Disks and Parallelization Practice

2025 Winter ECE353: Systems Software
Jon Eyolfson

Lecture 16
2.0.0

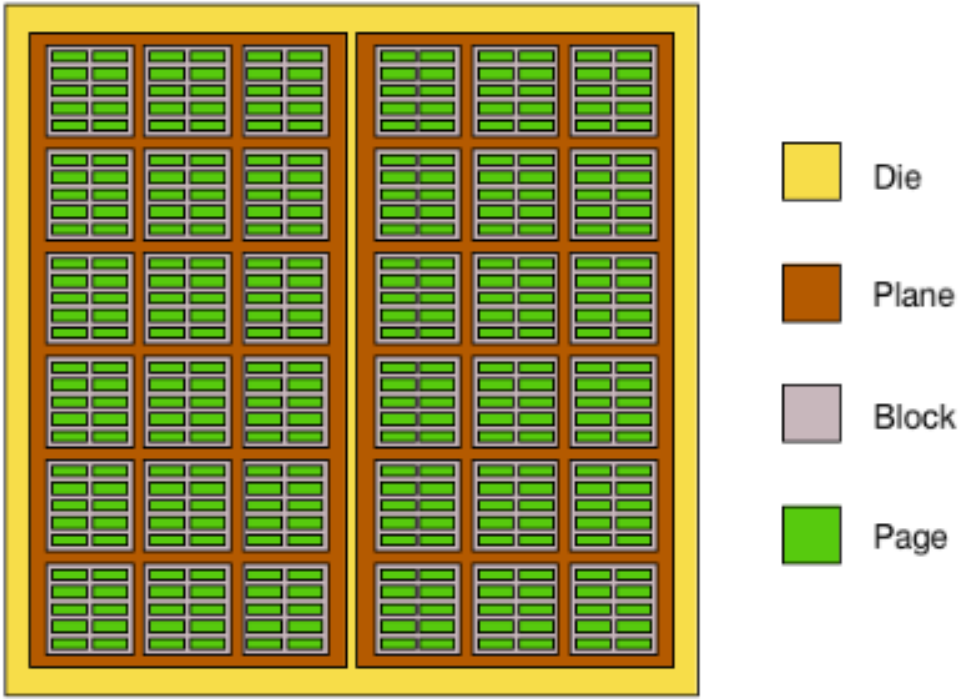
Solid State Drives (SSD) Are More Modern

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 HDDs

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)

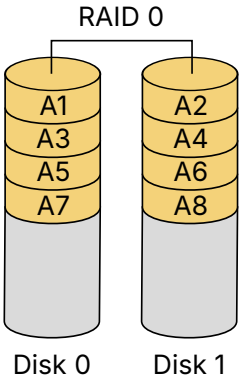
The OS can use the *TRIM* command to inform the SSD a block is unused
  The SSD can freely erase the block without moving overhead

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



RAID 0 is For Performance Only

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

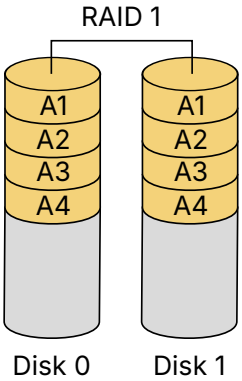
Pro

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



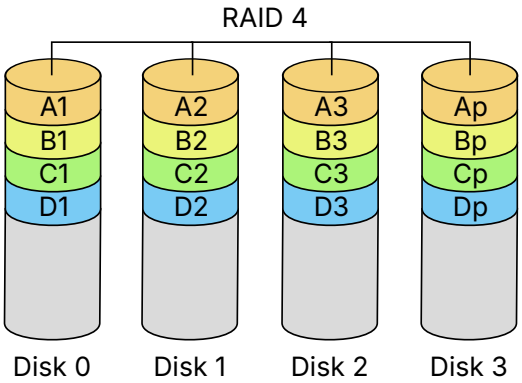
RAID 1 is Simple, But Wasteful

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



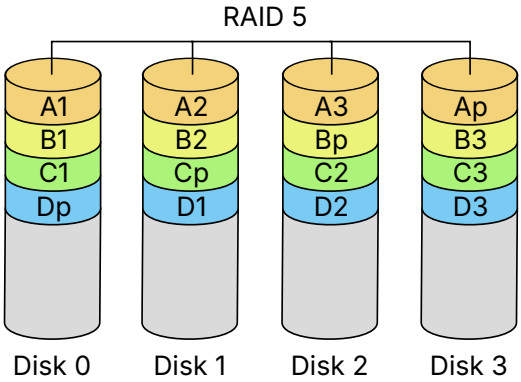
RAID 4 Can Use the Parity Drive to Recover

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

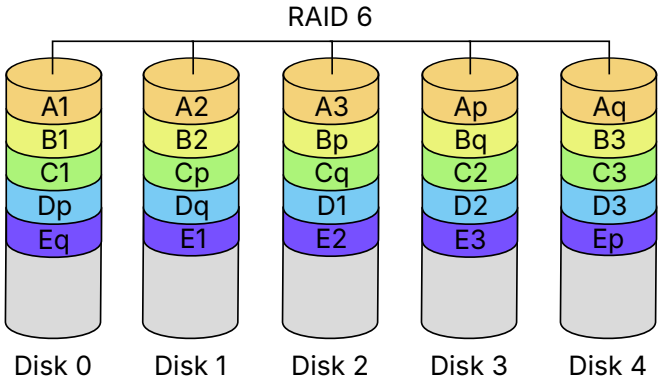


RAID 5 is an Improved Raid 4

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



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 topics: SSDs and RAID

- 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

Let's Practice Parallelization

We're Running a Bank Doing 10,000,000 Transfers

We have a simulation that can create varying number of accounts

Each account has a unique ID, and a balance (starting with $1 000)

We generate transfers between random accounts for 10% of their current balance

They must call `securely_connect_to_bank` before starting the transfer

Coding Example

Done live! We'll have data races and deadlocks

Our goal is to be faster than 11 seconds

You can find the template in the `examples` repository

To compile it, run the following commands:

```
cd lectures/16-disks-and-parallelization-practice # if not already there
meson setup build
meson compile -C build
```

Run the program using: `./banksim <num_accounts>`