The AFS file system

DRAWING PICTURES OF CLOUDS LONG BEFORE IT WAS TRENDY

Richard Brittain, 2021/08/22 Dartmouth College Research Cyberinfrastructure.

Acronyms

- Places
  - IBM International Business Machines
  - MIT Mass. Inst. of Technology
- Filesystems
  - AFS Andrew File System
  - NFS Network File System
  - NTFS New Technology File System
  - SMB Server Message Block
  - CIFS Common Internet File System
- Networking
  - WAN Wide Area Network
  - LAN Local Area Network
  - SAN Storage Area Network
  - FC Fibre Channel
  - RAID Redundant Array of Independent Disks
  - DNS Domain Name Server
  - TCP Transmission Control Protocol
  - UDP User Datagram Protocol
  - RPC Remote Procedure Call
- Application Services
  - KDC Key Distribution Center
  - PTS Protection database server
  - VLDB Volume Location Database
- Access Control
  - ACL Access Control List
  - RO Readonly
  - RW Read+Write

About me

- Grew up in Newcastle upon Tyne, UK
- Oxford University (Physics undergrad)
- Southampton University; Cornell University; Dartmouth College
  - Career grad student
  - Auroral Radio Physics; sounding rocket experiments; field work in Iceland, Norway, Alaska, Canada, Antarctica. Data acquisition systems
  - FORTRAN programmer
- Dartmouth College
  - Career sideslip into Unix Sysadmin and Research Support, Computing Services.

Timeline of Network Filesystems

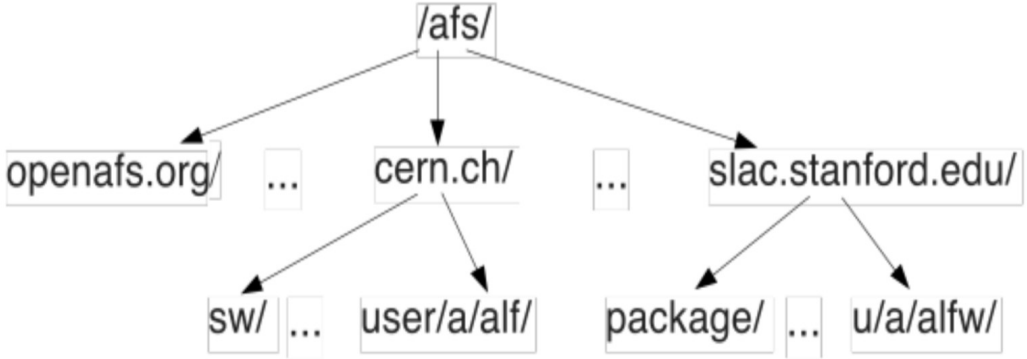
- 1983 SMB1 (IBM); Andrew project started at CMU; distributed computing
- 1988 Kerberos 4 (MIT); AFS (Carnegie-Mellon); AFS comes to Dartmouth; Project Northstar
- 1989 CMU spawns Transarc Corp to market AFS; NFS2 (Sun)
- 1990 Richard comes to Dartmouth
- 1993 Kerberos 5 (MIT); Arla project (AFS compatible) started in Sweden; NTFS (Microsoft)
- 1994 IBM acquires Transarc
- 1995 NFS3 (Sun)
- 1996 SMB1 -> CIFS (MS)
- 2000 IBM releases AFS code, OpenAFS 1.0; NFS4.0 (IETF)
- 2006 SMB2 (MS)
- 2009 Auristor founded
- 2010 NFS4.1 (IETF, Panasas)
- 2021 OpenAFS 1.8.8 (current release)

Core features of AFS

- Purely a software product
  - Supported on a wide variety of hardware
- AFS uses Kerberos v5 for authentication
- AFS client software
  - Cache manager
  - Universal mount /afs/cell (\\afs\cell)
- AFS metadata servers
  - High Availability features; distributed databases
  - Authorization (PTS) and Data location (VLDB) services
- File servers
  - Multiple servers; network distributed; data location independence (cloud!); WAN and LAN

AFS as a worldwide filesystem

- "AFS is a distributed filesystem that enables co-operating hosts (clients and servers) to efficiently share filesystem resources across both local area and wide area networks" (AFSWiki)



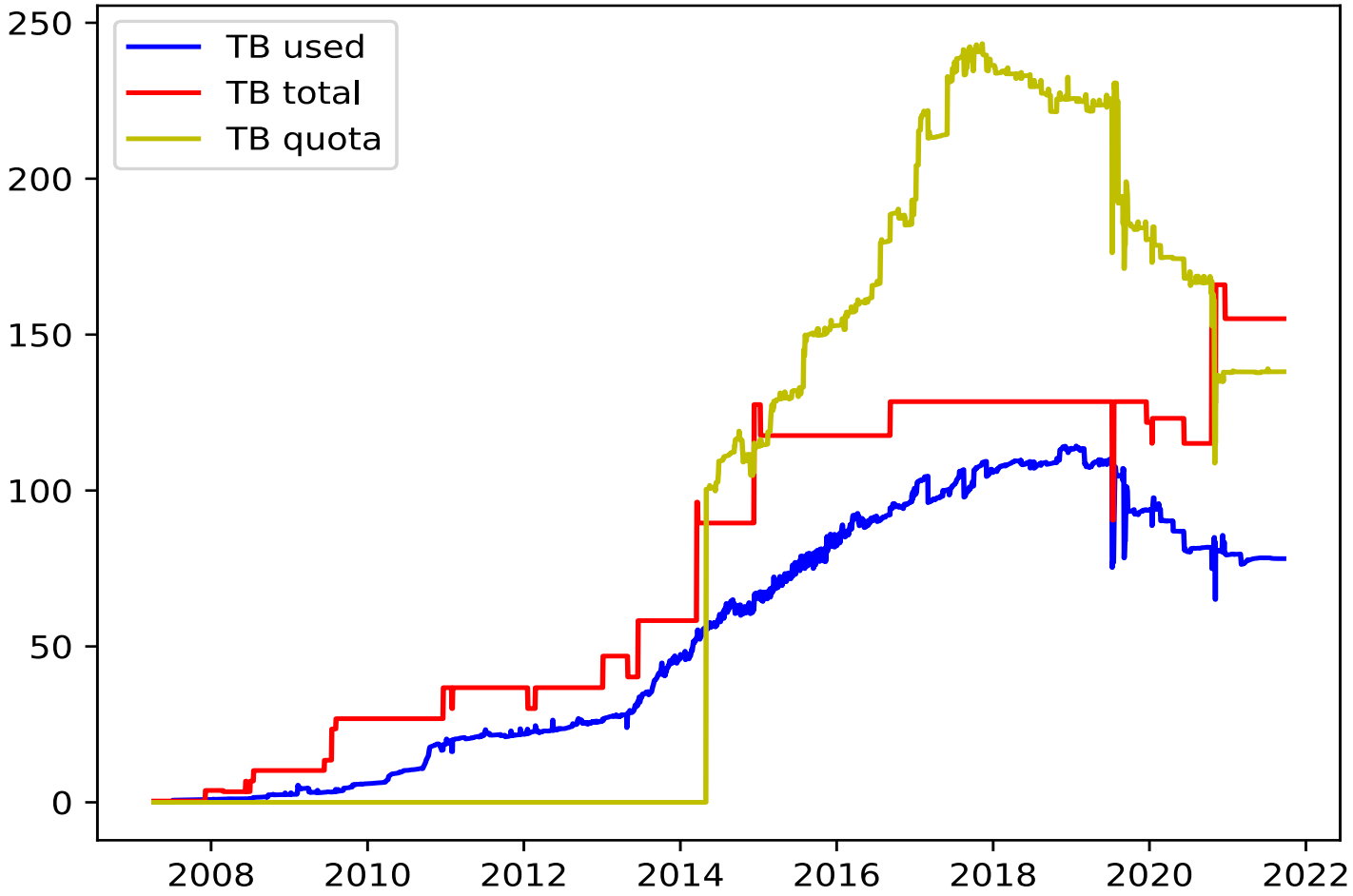
- `openafs.org`, `cern.ch` etc. are AFS Cells

- The highest level of organization is a *Cell*
- Cells usually named after DNS domains
- With appropriate credentials, a client can see other cells
- For legacy reasons, Dartmouth's cell is *northstar.dartmouth.edu*
- Everything in a cell shares the same authentication realm(s)
- Primary users are EDU, research labs, some large corporations

AFS use at Dartmouth

- Used initially as home directories for fleet of Unix workstations across campus (Project Northstar; Thayer School and IBM funding)
- Software installation and distribution; curricular software development
- First generation
  - 1 MB home quotas
  - 300 MB per file server, < 1GB total space
- Web server delivering content from AFS (Caligari, aka rcweb)
- Shared home directories for previously stand-alone research Unix servers
- 2008 Branded as RStor
  - Research storage volumes up to 2TB; 50GB homes
  - Expanded hardware, RAID
  - Billing
  - Improved locally developed backup system (*vos dump* to NetBackup)
- Trusts two Kerberos realms equally
  - RSTOR Kerberos realm (legacy usernames, MIT Kerberos)
  - KIEWIT Kerberos realm (NetID, MS Active Directory)
- PBS Department runs an independent cell (dbic.dartmouth.edu)

Over time: AFS/RStor at Dartmouth



- First 20 years we did not save usage metrics
- At least 6 generations of server hardware
- From 2008 we kept daily summary metrics as part of the backup process
- 2013, we dropped the price
- Mid-2014, we added total quota information
- In 2017, we start migrating data to DartFS

Architecture: overview

- File Servers supported on most variants of Unix and most available block storage (local disk, local RAID, FC SAN)
  - Server implements volumes, directories and files with metadata (ACLs) and file contents. Does not depend on host OS filesystem features. Not encrypted, but very obscure
  - Early versions played tricks with inodes for optimization
- Low overhead per client. High client:server ratios
- Unit of storage is the *Volume*.
  - Typically 1 per user; 1 per project, 1 per application install
  - Mounted in the filesystem tree
- The *Volume* is the unit for quotas, server migration and backup
  - Live migrations between servers
- Cellname, VolID, vnode combine to a globally unique identifier of a file
- Replicated volumes, multiple RO, single RW; synchronization on demand

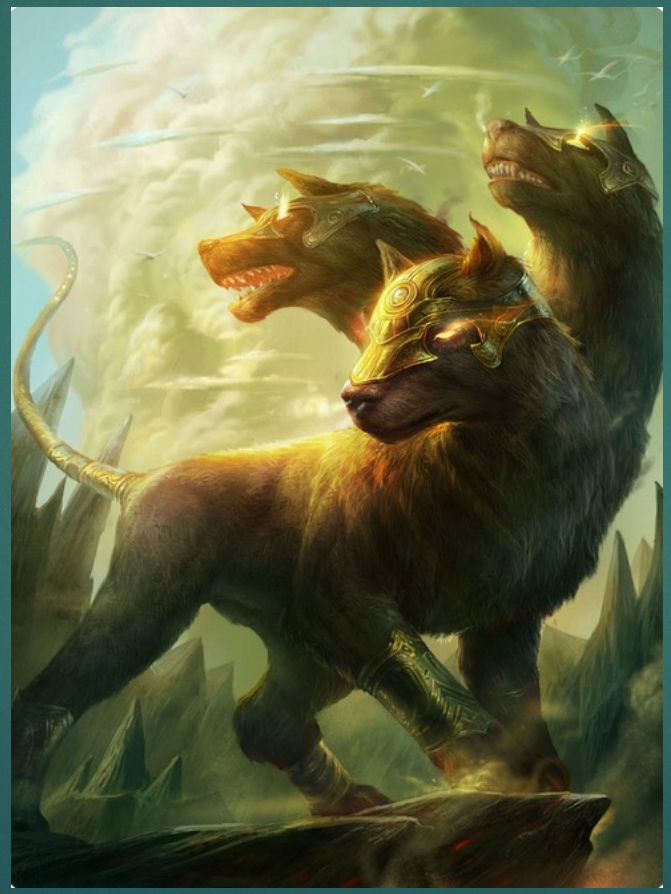
Architecture: clients

- Client cache manager (kernel module) presents a mounted filesystem with mostly POSIX semantics
- Uses Kerberos to authenticate users and file servers
  - Can leverage multiple Kerberos realms
- Use DNS to discover metadata servers (KDC, PTS, VLDB)
- Uses VLDB to discover location of files.
- Caches chunks of files on local disk
  - Stateful client
  - Uses callbacks to maintain coherency (weak)
- Write to local cache, flush to server on close(), or when cache full
- Identical view of storage from each client
- Client chooses between replicated volumes when available
- All traffic is client-initiated apart from callbacks
- Local root has NO access to tokens (generally), so no access to AFS home directories

Architecture: servers

- Configuration changes can be made from any client
  - Rarely need to directly log into servers
- Volumes can be moved from one server to another while active
  - Rebalance traffic
  - Add new servers
  - Drain servers for maintenance
- File servers can not access their own data, unless client is installed
- Metadata servers configured as high availability replicated databases
  - Implement the PTS database, mapping Kerberos identities to UID numbers, and group memberships
  - Implement the VLDB database, mapping mount points in the filesystem to file servers
  - 3 small VMs (not all on same host!)
  - RPC/UDP, Custom protocol (UBIK)

Authentication: Kerberos



Kerberos overview:

- Used daily by everyone with Active Directory authentication
  - AFS authentication is explicitly leveraging Kerberos features
- Kerberos provides *authentication*; not *authorization*
  - MS Active Directory blurs those lines. With AFS, fileserver access controls and PTS server provide authorization
- Mutual authentication of trusted parties in untrusted environments
  - Authenticates you as a user, *and also* the service you connect to
- Single Sign-On: authenticate to multiple services after an initial credential acquisition.
- Kerberos v5 uses symmetric key encryption
  - Public key encryption can be used for initial authentication
  - Secure generation and distribution of session keys

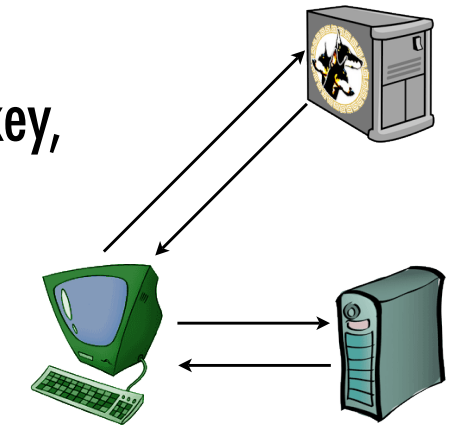
TGTs and service tickets

- Initial authentication creates a *Ticket Granting Ticket* (TGT). Compare it to a passport; limited lifetime; renewable; proves identity to other people without further documentation needed
- No passwords ever go on wire, even encrypted
- TGT is used to obtain a *Service Ticket* for a particular application. This contains an encrypted application server key
  - If the TGT is a passport, the Service Ticket is a visa stamp
- Service ticket is presented to the Application Server, which validates the enclosed encrypted key and authenticates the client
- AFS uses a special form of service ticket (Token) which is attached to *every* filesystem operation
  - When logging in to research Linux servers with NetID, AFS token is automatically generated
- Active Directory tickets are 10 hr. lifetime, renewable up to 30 days

Authenticating to a service

Authenticating to a service

- Client requests service ticket from KDC
- KDC returns service ticket and session key, encrypted with user's password
- Client sends service ticket and authenticator to application
- Application returns authenticator timestamp, encrypted with session key
  - thus proving that it knows the session key, and read the authenticator



Access Control Lists (ACL)

- NFS4 and SMB ACLs heavily influenced by AFS ACL design
  - Permission model in DartFS is similar, but more complex
- 7 permission bits deemed enough (NFS4/SMB has 14)
- AFS ACL on directories only – not per file
- Files always inherit permissions of the parent directory
- Creating a directory initially inherits the permissions of its parent
- Permissions apply to individuals or groups

Permission bits

Access control flags

- r read file contents
- l lookup (read directory contents, traverse directory)
- w write file contents (and append)
- i insert to directory (create file)
- d delete from directory (delete file)
- k lock file
- a administer (change the ACL, implicit for owner)

- POSIX bits (*ls –l*) are not accurate, but rarely a problem
- *chmod* +x works to set execute bit
- Negative permissions possible

Groups

- System groups
  - System:anyuser (public)
  - System:authuser (like domain user)
  - System:administrators (like root)
- Per-user group namespace
  - username:groupname
  - Users administer their own groups
- Groups can be nested

PAGs

- PAG (Process Authentication Group)
  - Separate credentials for groups of processes on same computer
- One token per cell per PAG
- Without PAGs, the cache manager uses local UID for identification, therefore a root account can switch UIDs and steal a token
- MacOS has no PAGs
- Windows has 1 PAG per login session
- Linux is trying hard to make PAGs impossible for desktop consoles

Problems with AFS

- 2TB volume limit (quota)
- Performance cannot saturate 10Gb links.
  - Disk arrays and network are now faster than the server software and network protocol
- Poor parallelization of processes within a given file server
- Desktop client (MacOS, Windows) increasingly difficult to build
  - Security features on desktop
  - Could not work with commercial support to build and distribute signed installers (Auristor, Sine Nomine)
- Lack of native backup tools. Native backup primitive is a volume dump which can be sent to NetBackup etc.
- File encryption is slow (doesn't leverage modern OS or hardware)
- Administration tools are a bit clunky
  - Lots of contributed tools to make things easier

Future of AFS

- OpenAFS is healthy, but mostly Linux oriented now
- Auristor. Commercial rewrite, backwards compatible, many new features and removes most of the limitations
- Red Hat kernel module (minimal client)
- MacOS and Windows clients from Auristor

Current Status of Northstar Cell