SLAC-PUB-11152

Part III: AFS - A Secure Distributed File System

Alf Wachsmann

Submitted to Linux Journal issue #132/April 2005

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

Work supported by Department of Energy contract DE–AC02–76SF00515.

Part III: AFS - A Secure Distributed File System

III.1 Introduction

AFS is a secure distributed global file system providing location independence, scalability and transparent migration capabilities for data. AFS works across a multitude of Unix and non-Unix operating systems and is used at many large sites in production for many years.

AFS still provides unique features that are not available with other distributed file systems even though AFS is almost 20 years old. This age might make it less appealing to some but with IBM making AFS available as open-source in 2000, new interest in use and development was sparked.

When talking about AFS, people often mention other file systems as potential alternatives. Coda (http://www.coda.cs.cmu.edu/) with its disconnected mode will always be a research project and never have production quality.
Intermezzo (http://www.inter-mezzo.org/) is now in the Linux kernel but not available for any other operating systems.
NFSv4 (http://www.nfsv4.org/) which picked up many ideas from AFS and Coda is not mature enough yet to be used in serious production mode.

This article presents the rich features of AFS and invites readers to play with it.

III.2 Features and Benefits of AFS

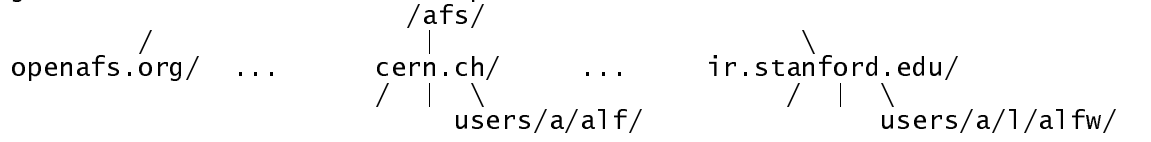
AFS client software is available for many Unix flavors: HP, Compaq, IBM, SUN, SGI, Linux. It is also available for Windows and Mac OS X.
This makes AFS the ideal file system for data sharing between platforms across local and wide area networks.

The following features were designed into AFS and not added as afterthoughts. This thorough design is one of the strength of AFS.

All AFS client machines have a local cache. A cache manager keeps track of users on a machine and handles the data requests coming from them. Data caching happens in chunks of files which are copied from an AFS file server to local disk. The cache is shared between all users of a machine and persists over reboots. This local caching reduces network traffic and makes subsequent access to cached data much faster.

The AFS file system is organized in a globally unique name space. Pathnames leading to files are not only the same wherever the data is accessed, the pathnames do not contain any server information. In other words, the AFS user is unaware on which file server the data is located. To make this work, AFS has a replicated data location database a client has to contact in order to find data. This is unlike NFS where the client has the information about the file server hosting a particular part of the NFS file system.

A global view of the AFS file space looks like this:



The different independent AFS domains are called "cells" and correspond to Kerberos realms.

A typical AFS pathname looks like this:

```
/afs/cern.ch/user/a/l/alf/Projects/
```

This pathname contains the AFS cell name but not the file server name.

This location independence allows AFS administrators to move data from one AFS server to another without any visible changes to users. It also make AFS very scalable. If you run out of space or network capacity on your AFS file servers, simply add another one and migrate data to the new server. Clients will not notice this location change. AFS also scales well in terms of number of clients per file server. On modern hardware, one AFS file server can server up to ~1000 clients without any problems.

For users, the AFS file space looks just like another file system they have access to. With the proper Kerberos credentials, they can access their AFS data from all over the world facilitating the globally unique name space.
Here an example: To be able to copy data from my Unix home directory at CERN (Switzerland) to my Unix home directory at SLAC (California) I first need to authenticate myself against the two different AFS cells:

```
% kinit --afslog alfw@ir.stanford.edu
alfw@ir.stanford.edu's Password:
% kinit -c /tmp/krb5cc_5828_1 --afslog alf@cern.ch
alf@cern.ch's Password:
```

AFS comes with a command "tokens" to show AFS credentials.

```
% tokens
Tokens held by the Cache Manager:

User's (AFS ID 388) tokens for afs@cern.ch [Expires Apr  2 10:30]
User's (AFS ID 10214) tokens for afs@ir.stanford.edu [Expires Apr  2 09:49]
   --End of list--
```

Now that I am authenticated, I can access my two AFS home directories:

```
% cp /afs/cern.ch/user/a/alf/Projects/X/src/hello.c \
     /afs/ir.stanford.edu/users/a/alfw/Projects/Y/src/.
```

On an AFS file server, the AFS data is stored on special partitions, called /vicepXX with XX ranging from "a" - "zz" allowing for a total of 256 partitions per server. Each of these partitions can hold data containers called volumes. Volumes are the smallest entity that can be moved around, replicated or backed up. Volumes then contain the directories and files.
Volumes need to be "mounted" inside the AFS file space to make them visible. These mount points look just like directories.

AFS is particularly well suited to serve read-only data like the /usr/local/ tree because AFS clients cache accessed data. To make this work even better and more robust, AFS allows for read-only clones of data on different AFS file servers. If one server hosting such a clone goes down, the clients will transparently fail-over to another server hosting another read-only copy of the same data. This replication technique can also be used to clone data across servers that are geographically far apart. Clients can be configured to prefer the close by copy and use the more distant copy as fall back. The openafs.org AFS cell, for example, is hosted on a server at Carnegie Mellon University in Pittsburgh, PA and on a server at the Royal Institute of Technology (KTH) in Stockholm, Sweden.

AFS provides a snapshot mechanism to provide backups. These snapshots are generated at a configurable time, like 2am, and work on a per volume basis. The snapshots can then be backed up to tape without interfering with user activities. They can also be provided to users via a simple mount point in their respective AFS home directories. This simple trick eliminates many user backup restore requests because the files in last night's snapshot are still visible in this special sub-directory (the mount point to the backup volume) in user's home directories.

The AFS communication protocol was designed for wide area networking. It uses

its own remote procedure call (RPC) implementation called Rx which works over UDP. The protocol retransmits only the single bad packet on a batch of packets and it allows a higher number of unacknowledged packets compared to other protocols.

AFS administration can be done from any AFS client. There is no need to logon to an AFS server. This allows to lock down the AFS server very tightly which is a big security plus. The location independence of AFS data also improves manageability. An AFS file server can be completely evacuated by moving all volumes to other AFS file servers. These moves are not visible to users. The empty file server can then undergo its maintenance like OS upgrade or hardware repair. Afterwards, all volumes can transparently moved back to the server.

Internally, AFS makes use of Kerberos to authenticate users. Out of the box this is Kerberos 4 but all major Kerberos 5 implementations are able to serve as a more secure substitute. AFS provides access control lists (ACLs) to restrict access to directories. Only Kerberos principals or groups of those can be put in ACLs. This is unlike NFS where only the Unix user ids are used for authorization. (Doesn't NFS stand for "No File Security"?)
An additional authorization service, the Protection Service (PTS), is used to keep track of individual Kerberos principals and groups of principals.

III.3 AFS Components

To make all these features work, AFS comes in several distinct parts:
The AFS client software that has to run on each computer that wants access to the AFS file space. The AFS server software is separated into four basic parts. It uses Kerberos for authentication, a Protection Server for authorization, a Volume Location server for location independence and two servers for data serving (File Server, Volume Server). All these different processes are managed on each AFS server by the Basic OverSeer (BOS) Server.
In addition to these necessary components, there are more service daemons available for AFS server maintenance and backup.
How to install an AFS server is beyond this article. Detailed instructions can be found at http://www.openafs.org/pages/doc/QuickStartUnix/auqbg000.htm

Because of all these different server components, the learning curve for AFS is very steep at the beginning. However, the payoff is rewarding and many sites cannot miss it any more. The day to day maintenance cost for AFS once a cell is installed is in the 25% FTE range even for very large installations.

For more information how AFS is used at various sites (including Morgan Stanley and Intel), have a look at the presentations given at the recent "AFS Best Practices Workshop" (http://www-conf.slac.stanford.edu/AFSBestPractices).

III.4 AFS Client Installation

You do not need your own AFS servers to try AFS yourself. Simply installing the OpenAFS client software and starting the AFS client daemon "afsd" with a special option allows to access the publicly accessible AFS space of foreign AFS cells.

The following section guides through the AFS client installation and shows how to add the necessary parameters to afsd.

The most difficult part in installing an AFS client is the necessary kernel module. If you are using Red Hat or Fedora, you can download RPMs from http://www.openafs.org/release/latest.html. For Fedora Core 1 the RPM is openafs-kernel-1.2.11-fc1.0.1.i386.rpm. For all other distribution or if you are running a different kernel, you need to download the source RPM and

compile the kernel module yourself. Be aware that you need the kernel sources available on the machine you do the compilation.

In addition to the kernel module, the AFS client needs a user space daemon (afsd) and the AFS command suite. These come in two additional RPMs: openafs-client-1.2.11-fc1.0.1.i386.rpm and openafs-1.2.11-fc1.0.1.i386.rpm.

The next step is to configure the AFS client for your needs. Firstly, you need to define the cell your computer should be member of. The AFS cell name is defined in the file /usr/vice/etc/ThisCell. If you do not have your own AFS servers, this name can be set to anything. Otherwise, it should be set to the name of the cell your AFS servers are serving.
The next parameter to look at is the local AFS cache. Each AFS client should have a separate disk partition for it but the cache can be put wherever you want. The location and size of the cache are defined in the file /usr/vice/etc/cacheinfo. The default location for the AFS cache is /usr/vice/cache and a size of 100 MB is plenty for a single user desktop or laptop computer. The cacheinfo file for this setting would look like this:

```
/afs:/usr/vice/cache:100000
```

This is the setting as it comes with the openafs-client RPM.

Next, configure the parameters for afsd, the AFS client daemon. They are defined in /etc/sysconfig/afs. Add the "-dynroot" parameter to the OPTIONS definition. This allows to start the AFS client without your own AFS servers.

Another option you want to add is "-fakestat". This parameter tells afsd to "fake" the stat(3) information of all entries in the /afs/ directory. Without this parameter, the AFS client would go out and contact each single AFS cell known to it (currently 133) if you do a long listing (/bin/ls -l) in the /afs/ directory.

Because AFS is using Kerberos for authentication, time needs to be synchronized on your machine(s). AFS used to have its own mechanism for this but it is outdated and should not be used any more. To switch it off, the flag "-nosettime" needs to be added to the OPTIONS definition in /etc/sysconfig/afs. If you don't have a time sync method, use NTP (http://www.ntp.org/).

After all changes, the new OPTIONS definition in /etc/sysconfig/afs should now look like this:

```
OPTIONS="$MEDIUM -dynroot -fakestat -nosettime"
```

The last step is to create the mount point for the AFS file system:

```
% sudo mkdir /afs
```

Now you can start the AFS client with

```
% sudo /etc/init.d/afs start
```

This takes a few seconds because afsd needs to populate the local cache directory before it can start. Since the cache is persistent over reboots, subsequent starts will be faster.

III.5 Explore AFS

Without your own AFS servers but with an AFS client configured as described above, you can now familiarize yourself with some AFS commands and explore the global AFS space yourself.

A quick test shows that you are not authenticated in any AFS cell:

```
% tokens

Tokens held by the Cache Manager:
```

```
          --End of list--
```
No credentials are listed. See above for an example where credentials are present.

The first thing you should do, is a long listing of the /afs/ directory. It will show all AFS cells known to your AFS client. Now change into the directory /afs/openafs.org/software/openafs and do a directory listing. You should see this:

```
        % ls -l
        total 10
        drwxrwxrwx    3 root     root         2048 Jan  7  2003 delta
        drwxr-xr-x    8 100      wheel        2048 Jun 23  2001 v1.0
        drwxr-xr-x    4 100      wheel        2048 Jul 19  2001 v1.1
        drwxrwxr-x   17 100      101          2048 Oct 24 12:36 v1.2
        drwxrwxr-x    4 100      101          2048 Nov 26 21:49 v1.3
```

Go deeper into one of these directories. For example

```
        % cd v1.2/1.2.10/binary/fedora-1.0
```

Have a look at the Access Control Lists (ACLs) in this directory with

```
        % fs listacl .
        Access list for . is
        Normal rights:
          openafs:gatekeepers rlidwka
          system:administrators rlidwka
          system:anyuser rl
```

This shows that two groups have all 7 possible privileges: read (r), lookup (l), insert (i), write (w), full file advisory lock (k), and ACL change right (a). The special group "system:anyuser" that comes with AFS has read (r) and lookup (l) rights which allows access to literally anybody.

To list the members of a group, use the "pts" (Protection Server) command:

```
        % pts member openafs:gatekeepers -cell openafs.org -noauth
        Members of openafs:gatekeepers (id: -207) are:
          shadow
          rees
          zacheiss.admin
          jaltman
```

The "-noauth" flag is used because this command is run without any credentials for this cell.

Special administrative privileges are necessary to explore the authentication part of AFS which is standard Kerberos, so I skip it here.

Now find out where the current directory is physically located:

```
        % fs whereis .
        File . is on hosts andrew.e.kth.se VIRTUE.OPENAFS.ORG
```

This shows that there are two copies of this directory available. One from "andrew.e.kth.se" and one from "VIRTUE.OPENAFS.ORG"

The command

```
        % fs lsmount /afs/openafs.org/software/openafs/v1.2/1.2.10/binary/fedora-1.0
        '/afs/openafs.org/software/openafs/v1.2/1.2.10/binary/fedora-1.0' is a mount point for volume '#openafs.1210.f10'
```

shows that this directory actually is a mount point for an AFS volume named "openafs.1210.f10".

Another AFS command allows to inspect volumes:

```
        % vos examine openafs.1210.f10 -cell openafs.org -noauth
```

This command examines the read-write version of volume "openafs.1210.f10" in AFS cell "openafs.org".

The output should look like this:

```
    openafs.1210.f10                  536871770 RW      25680 K  On-line
        VIRTUE.OPENAFS.ORG /vicepb
        RWrite  536871770 ROnly  536871771 Backup          0
        MaxQuota            0 K
        Creation    Fri Nov 21 17:56:28 2003
        Last Update Fri Nov 21 18:05:30 2003
        0 accesses in the past day (i.e., vnode references)

        RWrite: 536871770     ROnly: 536871771
        number of sites -> 3
           server VIRTUE.OPENAFS.ORG partition /vicepb RW Site
           server VIRTUE.OPENAFS.ORG partition /vicepb RO Site
           server andrew.e.kth.se partition /vicepb RO Site
```

This shows that this volume is hosted on server "VIRTUE.OPENAFS.ORG" in disk partition "/vicepb". The next line shows the numeric volume IDs for the read-write and the read-only volumes. It also shows some statistics. The last three lines show where the one read-write (RW Site) and the two read-only (RO Site) copies of this volume are located.

To find out how many other AFS disk partitions are on server VIRTUE.OPENAFS.ORG use the command

```
% vos listpart VIRTUE.OPENAFS.ORG -noauth
The partitions on the server are:
    /vicepa     /vicepb     /vicepc
Total: 3
```

which shows a total of 3 /vicep partitions.

To see what volumes are located in partition /vicepa on this server, execute

```
% vos listvol VIRTUE.OPENAFS.ORG /vicepa -noauth
```

This takes a while and eventually returns a list of 275 volumes. The first few lines of output look like this:

```
Total number of volumes on server VIRTUE.OPENAFS.ORG partition /vicepa: 275
openafs.10.src                    536870975 RW      11407 K On-line
openafs.10.src.backup             536870977 BK      11407 K On-line
openafs.10.src.readonly           536870976 RO      11407 K On-line
openafs.101.src                   536870972 RW      11442 K On-line
openafs.101.src.backup            536870974 BK      11442 K On-line
openafs.101.src.readonly          536870973 RO      11442 K On-line
```

Another command, "bos", communicates with a cell's Basic OverSeer (BOS) Server and finds out the status of that cell's AFS server processes:

```
% bos status VIRTUE.OPENAFS.ORG -long -noauth
Instance vlserver, (type is simple) currently running normally.
    Process last started at Sun Apr  4 00:00:36 2004 (1 proc starts)
    Command 1 is '/usr/afs/bin/vlserver'

Instance ptserver, (type is simple) currently running normally.
    Process last started at Sun Apr  4 00:00:36 2004 (1 proc starts)
    Command 1 is '/usr/afs/bin/ptserver'

Instance fs, (type is fs) currently running normally.
    Auxiliary status is: file server running.
    Process last started at Sun Apr  4 00:00:36 2004 (2 proc starts)
    Command 1 is '/usr/afs/bin/fileserver'
    Command 2 is '/usr/afs/bin/volserver'
    Command 3 is '/usr/afs/bin/salvager'
```

This reports for each of the different service processes that comprise an AFS server that currently everything is running normally.

Many more subcommands are available for the "fs", "pts", "vos", and "bos" commands. All of these AFS commands understand the "help" flag (no dash in front

of "help") to show all available subcommands. Use "fs <subcommand> -help" (with dash) to look at the syntax for a specific subcommand.

III.6 The Future of AFS

There are several enhancement projects on their way for AFS. The most important right now is to make AFS work with the 2.6 Linux kernels. These kernels don't export their syscall table any more which makes it impossible to create Process Authentication Groups (PAGs) to store credentials in a secure way. NFSv4 is another file system that depends on this feature.

Another project is to provide a "disconnected mode" that allows AFS clients to go off the network, continue to use AFS and once they reconnect the content of files in AFS space is re-synchronized.

III.7 Conclusion

While all the different parts of AFS can be overwhelming at first and the learning curve for setting up your own AFS cell is steep, the reward for using AFS in your infrastructure can be significant. Secure, platform independent world-wide file sharing are as attractive as serving your /usr/local/ area and all your Unix home directories. And all this comes with only minimal longterm administrative costs.

III.8 Further Reading

There are several good sources of information about AFS.

The OpenAFS Wiki at http://grand.central.org/twiki/bin/view/AFSLore/WebHome Especially the "Frequently Asked Questions" section.

There are several OpenAFS mailinglists. Information about how to subscribe and their archives can be found at https://lists.openafs.org/mailman/listinfo/

The only book about AFS, "Managing AFS: The Andrew File System" by Richard Campbell (Pearson Education POD, ISBN: 0138027293) is a little outdated and only describes the IBM/Transarc version of AFS but it shows in great detail how AFS works.