RAID

Computer Operating Systems, Fall 2023

Instructor: Travis McGaha

Head TAs: Nate Hoaglund & Seungmin Han

TAs:

Andy Jiang	Haoyun Qin	Kevin Bernat	Ryoma Harris
Audrey Yang	Jason hom	Leon Hertzberg	Shyam Mehta
August Fu	Jeff Yang	Maxi Liu	Tina Kokoshvili
Daniel Da	Jerry Wang	Ria Sharma	Zhiyan Lu
Ernest Ng	Jinghao Zhang	Rohan Verma	

Administrivia

- MILESTONE 1 IS DUE between Friday 11/10 – Tuesday 11/14
  - Expecting around 60% progress on this
  - Should have stand-alone PennFAT
  - Must meet with your TA in the specified window to demonstrate what you have implemented

- Second part of today's lecture will be PennOS office hours/AMA

Administrivia

- I synched a bunch of grades to canvas. **PLEASE CHECK THAT THEY ARE ACCURATE**
  - All check-ins
  - Project 0 & peer-eval
- Midterm Grades Posted
  - Regrade requests open, due Saturday night @ midnight
- Checkin – out tomorrow, due before Tuesday's lecture
  - Should be really short
  - "Do you remember pthreads?"

Poll Everywhere

pollev.com/tqm

- Any questions, comments or concerns from last lecture?

Lecture Outline

- RAID

RAID

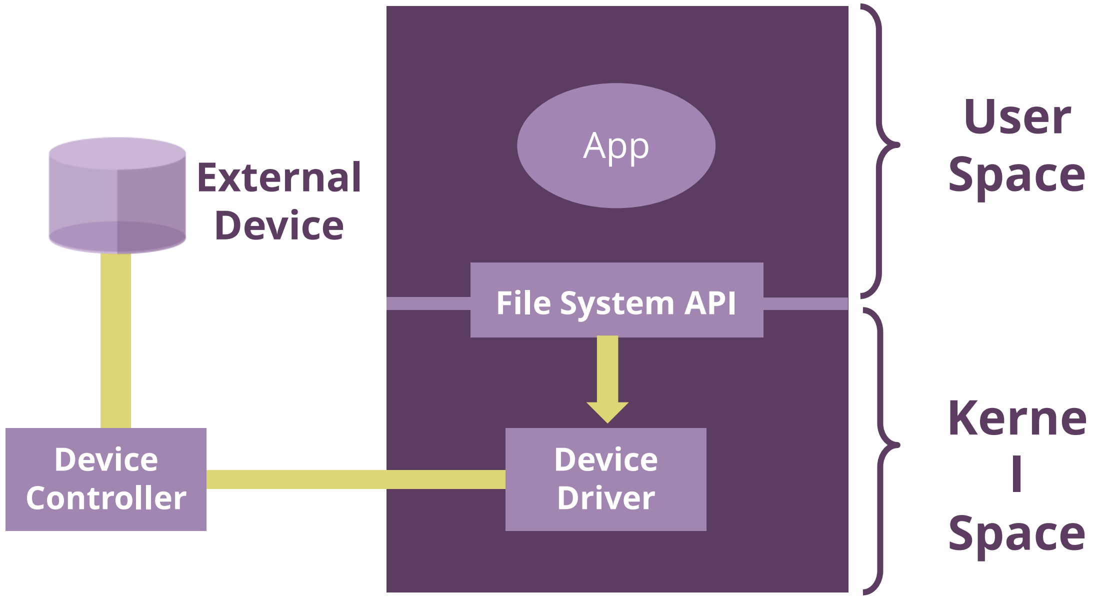
- **R**edundant **A**rray of **I**ndependent **D**isks
  - Taking multiple physical disk drives and combining them into one logical unit.
  - The OS Filesystem will talk to the RAID controller, which manages a suite of disks.

- Many variations ("levels") of RAID, but general idea is to spread out work across several physical disks

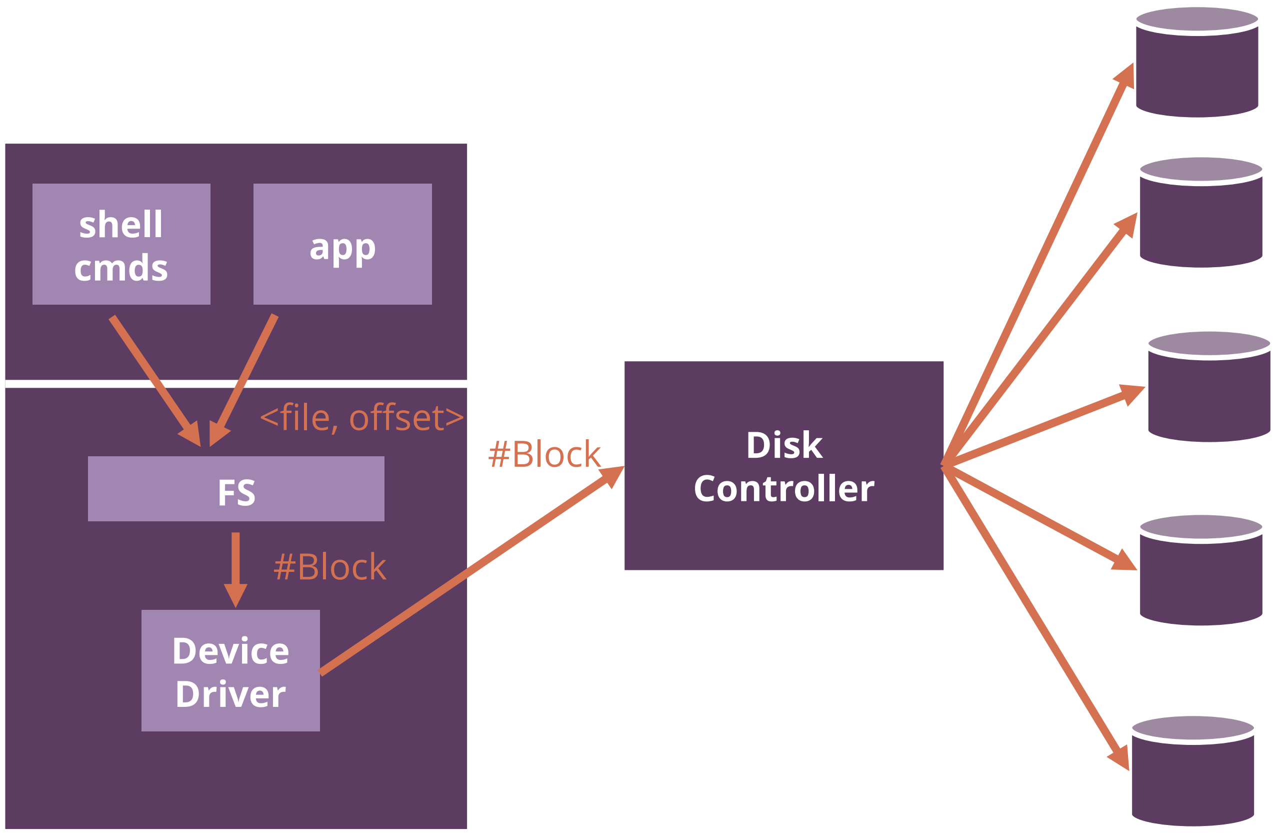
- Goals:
  - Aid in performance
  - Aid in data redundancy (error recovery)

Can implement this as Extra credit for PennOS

Previous Disk Architecture



RAID Disk Architecture



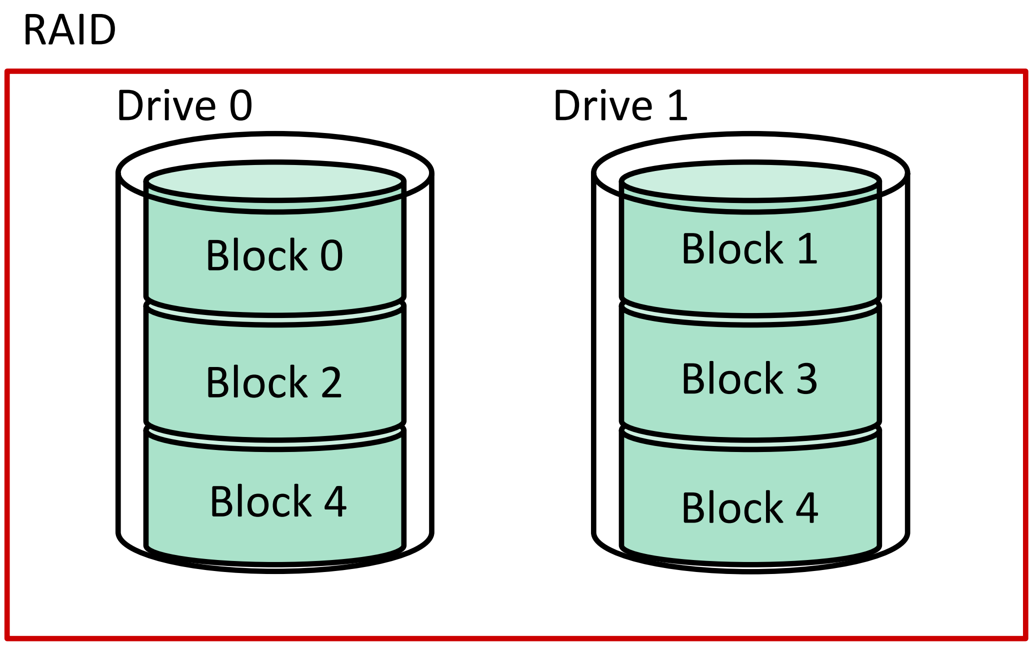
Striping

- Striping
- Basic idea: organize logically consecutive chunks of data across disk to improve performance
- This can be done at the bit-level, byte-level, sector-level or block level
- When the OS sends a read/write request to the RAID controller, it is split across all relevant disks by the RAID controller

RAID Level 0

- Array of Disks with block-level striping

- If we have a file that spans multiple blocks, we can split those blocks across our array of drives



Can be more than 2 disks, just keeping it small for slides

RAID Level 0

- Pros:
  - Not much overhead
  - This allows faster throughput on reads/writes to the same file.

    Normally if someone wanted to read block 0 and block 1, if it was all on one drive, the reads would occur one after the other

    With RAID 0, the read of block 0 and block 1 are split across both drives and can be completed in Parallel

- Cons:
  - More prone to losing or corrupting a file. Why?
    - If ANY disk a file is on fails, we lose the integrity of the file
  - Reading/writing a single block doesn’t get this sped up

Failure Assumption

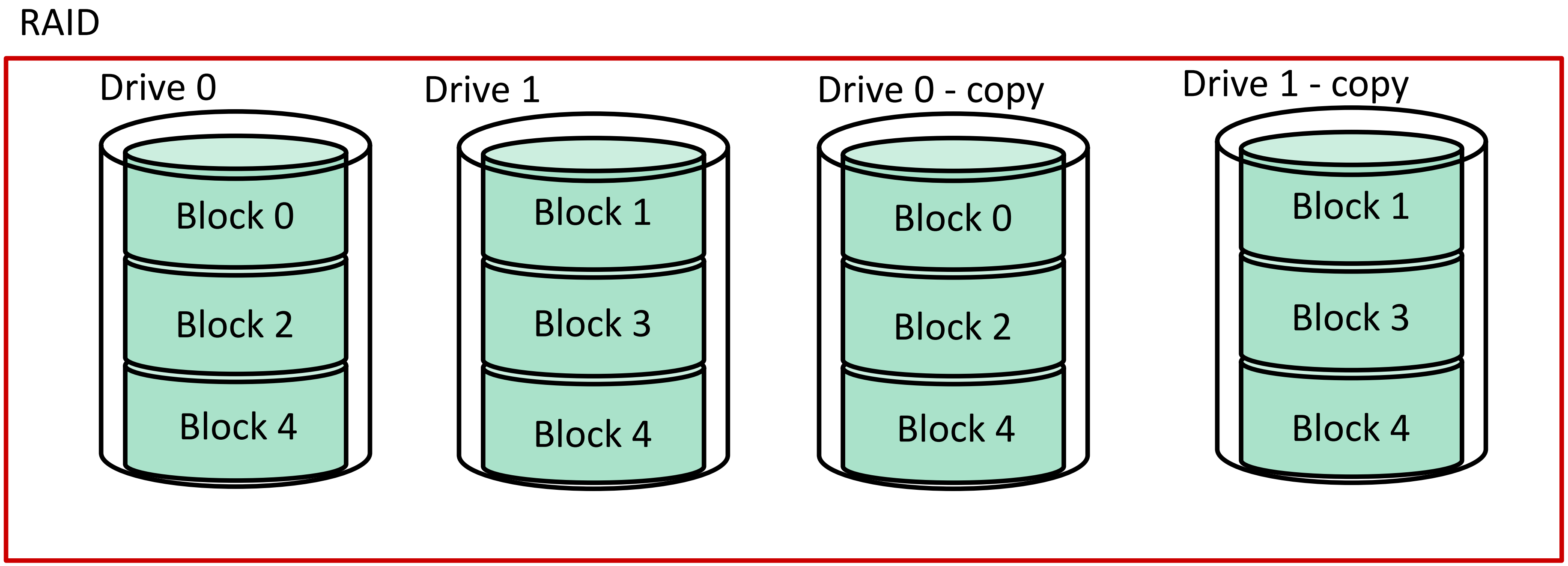
- For the RAID implementations that try to add data redundancy, we make an assumption on how errors occur.

- Assumption: when a data-center experiences a failure, only one disk is affected, and it is quickly replaced by the person managing the data-center

- If there are more than 1 disk failure for a single RAID, we have issues…
  - Other algorithms (in-addition to or in-place of RAID) may be implemented to counter these situations

RAID Level 1

- Do the same thing as RAID 0, but each drive has a duplicate "backup" drive



Poll Everywhere

pollev.com/tqm

- What pros and cons do you see with this model?
- Hint: this affects performance, space consumption and reliability.
- Hint2: are all drives the same speed?

RAID Level 1

- Pros:
  - Have data resiliency, unless something terrible happens

- Cons:
  - Twice as much space needed to store the same amount of data
  - A read/write of a block is as slows as the slowest of the two drives it is stored on

RAID Level 2

- Has not been used at least since 2014
  - Only mentioning it because it is interesting :P

- Called "Memory-style error correcting code organization"
  - Uses **bit-level** striping.
  - Some bits used for parity or error detection

- Example: if we write 32 bits of data, there are 7 bits of error correcting bits. Means this can be split up across 39 disks at most

RAID Level 2

- Advantadges:
  - Significant parrallelism
  - Error correcocting bits can recover all data if a disk crashes

- Disadvantadges:
  - Drives have to be synchronized for a write at the same time.
    If we are writing one block, that can use many disks and all disks using that block cannot handle other requests till the block is written.
  - Overhead in putting together reads from different disks
  - Crash recovery is much slower

RAID 3

- Similar to RAID 2, but stripes at the byte level

- Not as much overhead as RAID 2, and still allows for highly parallelized reads. In particular, sequential reads and writes are the best

Aside: XOR

- XOR is a boolean/bit level operation
- Exclusive OR, XOR, C operator **^**

A	**B**	**A ^ B**
0	0	0
0	1	1
1	0	1
1	1	0

- XOR is associative
  - `(A ^ B) ^ C == A ^ (B ^ C)`
- You can think of it as “1” if there is an odd number of “1” inputs.

Aside: XOR

- You can think of it as “1” if there is an odd number of “1” inputs.

A	**B**	**C**	**A ^ B ^ C**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- Some XOR identities:
  - A ^ A = 0
  - 0 ^ A = A
  - 1 ^ A = ~A (negation of A)

Poll Everywhere

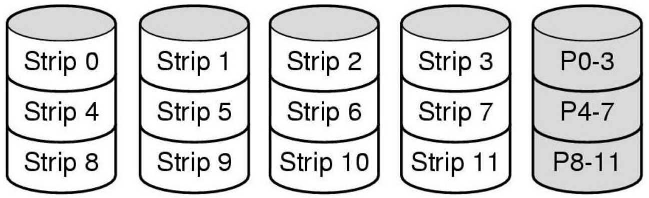
pollev.com/tqm

- Find the missing value:
  `0xC ^ 0x7 ^ ???? ^ 0x1 = 0x5`

- `1100 ^ 0111 ^ ???? ^ 0001 = 0101`

RAID 4

- RAID 4 is like RAID 0, but we have an extra disk dedicated to storing the parity (which you just sorta calculated!)

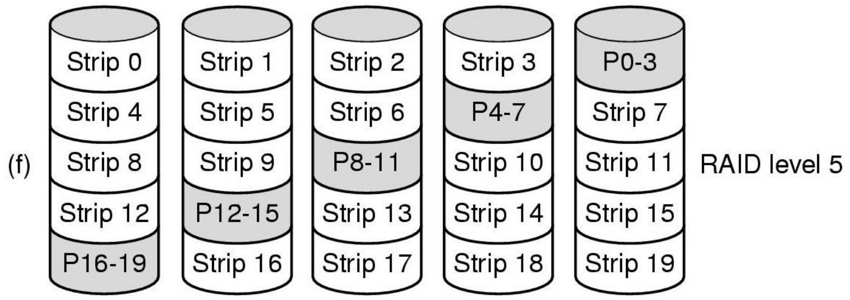
(e)


RAID level 4

- In this example:
  - the blocks dedicated to storing parity are sharded.
  - P0-3 is the XOR of strip 0, 1, 2, and 3.
- We only need to have 1 disk dedicated to error recovery
- **If one disk fails, we can recalculate it by using the parity and the data of the other disks**

RAID 5

- Same as RAID 4, but we distribute the parity across different disks.



- Why?
  - Whenever we write to a block, we must also update the parity. In RAID 4, this means every write had to go also go through the parity disk, which means we can't parallelize write requests :/
  - In RAID 5, we distribute the parity so that the workload of managing parity is spread across all disks.