OpenAFS: Ten Years of Open Source Storage Systems

Jeffrey Altman
OpenAFS Elder and Gatekeeper

What is OpenAFS?

- AFS is:
  - A globally accessible file system name space (/afs)
  - A family of file system RPCs used to manipulate the contents of the name space
  - A class of client and server applications that provide access to the name space
- OpenAFS is:
  - One of the oldest open source communities
  - One of the largest open source projects
  - The most widely deployed implementation of AFS

Talk Outline

- A brief history of OpenAFS
- Why is AFS still used given the resources devoted to CIFS, NFS, NFSv4, Lustre, and other network storage technologies?
- What makes our open source project successful?
- What is the future of the AFS protocol suite?
- What types of organizations depend on AFS?

A Brief History of OpenAFS

The Beginning of the Story …

- 1983-88 Andrew Project at Carnegie Mellon University creates AFS to support a distributed, heterogeneous, workstation based, computing environment
- Funded by IBM to compete with DEC funded Athena at MIT
- Unique properties of AFS at the time included:
  - High Scalability
  - Client side caching with coherency
  - Federated authentication model based on Kerberos
  - Usable Access Control model (not POSIX)
  - Location independent data access and volume management tools permitted zero downtime maintenance
  - Heterogeneous support including PCs

The Commercial Era

- 1988 Transarc is formed
  - Markets AFS as "the Enterprise File System"
  - Founding member of the Open Group
  - Must beat SunOS in order to beat NFS (v3)
  - Creates AFS4 which becomes the DCE File System (DFS)
- 1994 Transarc is acquired by IBM
  - AFS3 becomes a legacy product
  - DFS and the Encina Transaction Process Monitor become the focus of IBM Transarc Labs
  - AFS3 is bundled with WebSphere (1997)

The Road to Open Source

- 1996 to 2000, major educational and research institutions with source code licenses continue to develop enhancements to AFS but are only permitted to share them through IBM
- Lack of on-going development, and increased licensing costs, and pressure to migrate to other IBM products generates significant customer backlash
- 1997, Stacken Computer Club (Stockholm) implements Arla
- 1998, Derrick Brashear (OpenAFS Elder) requests source code access from IBM
  - First Rx source code is released as it is public domain
  - Later a commitment to release much of the AFS sources
- Aug 2000, IBM announces a plan to open source AFS
- 31 Oct 2000, IBM posts source code to DeveloperWorks

OpenAFS 10 Years and Counting

- OpenAFS was formed on 1 Nov 2000
- The original Elders represented IBM, Intel, Morgan Stanley, Carnegie Mellon, MIT, and UMichigan
- Since then ohloh.net says that OpenAFS has become one of the largest open source projects
  - 236 developers since inception (47 active in the last year)
    - Avg 670 commits/year for first 8 years, 1250 in 2009, on target for 1800 in 2010
  - Nearly 1 million lines of source code and 100,000 lines of user and developer documentation
  - All major operating systems (except mobile) are supported
  - Untold millions of end users (no way to measure)

How did OpenAFS reach Ten?

The Stars Were Not Aligned

- Enterprise Storage Systems are a fundamental building block that must be accessible from everywhere
- Selection of a storage solution is at a minimum a ten year strategic decision
- If there is any doubt that the operating systems the firm will rely on ten years from now will not be able to access the firm's data, the switch to a new technology must begin today
- With all the doubt surrounding the future of AFS in the late 1990s, how is it still here?

Once spoiled by AFS
There is no replacement

- Most if not all orgs that deploy AFS have considered migrating over the last fifteen years
- The risks of staying with a perceived to be dead technology were too great
- The risks and costs of transitioning were also significant
- BUT no other solution satisfies the operational requirements of the institutions that have an infrastructure built upon AFS

Operational Requirement: Location-Independence

- A file system must be distributed and support location-independence
  - It must be possible to migrate data sets while in use without the clients noticing
- This is required for continuous load balancing and permit evacuation of servers to permit hardware and operating system upgrades
- The AFS volume management capabilities satisfies these requirements

Operational Requirement: Authentication and Privacy

- Strong network authentication of users and processes is a necessity in most organizations due to audit requirements
- AFS support for Kerberos authentication was designed in from the start
- AFS has encryption but not as strong as would be desired but better than most other options
- The AFS Rx security class model permits alternatives to be added

Operational Requirements: Geographic Replication of Critical Data

- Critical organizational data must be geographically replicated for fault tolerance and business continuance
- Client failover must be transparent
- The replication mechanisms themselves must be replicated to ensure continuity of operations in case of a major outage

Operational Requirements: Atomic Publishing Model

- Organizations that deploy AFS become addicted to its built-in publishing model
- World visible readonly snapshots are generated within the file system name space from privately edited read-write volumes
- These snapshots are globally replicated

Operational Requirement: One File System for All Clients

- There must exists client support for one common distributed file system for all supported operating systems
  - Microsoft Windows, MacOS X, Linux, Solaris
  - IRIX, AIX, HP-UX
- Operating system support must exist from day of release to date of decommissioning

Operational Requirement: Fine Grained Access Control

- Better than Unix permissions
- Not necessarily POSIX
- Specific use cases
  - Insert (create but not modify nor delete)
  - Read (but not write nor delete)
  - List directories (but not read the contents)
  - Read and Write data (but not the permissions)
- User-defined groups that can be placed on ACLs
- Must be tied to the authentication identities

Operational Requirements: Global Accessibility and Federation

- Authenticated users must be able to access their data without use of VPNs
- Authenticated users must be able collaborate with authenticated entities from other institutions
- Authentication of foreign entities must not require issuance of local authentication accounts

Operational Requirement: Platform Specific Redirection

- In order to support a common file system paths for application binaries and configuration files regardless of OS/hardware platform, it is a requirement that the file system provide a redirection mechanism
- The AFS @sys system name list evaluation in symlink processing provides this capability
- It is a critical component for the deployment of a stateless computing infrastructure within a distributed file system

Operational Requirement: Platform Independence

- It is critical that the file system protocols and data formats be platform independent
- This permits the infrastructure to migrate to cheaper and more efficient systems as they become available from competing vendors
- Mixed deployments also provide a degree of protection against platform specific outages caused by hardware or software bugs, or denial of service attacks

Operational Requirement: Distributed Administration

- It must be possible to delegate management of name space subsets to different administrative groups
- Administration functionality must be scriptable in order to support higher level tools that
  - Globally manage distribution, replication, and restoration
  - Provide finer grained administrative functionality to non-administrative users based upon organizational roles

No Clear Cut Alternatives

- Given the set of operational requirements there simply are no clear cut alternatives
- There are dozens of distributed file systems. CIFS/Dfs, AFP, NFSv3, NFSv4, Lustre, GPFS and PanFS are just the tip of the iceberg
- While it is possible to construct a solution that supports all of the operational requirements with one or more file systems and higher level tools, there is nothing that jumps out and slaps you in the face

Transition Costs are Huge

- Any transition for a large organization will end up as a multi-million dollar project
  - Staff retraining
  - Documentation changes
  - Redevelopment of administrative processes and supporting tools
  - Decommissioning of platforms and applications that are not supported by the replacement
  - Support for both solutions in parallel for the length of the transition including
    - Double the hardware, double the data center capacity, increased staff requirements

Costs and Risks Provided an Opportunity for OpenAFS

- The risks and costs of a transition were a significant hurdle for existing users which in turn provided OpenAFS an opportunity
- However, there were many reasons to prevent new adoption of the technology

How Did OpenAFS Succeed?

Focus on the Users

- OpenAFS development has been evolutionary and revolutionary
- The community has focused on ensuring backward compatibility while improving performance and scalability
- Major release transitions have permitted rollback in case of unexpected disaster
- Day-0 support for first tier client platforms since 2005
  - Leopard, SnowLeopard, Vista, Win7

Every Project Needs Its Own Linus

- For OpenAFS, our Linus equivalent is Derrick Brashear
- Derrick has provided continuity, architectural guidance, and a dedication to review each and every one of the 7700+ submissions over the last ten years
- Supported by two other gatekeepers, Derrick ensures that contributions are incorporated (or not) in a timely manner which in turn permits the development community to grow
  - Even when our tool chain was working against us

The Tools Matter

- OpenAFS development was managed using a combination of CVS, a home grown patchset mechanism called DELTAs, and RT for patch submission and review
- It was extremely challenging to review submitted patches and obtain review from the community
- Pulling patches out of RT and building sandboxes using CVS on multiple platforms for testing was worse than pulling teeth

The New World: Git, Gerrit, and BuildBot

- At the 2008 Google Summer of Code Mentor Summit, we were introduced to the Gerrit Code Review tool built on top of Git
- OpenAFS had been attempting to migrate from CVS to Git for almost a year. Gerrit increased the priority of the transition a thousand fold
- Since the switch over to Git and Gerrit (thanks to Simon Wilkinson), the rate of developer participation has skyrocketed.
  - On target for a 150% increase in contributions over 18 months
- Buildbot has been integrated with Gerrit to perform build verification on multiple platforms prior to Gatekeeper review

OpenAFS Development Processes: The Old

- Most major development on OpenAFS is the result of a commercial development contract that may or may not involve a Gatekeeper
- Prior to Git/Gerrit, it was frequently the case that this work would be performed against a stale code base
- The customer wants OpenAFS stable + their feature
- This resulted in OpenAFS receiving the result of many months of work as one huge blob
  - Too late to perform extensive design review
  - What do you do when feature A conflicts with feature B?

OpenAFS Development Processes: The New

- Git provides for easy pulling of new commits and rebasing of changes. This makes it much easier to require developers to be current.
- Gerrit permits patchsets to be reviewed efficiently provided they aren't too large
  - OpenAFS now requires one change per patchset
  - Patchsets should be reviewable within an hour
  - No patchset should break the product on any platform
- OpenAFS strongly encourages incremental changes to be submitted early and often to reduce the costs of collisions between developers
  - This policy has a side effect of forcing collaboration between independent teams

Community Development and Trust

- Building community and creating market awareness is critical to the success of any product
  - Annual Best Practices Workshops for everyone
  - Periodic Hackathons for Developers
  - Promote Local User Groups
  - Community Outreach via Conference Presentations such as this one
- Be honest about the project weaknesses
- Do not promise what cannot be delivered
- Listen, Listen, Listen … to what your users need
  - Be willing to say 'no' to the developers
  - Simply because something is cool and can be built does not mean it belongs in the product

Standards are Important

- "AFS" is a name space, a protocol, a class of products
- "OpenAFS" is but one implementation among many
  - IBM AFS, Arla, kAFS, ….
- Even though "OpenAFS" is the gorilla in the room its code base cannot define the standard
- An independent standardization process has been defined (based loosely on the IETF / IANA model) to manage protocol registries and RPC standards
- New implementations are welcome

The Future of OpenAFS

Protocol Enhancements

- Paraphrasing Mike Kazar, the most efficient RPC is the one you don't have to issue
- The future of the AFS protocols build upon the strengths of the coherent caching preparing it for the mobile era
  - Extending the callback model to minimize the scope of cache invalidations and reduce the amount of redundant data requested from the file servers
  - Push as much work to the servers as possible thereby minimizing the transient data transmitted over the wire

Security Enhancements

- GSS-API Authentication
  - Kerberos v5, X.509, and SCRAM
- Kerberos Crypto Framework Encryption
  - RC4-HMAC, 3DES, AES-128, AES-256, and anything else that the IETF standardizes
- Departmental File Servers
- Privacy for anonymous connections and callback channels
- Close all known cache poisoning attack vectors

Many More Improvements

- The OpenAFS Roadmap includes a broad range of other funded improvements
  - Rx Transport Layer Throughput
  - Service Scalability
  - Byte range locking
  - Integration with Object Storage Systems
- Please see the OpenAFS Web Site
  - http://www.openafs.org/roadmap.html

Significant Deployments

U.S. Government

- Dept of Energy Labs
- U.S. High Performance Computing centers
- NASA
- Internal Revenue Service
- U.S. Geological Survey
- Naval Research Laboratory

Education

- Carnegie Mellon University
- Cornell University
- Duke
- Harvard
- MIT
- PSU
- UC-Berkeley and UC-Santa Barbara
- Stanford
- Univ of Michigan
- U of Wisconsin-Madison
- Iowa State University
- California Inst of Technology
- Univ of North Carolina
  - Chapel Hill
  - Charlotte (Mosaic)
- Univ of Stockholm
- KTH
- Univ of Edinburgh
- CERN
- Many many others

Commercial

- Morgan Stanley
- Pictage Photography Services
- General Electric Aircraft Engine Division
- Goldman Sachs
- Hitachi
- Qualcomm
- United Airlines
- Many that are not publicly known

AFS and Stateless Computing

Morgan Stanley

- Their challenge: Develop a distributed storage environment that would support global deployment of stateless client systems
  - No operating system pre-installed at boot
  - No applications pre-installed in an operating system image
- 1994 Aurora is deployed on UNIX
- 2002 Aurora model implemented on Microsoft Windows (minus network boot)
- Today, >100,000 hosts (clients and servers) managed with Aurora. More than 30,000 applications deployed and executed from AFS.

Virtualization vs Caching File Systems

- Caches are meant to reduce network traffic and load on the distributed storage infrastructure
- With virtualization the trend is towards many more client systems without more physical hardware
  - This places a strain on the storage servers
- Virtual disk images are often loaded from network storage. Large disk based caches within the VM generate more network traffic than they prevent
- The Fix: All network file system access must be performed through the hypervisor
  - Only then can caching be effective and the client explosion prevented

You have questions?

- I have answers!
- Take your best shot

Contact Info

Jeffrey Altman
President
Your File System Inc.
jaltman@your-file-system.com

Your File System