Using the Andrew File System with BSD

H. Meiland

May 4, 2006

Abstract

Since the beginning of networks, one of the basic idea's has been sharing of files; even though with the Internet as advanced as today, simple platform independent file sharing is not common. Why is the closest thing we use WebDAV, a 'neat trick over http', instead of a real protocol?

In this paper the Andrew File System will be described which has been (and is) the file sharing core of many universities and companies world-wide. Also the reason for it's relative unawareness in the community will be answered, and it's actual features and performance in comparison with alternative network filesystems. Finally some information will be given on how to use it with our favorite OS: BSD.

1 History

- 1984 Carnegie Mellon University, first release
- 1989 TransArc Corporation formed by part of original team members
- 1994 TransArc purchased by IBM
- 1997 Start of Arla development at stacken.kth.se
- 2000 IBM releases AFS in opensource (IBM License)
- 2000 http://www.OpenAFS.org
- 2006 good support for lot's of platforms, many new features etc.

2 Overview

2.1 User point of view

2.1.1 Global namespace

While discussing global filesystem, it is easy to dive into a organization, and explain wonderfull features like having replicas of often accessed data in branch-offices, and moving home-directories to local fileservers when moving employees between departments. An essential feature of AFS is often overlooked: a common root as accesspoint of all AFS stored data. Think about it, a simple `cd` will take you from here to the other side of the world, from private storage locations, to commercial companies to universities; all accessed through the same `/afs` mountpoint.
global path to files, e.g.

path	server location
/afs/andrew.cmu.edu	# Pittsburgh, USA
/afs/cern.ch	# Geneva, Switserland
/afs/gorlaeus.net	# Leiden, The Netherlands
/afs/openafs.org	# USA + Sweden

2.1.2 Getting access

As mentioned earlier, the security system of AFS is based on Kerberos; this means the first step to take when accessing files is to make sure you have the right credentials: `#>kinit <username>`, whether you have obtained kerberos tickets can be checked using `#>klist`.
The next step is not obvious, due to the early integration with kerberos, a path was chosen to integrate the actual authentication code in the kernel; hence the kerberos ticket has to be converted to a AFS token, and stashed into the kernel using `#>aklog` or `#>afslog` or `#>afs5log` or whatever toolname is current (check with your local sysadmin). To check whether this part has succeeded, use `#>tokens`, this will show you your current tokens.

client	direction	kerberos	afs
`#>kinit username@realm`	to kerberos		
	to client	krbtgt/realm@realm	
`#>afslog <cell>`	to kerberos		
	to client	afs/cell@realm	
krbtgt+afs/cell=token	to client		
`#>ls /afs/<cell>`+token	to afs		
	to client		files and dirs

2.1.3 ACL and quota

Like any grown up filesystem, AFS supports access control lists(ACL). These ACL's work on directory level, and allows the owner of a directory (or any user with administrative priviliges on that directory) to define groups and set permissions. The system administrator can set a quota on a volume, which can give a warning once the treshhold nearby.

2.1.4 Supported platforms

To be able to run a distributed filesyste among large group of users, it is very important to support a number of operating systems; fortunatly the current list of available clients is long enough to be of real intrest: *BSD, Linux, Microsoft Windows XP, Apple Mac OS X.

2.2 Admin point of view

2.2.1 Cell, Partitions and Volumes

For administrators it will be a learning curve to work with AFS. The view of the user is just that: an interpretation which is made available to the user, and which can even be completely different to an other group of users.
The underlaying architecture consists of the following elements: the administrative unit is called a cell, which is compareble to a dns domain. In these cells, the hardware consists of servers. Each server has partitions (mounted in the filesystem as `/vicepa` etcetera. On these partitions volumes can be stored. These volumes are the elements which can be used to build a view for the user. The same way as physical filesystems are mounted under a `/`, volumes can be mounted in the root of a cell: `/afs/<your_cell>/<volume>`.

2.2.2 Integration with Kerberos

Due to the early integration with Kerberos is the first versions, there are some issues and limitations grown due to backwards compatibility. AFS has been one of the first services with an integrated Kerberos server; but since the overall acceptance of Kerberos as a generic authentication service, this integrated server is dismissed in favor of the MIT or Heimdal versions of Kerberos. Although both versions work, some care has to be taken while creating the initial KeyFiles on the first AFS server in a cell. The biggest limitation is the encryption of the afs tickets; they can only be used when encrypted using (single) DES.

To create the KeyFile for your first afs server, the procedure for MIT or Heimdal are different. While the ktutil of Heimdal can easaly transform a

(Kerberos) keytab file to a (AFS) KeyFile[1]:

```
#>kadmin -p <username>/admin
kadmin> add --random-key afs/<cell>
kadmin> ext_keytab -k /tmp/afsv5key afs/<cell>
#>echo <cell> > /usr/afs/etc/ThisCell
#>ktutil copy /tmp/afsv5key AFSKEYFILE:/tmp/KeyFile
```

To create a KeyFile out of a MIT keytab, a utility called `asetkey` is required:[2]:

```
#>kadmin.local -q "ank -randkey afs"
#>kadmin.local -q "ktadd -e des-cbc-crc:afs3 -k /etc/krb5.keytab.afs afs"
#>asetkey add 4 /etc/krb5.keytab.afs afs
```

Because of the use of Heimdal Kerberos on most *BSD platforms, it might be required to do some tricks to create the correct KeyFile when using MIT Kerberos. To keep it all comprehensable you might like to install a seperate workstation based on MIT Kerberos just to create the KeyFile, and transfer it to the AFS servers using e.g. scp.

2.2.3 File locking

When rolling out AFS, be carefull of which applications are going to use this. AFS is not designed to support databases, since they make heavy use of byte-range-locking while AFS only supports file-locking. This is also something that can come up when users try to share their Microsoft Office documents, since the Microsoft applications tend to use byte-range-locking when users try to work on the same document simultaniously.

3 Alternatives

While looking at alternative distributed file systems, it is almost impossible to find one which implements enough functionalities to really compare with AFS. With the features defined in chapter 2 we could see if any other systems can compare.

3.1 Coda

Although Coda development started at CMU out of AFS2, it remains a research project, and not ready for production usage: *"I'd say a small user-base (20-30 users) and a few servers are pretty workable. (...) Don't expect*

[1] http://www.public.iastate.edu/ kula/talks/afs-bpw-2005/afs-bpw-2005-iowa.html
[2] http://www.seismo.ethz.ch/linux/afs/node6.html

to easily handle terabytes of data or a large group of non-technical oriented users."[3]

3.2 WebDAV

HTTP Extensions for Distributed Authoring – WEBDAV [4]

3.3 Microsoft DFS Namespace

The Microsoft DFS namespace (which used to be called Distributed File System) is an expansion to Windows file sharing services. It allows users to view the shared folders in a virtual tree, and connect to the closest available server. 3rd Party software for Mac OS X available[5].

3.4 Network File System version 4

The follow-up of the immense popular NFSv3; Main changes: Kerberos based authentication, client side caching, ACL support (optional), removal of portmapper, runs over TCP.

3.5 Other projects

- Server Message Block (SMB); not feasable due to firewall constraints
- Intermezzo; Project seems dead since june 2002[6]
- WebNFS; nfs without a portmapper, overrun by nfsv4

3.6 Comparison

feature	afs	coda	nfs.v4	smb/dfs
namespace	global		per server	per 'cluster'
acl	directory		file	file
wan				

4 Implementations: OpenAFS and Arla

The original source form TransArc is now maintaned by the OpenAFS developers, a group of developers mainly from the old customer base of TransArc/IBM who have years of experience in running afs cells. While the code is open source, the license of the code has still parts of the IBM open

[3] http://www.coda.cs.cmu.edu/misc/stability.html
[4] RFC 2518, Standards Track, Proposed Standard. February, 1999
[5] http://www.thursby.com/products/dave.html
[6] http://www.inter-mezzo.org/index.html

source license on it, and thus can not be intergrated into BSD's. During the commercial fase of afs, Arla was started as a project on the Royal Institute of Technology, Sweden (KTH) in 1993. In the next few years, the project slowed down a bit until 1997, when an rxkad implementation was written by Bjrn Grnvall and the project was put into a CVS server. The main focus is on free OS'ses such as Open- Free- and NetBSD and Linux.

5 Status OpenBSD

The OpenBSD team has supported AFS by integrating the Arla code since release 3.4 (Nov 1, 2003). Currently Arla 0.35.7 is supplied with OpenBSD 3.9 Release (Released May 1, 2006). To get it running, only the folowing trivial steps need to be taken:

```
#>echo afs=YES >> rc.conf.local
#>mkdir /afs
#>reboot
```

The 0.42 release from Arla has been tested on OpenBSD 3.8, to run correctly in 3.9 you will have to check out -current form cvs: installing arla-current from cvs:

```
> cp src.tar.gz sys.tzr.gz /usr/src
# cd /usr/src; tar zxvf src.tar.gz; tar zxvf sys.tar.gz
perform some kernel build magic up to make depend
> env CVS_RSH=ssh cvs -d anoncvs@anoncvs.stacken.kth.se:/stacken-cvs \
  checkout arla
# pkg_add autoconf-2.59.tgz ; pkg_add automake-1.9.6p0.tgz
# pkg_add libtool-1.5.18p2.tgz
> AUTOCONF_VERSION=2.59 AUTOMAKE_VERSION=1.9 autoreconf -f -i
> ./configure && make
# make install && mkdir /afs
# echo "/usr/arla/sbin/startarla" >> /etc/rc.securelevel
# reboot
```

The stable release of OpenAFS, version 1.4.0 will also compile and run in OpenBSD, Ober has created a port [7] and the following steps should get you running [8]:

```
# cd /usr/ports/net && tar zxvf openafs.tgz
# make && make install
# mkdir -p /etc/openafs ; mkdir -p /usr/vice/cache
```

[7] *http://www.linbsd.org/openafs.tgz*

[8] from *http://www.linbsd.org/afs_on_openbsd_client.html*

```
# cp ThisCell CellServDB /etc/openafs
# echo "/sbin/modload usr/local/lib/openafs/libafs.o" >> /etc/rc.securelevel
# echo "/usr/local/sbin/afsd -stat 4000 -dcache 4000 -daemons 6 \
  -volumes 256 -files 50000" >> /etc/rc.local
# mkdir -p -m 0755 /afs
# echo "/afs:/usr/vice/cache:198112" > /usr/vice/etc/cacheinfo
# reboot
```

6 Status FreeBSD

On versions in the Legacy release (5.x), Arla works out of the box. The plain configure, make, make install, startarla should give you access to all open AFS cells. Unfortunatly the 6.x release has seen some major changes in the VFS interface which have not been completely updated in the Arla code. At this time Arla will thus not function on the newer FreeBSD releases. Work is very much in progress to update to the new VFS interface.

7 Status NetBSD

- arla from ports
arla 0.42 running
- configure, make, make install, startarla

8 Future Work

The main focus points of Arla-on-BSD development seems to be the update to the FreeBSD VFS interface. It would be nice to see a update of the arla code in OpenBSD to a recent version, which would allow easier development on both sides. On the OpenAFS side, a working port of the client part hes been created for the OpenBSD platform. Once this is working, it should be feasable to bring this also the the other BSD's. Even though this won't be as easy as Arla development due to lisencing constraints.