OpenAFS

Derrick Brashear

OpenAFS Gatekeeper

shadow@openafs.org

What is AFS?

- AFS is used to refer to many things
  - A licensed product from IBM
  - A wide area distributed network filesystem
  - A file access protocol description
  - A filesystem namespace (/afs, traditionally)
  -

Quick History

- 1985 - CMU Andrew Project needed a filesystem.
- 1990 - Spun off as Transarc, AFS 3.0 released.
- 1991 - Talk of AFS 4.0 started. This became DCE DFS.
  - AFS was considered mature technology.
  - DFS progress at Transarc slowed changes in AFS.

Development tapers off

- IBM buys Transarc.
  - Many key personnel leave.
- No substantial changes after AFS 3.5
  - True threads and a portable fileserver backend were supposed to basically finalize the work.
  - However, existing deployments found moving to DCE cumbersome.

AFS EOL

- IBM gave many dates over time for the End of Life of AFS
- Existing customers pushed to have the software open-sourced if it were to be abandoned
- OpenAFS announced in late 2000.
  - Source available November 1.

Genesis of the OpenAFS idea

- IBM stopped providing support for new platforms.
  - Existing source licensees were allowed to share changes via IBM
  - New technology:
    - Disconnected AFS (offline cache) - CITI
  - Contributed ports:
    - NetBSD - Bill Sommerfeld/MIT SIPB
    - Linux 1.x - Derek Atkins/MIT SIPB

OpenAFS

- What is it?
  - An open source version forked from the IBM product in 2000.
  - It is broadly cross-platform.
  - It implements a caching client which speaks the AFS protocol for data access.
  - It includes a file server intended to operate as a black box, rather than re-exporting a local filesystem as NFS was traditionally used.
  - For platforms where native packages are provided, clients can provide a filesystem root dynamically, or mount a master top volume (usually root.afs)
  -

AFS Concepts

- Cell - all users, services and servers sharing one administration
- Volume - the unit of filesystem quota enforcement.
  - Think of it as a small UFS filesystem.
    - Others (ZFS) call it a dataset.
- FID - A way of describing a particular file of directory.
  - Volume, Vnode number, and Uniquifier.
    - Vnode numbers can be reused. A uniquifier increments once per vnode creation including reuse.

Design goals

- Support many clients from few servers
  - Clients cache data locally.
- Provide a coherent data “view”
  - Storing data will not return success until all interested parties are notified.
- Provide reliable data access
  - The concept of publishing readonly copies was added.
- Informed by experience with NFS.

AFS versus NFS

- Security (with RPCSEC GSS, NFS currently wins)
- Single namespace
- Site federation (cells)
- Dataset manageability
- Scalability (caching to shed load)

Network

- Built on UDP
  - In the 1980s, select() scaled badly.
- Rx RPC
  - A peer of SunRPC, reuses XDR and an extended rpcgen.
  - Adds per-packet authentication data.
    - In most current deployments only a cryptographic checksum on packets is done.

Authentication

- Symmetric key cryptography
  - A Kerberos 4 based service originally provided these to users.
    - OpenAFS replaces this with Kerberos 5.
  - fcrypt, a symmetric crypto algorithm using 56 bit DES keys from Kerberos, provides encryption.

Protocol

- RPC suites for each service protocol
  - Volume Location Server provides location data for AFS volumes.
  - Protection Server maps user and group names to IDs
  - Fileserver provides access-controlled data and file metadata fetch and store service.
  - Volume server is used to administer volume datasets.
  - Historic authentication and backup suites.

More on volumes

- The volume server gives you the ability to transparently migrate volumes.
  - Implemented by
    - sending most data to a new server
    - making the volume unavailable briefly to send “final” changes
    - then returning an error telling clients to find the volume elsewhere
- You can add and remove replicas on the fly
  - Same mechanics for noticing when one goes away.
  - Locations refreshed every 2 hours.

Namespace

- Coherent unified namespace
  - All machines see the same view.
    - Like an automounter but mount point objects plus VL service makes it explicitly embedded.
  - Cell federation
    - Access other cells at /afs/cell.name
      - This is by convention, not part of the protocol.
- Ad-hoc mount points
  - New in current clients. (/afs/.:mount/cell:volume/)

Database servers

- Volume location and protection servers are replicated
  - Can be located by fixed means (CellServDB file) or DNS AFSDB or SRV lookup.
  - Uses “ubik” simple database mechanism to provide replication to a quorum of servers.
    - No updates while there is not a quorum of “up” servers.
  - No file data read-write replication (yet).

Architecture - Server

- Portable POSIX filesystem backed black box data store (“vice partitions”).
- Multithreaded file and volume servers.
- Fast access control by caching authorization data on first receipt of client authentication data.
- Client host tracking done when RPCs received from new clients.
- Notification to interested clients on file changes (new “dataversion”).

Architecture - Client

- Client also implements a service
  - Callback service, used to receive server notification of file change.
    - Currently no details of changes are provided.
- Unreachable clients are notified of pending changes when they reappear.
  - Because of the UDP callback channel, unreachable can include NAT UDP port expiration.
    - 1.5 series OpenAFS works around this.

Client security

- What should have my credentials?
  - Process Authentication Groups (PAGs)
    - Can share with some of my processes
    - Can share with some other processes
    - Inherited across fork().
      - usually implemented by encoding in user's group list
      - requires stealing setgroups()/initgroups()

Cache

- Chunk-based fetching and storing
  - Allows use of files larger than the cache
- Persistent cache
  - Metadata is also persistently stored.
  - After restart, locally-resident data which is still current (same dataversion) can simply be reused.
  - IP address change does not invalidate cache.

Cache mechanics

- Memory- or disk-backed.
  - Disk-backed means VFS routines call into the KPI to access cache backing store files.
  - VOP_VGET + vn_rdwr

Protocol Assumptions

- Last writer wins
  - In the absense of locking, if you write last, you overwrite other previous changes.
- Store on fsync() or close()
  - Intended to work as through writing files is atomic.
  - This is violated if the client cache fills to permit the client to continue caching reads or writes.

Implementation details

- Loadable Kernel Module
  - Provides RPC system, AFS syscall, VFS, and a memory or local filesystem backed cache.
- Userspace helper (afsd)
  - Spawns services via syscall.
- Userspace utilities
  - Use syscall to communicate with kernel.

System Call

- Provides 3 kinds of services.
  - pioctl: ioctl-by-path.
    - But sometimes overloaded to tweak client configuration.
  - AFS system call: startup/shutdown helper.
  - iopen: legacy “open by inode number” for old platforms.

Issues you may notice

- Compromises to portability
  - Native interfaces are being included as time permits.
- Legacy interfaces
  - We've been slow to desupport old clients.
- Dated security
  - I'll get back to that.

OpenAFS on FreeBSD

- BSD support since the old NetBSD 1.0 days has been a sordid tale.
- A FreeBSD 5 era port in the tree.
- Renewed interest in updating the port.

An unusual consumer

- Locking issues
- KPI issues
  - Disk-backed cache means we have unusual issues; we call VOPs including VGET in the cache backing code.
    - And your KPI is a moving target

Where it stands today

- Works with 8-RELEASE
  - Currently uniprocessor, and with a memory cache.
    - Vnode locking is hard.
  - Fixes for the rest being worked on.
  - Where we go from there depends on interest.
    - Where we don’t want to go: in the kernel.
      - Insert sob story here.

Why you should want it.

- We're fixing our shortcomings.
  - Rx/TCP: funded, in progress.
  - RxGK (GSSAPI encryption): funded, in progress.
  - POSIX extended attributes: funded
  - Per-file ACLs: funded
  - Performance optimizations:
    - Extended callbacks: code exists. Needs RxGK
    - Locking delegation: being standardized
    - Parallel data access: being planned
    - Read-write replication: funded

Details

- RxGK
  - Provides a kernel-capable cipher suite.
  - Reuses heimdal crypto for userspace.
- Rx/TCP
  - Now in the works for 5 years.
  - Unlike before, this and several other things have been funded by the U.S. Department of Energy.

Details

- Current callbacks are of the granularity “something happened”
  - Extended callbacks include a reason and where possible the actual change.
    - This makes locking enhancements possible.
    - Because this can change the cache, authentication required.
    - Callback channel is not currently authenticated. This is added in RxGK.

Details

- Parallel data access
  - Replication means data already exists in more than one place.
  - Chunk-based access means you can easily parallelize reads.
- Might as well fetch from multiple sites in parallel.
  - The Arla AFS protocol client has done this for years.

When you can have it.

- The FreeBSD port updates are being worked on in “spare time”, ongoingly.
- The timeline for several of the funded enhancement items is interlocked.
- Most of the funded work will be completed by late next year.
- Some work will require standards approval before it can be issued in a release.

How you can help

- Try it.
  - If you find issues, report them.
  - If you fix issues, all the better.
- Help us fix it.
  - We’re looking for suggestions to better track kernel changes, and test against - CURRENTs ideally at least nightly.
  - If you have time to look at code, especially platform-specific code, tell us what we’re doing wrong.

How we can help you

- Suggestions?