A re-introduction to s3fs

Andrew Gaul
http://gaul.org/talks
23 January 2021

s3fs overview

- FUSE file system that allows POSIX applications to use S3 object storage
- Created in 2007 but significant improvements in recent years
- Compatible with wide variety S3 implementations: AWS, Backblaze, Ceph, GCS, Minio, and OpenStack Swift
- Many packages available: Amazon Linux, Debian, Fedora, FreeBSD, macOS, Raspberry Pi OS, Red Hat, and Ubuntu
- Works well for several use cases
  - But is a leaky abstraction with data consistency and performance pitfalls

Usage

Configure `$HOME/.aws/credentials` then:

```
$ s3fs bucket mnt
$ echo foo > mnt/foo
$ cat mnt/foo
foo
$ ls -l mnt/foo
-rw-rw-r--. 1 user user 4 Oct 31 00:00  mnt/foo
```

Can also access file data via AWS CLI:

```
$ aws s3 ls s3://bucket
2020-10-31 00:00:00          4 foo
$ aws s3 cp s3://bucket/foo -
foo
```

s3fs works well when

- Some POSIX compatibility is required
  - Supports most operations but lacks hard links and atomic renames
- Interoperating with other S3 applications
  - Uses the normal S3 object format
  - Stores UIDs, permissions, and xattr in S3 user metadata
- Sequential reads and writes on larger files
- Single directories have thousands of files, not millions
- High bandwidth and low latency to S3 bucket
  - e.g., client in the same AWS region

POSIX compatibility adds overhead

- Mapping POSIX onto S3 is like hammering a square peg in to a round hole
  - readdir issues a HEAD request for each file
  - Random writes amplify to 5 MB (minimum S3 multipart size)
  - Updating metadata copies entire object on S3 server
- Prefer native S3 applications when available
  - AWS CLI vs. ls or find
  - rclone vs. rsync
  - MySQL backups: xbstream vs. cp

Primitive multi-client coordination

- Close-to-open data consistency
- fsync does what you expect
- Multiple clients may cache stale data!
- Metadata freshness relies on stat cache timeouts
  - 15 minutes by default
- inotify does not work

Eventual consistency

- Traditional file systems guarantee strong consistency
- AWS recently guaranteed strong consistency (December 2020)
- Some S3 implementations only guarantee eventual consistency
  - `echo foo > file ; echo bar > file`
  - `cat file` can return foo or bar for some time!
- Worst case tests from 2014: 1 out of 1,000 operations
  - https://github.com/gaul/are-we-consistent-yet
- Caching can hide this behavior
- Make sure your workload can tolerate these semantics

Flags to tune performance

- `-o multireq_max` - number of parallel HEAD requests during readdir
- `-o use_cache` - cache GET requests in `/tmp`
- `-o parallel_count` - number of parallel PUT requests
- `-o multipart_size` - size of parallel PUT requests
- `-o max_dirty_data` - amount of temporary data to buffer locally before flushing to S3

Defaults tuned for lower-memory devices and containers.
Larger values can double write performance or more.

Changes in recent years

- Many concurrency, data corruption, and POSIX compatibility fixes
- More robust testing to prevent regressions
- Packages for all major distributions (2017-2018)
- 2-5x faster read, write, and readdir performance (2018-2019)
- Partial updates to objects via server-side copies (2019)
- Support for writing objects larger than local storage (2020)

Odds and ends

- Watch out for `updatedb` crawling the entire file system
  - Add `PRUNEPATHS` to `/etc/updatedb.conf`
- Writing to S3 also writes to `/tmp`
  - Control usage via `-o ensure_diskfree`
- Preserves UIDs and permissions
  - Defaults to `0750` and `0640` when `x-amz-meta-mode` not present
  - `-o umask` can override

Comparison with NFS

NFS is a better choice for workloads that require multi-client coordination, make small modifications, or list many files.

File system	NFSv4	s3fs
Concurrent access	strong semantics via leases	weak semantics
Write granularity	single byte	5 MB (minimum S3 multipart size)
readdir performance	one READDIR+ operation for many files	HEAD request for each file
AWS cost	$0.30 per GB-month	$0.02 per GB-month
S3 interoperable	no	yes

Alternatives

- goofys - gives up some POSIX compatibility for better performance, especially readdir
- s3ql - gives up S3 object interoperability and multi-client access for better metadata performance and other features
- See "Exploring trade-offs in S3 file systems" at Ohio LinuxFest 2020

Non-AWS clouds

- s3fs works with partial S3 implementations (no multipart)
- Cloud-specific FUSE sometimes works better
  - e.g., blobfuse (Azure), gcsfuse (Google)

Conclusion

- s3fs provides POSIX compatibility using S3 object storage
- POSIX imposes overhead - prefer native S3 applications when available
- Several important features and fixes in recent years
- Tune performance with a variety of flags
- Consider alternatives when your use case demands it

Questions?

Thank you!

http://gaul.org/talks
https://github.com/s3fs-fuse
🐦 @s3fsfuse