A Fast File System for UNIX

TOCS'84

201924169 Ju-Won Park

May 2, 2019

Problem statement

- The original 512-byte Unix file system offered very poor performance due to the small block size, limited read-ahead and mostly non-consecutive allocation of blocks for files and directories.
- There was a need for a faster file system which could atleast saturate the disk capacity.

Solution approach

- Several extensions to the original UNIX file system that improve the overall performance of the system and add a few new properties such as symbolic links and long file names.

Strong points

- Improved throughput by use of larger block size and intelligent storage allocation. Use of global data (including processor speed, controller capabilities) to make storage allocation decisions.
- Improved reliability by replication of superblock in multiple cylinder groups. This improves the chance of recovery in the event of a drive failure.
- Use of fragments to reduce the impact of wastage of disk space due to bigger block size for smaller files.
- New features are generated like long file names, file locking, symbolic links that cross volumes, atomic rename, quotas.

Weak points / Limitations

- This paper is difficult to have serious problems. Even if they didn't fully solve all the problems they took an existing system and provided substantial improvements with few drawbacks.

Questions

- I am wondering about the increased block size is fixed and so big (8KB) that it practically necessitated the introduction of a complication: the use of fragments to keep from wasting too much space.

New ideas / Comments

- I think the paper describes simple but very efficient ways of improving file system throughput and reliability. The fact that most of the concepts are still used in file systems 20+ years later says something. The only concept that probably is not so relevant today is the use of fragments - with the disk space costing nothing, savings provided by fragments isnt worth the complications it introduces.