Section 9

Eric Wu (ericwu@cs)

Today

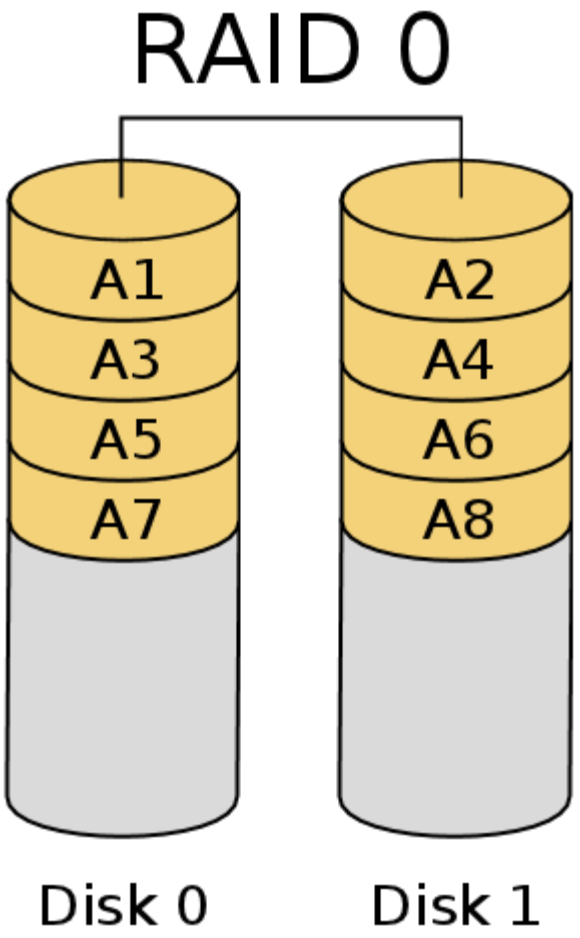
- RAID
- More File Systems
- Project 4

RAID

- Redundant Array of Inexpensive Disks
- Improves performance and reliability of storage

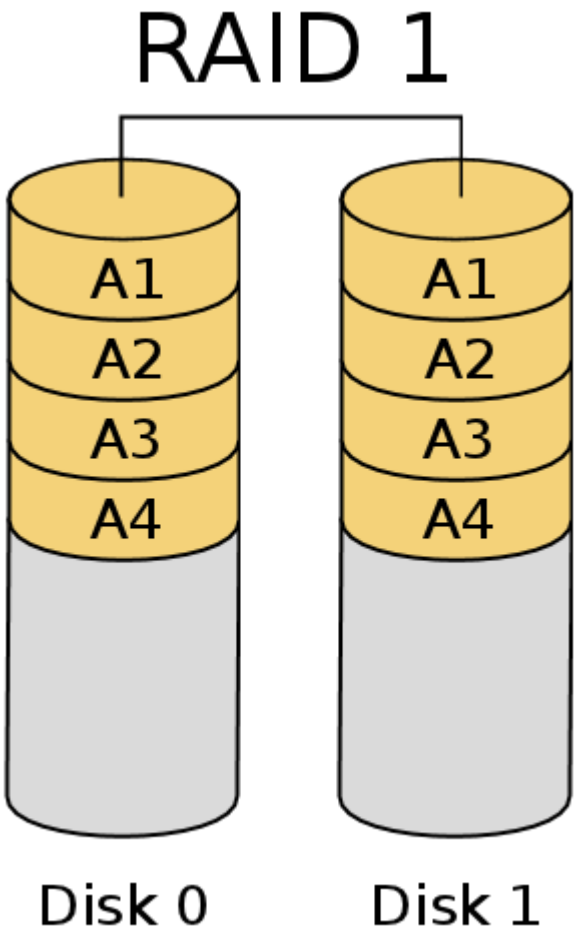
RAID Level 0

- Striping files/blocks across multiple disks
  - No redundancy
  - No fault tolerance



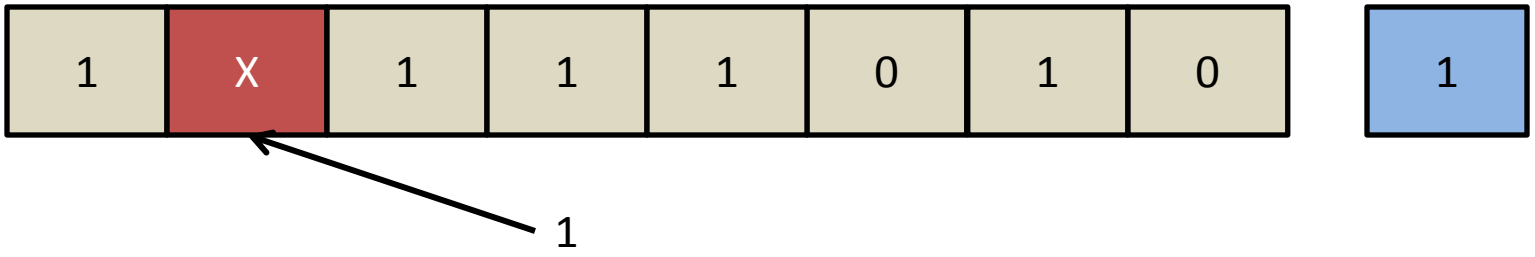
RAID Level 1

- Mirroring files/blocks across multiple disks
  - Redundancy
  - Fault tolerance
- Read from any disk
- Write to all disks



Parity

- Error detection and correction
- For each byte, add a bit whose value is 1 if the number of 1's in the byte of data is even.
- Can reconstruct any one corrupted bit in the byte of data.



RAID Level 2-4

- Varying levels of parity using a parity disk
- Striping on bits (2), bytes (3), or blocks (3)
- On write, separate writes to each disk, plus parity disk.
- On read, must read from all disks.
- Cheaper storage cost for fault tolerance.

RAID Level 5

- Like RAID Levels 2-4, but distributes parity data among all disks
- Better performance
  - Why?
- Better fault tolerance
  - Why?

More File Systems

- Last time we covered Unix File System and File Allocation Table
  - FAT uses table to look up data clusters
  - UFS uses i-nodes instead to more efficiently use space
- Today we will look at:
  - Journaling FS
  - Log-based FS

Crash Recovery?

- FAT and Unix FS both use caches in memory
  - …but what if a crash occurs?

Crash Recovery?

- FAT and Unix FS both use caches in memory
  - …but what if a crash occurs?
  - Bad news!

http://networkequipment.net/2011/04/04/nasa-network-could-be-exposed-to-cyber-attack/

Journaling FS

- Keeps a (sequential) log of all writes in a file on disk.
  - When changes to target files occur, write them to log instead of to the actual file data locations.
  - Execute the changes in the log sequentially whenever the disk is ready.

Redo Logging

- Log: an append-only file containing log records
  - <start t>
    - transaction t has begun
  - <t,x,v>
    - transaction t has updated block x and its new value is v
      - Can log block “diffs” instead of full blocks
  - <commit t>
    - transaction t has committed – updates will survive a crash

In case of crash…

- Read the log file
  - Ignore all uncommitted transactions
  - Redo only committed transactions
    - Do these all in chronological order
    - Ensures *atomicity* of the data being written

Log-based FS

- Journaling FS uses a log… why not make the file system a log?!
- Writes are all performed sequentially on disk
  - i-nodes also go into the log!
- Reading uses i-node map at beginning of log to locate i-nodes

Space Management

- File updates and deletes are costly
- Log can be compressed periodically to allocate more space

What about crashes?

- Log-based FS is a log!
  - Log will always be in a consistent state
- Recovery ends at the last write checkpoint
  - Toss all unfinished transactions
  - Update the i-node map to the latest checkpoint

Project 4

- More hints…
  - Changing volume label should NOT trigger an immediate change to the file system (except the volume label itself)
  - Consider where **dirent** structs are swapped. Look at the FatSetRenameInfo function.
  - Discuss!