CS 5600 Project Proposal

Team: Rene Adaimi, Brian Hayes and Christopher Stadler 03/19/2018

Our goal is to build a log-structured file-system (called s3LogFS) using Amazon S3 as the storage medium. It will be mountable in Linux using FUSE (https://github.com/libfuse/libfuse). The user will then be able to take advantage of the reliability and scalability of S3 without needing to use S3-specific tools to access their data. Note that we are not attempting to build a filesystem that can be mounted by multiple clients concurrently.

Novelty

There are several existing projects that use S3 as the basis for a filesystem. Our project will be differentiated primarily by its log structure. Existing projects either create one S3 object per file (https://github.com/s3fs-fuse/s3fs-fuse) or break files into multiple objects (https://github.com/s3ql/s3ql). Using a log structure we will instead store each segment as its own object in S3. Since a single segment may contain changes to many files this will reduce the number of requests to S3 for workloads that make small writes to multiple files. Amazon charges for each request (https://aws.amazon.com/s3/pricing/#Request_Pricing) so this could reduce cost to the user.

Challenges

We believe the main challenge is correctly implementing the log-structure in the time available, as it is fairly complex. Since we are using S3 for storage un-optimized performance will likely not be very good, so another challenge will be designing and implementing caching on top of the log-structure to achieve more reasonable performance. We are also choosing to implement the filesystem in Python as this should make development much faster, but this could make performance optimization difficult.

Milestones

1. Document design.
2. Implement s3LogFS S3 API.
3. Implement client side s3LogFS with common POSIX functions.
4. Integrate s3LogFS with FUSE and mount as filesystem.
5. Implement Garbage Collection.
6. Implement Crash Recovery.
7. Benchmark s3LogFS vs local and other remote FS.
8. Expand features, POSIX support and Performance tuning.
9. Write results of benchmarking and outline conclusions.

Grading

1. File System Design - 10%
2. Core File System Implementation - 50%
3. Garbage Collection - 10%
4. Crash Recovery - 10%
5. Analysis of s3LogFS w/ Conclusions - 10%
6. Expanded Features & Information - 10%

Expanded Features & Information: file versioning and restore, time travel, improved RAM/local disk cache, leveraging other Amazon services for performance and/or providing a clear list of known issues with how if given more time we would resolve them.