The Design and Implementation of a Log-structured File System

TOCS'92

201924169 Ju-Won Park

May 19, 2019

Problem statement

- Disk Hardware File System speeds haven't kept pace with rapidly increasing raw processing power.
- As a result, we need to use existing hardware in innovative ways to compensate. Random disk access, as in the Unix FFS, is expensive.

Solution approach

- The paper introduces the Sprite Log-based File System.
- The Log-based File System is innovative in buffering and serializing writes to reduce required write seeks, increasing write speed, and not impacting read speed.

Strong points

- LFS' method of buffering and sequentially writing to the head of the log greatly increased write bandwidth by reducing required seek-on-write, and introduced a new file system paradigm.
- The crash recovery mechanisms are also significant contributions, including the system of frequent checkpoints to assist in rebuilding the required ancillary data structures.

Weak points / Limitations

- The paper does not discuss redundancy at all.
- Internal fragments may occur for file size less than 10 blocks.
- LFS does not provide a guarantee fully-atomic save operations.

Questions

- How to deal with new hardware paradigms and their restrictions?
- How does the LFS design simplify crash recovery as compared to FFS or UFS?

New ideas / Comments

- LFS improves crash recovery because you know that in a log-structured storage the most recent changes are at the end of the log. In a FFS or UFS style system you would need to rescan all the metadata to tell where changes occurred. With LFS you would simply look at the latest checkpoint to see which segments are at the "end" of the log.