AURISTOR

YOUR DATA · YOUR FILE SYSTEM

THE FUTURE OF AFS®

STATUS REPORTS: AURISTOR FILE SYSTEM, KAFS, AND OPENAFS

LINUX CONTAINERS

TENNESSEE OPEN RESEARCH STORAGE CLOUD (TORC)

JEFFREY ALTMAN (JALTMAN@AURISTOR.COM)

JEFFREY ERIC ALTMAN
CEO/CTO, AURISTOR, INC.

- The Kermit Project – Columbia University
  - C-Kermit (Unix) and Kermit 95 (OS/2, Windows)
  - Internet Kermit Service
- Internet Access Methods
  - Java-based Person to Person collaboration software
- Project JXTA
  - Founding Board Member
- Internet Engineering Task Force (IETF)
  - Secure Remote Password, TELNET START_TLS, FTP AUTH TLS, SSHv2
  - Security Area Advisory Group
  - Former Kitten Working Group Chair
- OpenSSL
- MIT Kerberos
  - former core developer
  - Kerberos for Windows team lead
- Heimdal Kerberos
  - Project Owner
- Network Identity Manager
- OpenAFS
  - Elder Emeritus
  - The last of the Gatekeepers

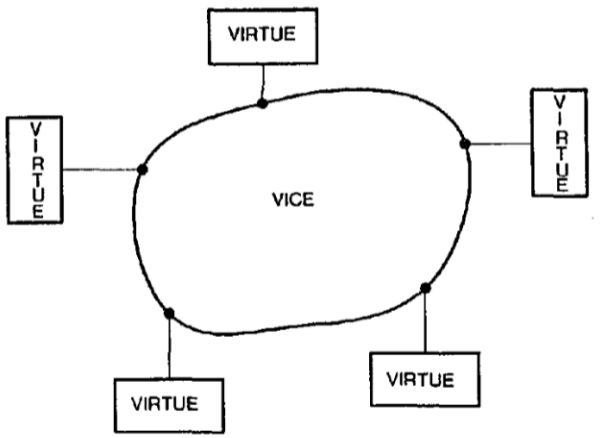
The /afs® Vision

- Location Transparency: one name space; data mobility without service interruption
- User Mobility: access from any device
- Security: Flexible model for authentication, privacy, data protection and access control
- High Availability: Temporary loss to small groups for short time periods
- Integrity: No need for user initiated backups; users trust the system
- Heterogeneity: Multiplatform; one file system for all operating environments
- Self service: Low Help Desk costs
- Atomic Publishing: Software, documentation, web sites, ..
- Real time collaboration: Distributed File Locking; Real-time full data coherency
- Distributed Administration: administer from any device

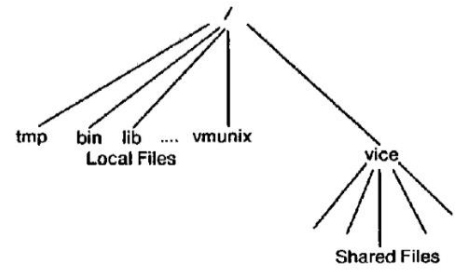
AFS is 31 years old

In 1985, CMU's VICE file system was decades ahead of its time:

"A comparison with other distributed file systems reveals that although this design has individual features in common with some of the other systems, it is unique in the way it combines these features to produce a total design. It is further distinguished from all the other systems in that it does not rely on the trustworthiness of all network nodes"



Each *Virtue* is an individual workstation. *Vice* has fine structure that is shown in detail in Figure 2-2. This diagram is certainly not to scale, since Vice will encompass an entire campus!



The files in the "/vice" subtree are common to all the workstations in the system. Certain directories and files in the local name space, such as "/bin" and "/lib", are symbolic links into "/vice". All other local files are inaccessible to other workstations.

Figure 3-2: A Unix Workstation's View of the File System

Use cases limited due to 30 Years of Technical Debt

- Limited network throughput
- Increased call processing latency
- Decreased service reliability and availability
- Elevated risk of distributed deadlocks
- Inability to use full capability of available hardware
- Failure to keep up with competing technologies
- Multi-producer, multi-consumer work flows cannot be supported

That /afs is still in use today is a credit to the uniqueness of the vision and the strength of its architecture.

AuriStor's Vision for /afs

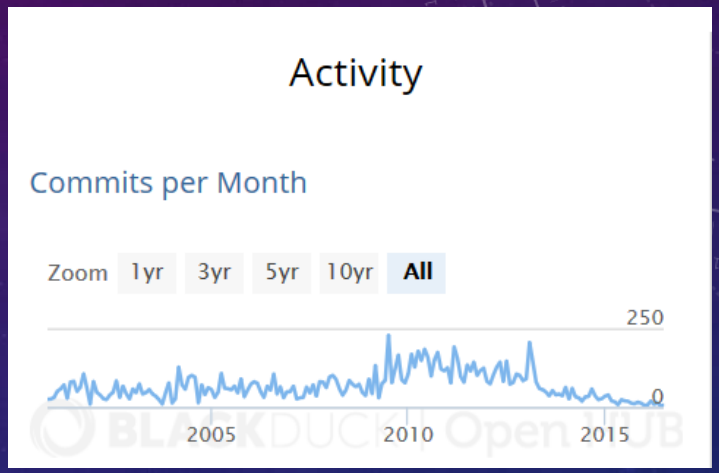
- Application Transparency
  - Be a First Class file system on all major OSes
  - Client support built into Linux kernel
- Embrace multi-producer, multi-consumer work flows
- Provide persistent storage for Container deployments
- Extended Integrity: Disaster Recovery
- Be performance competitive
  - Lustre, GPFS, Panasas, NFSv4, …
- Best of breed data security
  - Wire privacy and integrity protection policy guarantees
  - Combined identity authentication
  - Multi-factor authorization
  - Geographic isolation
- Improved Ease of Use

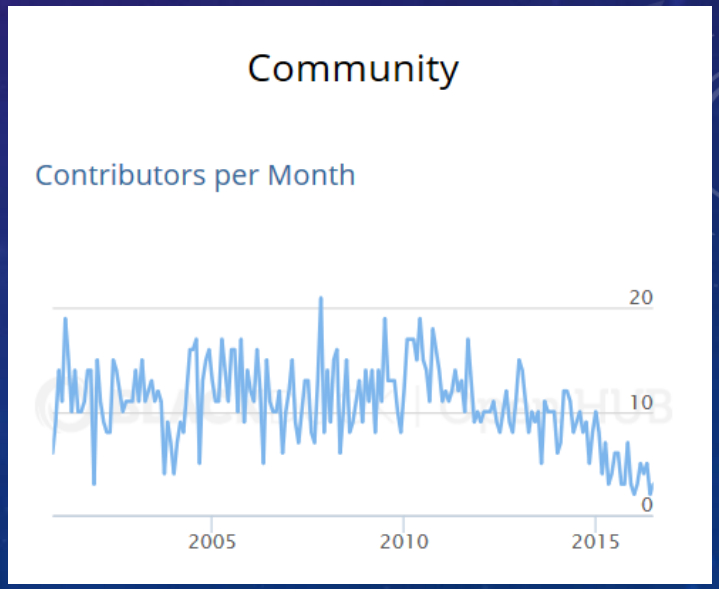
OPENAFS

Status Report

16 Years of OpenAFS

- Five years between each major release 1.2 -> 1.4 -> 1.6 -> 1.8
- Major system rewrites are few and far between
- “Contractor Support Model” led to many small and localized changes
- A lack of consistent vision and quality control
- Few incentives to invest in the next 30 years
- Decreasing year over year development activity





OPENAFS 1.6

- struggling to stay afloat
- trying to issue occasional releases fixing the most urgent issues and maintain parity with the latest Linux kernels
- 1.6.19pre1 imminent
  - Linux 4.8 kernel support
- anything else will be subject to available resource which have crossed the "lower critical" threshold

OPENAFS 1.8 AND BEYOND

- OpenAFS master has accumulated many improvements over the 1.6 series since January 2011
  - Code cleanup (warnings removed, dead code removed, private data structures removed from public headers)
  - conversion to shared libraries built using libtool, Heimdal's libroken, and Heimdal's libhcrypto.
  - new token apis to support non-rxkad tokens
  - Ubik conversion to pthreads (death to LWP)
  - rx optimizations (atomics, red/black trees, Bob Jenkin's hash functions, and more)
- Status update and list of release blockers are at

  http://lists.openafs.org/pipermail/openafs-devel/2016-September/020355.html
- More code review is needed at https://gerrit.openafs.org for the listed changes, and to help assemble the release notes at

  https://gerrit.openafs.org/12393
- Once the 1.8 branch is created, master will be open to start receiving new features. Desired features include rxgk and IPv6 support, but at present no organization or individual has indicated that they will contribute code contributions towards those goals.
- In the absence of such contributions, OpenAFS will likely see only incremental improvements on the 1.8 branch with no clear prospect of a 2.0 series.

OPENAFS CORE WEAKNESSES

- Rx network protocol
  - Congestion control
  - Resource contention
  - IPv4 only; design flaws limit packet size and window size
  - Fcrypt (56-bit weaker than DES)
- UBIK distributed database
  - Quorum Establishment
  - Database Synchronization
- Volserver Transaction Management
- Volume Clone Management
- Volserver Thread Safety
- File Server Copy on Write operations
- Cache Coherency
- Signal Handling Race conditions
- Security
  - Privacy – Extensive Metadata leakage
  - Anonymous communication
- Quality Control Issues
  - Lack of test suites
- Inability to maintain Day Zero compatibility with new OS releases

AF_RXRPC, KAFS, AND KAFS-UTILS

Status Report

THE KAFS PROJECT
THE NATIVE LINUX KERNEL /AFS CLIENT

- The primary goal of the kAFS project is to provide a filesystem within the Linux kernel that can communicate with AFS and AuriStorFS servers.
- With the sources being in the upstream Linux kernel, kAFS can take advantage of GPL-only kernel APIs and can be fixed up by developers making wholesale Linux VFS interface changes.
- kAFS comprises four in-kernel components:
  - **kAFS** itself. This is a network filesystem that's used in much the same way as any other network filesystem provided by the Linux kernel, such as NFS.
  - **AF_RXRPC**. This is a network protocol that provides the network interface for kAFS to use and also offers userspace access via the socket interface.
  - **Kernel keyrings**. This facility's primary purpose is to retain and manage the authentication tokens used by kAFS and AF_RXRPC, though it has been kept generic enough that it has been adapted to a variety of roles within the kernel.
  - **FS-Cache**. This is an interface layer acts as a broker between any network filesystem and local storage, allowing retrieved data to be cached. It can be used with NFS, CIFS, Plan9 and Ceph in addition to kAFS.
- and two userspace components:
  - **kafs-utils**. This is a suite of AFS management tools, written in Python3, that use AF_RXRPC from userspace as a transport to communicate with a server.
  - **keyutils**. This is a set of utilities for manipulating the keyrings facility from userspace.
- See https://www.infradead.org/~dhowells/kafs/ for detailed status and example code.

STATUS OF AF_RXAFS DEVELOPMENT

- Implemented features
  - Usable from userspace via socket(AF_RXRPC, ...)
  - Supports client and server calls over the same socket
  - Kauth, Kerberos 4 and Kerberos 5 security
    - plain, partial and full encryption
  - IPv6 support
  - Kernel tracepoints
  - Slow start congestion management

- Features that need to be added for AuriStorFS
  - Service upgrade
  - YFS-RXGK security class

STATUS OF KAFS DEVELOPMENT

- Implemented features
  - Reading and writing
  - Advisory file locking
  - Encryption and authentication
  - Automounting of mountpoints
  - Failover (DB and FILE services)
  - AFSDB and SRV DNS record lookup

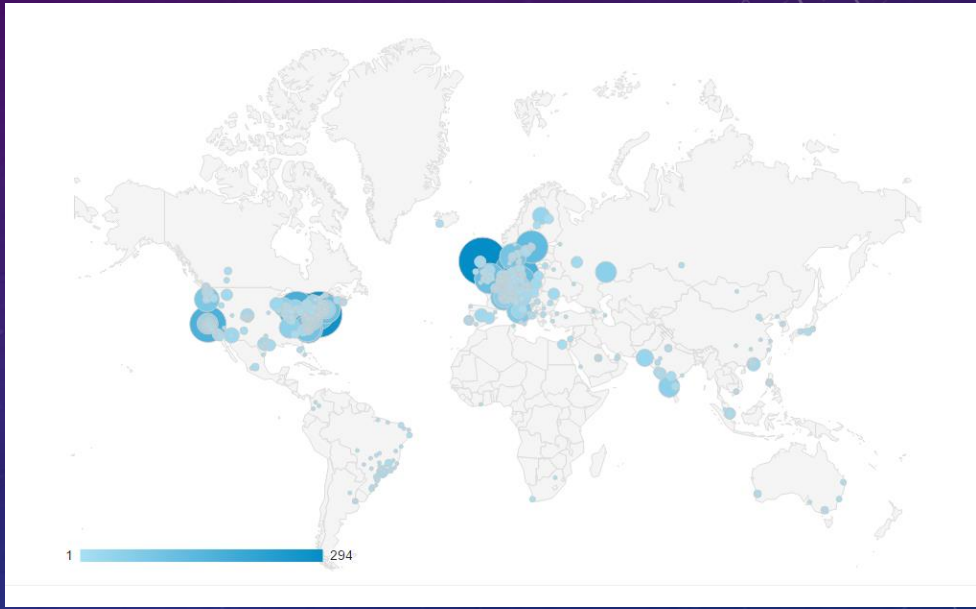
- Features still to be added
  - Kernel tracepoints.
  - Userspace tool interface (substitute for pioctl)
  - Notifications (inotify and other GPL_ONLY friends)
  - Bulk read and write for large files
  - IPv6 support. This requires:
    - New AFS3 Standardization of RPCs that can transmit IPv6 addresses in addition to IPv4 addresses, or
      Implementation of AuriStorFS RPCs that include IPv6 endpoints
  - AuriStorFS RX service RPCs and upgrades
  - Handle volumes being moved or being busy
  - @sys symlink handling

AURISTOR FILE SYSTEM

Status Report

AURISTOR FILE SYSTEM

- U.S. Dept of Energy
  SBIR 2008 - 2011
- Worldwide deployments began in April 2016
- Satisfies EU Privacy Requirements
- Zero Flag Day conversion from IBM AFS and OpenAFS



AURISTOR FILE SYSTEM KEY FEATURES

- Security
  - Combined identity authn
  - Multi-factor authz
  - Volume and File Server policies
  - Data privacy for all services
  - Need to know policies applied to all metadata and data (EU Privacy Laws)
- Networking
  - IPv6
  - Can saturate multiple bonded 10Gbit NICs
- File Server and DB Server Performance and Scale
- UBIK DB quorum
  - Accelerated establishment
  - Membership scaling up to 80 per cell
- Supports multi-producer, multi-consumer work flows
- 2038 compatible
- Simplified configuration management
- Coverity analyzed
- Robust test suite
- Docker container deployments
- Optimized Linux clients
- Servers run as non-privileged user; not “root”

AURISTOR FILE SYSTEM
EXTENDED ACCESS CONTROL MODEL

- Combined Identity Authentication (sequence of identities)
  - Anonymous User or Authenticated User or Authenticated Machine
  - Anonymous User and Authenticated Machine
  - Authenticated User and Authenticated Machine
- User Centric, Constrained Elevation Access Control Entries
  - Grant permissions to matching prefix substrings of the combined identity sequence
  - Grant permissions based upon membership in more than one group
- Permits elevated access from trusted machines
- When negative access control entries are used, revokes permissions from untrusted but authenticated machines

AURISTOR FILE SYSTEM INTENDED USE CASES

- Cross-platform home and project directories
  - Local and remote access
- Open Data Sets
  - Long term storage
  - Direct access from remote locations (No need for GridFTP)
- Management of Data by Classification
- Software Distribution
  - Direct from the name space (@sys)
  - Linux Containers
- Secure Federated Global Name Space
- Extending Storage infrastructure across
  - Internal
  - DMZ
  - Cloud hosting providers

LINUX CONTAINERS

- The future of software deployment but has challenges when it comes to data access
  - No common file system namespace hampers portability
  - No shared file system across containers
    - No work around when executing in multiple hosts or data centers
  - Network file system access requires exposing secrets to the container

- /afs is a perfect file system name space for use from within containers
  - Requires the availability of /afs in the Linux distributions
    - Calling out Scientific Linux maintainers
    - For RHEL, SUSE, Debian, Ubunutu, others must bring kAFS and AF_RXRPC to production status
  - Requires the ability to start containers with a predefined network identity and secret
    - Will work with AuriStorFS Extended Authz model to restrict data access to container instance executing on a trusted host

TENNESSEE OPEN RESEARCH CLOUD (TORC)

- Vanderbilt University, UT Knoxville and UT Memphis submitted a US$3M Innovative Integrated Storage Resources grant request
  - NSF Campus Cyberinfrastructure (Solicitation 16-567)
  - https://www.auristor.com/documentation/TORC.pdf
- Uses AuriStorFS to provide a secure global federated cross-platform name space backed by Vanderbilt University ACCRE's Logistical Storage (L-Store)
  - http://www.lstore.org/
  - Intended to provide enhanced storage access for researchers requiring 100gbit or better throughput to large data sets
  - AuriStorFS manages all name space security and metadata
  - L-Store provides parallel high speed direct access from clients to the backing store (when available)
    - AuriStor File Servers provide access for non-L-Store accessible clients
  - All AFS transaction, security and cache coherency semantics are preserved

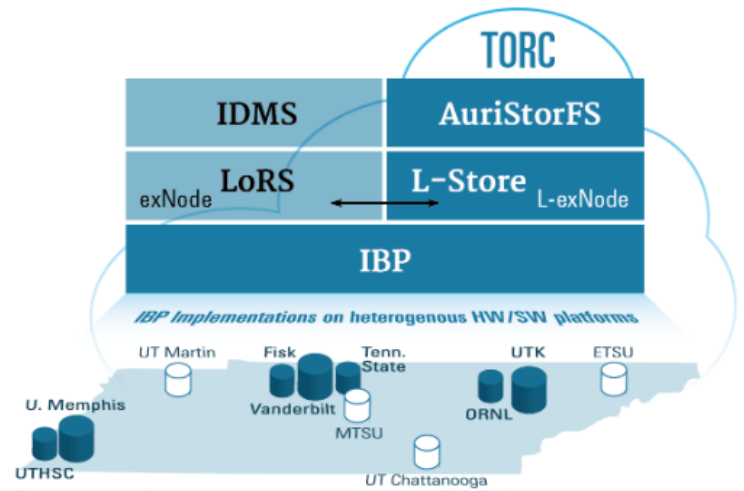


Figure 1: Simplified picture of the TORC stack and the deployment of depots in TORC's core (dark blue) and prospective partnering sites (white).

AURISTOR

YOUR DATA · YOUR FILE SYSTEM

THE FUTURE OF AFS®

STATUS REPORTS: AURISTOR FILE SYSTEM, KAFS, AND OPENAFS USE CASES

TENNESSEE OPEN RESEARCH STORAGE CLOUD (TORC)