MountableHDFS

Mounting HDFS

```
[machine1] ~ > df -kh /export/hdfs/
Filesystem            Size  Used Avail Use% Mounted on
fuse                  4.1P  642T  3.5P  21% /export/hdfs
[machine1] ~ > ls /export/hdfs/
home  tmp  Trash  user  usr  var
```

These projects (enumerated below) allow HDFS to be mounted (on most flavors of Unix) as a standard file system using the mount command. Once mounted, the user can operate on an instance of hdfs using standard Unix utilities such as 'ls', 'cd', 'cp', 'mkdir', 'find', 'grep', or use standard Posix libraries like open, write, read, close from C, C++, Python, Ruby, Perl, Java, bash, etc.

All, except HDFS NFS Proxy, are based on the Filesystem in Userspace project FUSE (http://fuse.sourceforge.net/). Although the Webdav-based one can be used with other webdav tools, but requires FUSE to actually mount.

Note that a great thing about FUSE is you can export a fuse mount using NFS, so you can use fuse-dfs to mount hdfs on one machine and then export that using NFS. The bad news is that fuse relies on the kernel's inode cache since fuse is path-based and not inode-based. If an inode is flushed from the kernel cache on the server, NFS clients get hosed; they try doing a read or an open with an inode the server doesn't have a mapping for and thus NFS chokes. So, while the NFS route gets you started quickly, for production it is more robust to automount fuse on all the machines you want access to hdfs from.

Also, since NFS often reorders writes, you can have write failures when you use the NFS -> FUSE -> HDFS route. HDFS requires that writes be sequential.

Projects

- contrib/fuse-dfs is built on fuse, some C glue, libhdfs and the hadoop-dev.jar
- fuse-j-hdfs is built on fuse, fuse for java, and the hadoop-dev.jar
- hdfs-fuse - a google code project is very similar to contrib/fuse-dfs
- webdav - hdfs exposed as a webdav resource
- mapR - contains a closed source hdfs compatible file system that supports read/write NFS access
- HDFS NFS Proxy - exports HDFS as NFS without use of fuse. Supports Kerberos and re-orders writes so they are written to hdfs sequentially.
- native-hdfs-fuse - a FUSE implementation in C that supports random writes

While not complete filesystem implementations (FUSE or otherwise), the following projects could be useful when building one:

- hadoofus - an implementation of the libhdfs API in C for hadoop 0.20.203 to 1.0.3

Supported Operating Systems

Linux 2.4, 2.6, FreeBSD, NetBSD, MacOS X, Windows, Open Solaris. See: http://fuse.sourceforge.net/wiki/index.php/OperatingSystems

Fuse-DFS

Supports reads, writes, and directory operations (e.g., cp, ls, more, cat, find, less, rm, mkdir, mv, rmdir). Things like touch, chmod, chown, and permissions are in the works. Fuse-dfs currently shows all files as owned by nobody.

Contributing

It's pretty straightforward to add functionality to fuse-dfs as fuse makes things relatively simple. Some other tasks require also augmenting libhdfs to expose more hdfs functionality to C. See contrib/fuse-dfs JIRAs

Requirements

- Hadoop with compiled libhdfs.so
- Linux kernel > 2.6.9 with fuse, which is the default or Fuse 2.7.x, 2.8.x installed. See: http://fuse.sourceforge.net/ or even easier: http://dag.wieers.com/rpm/packages/fuse/
- modprobe fuse to load it
- fuse-dfs executable (see below)
- fuse_dfs_wrapper.sh installed in /bin or other appropriate location (see below)

BUILDING

(for 0.23 and up see here)

1. in HADOOP_HOME: `ant compile-c++-libhdfs -Dlibhdfs=1` (EDIT: March15 2010 I had to add -Dislibhdfs=1 to get this to work) 2. in HADOOP_HOME: `ant package` to deploy libhdfs` 3. in HADOOP_HOME: `ant compile-contrib -Dlibhdfs=1 -Dfusedfs=1`

NOTE: for amd64 architecture, libhdfs will not compile unless you edit the Makefile in src/c++/libhdfs/Makefile and set OS_ARCH=amd64 (probably the same for others too). See HADOOP-3344

Common build problems include not finding the libjvm.so in JAVA_HOME/jre/lib/OS_ARCH/server or not finding fuse in FUSE_HOME or /usr/local.

CONFIGURING

Look at all the paths in fuse_dfs_wrapper.sh and either correct them or set them in your environment before running. (note for automount and mount as root, you probably cannot control the environment, so best to set them in the wrapper)

INSTALLING

1. `mkdir /export/hdfs` (or wherever you want to mount it)

2. `fuse_dfs_wrapper.sh dfs://hadoop_server1.foo.com:9000 /export/hdfs -d` and from another terminal, try `ls /export/hdfs`

If 2 works, try again dropping the debug mode, i.e., -d

(note - common problems are that you don't have libhdfs.so or libjvm.so or libfuse.so on your LD_LIBRARY_PATH, and your CLASSPATH does not contain hadoop and other required jars.)

Also note, fuse-dfs will write error/warn messages to the syslog - typically in /var/log/messages

You can use fuse-dfs to mount multiple hdfs instances by just changing the server/port name and directory mount point above.

DEPLOYING

in a root shell do the following:

1. add the following to /etc/fstab

```
fuse_dfs#dfs://hadoop_server.foo.com:9000 /export/hdfs fuse -oallow_other,rw,-ousetrash 0 0
```

2. Mount using: `mount /export/hdfs`. Expect problems with not finding fuse_dfs. You will need to probably add this to /sbin and then problems finding the above 3 libraries. Add these using ldconfig.

Fuse DFS takes the following mount options (i.e., on the command line or the comma separated list of options in /etc/fstab (in which case, drop the -o prefixes):

```
-oserver=%s  (optional place to specify the server but in fstab use the format above)
-oport=%d (optional port see comment on server option)
-oentry_timeout=%d (how long directory entries are cached by fuse in seconds - see fuse docs)
-oattribute_timeout=%d (how long attributes are cached by fuse in seconds - see fuse docs)
-oprotected=%s (a colon separated list of directories that fuse-dfs should not allow to be deleted or moved - e.
g., /user:/tmp)
-oprivate (not often used but means only the person who does the mount can use the filesystem - aka !
allow_others in fuse speak)
-ordbuffer=%d (in KBs how large a buffer should fuse-dfs use when doing hdfs reads)
ro
rw
-ousetrash (should fuse dfs throw things in /Trash when deleting them)
-onotrash (opposite of usetrash)
-odebug (do not daemonize - aka -d in fuse speak)
-obig_writes (use fuse big_writes option so as to allow better performance of writes on kernels >= 2.6.26)
```

The defaults are:

```
entry,attribute_timeouts = 60 seconds
rdbuffer = 10 MB
protected = null
debug = 0
notrash
private = 0
```

EXPORTING

Add the following to /etc/exports:

```
/export/hdfs *.foo.com(no_root_squash,rw,fsid=1,sync)
```

NOTE - you cannot export this with a FUSE module built into the kernel

- e.g., kernel 2.6.17. For info on this, refer to the FUSE wiki.

RECOMMENDATIONS

1. From /bin, `ln -s $HADOOP_HOME/contrib/fuse-dfs/fuse_dfs* .`

2. Always start with debug on so you can see if you are missing a classpath or something like that.

3. use -obig_writes

KNOWN ISSUES

1. if you alias `ls` to `ls --color=auto` and try listing a directory with lots (over thousands) of files, expect it to be slow and at 10s of thousands, expect it to be very very slow. This is because `--color=auto` causes ls to stat every file in the directory. Since fuse-dfs does not cache attribute entries when doing a readdir,

this is very slow. see HADOOP-3797

2. Writes are approximately 33% slower than the DFSClient. TBD how to optimize this. see: HADOOP-3805 - try using -obig_writes if on a >2.6.26 kernel, should perform much better since bigger writes implies less context switching.

3. Reads are ~20-30% slower even with the read buffering.

Fuse-j-HDFS

see HADOOP-4 and here is the code fuse-j-hadoopfs

HDFS-FUSE

see http://code.google.com/p/hdfs-fuse/

Webdav

Boris Musykantski wrote (on hadoop-core-user@)

Some time ago there was a discussion on this list about exposing HDFS through webdav. The almost complete code is in https://issues.apache.org/jira/browse/HADOOP-496. We re-touched it a bit, updated it to work with 0.16 and 0.17 and put some support for authentications and permissions.

The result is here : http://www.hadoop.iponweb.net/ together with installation instructions and a small "demo" server to allow you to check compatibility of the server with different clients. (scroll all the way down if you are not interested in our thoughts on hadoop framework in general)

We tested this with WinXP and Win2003 server and plan to use this in production, so, quite naturally we are very interested in any feedback. Also I'd want to thank the authors of the HADOOP-496 for the terrific work.

HDFS NFS Proxy

Exports the HDFS system as NFS. Written entirely in java and uses the HDFS java API directly.

https://github.com/brockn/hdfs-nfs-proxy

Hadoofus

https://github.com/cemeyer/hadoofus

The hadoofus project is an HDFS (Hadoop Distributed File System) client library. It is implemented in C and supports RPC pipelining and out-of-order execution.

It provides a C API for directly calling Namenode RPCs and performing Datanode block read and write operations, as well as a libhdfs-compatible interface (libhdfs_hadoofus.so).

It also includes a Python wrapper module, implemented in Cython.

Note: This library currently supports the HDFS protocol as spoken by Apache Hadoop releases 0.20.203 through 1.0.3.

native-hdfs-fuse

https://github.com/remis-thoughts/native-hdfs-fuse

Unlike most other FUSE HDFS implementations this implementation doesn't use libhdfs or otherwise start a JVM - it constructs and sends the protocol buffer messages itself. The implementation supports random file writes too.