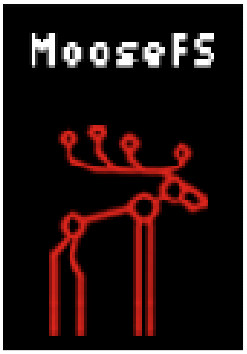
Installing MooseFS
Step by Step Tutorial



Michał Borychowski
MooseFS Support Manager
contact@moosefs.org

march 2010
Gemius SA

Overview

Below we present a step-by-step installation process of MooseFS system on Linux platform. We assume that the system will be run by a user `mfs` which belongs to `mfs` group. We will use FHS (Filesystem Hierarchy Standard) compliant paths, and source archive `mfs-1.6.15.tar.gz` would be placed in `/usr/src`. We'll show how to install the system on separate dedicated machines and how to make a test install on one server.

The latest stable release of MooseFS can be downloaded from http://sourceforge.net/projects/moosefs/, and on user computers you would need FUSE package which can be downloaded from http://sourceforge.net/projects/fuse/.

MooseFS install process on dedicated machines

We assume that our machines have following IP addresses:

- Master server: 192.168.1.1
- Metalogger server: 192.168.1.2
- Chunk servers: 192.168.1.101 and 192.168.1.102
- Users' computers (clients): 192.168.2.x

Master server installation

When installing the master server we disable in `./configure` installation of a chunk server (`--disable-mfschunkserver`) and of a client (`--disable-mfsmount`). We do the following steps:

```
#groupadd mfs
#useradd -g mfs mfs
#cd /usr/src
#tar -zxvf mfs-1.6.15.tar.gz
#cd mfs-1.6.15
#./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var/lib --with-default-user=mfs
--with-default-group=mfs --disable-mfschunkserver --disable-mfsmount
#make
#make install
```

Sample configuration files will be created in `/etc` with the extension `.dist`. We'll use these files as our target configuration files:

```
#cd /etc
#cp mfsmaster.cfg.dist mfsmaster.cfg
#cp mfsmetalogger.cfg.dist mfsmetalogger.cfg
#cp mfsexports.cfg.dist mfsexports.cfg
```

If we would like to change any of the settings we should uncomment the given line and write a different value. For the lines which are commented out the system will use built-in default values.

File `mfsmaster.cfg` contains master server settings, here we leave it unchanged. You can find out more information about this file in the man pages (`man mfsmaster.cfg`).

File `mfsexports.cfg` specifies which users' computers can mount the file system and with what privileges. For our example we'll specify that only machines addressed as 192.168.2.x can use the whole structure of MooseFS resources (`/`) in read/write mode. In the first line which is not commented out we change an asterisk (`*`) to `192.168.2.0/24` so that we have:

```
192.168.2.0/24            /     rw,alldirs,maproot=0
```

Binary metadata file and changelog text files are kept in a folder set during compilation as `localstatedir` – in our example the folder is: `/var/lib/mfs`. At the first installation an empty metadata file is created with a name of `metadata.mfs.empty`, which we change to `metadata.mfs`:

```
#cd /var/lib/mfs
#cp  metadata.mfs.empty metadata.mfs
```

We alse need to specify in `/etc/hosts` that name `mfsmaster` refers to the machine of 192.168.1.1 address:

```
192.168.1.1    mfsmaster
```

At this moment it is possible to run the master server (server would be run as a user given in the install configuration, in our case it is `mfs`):

```
#/usr/sbin/mfsmaster start
```

In a production environment it would be necessary to set up automatic start of `mfsmaster` process at system start.

We can now also run CGI monitor which shows current system status in a browser:

```
#/usr/sbin/mfscgiserv
```

Information should now be available under http://192.168.1.1:9425/ (for the moment there would be no data about chunk servers).

Backup server (metalogger) installation

Machine used to install the backup server should be as much strong as the master server (at least with amount of RAM). In case of the master server failure, after importing changelogs to the metadata file, the metalogger server would be able to take over functions of the managing server (more about this can be read at http://www.moosefs.org/mini-howtos.html#redundant-master).

Metalogger installation is very similar to that of the master server. We issue the following commands:

```
#groupadd mfs
#useradd -g mfs mfs
#cd /usr/src
#tar -zxvf mfs-1.6.15.tar.gz
#cd mfs-1.6.15
```

```
#./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var/lib --with-default-user=mfs
--with-default-group=mfs --disable-mfschunkserver --disable-mfsmount
#make
#make install

#cd /etc
#cp mfsmetalogger.cfg.dist mfsmetalogger.cfg
```

Similarly, in `/etc/hosts` we add:

```
192.168.1.1   mfsmaster
```

Now we are ready to start the backup server process:

```
#/usr/sbin/mfsmetalogger start
```

In a production environment you should set up automatic start of `mfsmetalogger`.

Chunk servers installation

We issue the following commands on the machines which are to be chunks servers:

```
#groupadd mfs
#useradd -g mfs mfs
#cd /usr/src
#tar -zxvf mfs-1.6.15.tar.gz
#cd mfs-1.6.15
#./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var/lib --with-default-user=mfs
--with-default-group=mfs --disable-mfsmaster
#make
#make install
```

Now we similarly prepare configuration files of the chunk server:

```
#cd /etc/
#cp  mfschunkserver.cfg.dist mfschunkserver.cfg
#cp  mfshdd.cfg.dist mfshdd.cfg
```

For our test installation we leave `mfschunkserver.cfg` unchanged; You can find out more information about this file in the man pages (`man mfschunkserver.cfg`).

In a `mfshdd.cfg` file we give locations in which we have mounted hard drives / partitions purposed for the chunks of the system. It is recommended that they are used exclusively for the MooseFS - this is necessary to manage the free space properly. Let's assume we'll use `/mnt/mfschunks1` and `/mnt/mfschunks2` locations, so we add these two lines to `mfshdd.cfg` file:

```
/mnt/mfschunks1
/mnt/mfschunks2
```

Before we start chunks server we have to make sure that the user `mfs` has rights to write in the mounted partitions (which is necessary to create a `.lock` file):

```
#chown -R mfs:mfs /mnt/mfschunks1
#chown -R mfs:mfs /mnt/mfschunks2
```

Similarly we add the following line in `/etc/hosts` file:

```
192.168.1.1    mfsmaster
```

Now we are ready to start the chunk server:

```
#/usr/sbin/mfschunkserver start
```

We repeat the same steps for each chunk server we want to use for data storing in MooseFS system.

Now at http://192.168.1.1:9425/ is available full information about the system, including the master server and chunk servers.

Users' computers installation

In order to mount a file system based on MooseFS it is necessary that users' computers have FUSE package (at least in version 2.6, recommended >=2.7.2). If it is not present, you have to install it. One of the options is to compile it from sources – you can get them from http://sourceforge.net/projects/fuse/:

```
#cd /usr/src
#tar -zxvf fuse-2.8.3.tar.gz
#cd fuse-2.8.3
#./configure
#make
#make install
```

In order to install `mfsmount` package we do the following steps:

```
#tar -zxvf mfs-1.6.15.tar.gz
#cd mfs-1.6.15
#./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var/lib --with-default-user=mfs
--with-default-group=mfs --disable-mfsmaster
--disable-mfschunkserver
#make
#make install
```

In a `/etc/hosts` file we add this line:

```
192.168.1.1    mfsmaster
```

Let's assume that we will mount the system in a `/mnt/mfs` folder on a client's machine. We issue the following commands:

```
#mkdir -p /mnt/mfs
#/usr/bin/mfsmount /mnt/mfs -H mfsmaster
```

Now after issuing the `df -h | grep mfs` command we should get information similar to this:

```
/storage/mfschunks/mfschunks1
                      2.0G   69M  1.9G   4% /mnt/mfschunks1
/storage/mfschunks/mfschunks2
                      2.0G   69M  1.9G   4% /mnt/mfschunks2
mfs#mfsmaster:9421    3.2G     0  3.2G   0% /mnt/mfs
```

Installing MooseFS on one server

If you want to make a test installation of the system on one machine, you should follow the following steps. We won't install metalogger process here and we assume that the server has 192.168.1.1 address.

In order to mount a file system based on MooseFS it is necessary that users' computers have FUSE package (at least in version 2.6, recommended >=2.7.2). If it is not present, you have to install it. One of the options is to compile it from sources – you can get them from http://sourceforge.net/projects/fuse/:

```
#cd /usr/src
#tar -zxvf fuse-2.8.3.tar.gz
#cd fuse-2.8.3
#./configure
#make
#make install
```

Now we start installation of MooseFS:

```
#groupadd mfs
#useradd -g mfs mfs
#cd /usr/src
#tar -zxvf mfs-1.6.15.tar.gz
#cd mfs-1.6.15
#./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var/lib --with-default-user=mfs
--with-default-group=mfs
#make
#make install
```

It is important to dedicate file systems used by MooseFS chunks exclusively to it - this is necessary to manage the free space properly. MooseFS does not take into account that a free space accessible to it could be taken by other data. If it's not possible to create a separate disk partition, filesystems in files can be used. For needs of this test install we will prepare two 2GB files (located in `/storage/mfschunks`), we'll format them as `ext3` and mount as `/mnt/mfschunks1` and `/mnt/mfschunks2`.

```
#mkdir -p /storage/mfschunks

#dd if=/dev/zero of=/storage/mfschunks/mfschunks1 bs=1024 count=1
seek=$((2*1024*1024-1))
#mkfs -t ext3 /storage/mfschunks/mfschunks1
#mkdir -p /mnt/mfschunks1
#mount -t ext3 -o loop /storage/mfschunks/mfschunks1 /mnt/mfschunks1

#dd if=/dev/zero of=/storage/mfschunks/mfschunks2 bs=1024 count=1
seek=$((2*1024*1024-1))
#mkfs -t ext3 /storage/mfschunks/mfschunks2
#mkdir -p /mnt/mfschunks2
#mount -t ext3 -o loop /storage/mfschunks/mfschunks2 /mnt/mfschunks2
```

Before we start chunks server we have to make sure that the user `mfs` has rights to write in the mounted partitions (which is necessary to create a `.lock` file):

```
#chown –R mfs:mfs /mnt/mfschunks1
#chown –R mfs:mfs /mnt/mfschunks2
```

Sample configuration files will be created in `/etc` with the extension `.dist`. We'll use these files as our target configuration files:

```
#cd /etc
#cp mfsexports.cfg.dist mfsexports.cfg
#cp mfsmaster.cfg.dist mfsmaster.cfg
#cp mfschunkserver.cfg.dist mfschunkserver.cfg
#cp mfshdd.cfg.dist mfshdd.cfg
```

Files `mfsexports.cfg` and `mfsmaster.cfg` refer to master server settings and files `mfschunkserver.cfg` and `mfshdd.cfg` to chunk server.

File `mfsexports.cfg` specifies which users' computers can mount the file system and with what privileges. For our example we'll specify that only machines addressed as 192.168.1.x can use the whole structure of MooseFS resources (`/`) in read/write mode. In the first line which is not commented out we change an asterisk (`*`) to `192.168.1.0/24` so that we have:

```
192.168.1.0/24            /    rw,alldirs,maproot=0
```

In a `mfshdd.cfg` file we give locations of our filesystems in files (if you have independent partitions / hard drives, write them here):

```
/mnt/mfschunks1
/mnt/mfschunks2
```

For our example we leave other options of `mfsmaster.cfg` and `mfschunkserver.cfg` unchanged.

Binary metadata file and changelog text files are kept in a folder set during compilation as `localstatedir` – in our example the folder is: `/var/lib/mfs`. At the first installation an empty metadata file is created with a name of `metadata.mfs.empty`, which we change to `metadata.mfs`:

```
#cd /var/lib/mfs
#cp  metadata.mfs.empty metadata.mfs
```

We also add to `/etc/hosts` the following line:

```
192.168.1.1   mfsmaster
```

At this moment it is possible to run processes of master server, CGI monitor and chunk server:

```
#/usr/sbin/mfsmaster start
#/usr/sbin/mfscgiserv
#/usr/sbin/mfschunkserver start
```

Information about the current state of the system will be available under http://192.168.1.1:9425/.

Now we'll mount the system as `/mnt/mfs` folder:

```
#mkdir -p /mnt/mfs
#/usr/bin/mfsmount /mnt/mfs -H mfsmaster
```

After issuing the `df -h | grep mfs` command we should get information similar to this:

```
/storage/mfschunks/mfschunks1
                      2.0G   69M  1.9G   4% /mnt/mfschunks1
/storage/mfschunks/mfschunks2
                      2.0G   69M  1.9G   4% /mnt/mfschunks2
mfs#mfsmaster:9421    3.2G     0  3.2G   0% /mnt/mfs
```

Basic MooseFS Usage

Let's create `folder1` in `/mnt/mfs`, in which we would store files in one copy (setting `goal=1`):

```
mkdir -p /mnt/mfs/folder1
```

and `folder2`, in which we would store files in one copy (setting `goal=2`):

```
mkdir -p /mnt/mfs/folder2
```

The number of copies for the folder is set with the `mfssetgoal -r` command:

```
#mfssetgoal -r 1 /mnt/mfs/folder1
/mnt/mfs/folder1:
 inodes with goal changed:                       0
 inodes with goal not changed:                   1
 inodes with permission denied:                  0

#mfssetgoal -r 2 /mnt/mfs/folder2
/mnt/mfs/folder2:
 inodes with goal changed:                       0
 inodes with goal not changed:                   1
 inodes with permission denied:                  0
```

Let's copy now a file to both folder:

```
cp /usr/src/mfs-1.6.15.tar.gz /mnt/mfs/folder1
cp /usr/src/mfs-1.6.15.tar.gz /mnt/mfs/folder2
```

The command `mfscheckfile` is used for checking in how many copies the given file is stored. For the `folder1` we have one copy stored in one chunk:

```
#mfscheckfile /mnt/mfs/folder1/mfs-1.6.15.tar.gz
/mnt/mfs/folder1/mfs-1.6.15.tar.gz:
1 copies: 1 chunks
```

And in the `folder2` the file `mfs-1.6.15.tar.gz` is saved in two copies:

```
#mfscheckfile /mnt/mfs/folder2/mfs-1.6.15.tar.gz
/mnt/mfs/folder2/mfs-1.6.15.tar.gz:
2 copies: 1 chunks
```

Additional information. When all the processes are installed on one server you would see that even for the `goal=2` the files are saved in just one copy – this is a proper behavior because in spite of two disks we have one chunk server.

You can read more information about usage and commands of MooseFS on this page:
http://www.moosefs.org/reference-guide.html#using-moosefs

We also recommend to read the FAQ page:
http://www.moosefs.org/moosefs-faq.html

Stopping MooseFS

In order to safely stop the MooseFS cluster you have to do the following steps:

- Unmount the file system on all machines using `umount` command (in our examples it would be: `umount /mnt/mfs`)
- Stop the chunk server processes: `/usr/sbin/mfschunkserver stop`
- Stop the metalogger process: `/usr/sbin/mfsmetalogger stop`
- Stop the master server process: `/usr/sbin/mfsmaster stop`