Data Integrity and Protection

CSC 343, Operating Systems

Topics covered in this lecture

- Data Integrity and Protection

This slide deck covers chapters 45 in OSTEP.

Disk Failure Modes

- Common and worthy failures are frequency of latent-sector errors (LSEs) and block corruption.
  - Latent-sector errors arise when a disk sector has been damaged in some way
  - Block corruption is where data becomes corrupt in a way undetectable by the disk itself.

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

Disk Failure Modes

- Frequency of latent-sector errors (LSEs)
  - Costly drives with more than one LSE are as likely to develop additional LSEs
  - For most drives, annual error rate increases in year two
  - LSEs increase with disk size
  - Most disks with LSEs have less than 50
  - Disks with LSEs are more likely to develop additional LSEs
  - There exists a significant amount of spatial and temporal locality
  - Disk scrubbing is useful (most LSEs were found this way)

Disk Failure Modes

- Block corruption:
  - Chance of corruption varies greatly across different drive models
  - Effects of age are different across models
  - Workload and disk size have little impact on corruption
  - Most disks with corruption typically have very few corruptions
  - Corruption is not independent with a disk or across disks in a RAID
  - There exists spatial locality, and some temporal locality
  - There is a weak correlation with LSEs

Handling Latent Sector Errors

- Latent sector errors are easily detected and handled
- Using redundancy mechanisms:
  - In a mirrored RAID or RAID-4/RAID-5 system based on parity, the system should reconstruct the block from the other blocks in the parity group.

Detecting Corruption: The Checksum

- How can a client tell that a block has gone bad?
- Using checksum mechanisms:
    - This is a function that takes a chunk of data as input and computes a small summary of the content of the data

Common Checksum Functions

- Different functions are used to compute checksums
  - A simple checksum function is based on exclusive or (XOR): divide the data into equal-sized bitstring (with padding if necessary) and keep a running bitwise XOR.
  - XOR is a reasonable checksum but has limitations; when two bits in the same position within checksumed unit change, the checksum will not detect the corruption.

Common Checksum Functions

- Addition Checksum
  - Compute the 2's complement addition over each chunk of the data
  - This approach has the advantage of being fast.
- Fletcher Checksum
  - Compute two check bytes, $s1$ and $s2$
  - Assuming a block D consists of bytes $d_1, \ldots, d_n$
    - $s_1 = s_1 + d_i \bmod 255$ (compute over all d_i)
    - $s_2 = s_2 + s_1 \bmod 255$ (again, compute over all d_i)
- Cyclical redundancy check (CRC)
  - Treat D as if it is a large binary number and divide it by an agreed upon value; the remainder of this division is the value of the CRC

Checksum Layout

- The disk layout without checksum

- The disk layout with checksum

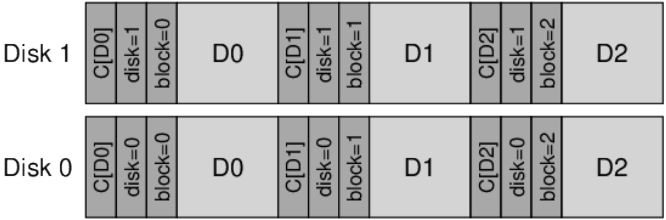
  - Store the checksums packed into 512-byte blocks

Using Checksums

- When reading a block D, the client reads its checksum from disk $Cs(D)$, stored checksum
- Compute the checksum over the retrieved block D, computed checksum $Cc(D)$
- Compare the stored and computed checksums
  - If they are equal the data is safe
  - If are not equal, the data has changed since the time it was stored.

Other Problems

- Modern disks have a couple of unusual failure modes that require different solutions
- Misdirected writes arise in disk and RAID controllers when the data is written correctly, but to the incorrect location



- Lost writes occur when the device informs the upper layer that a write has completed, but in fact is never written.

Scrubbing

- When do these checksums actually get checked?
  - Most data is rarely accessed, and thus remains unchecked
- To remedy this problem, many systems utilize disk scrubbing
  - Periodically read through every block of the system
  - Check whether checksums are still valid
  - Reduce the chance that all copies of certain data become corrupted

Overhead of Checksumming

- Two distinct kinds of overhead: space and time
- Space overhead
  - Disk: a typical ratio might be an 8 byte checksum per 4 KB data block; a 0.19% on-disk space overhead
  - Memory: this overhead is short-lived and not much of an issue
- Time overhead
  - The CPU must compute the checksum of each block; to reduce CPU overhead combine data copying and checksumming into one activity