OpenAFS: Ten Years of Open Source Storage Systems

Jeffrey Altman
OpenAFS Elder and Gatekeeper
Your File System, Inc. Founder
Talk Outline

- 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?
OpenAFS 10 Years and Counting

- OpenAFS was formed on 1 Nov 2000
- The original Elders represented IBM, Intel, Morgan Stanley, Carnegie Mellon U., MIT, and U. Michigan
- Initial US$105,000 contribution from USENIX, Morgan Stanley and Intel to cover costs of merging source trees
- Since then ohloh.net says that OpenAFS has become one of the largest open source projects
  - 248 developers since inception (55 active in the last year)
    - Avg 670 commits/year for first 8 years, 815 in 2008, 1250 in 2009, 1873 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 numbers of end users (no way to measure)
How did OpenAFS reach Ten?
(Or, why isn’t it dead by now?)
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 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, why is it still in use?
Once spoiled by AFS functionality, 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 technology that is perceived to be dead are too great
- The risks and costs of transitioning are also significant
- BUT no other solution satisfies all of the institutional operational requirements
- AFS was ahead of its time in 1985 and remains so to this day
Operational Requirement #1: 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
- Required for continuous load balancing and to permit evacuation of servers during hardware and operating system upgrades
Operational Requirement #2: Authentication and Privacy

- Strong network authentication of users and processes is a necessity in many 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 Requirement #3: Geographic Data Replication

- 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 Requirement #4: 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
  - Application binaries
  - Web server content
  - Documentation Sets
Operational Requirement #5: One File System for All Clients

- There must exist client support for one common distributed file system for all supported operating systems
  - Microsoft Windows, MacOS X, Linux, Solaris
  - IRIX, AIX, HP-UX
  - Other ...
- Operating system support must exist from day of release to date of decommissioning
Operational Requirement #6: 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 Requirement #7: Global Accessibility and Federation

• Authenticated users must be able to access their data without use of VPNs
• Authenticated users must be able to collaborate with authenticated entities from other institutions
• Authentication of foreign entities must not require issuance of local authentication accounts
• Multiple authentication names should be able to refer to the same authorization identity
  • user@OPENAFS.ORG
  • user@AD.OPENAFS.ORG
Operational Requirement #8: Platform Specific Specific Redirection

- Must support common file system paths for application binaries and configuration files regardless of OS/hardware platform
- 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 #9: 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 #10: 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 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
## File System Comparison

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>Volume Management</th>
<th>Filesystem snapshots</th>
<th>POSIX Extended Attributes</th>
<th>Transport</th>
<th>Scalability</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPENAFS</td>
<td>Yes</td>
<td>Limited</td>
<td>No</td>
<td>UDP IPv4</td>
<td>Yes</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

**OPENAFS NOTES**
- Transparent movement of data.
- Typically one “backup”.
- TCP support planned.
- Thousands of clients per server in practice.
- No parallel access today. Limited by transport.

| LUSTRE           | No                | No                   | Yes                       | TCP IPv4  | Yes         | High        |

**LUSTRE NOTES**
- Online data migration planned.
- Was planned for 3.0.
- 30000 clients per node.
- Optimized; Uses object-based storage.

| NFS V4           | Extension         | No                   | Yes                       | TCP       | Yes         | Varies      |

**NFS V4 NOTES**
- Optional to implement.
- IPv4, IPv6 standardized
- pNFS extension, TCP allow good performance.

| ZFS              | Yes               | Yes                  | Yes                       | N/A       | N/A         | High        |

**ZFS NOTES**
- Local only.
- Uses mirroring and striping to achieve high bandwidth.

| YFS              | Yes               | Limited              | No                        | UDP, TCP; IPv4, IPv6 | Yes         | High        |

**YFS NOTES**
- Striping: Q3 2011
- More than OpenAFS; Q3 2011
- Q3 2011
- TCP Q1 2012 IPv6 Q1 2012
- Asynchronous threading model; 60,000 clients / server Q1 2012
- Transport, threading, OSD; Q3 2010-11
## File System Comparison (cont.)

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>Locking</th>
<th>Replication</th>
<th>Object Storage Integration</th>
<th>Security</th>
<th>Authentication</th>
<th>Open Source</th>
<th>Commercial Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPENAFS</td>
<td>Advisory</td>
<td>Read-Only</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>OPENAFS NOTES</strong></td>
<td>Whole file only.</td>
<td>Read-Write planned.</td>
<td>Integration to begin soon.</td>
<td>56 bit fcrypt.</td>
<td>KScrypto, 2010</td>
<td>Kerberos 4 and Kerberos 5.</td>
<td>IBM Public License V1.0.</td>
</tr>
<tr>
<td>LUSTRE</td>
<td>Yes</td>
<td>Local</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>LUSTRE NOTES</strong></td>
<td>No lockf /flock yet.</td>
<td>RAID, not multi-server yet.</td>
<td>That’s largely the point!</td>
<td>1.8</td>
<td>Kerberos support in 1.8</td>
<td>GPL.</td>
<td>Oracle</td>
</tr>
<tr>
<td>NFS V4</td>
<td>Yes</td>
<td>Extension</td>
<td>Extension</td>
<td>Yes</td>
<td>Yes</td>
<td>Available</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>NFS V4 NOTES</strong></td>
<td>Mandatory and Advisory. Not widely available.</td>
<td>In pNFS/NFS v4.1.</td>
<td>GSSAPI RPC.</td>
<td>GSSAPI / Kerberos 5.</td>
<td>Citi reference implementation is GPL.</td>
<td>Typically from OS vendor.</td>
<td></td>
</tr>
<tr>
<td>ZFS</td>
<td>Yes</td>
<td>Manual</td>
<td>Extension</td>
<td>N/A</td>
<td>N/A</td>
<td>Available</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>ZFS NOTES</strong></td>
<td>Mandatory and Advisory. Using zfs send/receive.</td>
<td>Block-based ZFS.</td>
<td></td>
<td></td>
<td></td>
<td>Typically from OS vendor.</td>
<td></td>
</tr>
<tr>
<td>YFS</td>
<td>Yes</td>
<td>Read-Write &amp; Read-Only</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>YFS NOTES</strong></td>
<td>Q2 2011</td>
<td>Q2 2011</td>
<td>Q3 2011</td>
<td>RFC3961, Q2 2011</td>
<td>GSSAPI / Kerberos 5, X.509; Q2 2011</td>
<td>IBM Public License V1.0 + BSD</td>
<td>Your File System</td>
</tr>
</tbody>
</table>
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
IBM Advanced Distributed File System (ADFS)

- By 2004, some within IBM realized that AFS and DFS customers (internal and external) were not migrating to alternative IBM storage solutions
- ADFS was an attempt to provide a successor file system for both AFS and DFS that would have the simplicity of AFS with the power of DFS
  - >2GB files, Kerberos v5, byte range locks, per-file ACLs, better threading
- Better than NFSv4
  - Replication, transparent data movement, global namespace, proven code base
- Unfortunately, the product never saw the light of day
How Did OpenAFS Succeed?
Focus on the Users

- OpenAFS development has been evolutionary not revolutionary
- The development community has focused on ensuring backward compatibility while improving performance and scalability
- Major release transitions have permitted rollback in case of unexpected disaster
- o-Day support for first tier client platforms since 2005
  - Leopard, SnowLeopard, Vista, Win7
Adjusting to a changing environment

- Networking Changes
  - Increased use of wide area network connections
  - Greater client mobility (laptops, wifi, cellular)
  - Protection against denial of service attacks
  - Network Address Translators / Port Mapping
  - Multi-homed clients
  - Split horizon addressing
  - Classless Addressing
- Dynamic root volumes
- Bulk RPCs to reduce number of round trips
- Kerberos v5 authentication
- Large file support
More efficient cache managers

- The fastest RPC is the one that is not issued
- Avoid queries for data that cannot exist
  - Chunks at end of file
- Do not query for status information that should already be known
  - Readonly volume versioning
- Do not discard data that is known to be current
  - Incremented data version in response to Store operation
- Reduce copies between AFS and operating system page cache
Linux page cache read performance: AFS should match ext3 below 1GB
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 actively being pursued
The Future of OpenAFS
The Road Map

- The OpenAFS road map includes a broad range of funded improvements
  - Rx Transport Layer Throughput
  - Security Enhancements
  - Server Scalability
  - Missing first class file system functionality
  - Read write Replication

- Please see the OpenAFS Web Site
  - http://www.openafs.org/roadmap.html
Name Space Expansion

- 32-bit -> 64-bit File Identifier components
  - Up to $2^{63}-1$ volumes and objects in each volume
- Longer volume names
- Unlimited directory contents
- Time resolution from 1s to 100ns
Improved AFS Cache Coherency Algorithms

• AFS relies on an unauthenticated generic callback message to enforce cache coherency
  • 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
Read Write Replication

- Replication is a requirement for disaster recovery
- AFS Read-only replication is designed for publishing
- To support backward compatibility and reduce client network traffic:
  - Single master with multiple replicas
  - Master file server issues locks and accepts Data Stores
  - File server writes to replicas in background
  - Clients request status info from Master (if file opened for write) or from replicas (if file opened for read)
  - Clients read data from anywhere but fallback to Master if data version is old
Popular Feature Requests

- Disconnected Operations
- End-to-end data integrity
- Integrated Search
- Global Cell Replication Services
- Virtual Machine Integration
Your File System, Inc.

- Founded in 2007, Your File System, Inc. is funded by the U.S. Department of Energy to develop a successor to AFS
- A prototype will be ready by the end of 2011
- Full backward compatibility for deployed AFS clients
- Backend servers are incompatible with AFS servers
  - AFS volumes can be imported without modification
  - Existing AFS management tools can be used
- Core functionality will be open sourced through OpenAFS
AFS and Stateless Computing
Morgan Stanley’s Aurora

- 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

- 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
Founder
Your File System Inc.
jaltman@your-file-system.com