

CSE 451: Operating Systems

Winter 2013

Redundant Arrays of Inexpensive Disks (RAID) and OS structure

Gary Kimura

The challenge

- Disk transfer rates are improving, but much less fast than CPU performance
- We can use multiple disks to improve performance
 - by *striping* files across multiple disks (placing parts of each file on a different disk), we can use parallel I/O to improve access time
- Striping reduces reliability
 - 100 disks have 1/100th the MTBF (mean time between failures) of one disk
- So, we need striping for performance, but we need something to help with reliability / availability
- To improve reliability, we can add redundant data to the disks, in addition to striping

RAID

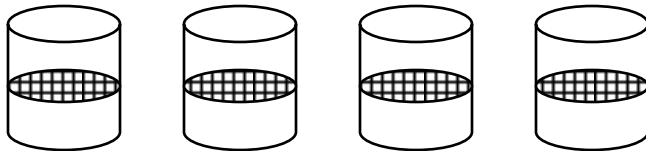
- A **RAID** is a **Redundant Array of Inexpensive Disks**
- Disks are small (physically) and cheap, so it's easy to put lots of disks (10s to 100s) in one box for increased storage, performance, and availability
- Data plus some redundant information is striped across the disks in some way
- How striping is done is key to performance and reliability

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

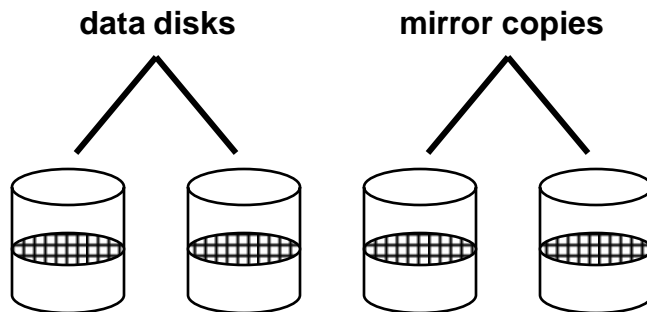
RAID Level 0

- RAID Level 0 is a non-redundant disk array
- Files are striped across disks, no redundant info
- High read throughput
- Best write throughput (no redundant info to write)
- Any disk failure results in data loss; sometimes a file, sometimes the entire *volume*



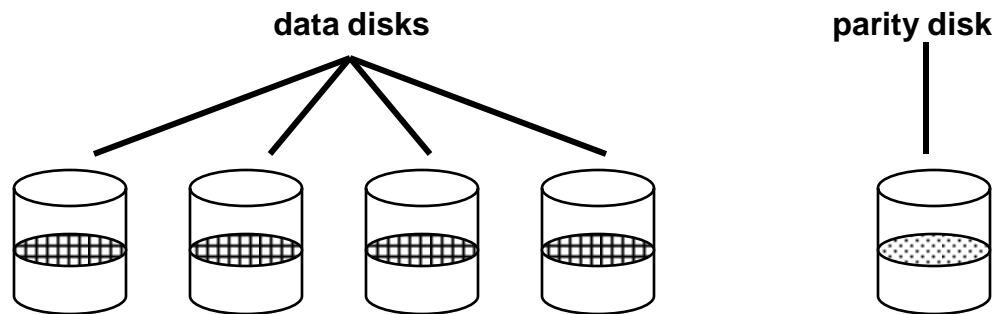
RAID Level 1

- RAID Level 1 is mirrored disks
- Files are striped across (half) the disks
- Data is written to multiple (two) places – data disks and mirror disks
- On failure, just use the surviving disk(s)
- Factor of N (2x) space expansion

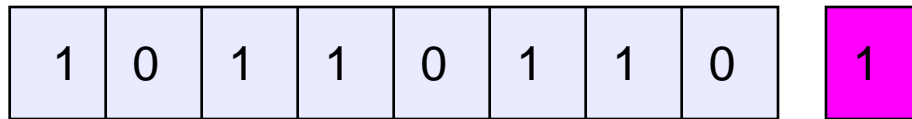


RAID Levels 2, 3, and 4

- RAID levels 2, 3, and 4 use ECC (error correcting code) or parity disks
 - E.g., each byte on the parity disk is a parity function of the corresponding bytes on all the other disks
- A read accesses all the data disks
- A write accesses all the data disks plus the parity disk
- On disk failure, read the remaining disks plus the parity disk to compute the missing data



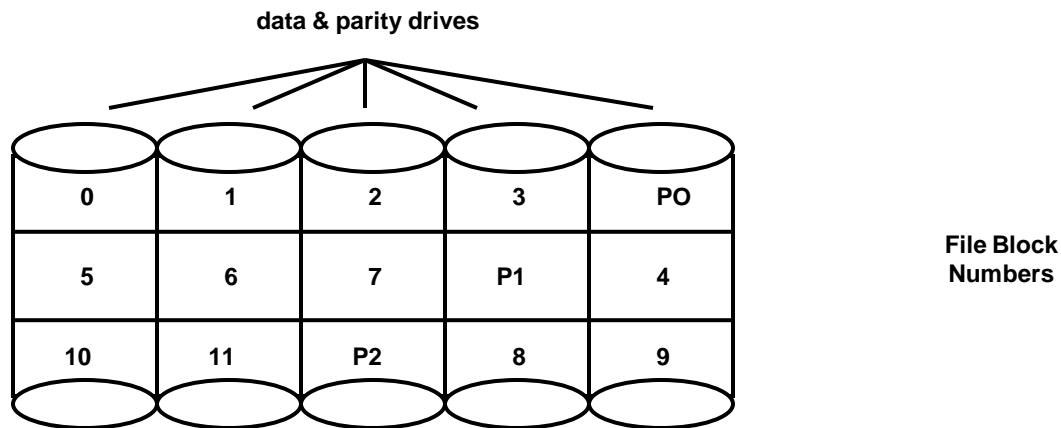
Refresher: What's parity?



- To each byte, add a bit set so that the total number of 1's is even
- Any single missing bit can be reconstructed
- (Why does memory parity not work quite this way?)

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
 - parity disk is not a hot spot

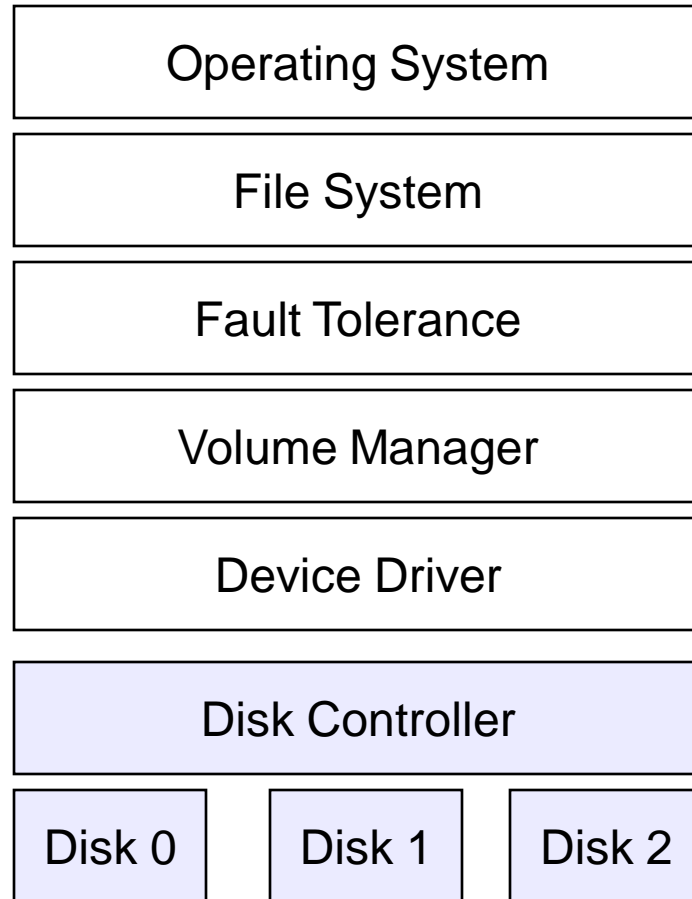


Management

- Multiple spindles complicates things
 - Visible to: Apps? File system?
 - Allocation of space (per user on a “file server”)
 - SANs
 - Differing disk sizes
- Partitions!
 - Logical division of spindle into blocks
 - Assemble volumes from partitions

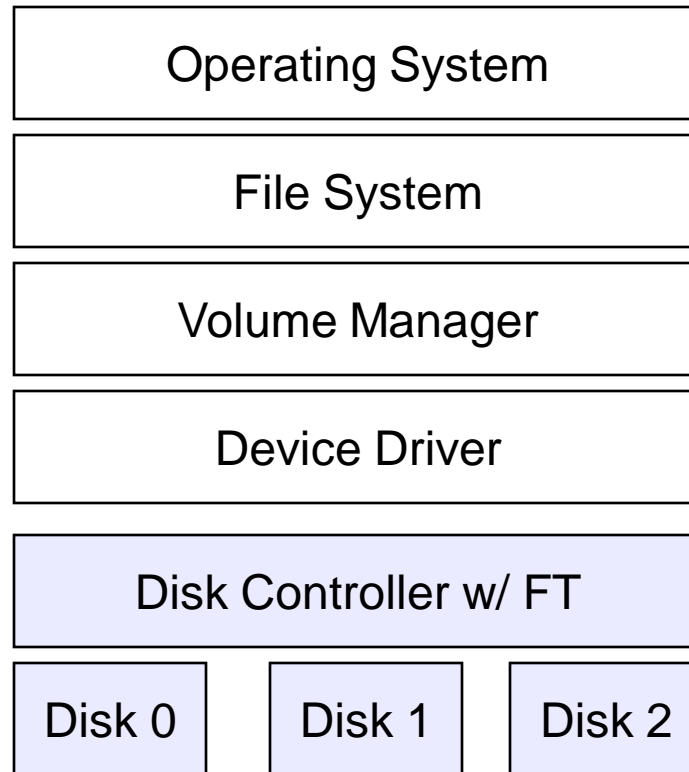
OS and Hardware Layering

- Who needs to know?
 - FS: probably not



OS and Hardware Layering

- Why put FT into hardware?
 - Speed
 - Simplicity of OS
- Why not?
 - Fixes
 - Cvl data



One tiny problem with RAID and multiple spindles...

- How do you boot?
 - ROM on motherboard doesn't have a lot of room for logic
 - Boot zone on disk doesn't have a lot of room for logic (and needs to be replicated, striped, etc.)
 - Put partition info into "well-known place"
 - Put RAID info into "well-known place"
 - Still... too complicated
- "Real systems" ...
 - Don't boot from multi-partition volumes
 - Boot from RAID 0, at most RAID 1 (simple failover)