AURISTOR

YOUR DATA · YOUR FILE SYSTEM

AFS UPDATE – SPRING 2018

AURISTOR FILE SYSTEM AND LINUX KERNEL AFS/AF_RXRPC

PRESENTED BY JEFFREY ALTMAN
(JALTMAN@AURISTOR.COM)

2016 ACM SYSTEM SOFTWARE AWARD RECIPIENTS "THE ANDREW FILE SYSTEM DEVELOPERS"

- **Mahadev Satyanarayanan**, **Michael L. Kazar**, **Robert N. Sidebotham**, **David A. Nichols**, **Michael J. West**, **John H. Howard**, **Alfred Z. Spector** and **Sherri M. Nichols**, recipients of the ACM Software System Award for developing the Andrew File System (AFS). AFS was the first distributed file system designed for tens of thousands of machines, and pioneered the use of scalable, secure and ubiquitous access to shared file data. To achieve the goal of providing a common shared file system used by large networks of people, AFS introduced novel approaches to caching, security, management and administration. AFS is still in use today as both an open source system and as the file system in commercial applications. It has also inspired several cloud-based storage applications.
- The 2016 Software System Award recipients designed and built the Andrew File System in the 1980's while working as a team at the Information Technology Center (ITC), a partnership between Carnegie Mellon University and IBM. Many other universities soon integrated AFS before it was introduced as a commercial application. Many of this year's recipients also contributed to two foundational AFS papers: The ITC Distributed File System: Principles and Design, published in Proceedings of ACM SOSP 1985 and Scale and Performance in a Distributed File System, published in Proceedings of ACM SOSP 1987.
- *The ACM Software System Award is presented to an institution or individual(s) recognized for developing a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both. The Software System Award carries a prize of $35,000. Financial support for the Software System Award is provided by IBM.*
  - https://awards.acm.org/about/2016-technical-awards
- ACM System Award Video - https://youtu.be/02UOfE1ePfc

AF_RXRPC, KAFS, AND KAFS-UTILS

Status Report

THE KAFS PROJECT
WHY A NATIVE LINUX KERNEL /AFS CLIENT?

- IBM AFS, OpenAFS and AuriStorFS are license incompatible with Linux kernel
- New Linux kernel releases every six to eight weeks
  - Constant changes to internal kernels interfaces – few compatibility guarantees
  - Interface change developers are only responsible for fixing in-tree source code
  - Without stable interface semantics, function call changes can silently introduce data corruption – let alone break /afs behavior
  - Mainline changes have full transparency – Distribution kernels do not
  - Too many variants to keep track of
- As an integrated part of the mainline Linux kernel
  - kAFS can leverage GPL-only kernel APIs and benefits from all Linux mainline development processes
    - Tree wide Linux VFS interface changes must be applied to kAFS at the same time as other in-tree file systems
    - KASAM and Coverity testing is performed against kAFS
    - Code review by a much broader community of storage and network stack developers

THE KAFS PROJECT
THE NATIVE LINUX KERNEL /AFS CLIENT

- The kAFS project provides access to the /afs global file namespace or to individual AFS volumes from the Linux kernel without any third party kernel or FUSE modules.
- kAFS shares no source code with prior RX and AFS implementations.
- 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 socket family implementation of the RX RPC protocol for use by kAFS and userspace applications.
  - **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 family, CIFS, Plan9 and Ceph in addition to kAFS.
- and three userspace components:
  - **kafs-client**. This will be the configuration and systemd integration for automatically mounting the kAFS filesystem and tools for managing authentication.
  - **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_RXRPC DEVELOPMENT (LINUX 4.18)

- Implemented features
  - Usable from userspace
    - `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 and fast restart congestion management
  - Service upgrades
- Features that need to be added for AuriStorFS
  - YFS-RXGK security class

STATUS OF KAFS DEVELOPMENT (LINUX 4.18)

- Implemented features:
  - POSIX syscalls
  - Advisory file locking
  - Encryption and authentication
  - Automounting of mountpoints
  - Failover (DB and FILE services)
  - Kernel tracepoints.
  - DNS SRV and AFSDB record lookup
  - Busy volume failover
  - Path substitution (@sys and @cell)
  - IPv6 support (requires AuriStorFS DB and FILE)
  - AuriStorFS RX service RPCs
  - Dynamic root /afs mount
    - `mount -t afs none /afs -o dyn`
  - Direct volume mount
    - `mount -t afs "#grand.central.org:root.cell" /mnt`

https://www.infradead.org/~dhowells/kafs/todo.html

LINUX AFS NEXT STEPS

- Linux AFS will significantly reduce the development burden on the OpenAFS community
- File feature requests with your preferred Linux distribution
  - For Red Hat, must first be distributed in Fedora
  - It will not be in RHEL 8.0!
    - No technical barrier to addition in a subsequent release

AURISTOR FILE SYSTEM

Status Report

AURISTOR FILE SYSTEM
AFS WITHOUT FEAR!

- The global /afs file namespace has a long history of use within the scientific computing community
  - Federated authentication
  - Home and shared Project directories
  - Software distribution
  - Cross platform distributed access to files and directory trees over the WAN
  - Anything that benefits from atomic publication and/or readonly replication
  - Global data replication and live data migration
- /afs now also handles work flows that require
  - Hundreds of nodes modifying a common set of objects (multiple writer, multiple reader)
  - Hundreds of processes on a single multi-processor thread client system
  - Robust multi-factor authorization and security (auth/integ/priv) requirements

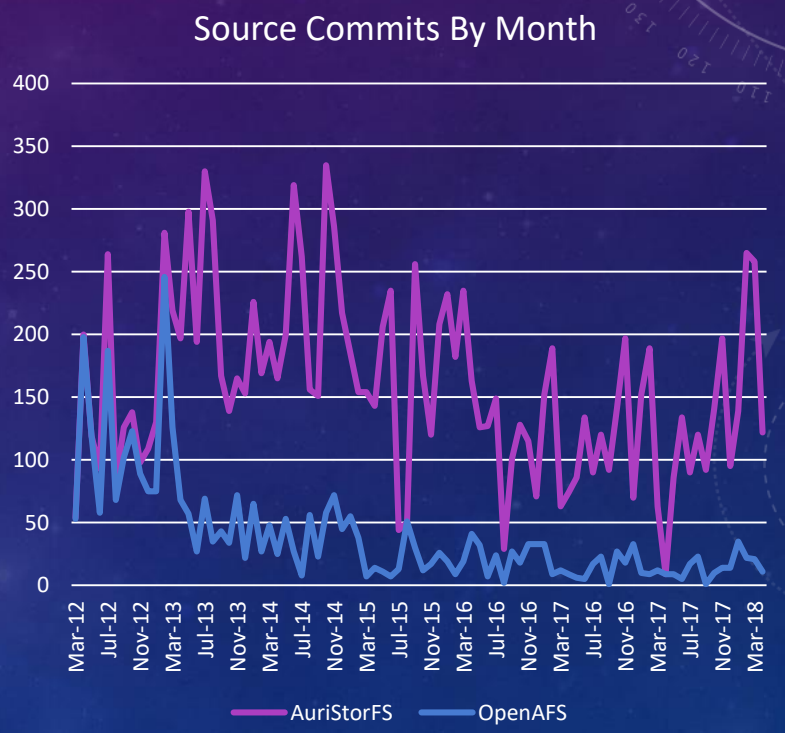
Our partners push the limits of /afs without fear!
No more walking on egg shells.

AURISTOR FILE SYSTEM
MAJOR MILESTONES SINCE HEPIX 2016

- Unlimited data transfer RX calls
  - March 19th – World's first RX call to send more than 5.63TB
- 16 Zettabyte Volume Quotas
- World's largest AFS volume – 250TB
  - Fully functional – migration, replication, backup, restore
- 500,000 sustained RX connections per fileserver
- Linux cache manager scaling beyond 64 simultaneous processor threads with minimal resource contention.
  - Starting from an empty cache, 64 instances of the same Python process reading 6500 class files from /afs required just 3s longer to start (30s) than a single instance (27s).
  - The same workload used to take more than 13 minutes to start on the same hardware and OS due to resource contention.

AURISTOR FILE SYSTEM DEVELOPMENT BY THE NUMBERS 2017

- 8 contributing developers
- 13,638 code review submissions
- 196 software errors detected by Coverity
- 1283 commits
- 1395 files changed, 76,118 insertions, 61,391 deletions
- 258 new tests added (~6400 total)
- 165,116 continuous integration builds spanning 16 different platforms
- 19 tagged releases
- 8332 Linux kernel module



AURISTOR FILE SYSTEM
LATEST LINUX PLATFORM SUPPORT

- Distributions:
  - Red Hat Enterprise Linux 7.5 (and clones)
  - Red Hat Enterprise Linux 6.10 Beta
  - Red Hat Fedora 28
  - Debian Stretch
  - Ubuntu 18.04 LTS
- New Architectures
  - PPC-LE
  - aarch64
- End of support
  - Red Hat Enterprise Linux 5.x
  - Fedora 26

AURISTOR FILE SYSTEM MELTDOWN AND SPECTRE

- December 29th 2017 -- testing KPTI changes merged by Linus Torvalds
  - Potential performance hit of 15% or more depending upon
    - Processor family support for PCID
    - Linux kernel version
    - syscall pattern – does syscall result in loss of processor time slice?
  - AuriStorFS lock free coding practices particularly hard hit
- January 2nd 2018 AuriStor notified partners of embargoed security flaw
- March 6th 2018 – First mitigations shipped to customers
  - 50% reduction in syscalls to send RX packets
  - 50% reduction in syscalls performed by UBIK
  - Reductions in fileserver and volserver read and write syscalls
  - Customers urged to upgrade to RHEL 6.7 or later; or RHEL 7.0 or later -- enables
    - 90% reduction in syscalls to receive RX packets
    - Up to 16 RX listener threads (data pumps)

AURISTOR RX SECURITY
NEW CRYPTO ENGINE

- Today, AuriStorFS leverages Operating System vendor's crypto engine
  - FIPS certificate, speed, access to hardware accelerators
  - Integrated support into Heimdal's hcrypto framework
    - Microsoft Crypto Next Gen (WinCNG), Apple Common Crypto, Solaris PKCS#11, Linux OpenSSL
- Simon Wilkinson's new crypto library leverages the latest x86/x64 instructions
  - Intel **Advanced Encryption Standard New Instructions (AES-NI)**
  - **Streaming Single Instruction Multiple Data Extensions (SSE, SSE2, SSE3)**
  - Advanced Vector Instructions (AVX, AVX2)
- to encrypt, decrypt, sign and verify RX packets at a high level of efficiency
- Compared to rfc3961 kernel implementation contributed by AuriStor to OpenAFS in 2010; and userland OSX Common Crypto (built from OpenSSL)
- Faster crypto -- Reduced latency

AURISTOR RX SECURITY
NEW CRYPTO ENGINE - RESULTS

- Measurements were performed on a MacBook Air
  - macOS Sierra
  - Intel(R) Core(TM) i5-4250U CPU @ 1.30GHz (Haswell 22nm)
    - Intel® SSE4.1, Intel® SSE4.2, Intel® AVX2
    - Intel® AES New Instructions
- Time to compute 5,000,000 rxgk packets in a single thread
  - Worst case for encryption is MIT with builtin crypto at 203 seconds (or 24,630 packets/sec)
  - Best case is 23 seconds (or 217,391 packets/sec)

Faster than …	RFC3961 (kernel)	CommonCrypto	MIT Krb5 (built-in)	MIT Krb5 (OpenSSL)
Encryption	6x	3x	9x	3.5x
Decryption	5x	5x	14x	6x
Sign and Verify	3x	2x	9x	2x

AURISTOR FILE SYSTEM VOLUME TRANSACTION LIFECYCLE IMPROVEMENTS

- New or updated commands:
  - “vos status”, “vos movesite”, “vos copysite”, “vos eachfs”, “vos eachvl”, “vos eachvol”
- Improved behavior for system administrators
  - YFSVOL RPC suite removes race conditions associated with volume transactions
  - volservers automatically garbage collect temporary or partial volumes created by interrupted or failed “vos” commands
  - Read-only volumes can be moved between servers or partitions on the same server
  - Read-write volume moves recreate the backup volume on the destination
    - Avoids loss of nightly backup if move occurs between snapshot and backup
  - Improved status reporting
    - Bytes sent and received for each call bound to a transaction
    - Search transactions by volume group
  - Volserver validates volume transaction metadata against location database for consistency checks

AURISTOR FILE SYSTEM
GENERAL DATA PROTECTION REGULATION (GDPR)

- AuriStorFS blocks bulk queries of Protection and Location Service content ….
  - Except by Read-write and Read-only super users
- Areas of privacy concern:
  - Protection Service stores User Identities and Aliases
  - Volume names
    - Does your cell label volumes with the user's name?
  - File system directory entry names
    - Are mount points or symlinks to personal volume root directories labeled with the user's name?
  - Audit logs record Authentication Identities and Network Endpoints
    - GSS Display Names or Kerberos v5 principals
  - Debug logs might contain volume names and directory entry names
  - RX and CM Debug interfaces expose (to anonymous callers)
    - Volume names and IDs
    - Network Endpoints of recent connections and peers
    - Accessed cell information

AURISTOR FILE SERVER WORKAROUND FOR IBM/OPENAFS CLIENT RXAFSCB DEADLOCKS

- All IBM and OpenAFS Unix cache managers are vulnerable to callback service deadlocks
  - A deadlocked client will be unresponsive to all RXAFSCB RPCs
  - FS times out after 120 seconds; responds to failed RXAFSCB RPC by failing inbound RPCs with VBUSY.
  - CMs retry VBUSY failures 100 times before failing the VFS operation
  - A deadlock can resolve itself after approximately 2.5 to 3 hours

- AuriStorFS Fileserver
  - detects deadlocked RXAFSCB and fails inbound RXAFS RPC to prevent retries and force immediate VFS operation failure

AURISTOR BUG FIXES SUBMITTED TO OPENAFS

- The AuriStor team no longer develops OpenAFS but continues to protect our partners
  - Ensure that changes introduced by the OpenAFS salvager will be noticed by backup applications. (OpenAFS 1.6.21)
  - rx window size handling flaw leads to periodic 100ms delays. (OpenAFS 1.6.21)
  - remote triggered rx assertion failure http://www.openafs.org/pages/security/#OPENAFS-SA-2017-001
  - SRXAFS_InlineBulkStatus fails to set AFSFetchStatus.InterfaceVersion on error https://gerrit.openafs.org/#/c/13067/
  - “vos move” failures if VTDeleteOnSalvage flagged volumes are missing https://gerrit.openafs.org/#/c/12976/
  - CVE-2018-7168 – RXAFS_StoreACL deprecation to prevent assignment of incorrect ACLs https://gerrit.openafs.org/#/c/12942/

OPENAFS 1.8
AURISTOR CONTRIBUTIONS

- From 2008 through 2011, AuriStor (fka Your File System) was awarded Phase I & II SBIR Commercialization grants from U.S. Dept of Energy
- OpenAFS 1.8 includes all of the work that was developed by AuriStor with support by U.S. DoE.
  - Significant RX improvements
  - Much of the pthreading work
  - Much of the conversion to libtool generated shared libraries
  - Fixes for many hundreds of Coverity detected coding errors
  - Support for External Source trees
  - Conversion to Heimdal hcrypto and roken portability library
  - Kernel mode RFC3961 crypto library
  - OpenAFS Portable Runtime library (red/black trees, Jenkins hashing, atomics)
  - New extended KeyFile format

OPENAFS 1.8
LINUXBOX CONTRIBUTIONS

- LinuxBox (aka Matt Benjamin) contributed:
  - Cache Bypass
    - Similar to an automatic Direct I/O for read operations
    - Prevents storing data in the AFS cache for files larger than a certain size
    - The intended purpose is to avoid cache manager bottlenecks and flushing of cache content for small files and directories
  - File Server RX connection pools
    - Increases the number of RX calls (from 4 to 12) that can be issued in parallel from a single Process Authentication Group (AFS token set)
    - This increases the number of background FetchData RPCs that can be issued in parallel ….
    - *This change can result in a 200% increase in fileserver load!!!!*
    - OpenAFS 1.8 does not include any improvements in fileserver RPC scaling

AURISTOR

YOUR DATA · YOUR FILE SYSTEM

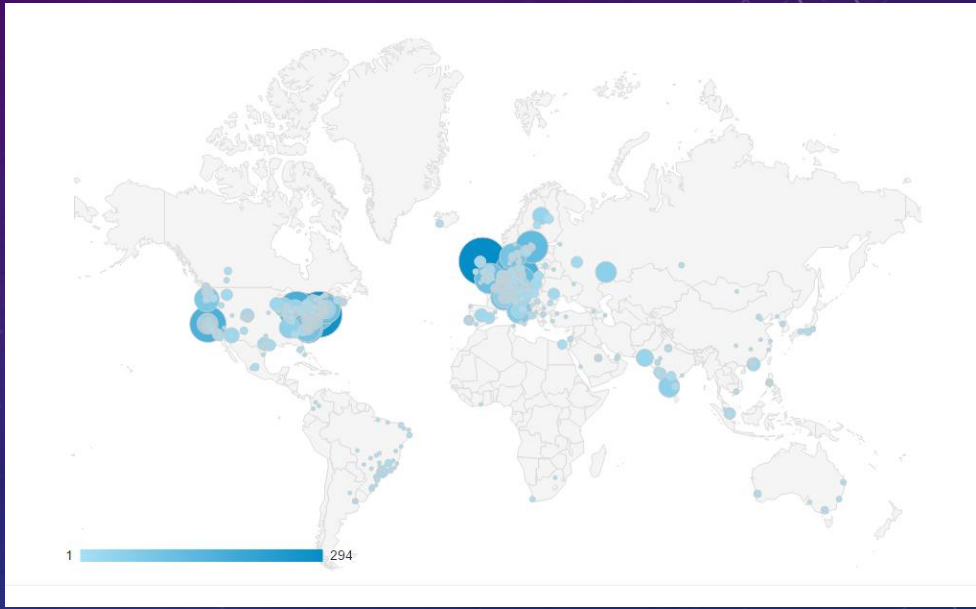
END OF AFS UPDATE – SPRING 2018

AURISTOR FILE SYSTEM AND LINUX KERNEL AFS/AF_RXRPC

PRESENTED BY JEFFREY ALTMAN
(JALTMAN@AURISTOR.COM)

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”
- Arbitrary file server vice partition backing store
  - Any POSIX file system – local or remote
  - Object stores

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