text{old})$
 - If $C2(\text{new}) != C2 (\text{old}) \rightarrow \text{Flip the old parity bit}$

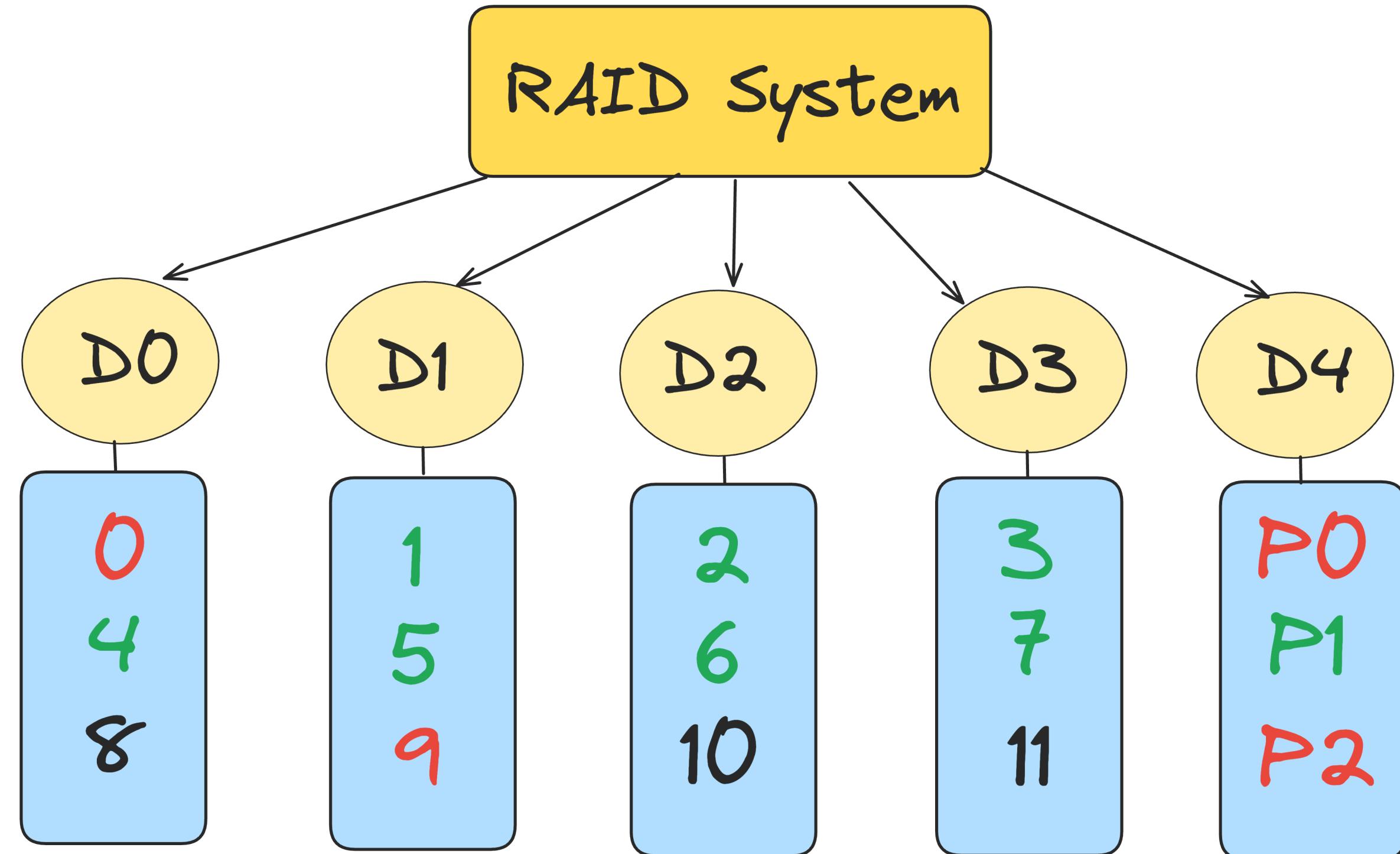


Small-write Problem

- The parity disk can be a bottleneck

- Example: Writes to 0 and 9

- Disk 0 and Disk 1 can be accessed in parallel
 - Disk 4 prevents any parallelism



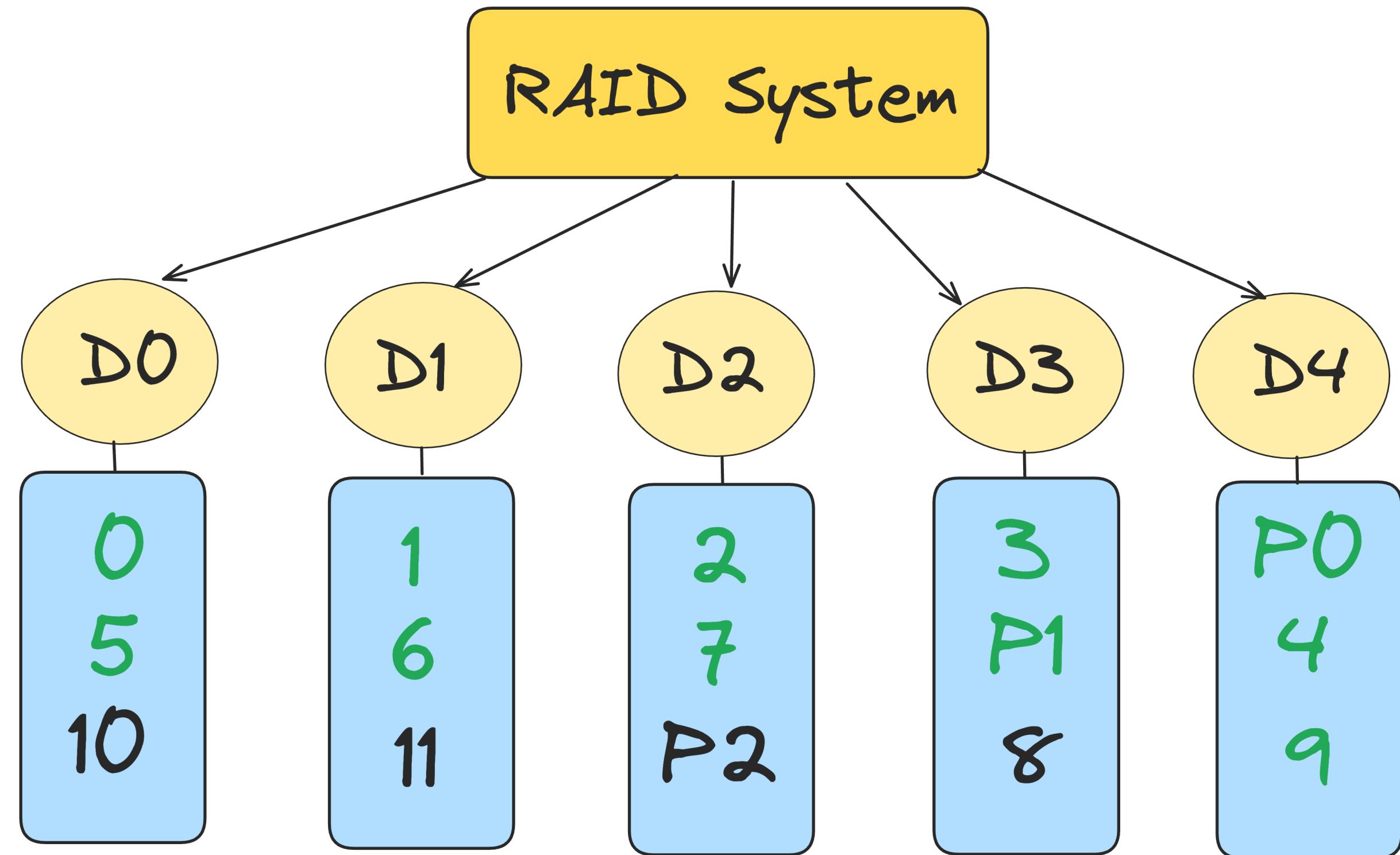
- RAID-4 under random workload, small writes is $(R/2)$ MB/s - **terrible!**
 - How to improve further?

I/O Latency in RAID-4

- A single read
 - Equivalent to latency of single disk request
- A single write
 - Two reads + Two writes
 - Data block + parity block
 - The reads and writes can happen in parallel
 - Total latency is twice that of single disk



RAID Level 5: Rotating Parity



- Addresses the small-write problem
- Similar to RAID-4 except that keeps rotating the parity block
- Removes the parity-disk bottleneck for RAID-4

RAID-5 Analysis

- **Capacity and reliability** identical to RAID-4
- Sequential read and write performance similar to RAID-4
- Random read performance is little better (utilize all disks)
- Random write performance
 - Here the write requests can be parallelized as parity is not bottleneck
 - Given large number of random write requests, all disks can be evenly kept busy, total bandwidth = $(N/4)*R$ MB/s. Still 4 I/O operations (as parity is there)



Summarizing RAIDS

- **Performance** and do not care about reliability -> RAID-0 (Striping)
- Random I/O performance and **reliability** -> RAID-1 (Mirroring)
- **Capacity** and **Reliability** -> RAID-5
- **Sequential** I/O and Maximise **Capacity** -> RAID-5





Thank you

Course site: karthikv1392.github.io/cs3301_osn

Email: karthik.vaidhyanathan@iiit.ac.in

Twitter: [@karthyishere](https://twitter.com/karthyishere)

