Persistence: Data Integrity and Protection

CS 537: Introduction to Operating Systems

Louis Oliphant

University of Wisconsin - Madison

Fall 2023

Administrivia

- Project 7 due Dec 8th @ 11:59pm
- Projects 5 & 6 Graded
  - Issues with grading? Fill out form sent on Piazza
- Midterm 3 scheduled for Dec 12th in-class
  - Alternate time Dec 18th @ 12:25pm (email me)
  - Alternate time also for McBurney accommodations
  - Location Van Vleck B102

Review LFS & SSD

Log File System

- Layout on disk – checkpoint region, segments (data, inodes, imap, segment summary),
- Memory caching – imap and buffered writes
- Garbage Collection – block liveness, which blocks to clean
- Crash Recovery – multiple CRs, roll forward

SSD

- Physical Storage System
- Flash-based Operations
  - Read (a page), Erase (a block), Program (a page)
- Log-Structured FTL
- Garbage Collection / Mapping Tables
- Wear Leveling / Over Provisioning

Quiz 21 LFS

https://tinyurl.com/cs537-fa23-q21

Data Integrity & Protection

- Disk Failure Modes
- Handling Latent Sector Errors
- Detecting Corruption
  - Checksum Functions
    - xor, addition, Fletcher checksum, CRC
  - Cheksum Layout
- Misdirected Writes
- Lost Writes
- Scrubbing

Disk Failure Modes

- Entire Disk Fails (considered in RAID systems)
- **Latent Sector Errors**
  - Performing I/O to a sector returns an error (unable to read/write to sector)
  - Sector has been damaged
- **Block Corruption**
  - Performing I/O to sector returns bad data

	Cheap	Costly
LSEs	9.40%	1.40%
Corruption	0.50%	0.05%

3 year study of 1.5 million disk drives

Latent Sector Errors

- In RAID systems, when LSE detected, use redundancy of data to return correct data
- When full-disk error occurs and reconstruction of disk is happening, if LSE occurs, data can not be recovered
  - Because of this, extra degree of redundancy can be used (e.g. two parity blocks)

Detecting Corruption – Checksums

- **Detection** is the key problem (unlike LTEs), once detected recovery is the same as LTEs
- **Checksum** functions are used and values stored with data to detect corruption
  - For each 4KB block, calculate a 4 or 8 byte value

Common Checksum Functions - XOR

- e.g. bitwise xor of 16-byte block to calculate 4-byte checksum

```
0x 365e c4cd ba14 8a92 ecef 2c3a 40be f666

0011 0110 0101 1110   1100 0100 1100 1101
1011 1010 0001 0100   1000 1010 1001 0010
1110 1100 1110 1111   0010 1100 0011 1010
0100 0000 1011 1110   1111 0110 0110 0110
-----------------------------------------
0010 0000 0001 1011   1001 0100 0000 0011

0x 201b9403
```

- Has limitations, if two bits in same unit changed, will not be detected

Common Checksum Functions - Addition

- Addition, ignoring overflow, e.g. 4-byte block, calculate 1-byte checksum

```
 0x d8c26b42

  1101 1000
  1100 0010
  0110 1011
+ 0100 0010
  ---------
  0100 0111

  0x 47
```

- Detects many changes, not good if the data is shifted

Common Checksum Functions - Fletcher checksum

- Computes two check bytes, s1 and s2, looping over all bytes, d_i, of data:
  - $s1 = (s1 + d_i) \mod 255$
  - $s2 = (s2 + s1) \mod 255$
- Detects all single-bit and double-bit errors and many burst errors
- Example of 4-byte block, calculating S1 and S2 bytes

Data	S1	S2
	0	0
216	216	216
194	410	626
107	517	1143
66	583	1726
—	—	—
%255	73	196

Common Checksum Functions - Cyclic Redundancy Check (CRC)

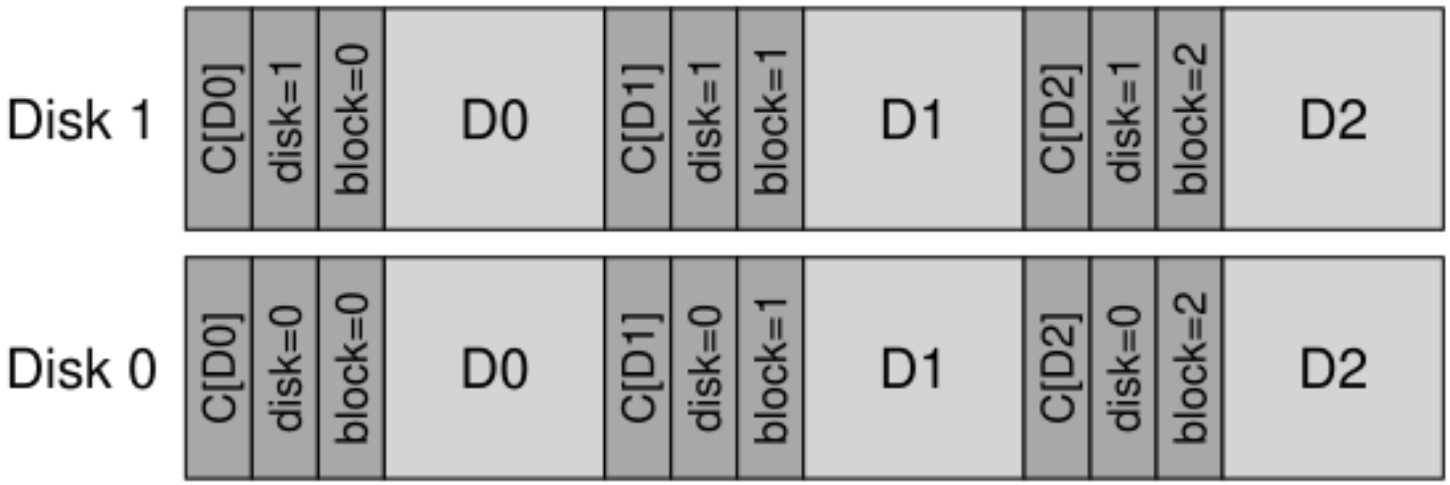
- Treat entire data block as one large binary number, D.
- Divide by an agreed upon value, k.
- The remainder of the division is the CRC value.

Using checksums

- **Collisions** are possible with all checksum functions
  - two non-identical data blocks have identical checksum values
- Use checksum when data is loaded
  - Recalculate checksum and see if matches stored checksum value

Misdirected Writes

- Write data to disk correctly, but to wrong location
- Can't detect with just a checksum
- Write a **physical identifier** along with checksum to detect



Lost Writes

- Device informs the FS that a write is complete, but in fact it never persisted
- Possible solution: **read-after-write**, but this doubles I/O
- Another solution: store checksum in different location, e.g. with file's inode

Scrubbing

- Data is checked when it is loaded, but what about data that is rarely loaded?
- **Scrubbing** is the process of periodically reading through *every* block of a system and verifying checksums.
- Typically systems schedule scans on a nightly or weekly basis.

Quiz 22: Data Integrity

- Open quiz (you can submit as often as you like and get immediate feedback)

https://tinyurl.com/cs537-fa23-q22

Student Projects

- SSD benchmarking
  - Weitao Su and Wenpei Shao (support from Vojtech Aschenbrenner)
- xv6 Console
  - Arthur Wang
- MadDisk
  - Forrest Dai and Ying-Fang Jaw (Hack-a-thon project)