fs123

A scalable, read-only, network filesystem with pervasive caching

John Salmon
D. E. Shaw Research
PDSW 2019
November 18, 2019
The Problem

- We have >15 petabytes of simulation data and >1 terabyte of code/binaries
  - Growing at ~10 terabytes/day and ~10 new software packages/versions/day.
- POSIX is non-negotiable for executables
- Read-only access is sufficient
- Three widely distributed data centers, remote workers, laptops, …
- NFS is a non-starter
The solution: fs123

• Read-only distributed POSIX filesystem
• How does it work?
  • Loosely-coupled client-server protocol built on HTTP
  • Client implements a Filesystem in USErspace (FUSE) filesystem
  • HTTP origin server exports a backend POSIX filesystem

• That’s it!
The fs123 protocol: map FUSE callbacks to HTTP

FUSE client gets callback from kernel:
    fuse_lowlevel_ops::getattr(req, ino, fi)
FUSE client translates that into:
    HTTP GET http://server/anything/fs123/7/2/a/some/file
Origin server replies with:
    HTTP 200: cache-control: max-age=86400,
               errno=0, uid=503, gid=503, mtime=1573923416, …
FUSE client translates the reply into:
    fuse_reply_attr(ino, &stat, timeout=86400)
The software

• A single client binary (no special permission required)
  $ fs123p7 mount http://thesalmons.org:8888/ mtpt

• A single server binary (no special permission required)
  $ fs123p7exportd –port 8888 –export-root=/public/stuff

• About 10k lines of C++, available on github (2-clause license):
  https://github.com/DEShawResearch/fs123

• In production. Critical to our day-to-day scientific operations.
Why HTTP?

• Inherently wide-area

• Resilient on intermittent networks

• Standardized cache-management strategies

• Well understood by sysadmins
Well understood by sysadmins
Caching is essential for scalability

- **Kernel caches**
  - indispensable, require careful management

- **Client-side disk caches**
  - great for hiding network latency and coming back quickly after reboots

- **Proxy caches (e.g., Varnish, Squid)**
  - essential for wide-area operation
Caching would be easy if the data were immutable

• HTTP Cache-control (RFC 7234) allows proxies to work read-only, mutable data

• fs123 adheres to RFC 7234 for its kernel and disk cache

• RFC 7234 is not quite enough:
  • Monotonic validator: “The file you’re asking about has changed since the last time you asked about it, so you (the client filesystem) should flush everything you have cached about its contents”.
  • Estale cookie: “The file you’re asking about (by name) has disappeared (inode) since the last time you asked about it, so any attempt to see more of it must fail with errno=ESTALE.”
IF you’re comfortable running a static Linux binary from my personal URL:

$ wget https://thesalmons.org/fs123/fs123p7 && chmod +x fs123p7
$ mkdir mtpt c
$ ./fs123p7 mount –oFs123CacheDir=c http://thesalmons.org:8888 mtpt

# look around in mtpt: ls, find, cat, emacs (read-only)
# if you feel lucky, and have devel versions of libevent, libcurl libsodium
$ mkdir build; pushd build
$ make –f ../mtpt/GNUmakefile

# it’s a fuse daemon. To shut it down, do:
$ fusermount –u ../mtpt # or, in a pinch, pkill -9 fs123p7