CSE 451: Operating Systems

Spring 2022

Module 13

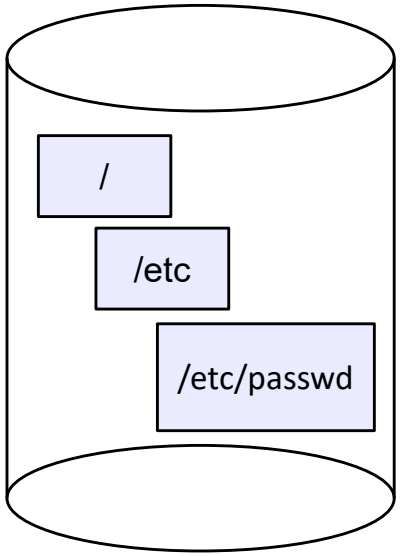
Redundant Arrays of Inexpensive Disks (RAID)

John Zahorjan

Background

- We start with very static formatting
  - Superblock/inode/free block map locations are absolute positions on disk

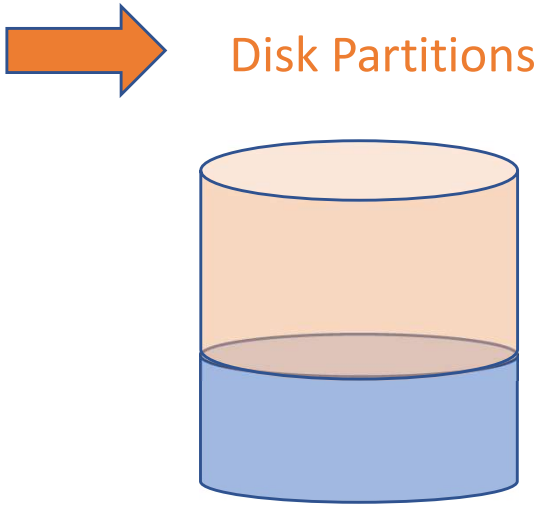
- 1 disk = 1 file system



Disk 1
(C drive)

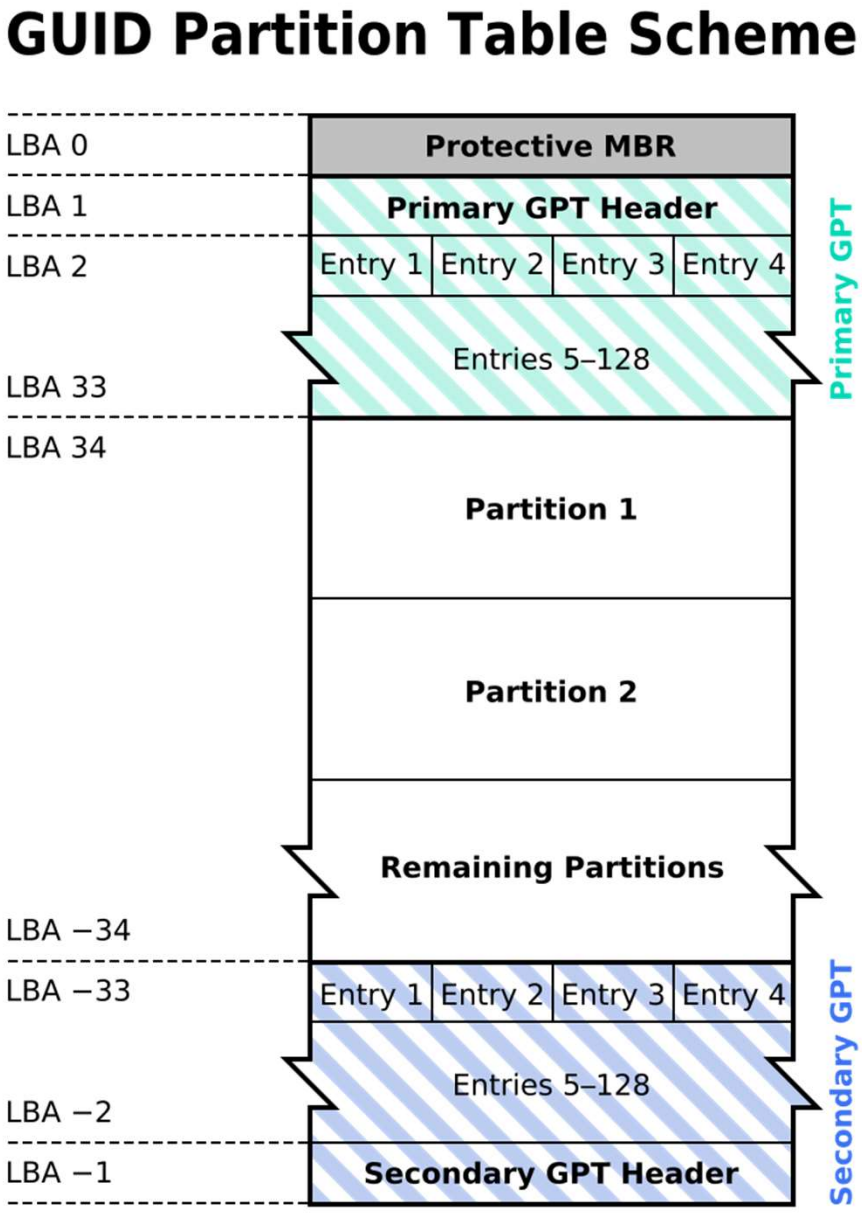
Background (cont.)

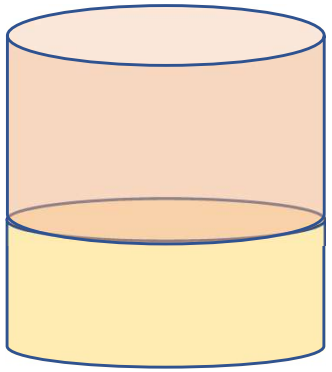
- One file system = one disk was too rigid
  - If file system was corrupted, you lost everything
  - Could have only one file system
    - One blocksize, for instance
  - Backups are often done on a file system, so schedule for most important and least important data would be the same
    - This is a logical backup – a backup of the ADT that is the file system
  - Disk backups (copying the disk at the block level) involved the entire disk



One device, two partitions

Partitions



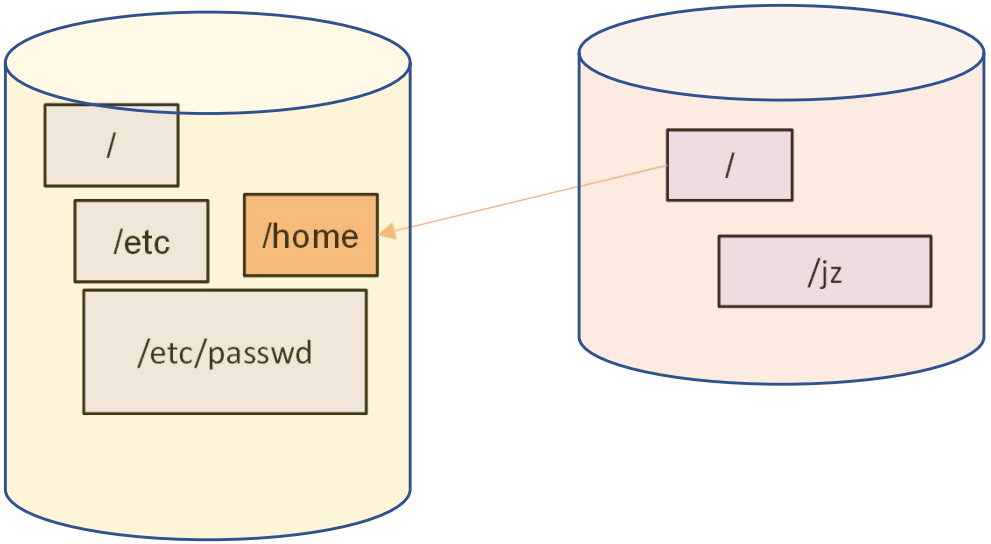


GUID partition entry format

Offset	Length	Contents
0 (0x00)	16 bytes	Partition type GUID (mixed endian[6])
16 (0x10)	16 bytes	Unique partition GUID (mixed endian)
32 (0x20)	8 bytes	First LBA (little endian)
40 (0x28)	8 bytes	Last LBA (inclusive, usually odd)
48 (0x30)	8 bytes	Attribute flags (e.g. bit 60 denotes read-only)
56 (0x38)	72 bytes	Partition name (36 UTF-16LE code units)

Using Multiple Partitions (Linux)

- There is a single tree of names
  - So there is a single root, "/"
  - (Contrast with Windows, which has a forest: C:\, D:\, ...)
- Use "mount" to extend file system namespace to span multiple devices (or partitions)
  - *$ mount /dev/sda2 /home*



Disks or partitions

/etc/fstab

Configuration file for use during boot

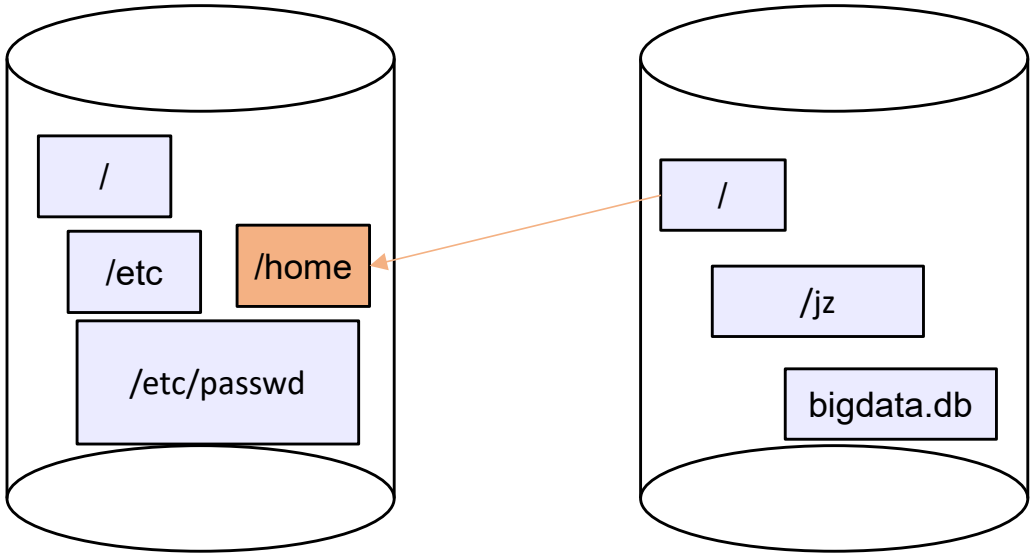
```
# /etc/fstab
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#

UUID=8e3edd2f-eb93-4876-9d76-a929a5ac6fd9 /                       zfs     rw,nodev         0 0
UUID=d3b5ced1-a961-40a8-8585-acf3a2676949 /boot                   ext4    rw,nosuid,nodev 1 2
UUID=82f54c67-e64c-444f-8296-4961b20e283e /tmp                    zfs     rw,nosuid,nodev 0 0
UUID=8b1dc520-3a63-4c48-9bd4-871894cf9cdb /var                    zfs     rw,nosuid,nodev 0 0
UUID=2b2fd4ea-0c2e-4e3e-a4c0-8a7ce31e0347 swap                    swap    defaults         0 0
```

(Finally...) RAID

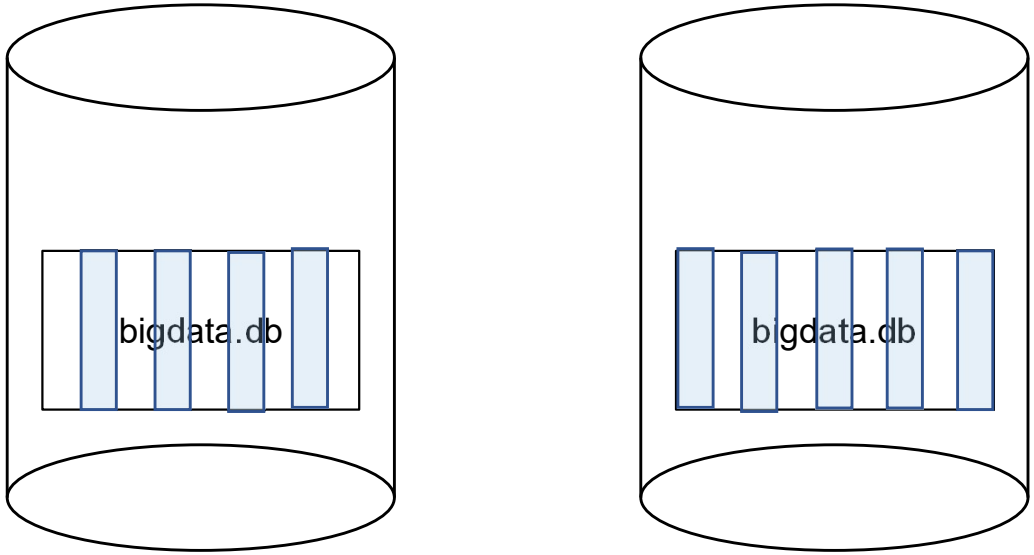
- Redundant Array of Inexpensive Disks
- “Disks are cheap” means bytes are cheap
- Bytes are cheap means you can afford to waste them if it helps you achieve other goals
- What goals are there, besides capacity?
  - Performance

Two disks vs. one



- How is “peak performance” affected?
  - Are read times cut in half? Is write throughput doubled?
- Can we do better?

Improving “Single Threaded” Performance



- If we locate data of individual files on multiple devices, we can improve read/write peak performance even for individual files
- This is called striping

Raid Basic Idea

- Improve performance by *striping* individual files across multiple disks
  - we can use parallel I/O to improve access time even when overall I/O demand is bursty/low
  - but...
- More disks → more disk failures
  - 10 disks have about 1/10th the MTBF (mean time between failures) of one disk, and...
  - if files are striped, any single disk failure causes loss of every file
- So, we want striping for performance, but we need something to help with reliability

Reliability through Redundancy

- The issue: disk failure
  - not software failure (it's not journaling/crash tolerance)
  - not user error (it's not backup)
- To achieve reliability, add redundant data that allows a disk failure to be tolerated
  - We'll see how in a minute
- At the scales we're currently considering (tens of disks), it's typically enough to be resilient to the failure of a single disk
  - What are the chances that a second disk will fail before you've replaced the first one?
    - Er, it has happened to us!
- So:
  - Obtain performance from striping
  - Obtain reliability from redundancy

RAID

- RAID: Redundant Array of Inexpensive Disks

- Disks are cheap, so it's easy to put lots of disks (10s, say) in one box for increased storage, performance, and availability

- Data plus some redundant information is striped across the disks in some way

- How striping is done is key to performance and reliability

RAID Implementation

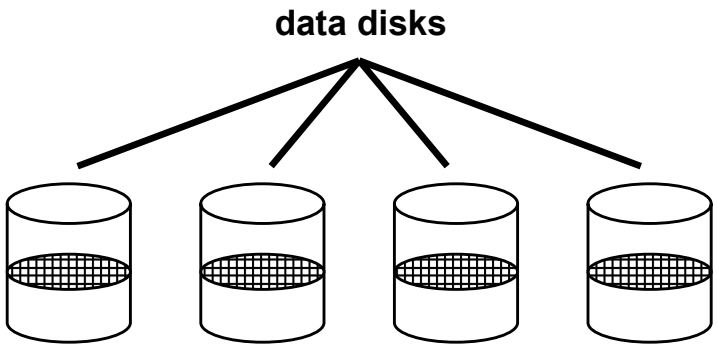
- Option A: hardware
  - The hardware RAID controller deals with this
    - From the OS's perspective, the multi-disk RAID looks like one big array of blocks
- Option 2: software
  - A low level layer of the OS knows there are multiple disks, but presents them to upper layers as a single block device
    - That is, it does what the hw RAID controller does
- It doesn't matter to what follows which approach is used

Some RAID tradeoffs

- Granularity
  - fine-grained: stripe each file over all disks
    - high throughput for the file
    - limits transfer to 1 file at a time
  - coarse-grained: stripe each file over only a few disks
    - limits throughput for 1 file
    - allows concurrent access to multiple files
- Redundancy
  - uniformly distribute redundancy information on disks
    - avoids load-balancing problems
  - concentrate redundancy information on a small number of disks
    - partition the disks into data disks and redundancy disks

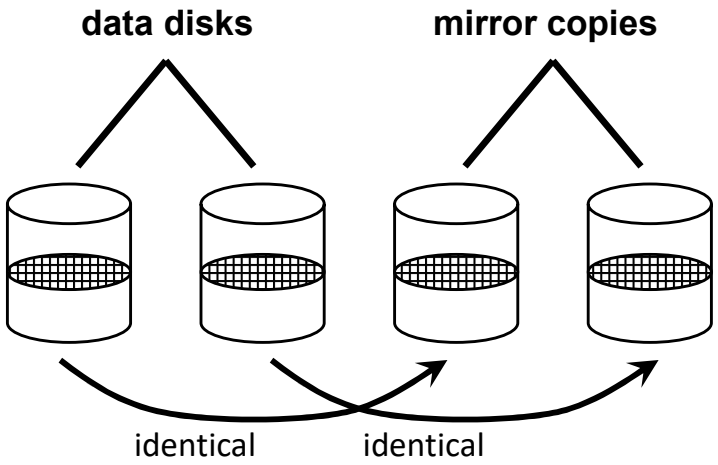
RAID Level 0: Non-Redundant Striping

- RAID Level 0 is a non-redundant disk array
- Files/blocks are striped across disks, no redundant info
- High (single-file) read throughput
- Best write throughput (no redundant info to write)
- Maximum use of disk capacity
- Any disk failure results in data loss

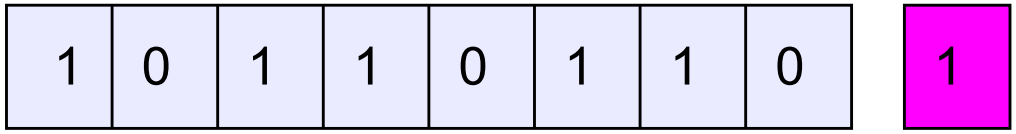


RAID Level 1: Mirrored Disks

- Files are striped across half the disks, and mirrored to the other half
  - 2x space expansion
- Reads: Read from either copy
  - read time is fastest read among copies
- Writes: Write both copies
  - write time is slowest write among copies
- On single drive failure, just use the surviving disk during repair
  - If two disks fail, you rely on luck…



Prelude to RAID Levels 2-5: A parity refresher



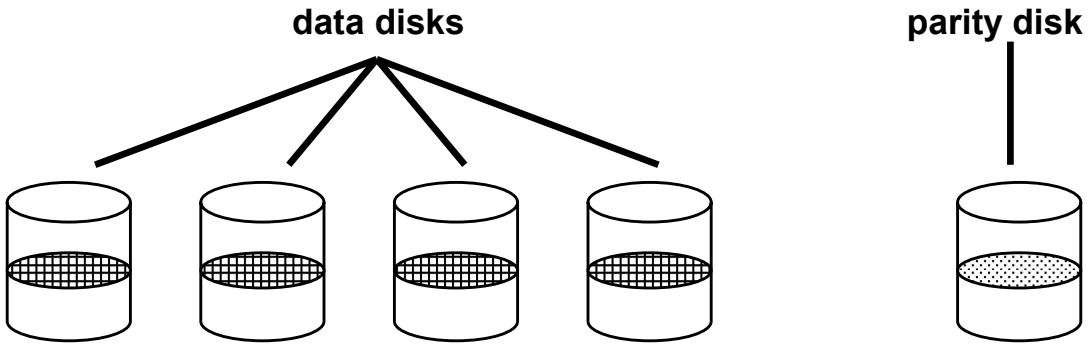
- To each byte, add a bit whose value is set so that the total number of 1's is even
- Can detect any odd number of bit errors
  - If an odd number of bits have their values flipped, the overall parity won't be even, so you'll know something is wrong
- Can correct a single error if the error is a bit "goes missing"
  - (next slide)
- More sophisticated schemes, called ECC (error correcting codes), can correct multiple bit errors at the cost of requiring more "extra bits"

RAID Levels 2, 3,a and 4: Striping + Parity Disk

- RAID levels 2, 3, and 4 use parity or ECC disks
  - e.g., each byte on the parity disk is a parity function of the corresponding bytes on all the other disks
  - details between the different levels have to do with kind of ECC used, and whether it is bit-level, byte-level, or block-level

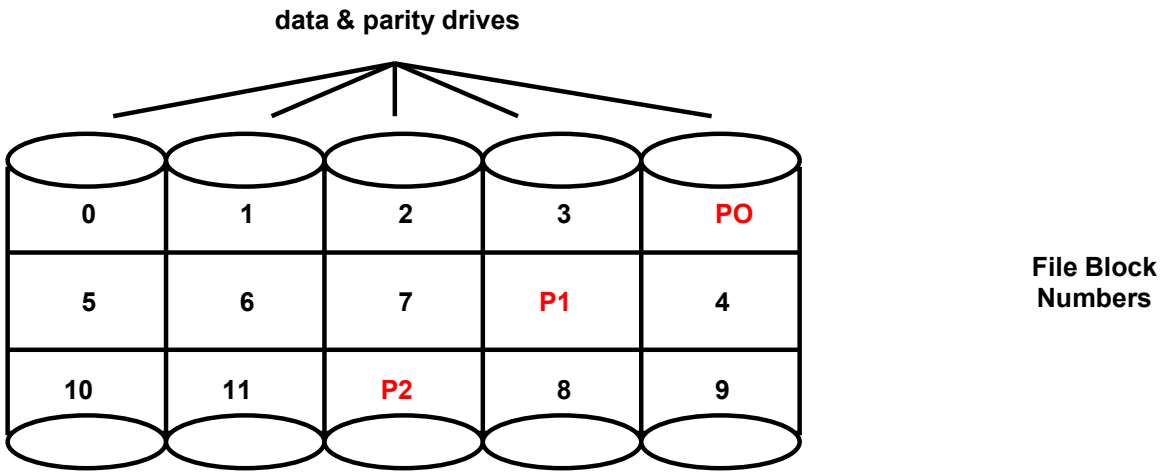
- A read accesses all the data disks
- A write accesses all the data disks plus the parity disk

- To recover from a single disk failure, read the remaining disks (including the parity) disk to compute the missing data



RAID Level 5

- RAID Level 5 uses block interleaved distributed parity
- Like parity scheme, but distribute the parity info (as well as data) over all disks
  - for each block, one disk holds the parity, and the other disks hold the data



- Significantly better performance
  - every write of a block must modify the corresponding parity block
    - new parity block = (old data block) ^ (new data block) ^ (block parity block)
  - parity disk is not a hot spot

RAID Level 6

- Basically like RAID 5 but with replicated parity blocks so that it can survive two disk failures.
- Useful for larger disk arrays where multiple failures are more likely.

RAID Summary

- Why use multiple disks (vs. one bigger disk)?
- What kinds of errors is RAID designed to protect against?
- If you have RAID, do you need journaling?
- If you have RAID, is a log structured file system of any use?
- If you have RAID, do you need file system backups?
- Is there any realistic situation in which you might lose “too many” disks at once?
  - For example, all of them?