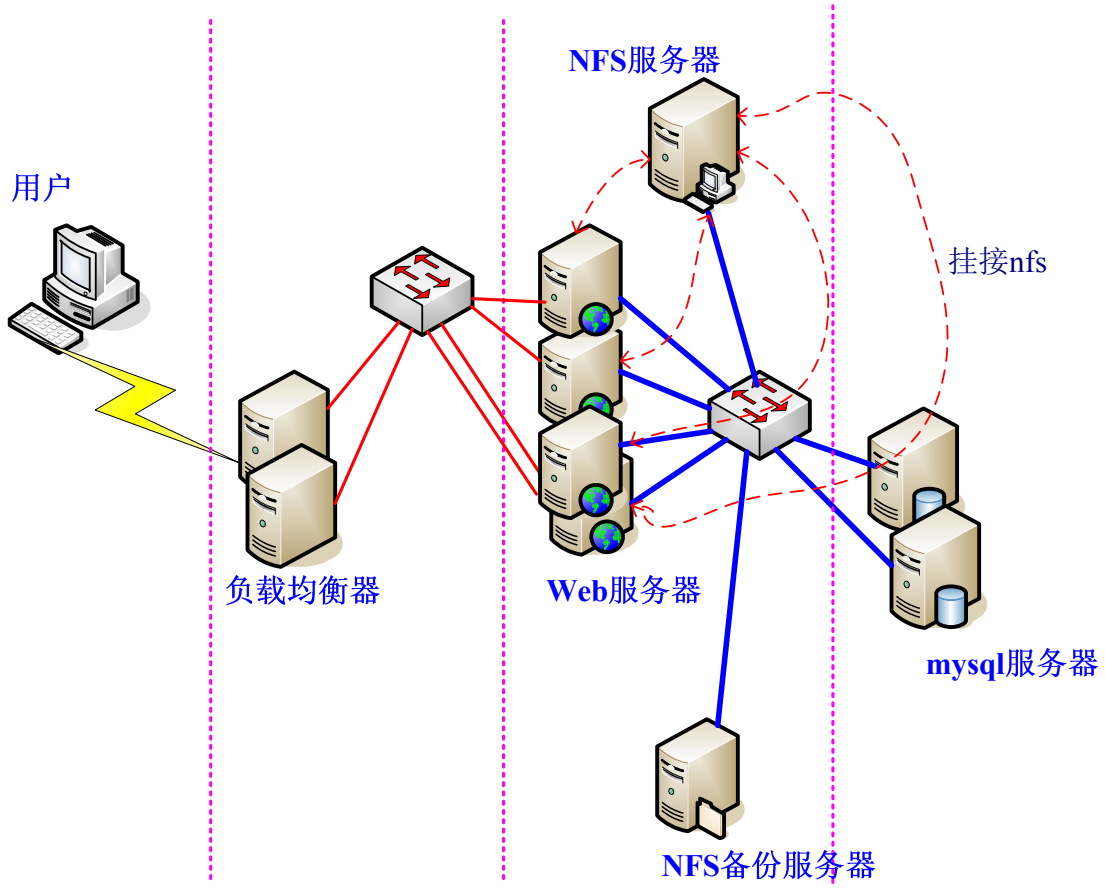
分布式文件系统 MFS(moosefs)实现存储共享(第二版)

作者:田逸(sery@163.com)

由于用户数量的不断攀升,我对访问量大的应用实现了可扩展、高可靠的集群部署（即 lvs+keepalived 的方式），但仍然有用户反馈访问慢的问题。通过排查个服务器的情况，发现问题的根源在于共享存储服务器 NFS。在我这个网络环境里，N 个服务器通过 nfs 方式共享一个服务器的存储空间，使得 NFS 服务器不堪重负。察看系统日志，全是 nfs 服务超时之类的报错。一般情况下，当 nfs 客户端数目较小的时候，NFS 性能不会出现问题；一旦 NFS 服务器数目过多，并且是那种读写都比较频繁的操作，所得到的结果就不是我们所期待的。下面是某个集群使用 nfs 共享的示意图：



这种架构除了性能问题而外，还存在单点故障，一旦这个 NFS 服务器发生故障，所有靠共享提供数据的应用就不再可用，尽管用 rsync 方式同步数据到另外一个服务器上做 nfs 服务的备份，但这对提高整个系统的性能毫无帮助。基于这样一种需求，我们需要对 nfs 服务器进行优化或采取别的解决方案，然而优化并不能对应对日益增多的客户端的性能要求，因此唯一的选择只能是采取别的解决方案了;通过调研,分布式文件系统是一个比较合适的选择。采用分布式文件系统后，服务器之间的数据访问不再是一对多的关系（1 个 NFS 服务器，多个 NFS 客户端），而是多对多的关系，这样一来，性能大幅提升毫无问题。

到目前为止,有数十种以上的分布式文件系统解决方案可供选择,如 lustre,hadoop,Pnfs 等等。我尝试了 PVFS,hadoop,moosefs 这三种应用，参看了 lustre、KFS 等诸多技术实施方法，最后我选择了 moosefs（以下简称 MFS）这种分布式文件系统来作为我的共享存储服务器。为什么要选它呢？我来说说我的一些看法：

1、 实施起来简单。MFS 的安装、部署、配置相对于其他几种工具来说，要简单和容易得多。看看 lustre 700 多页的 pdf 文档，让人头昏吧。
2、 不停服务扩容。MFS 框架做好后，随时增加服务器扩充容量；扩充和减少容量皆不会影响现有的服务。注：hadoop 也实现了这个功能。
3、 恢复服务容易。除了 MFS 本身具备高可用特性外，手动恢复服务也是非常快捷的，原因参照第 1 条。
4、 我在实验过程中得到作者的帮助，这让我很是感激。

MFS 特性（根据官方网站翻译）

★ 高可靠性（数据能被分成几个副本存储在不同的计算机里）

```
[root@mysql-bk serydir]# mfsfileinfo bind-9.4.0.tar.gz
bind-9.4.0.tar.gz:
        chunk 0: 00000000001D1A6B_00000005 / (id:1907307 ver:5)
                copy 1: 192.168.0.71:9422
                copy 2: 192.168.0.73:9422
                copy 3: 192.168.0.74:9422
```

★ 通过增加计算机或增加新的硬盘动态扩充可用磁盘空间

★ 可以设置删除文件的空间回收时间

```
[root@mysql-bk serydir]# mfsgettrashtime bind-9.4.0.tar.gz
bind-9.4.0.tar.gz: 600
```

文件被删除 10 分钟后（600 秒），才真正删除文件，回收磁盘空间。

★ 为文件创建快照

MFS 文件系统的组成

1、 元数据服务器。在整个体系中负责管理管理文件系统，目前 MFS 只支持一个元数据服务器 master，这是一个单点故障，需要一个性能稳定的服务器来充当。希望今后 MFS 能支持多个 master 服务器，进一步提高系统的可靠性。
2、 元数据日志服务器。备份 master 服务器的变化日志文件，文件类型为 changelog_ml.*.mfs。当元数据服务器数据丢失或者损毁，可从日志服务器取得文件进行恢复。
3、 数据存储服务器 chunkserver。真正存储用户数据的服务器。存储文件时，首先把文件分成块，然后这些块在数据服务器 chunkserver 之间复制（复制份数可以手工指定，建议设置副本数为 3）。数据服务器可以是多个，并且数量越多，可使用的“磁盘空间”越大，可靠性也越高。
4、 客户端。使用 MFS 文件系统来存储和访问的主机称为 MFS 的客户端，成功挂接 MFS 文件系统以后，就可以像以前使用 NFS 一样共享这个虚拟性的存储了。

元数据服务器安装和配置

元数据服务器可以是 linux,也可以是 unix,你可以根据自己的使用习惯选择操作系统,在我的环境里,我是用 freebsd 做为 MFS 元数据的运行平台。GNU 源码，在各种类 unix 平台的安装都基本一致。

（一） 安装元数据服务

1、下载 GNU 源码

wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz

2、解包 tar zxvf mfs-1.6.11.tar.gz

3、切换目录 cd mfs-1.6.11
4、创建用户 useradd mfs –s /sbin/nologin
5、配置 ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
6、编译安装 make ; make install

（二） 配置元数据服务
元数据服务器的配置文件被放置于安装目录/usr/local/mfs/etc。与 mfs-1.5.12 版本不同的是：mfs-1.6.x 版安装完成只有模版文件，其后缀形如 mfsmaster.cfg.dist。为了使 mfs master 正常工作，需要两个配置文件 mfsmaster.cfg 及 mfsexports.cfg,前者为主配置文件，后者为权限控制文件（mfs 客户端挂接时使用）。

（1）主配置文件 mfsmaster.cfg,可直接从模版文件拷贝而来，打开这个配置文件/usr/local/mfs/etc/mfsmaster.cfg，看看都有哪些内容：

```
# WORKING_USER = mfs
# WORKING_GROUP = mfs
# SYSLOG_IDENT = mfsmaster
# LOCK_MEMORY = 0
# NICE_LEVEL = -19

# EXPORTS_FILENAME = /usr/local/mfs/etc/mfsexports.cfg

# DATA_PATH = /usr/local/mfs/var/mfs

# BACK_LOGS = 50

# REPLICATIONS_DELAY_INIT = 300
# REPLICATIONS_DELAY_DISCONNECT = 3600

# MATOML_LISTEN_HOST = *
# MATOML_LISTEN_PORT = 9419

# MATOCS_LISTEN_HOST = *
# MATOCS_LISTEN_PORT = 9420

# MATOCU_LISTEN_HOST = *
# MATOCU_LISTEN_PORT = 9421

# CHUNKS_LOOP_TIME = 300
# CHUNKS_DEL_LIMIT = 100
# CHUNKS_WRITE_REP_LIMIT = 1
# CHUNKS_READ_REP_LIMIT = 5

# REJECT_OLD_CLIENTS = 0
```

```
# deprecated, to be removed in MooseFS 1.7
# LOCK_FILE = /var/run/mfs/mfsmaster.lock
```

尽管每行都被注释掉了，但它们却是配置文件的默认值，要改变这些值，需要取消注释，然后明确指定其取值。接下来说明一下其中一些项目的含义。

- EXPORTS_FILENAME = /usr/local/mfs/etc/mfsexports.cfg 权限控制文件的存放位置。
- DATA_PATH = /usr/local/mfs/var/mfs 数据存放路径，只元数据的存放路径。那么这些数据都包括哪些呢？进目录看看，大致分 3 种类型的文件：

```
-rw-r-----  1 mfs  mfs    1640316 Mar 27 08:00 changelog.7.mfs
-rw-r-----  1 mfs  mfs     845817 Mar 27 07:00 changelog.8.mfs
-rw-r-----  1 mfs  mfs     672211 Mar 27 06:00 changelog.9.mfs
-rw-r-----  1 mfs  mfs  310664388 Mar 27 14:00 metadata.mfs.back
-rw-r-----  1 mfs  mfs     608088 Mar 27 14:00 stats.mfs
```

  这些文件也同样要存储在其他数据存储服务器的相关目录。
- MATOCS_LISTEN_PORT = 9420 MATOCS--master to chunkserver，即元数据服务器使用 9420 这个监听端口来接受数据存储服务器 chunkserver 端的连接。
- MATOML_LISTEN_PORT = 9419 MATOML---master to metalogger，用于备份元数据服务器的变化日志。注：Mfs-1.5.12 以前的版本没有这个项目。
- MATOCU_LISTEN_PORT = 9421 元数据服务器在 9421 端口监听，用以接受客户端对 MFS 进行远程挂接（客户端以 mfsmount 挂接 MFS）
- 其他部分看字面意思都不难理解。还有几个与时间有关的数值，其单位是秒。

这个配置文件，不必做修改就能工作了。

（2）配置文件/usr/local/mfs/etc/mfsexports.cfg,也可直接从模版文件复制而来。这个文件的内容，十分类似 NFS 服务器的 exports 文件．实际配置时，可参照这个文件的默认行来修改以满足自己的应用需求．我的 mfsexports.cfg 文件的内容为：

```
192.168.93.0/24     /     rw
```

（3）复制文件

cp /usr/local/mfs/var/mfs/metadata.mfs.empty /usr/local/mfs/var/mfs/metadata.mfs

这是一个 8 字节的文件，为 mfs-1.6.x 新增项目。

（三） 元数据服务器 master 启动

元数据服务器可以单独启动，即使没有任何数据存储服务器（chunkserver）也是能正常工作的，因此当我们安装配置完 MFS 后，即可启动它。执行命令 /usr/local/mfs/sbin/mfsmaster start ，如果没有意外，元数据库服务器就应该作为一个守护进程运行起来。现在我们可以通过 3 个方面来检查一下 MFS master 的运行状况：

1、 检查进程

```
mfs-ctrl# ps aux | grep mfsmaster | grep -v grep
mfs   29647  0.2 27.0 1173960 1126884  ??  S    11Mar09 139:15.90 sbin/mfsmaster start
```

2、 检查网络状态

```
mfs-ctrl# netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0     52  192.168.0.19.22        192.168.1.252.50265    ESTABLISHED
tcp4       0      0  192.168.0.19.9421      192.168.0.31.24328     ESTABLISHED
tcp4       0     94  192.168.0.19.9420      192.168.0.74.54171     ESTABLISHED
tcp4       0     94  192.168.0.19.9420      192.168.0.73.49985     ESTABLISHED
tcp4       0      0  192.168.0.19.9421      192.168.0.38.23232     ESTABLISHED
tcp4       0      0  192.168.0.19.9421      192.168.0.77.46630     ESTABLISHED
tcp4       0      0  192.168.0.19.9420      192.168.0.71.35259     ESTABLISHED
tcp4       0      0  192.168.0.19.9421      192.168.0.151.36031    ESTABLISHED
tcp4       0     13  192.168.0.19.9421      192.168.0.30.11501     ESTABLISHED
tcp4       0      0  *.9421                 *.*                    LISTEN
tcp4       0      0  *.9420                 *.*                    LISTEN
```

3、 检查系统日志

```
mfs-ctrl# tail -f /var/log/messages
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: inodes: 5902006
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: dirnodes: 5005394
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: filenodes: 896612
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: chunks: 899177
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: chunks to delete: 0
Mar 27 08:13:00 mfs-ctrl mfsmaster[29647]: chunkservers status:
```

MFS 的日志会直接写入系统日志。当我们增加数据存储服务器（chunkserver）或数据存储服务器（chunkserver）处故障时，都能在系统日志找到这些记录。注意，这个日志跟元数据变化日志不是一回事情。

（四）、关闭元数据服务器

关闭元数据服务器，务必使用 /usr/local/mfs/sbin/mfsmaster –s 这种方式，如果直接使用 kill 杀死进程，将导致下次启动时出现找不到相关文件，而不能正常启动服务器。这个一定要谨慎。当然，如果发生了这个事情，还是可以通过 mfsmetastore 来恢复的。

元数据日志服务器安装和配置

元数据日志服务为 mfs 1.6 以后版本新增的服务，即可以把元数据日志保留在元数据服务器，也可以单独存储。为保证其可靠性，最好单独放置。需要注意的是，源数据日志守护进程跟元数据服务器（master）在同一个服务器上，备份元数据日志的服务器作为它的客户端，从元数据服务器取得日志文件进行备份。

(一)安装元数据日志服务器 metalogger

1、下载 GNU 源码

wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz

2、解包 tar zxvf mfs-1.6.11.tar.gz

3、切换目录 cd mfs-1.6.11

4、创建用户 useradd mfs –s /sbin/nologin

5、配置 ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs

6、编译安装 make ; make install

（二）元数据日志服务（metalogger）配置

该服务仅需要一个配置文件，这里我们只需要从模板文件复制一个，然后稍微加以修改即可，下面是我的某个 metalogger 的配置文件：

```
[root@hynfs-2 etc]# more mfsmetalogger.cfg
# WORKING_USER = mfs
# WORKING_GROUP = mfs
# SYSLOG_IDENT = mfsmetalogger
# LOCK_MEMORY = 0
# NICE_LEVEL = -19

# DATA_PATH = /usr/local/mfs/var/mfs

# BACK_LOGS = 50
# META_DOWNLOAD_FREQ = 24

# MASTER_RECONNECTION_DELAY = 5

  MASTER_HOST = 192.168.93.18
  MASTER_PORT = 9419

# MASTER_TIMEOUT = 60

# deprecated, to be removed in MooseFS 1.7
# LOCK_FILE = /var/run/mfs/mfsmetalogger.lock
```

这个配置文件，唯一需要修改的地方就是 MASTER_HOST，它的值必须是元数据服务器的主机名或者ｉｐ地址。另外，为方便大家进一步理解，我把配置文件里其他几个项目简单的说明一下：

（１）SYSLOG_IDENT = mfsmetalogger　元数据日志服务运行时，在系统日志输出的标识，下面给出一段系统日志：

```
[root@hynfs-2 mfs]# tail -f /var/log/messages
Jan 11 15:21:00 hynfs-2 mfsmetalogger[21057]: sessions downloaded 346B/0.000253s (1.368 MB/s)
Jan 11 15:22:00 hynfs-2 mfsmetalogger[21057]: sessions downloaded 346B/0.000195s (1.774 MB/s)
Jan 11 15:23:00 hynfs-2 mfsmetalogger[21057]: sessions downloaded 346B/0.000247s (1.401 MB/s)
Jan 11 15:24:00 hynfs-2 mfsmetalogger[21057]: sessions downloaded 346B/0.000150s (2.307 MB/s)
```

（２）DATA_PATH = /usr/local/mfs/var/mfs　从元数据服务器(master)抓回文件，然后进行存放的路径。

（３）BACK_LOGS = 50　存放备份日志的总个数为５０，超出５０则轮转。在做元数据恢复时，仅仅需要最近的那个日志文件备份，因此默认的日志个数就足够了，这也保证了日志备份不会写满整个分区。

（４）META_DOWNLOAD_FREQ = 24　元数据备份文件下载请求频率。默认为２４小时，即每隔一天从元数据服务器(MASTER)下载一个 metadata.mfs.back 文件。当元数据服务器关闭或者出故障时，matedata.mfs.back 文件将消失，那么要恢复整个 mfs,则需从 metalogger 服务器取得该文件。请特别注意这个文件，它与日志文件一起，才能够恢复整个被损坏的分布式文件系统。

（三）元数据日志服务（metalogger）运行及关闭

１、启动过程为：

```
/usr/local/mfs/sbin/mfsmetalogger start
```

```
working directory: /usr/local/mfs/var/mfs
lockfile created and locked
initializing mfsmetalogger modules ...
mfsmetalogger daemon initialized properly
```

启动过程如果不能跟元数据服务器进行通信的话，系统会给出错误信息。

２、关闭服务，执行命令 **/usr/local/mfs/sbin/mfsmetalogger stop**

３、检查服务的运行状况。从两个方面看，一个是元数据服务器，另一个是本身的数据生成情况。

◆察看元数据服务器网络连接，可以看见日志服务器连接到元数据服务器的 tcp 9419 端口。

◆查看日志服务器的工作目录，正常情况应该看见已经有文件生成了（从元数据服务器获取过来的）。可以手动从元数据服务器复制一个日志文件过来比较文件的内容。

数据存储 chunkserver 服务器的安装配置

数据存储服务器 chunkserver 也是可以运行在各种类 unix 平台的，因此不再多说。一个 MFS 环境到底能集群多少服务器，作者的说法是上 PB 容量，个人建议，最好 3 台以上；并且专门用来做存储，不要把它跟 master 搞到一个机器（理论上没问题，实现也是可以的，但这不是一个好策略）。因为每个数据存储服务器的安装和配置都是相同的，所以只需按照一个服务器的操作就可以了。

（一）、安装数据存储服务器 chunkserver

1、下载 GNU 源码

wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz

2、解包 tar zxvf mfs-1.6.11.tar.gz

3、切换目录 cd mfs-1.6.11

4、创建用户 useradd mfs –s /sbin/nologin

5、配置 ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs

6、编译安装 make ; make install

（二）配置数据存储服务器 chunkserver

数据存储服务器有 2 个配置服务器需要修改，一个是主配置文件 mfschunkserver.cfg ,另一个配置文件是 mfshdd.cfg。每个服务器用来分配给 MFS 使用的空间最好是一个单独的硬盘或者一个 raid 卷，最低要求是一个分区。作者举的例子是创建一个大文件，然后挂接在本地，这不是个好主意，只能用来做实验了。

1、修改配置文件 /usr/local/mfs/etc/mfschunkserver.cfg。下面是修改了的配置文件：

```
#WORKING_USER = mfs
#WORKING_GROUP = mfs

# DATA_PATH = /usr/local/mfs/var/mfs
# LOCK_FILE = /var/run/mfs/mfschunkserver.pid
# SYSLOG_IDENT = mfschunkserver

# BACK_LOGS = 50

# MASTER_RECONNECTION_DELAY = 30
```

```
MASTER_HOST = 192.168.0.19
MASTER_PORT = 9420

# MASTER_TIMEOUT = 60

# CSSERV_LISTEN_HOST = *
# CSSERV_LISTEN_PORT = 9422

# CSSERV_TIMEOUT = 60

# CSTOCS_TIMEOUT = 60

# HDD_CONF_FILENAME = /usr/local/mfs/etc/mfshdd.cfg
```

这个配置文件里，没有注释符号“#”就是被修改过的项了，接下来是里面某些项的含义说明：

- MASTER_HOST = 192.168.0.19 元数据服务器的名称或地址，可以是主机名，也可以是 ip 地址，只要数据存储服务器能访问到元数据服务器就行。
- LOCK_FILE = /var/run/mfs/mfschunkserver.pid 与元数据服务器 master 的处理完全相同.
- CSSERV_LISTEN_PORT = 9422 CSSERV—chunkserver,这个监听端口用于与其它数据存储服务器间的连接，通常是数据复制。
- HDD_CONF_FILENAME = /usr/local/mfs/etc/mfshdd.cfg 分配给 MFS 使用的磁盘空间配置文件的位置。

2、修改配置文件/usr/local/mfs/etc/mfshdd.cfg。在我的服务器上，只有一个 1T 的 SATA 硬盘，分了一个 800G 容量的分区来做为 MFS 存储服务的组成部分。为了使 mfs 拥有写目录的权限，需要修改目录的属主。我的服务器的分区挂接点是 /data , 用 chown –R mfs:mfs /data 把属主改变。因为我的每个服务器只需贡献一个分区做为 MFS,因此配置文件只需要如下一行内容就可以了：

```
/data
```

这个文件默认情况下有好几行，我们最好把它删掉，因为按常规情况用注释符号“#”好像不起作用。

（三）启动数据存储服务器 chunkserver

在数据存储服务器 chunkserver 执行命令 /usr/local/mfs/sbin/mfschunkserver start 启动数据存储守护进程.通过以下几种方式来检查 chunkserver 的运行状态.

1、 查看进程 ps aux | grep mfschunkserver

2、 查看网络状态，正常情况下应该看见 9422 处于监听状态，如果有其他数据存储服务器 chunkserver 在同一个元数据服务器 master 管理下运行的话，应该能看见其他 chunkserver 跟本机的连接情况:

```
[root@mfs-1 ~]# netstat -an| grep 9422|grep EST
tcp        0     21 192.168.0.71:9422          192.168.0.73:60494          ESTABLISHED
```

3、 查看**元数据服务器**的系统日志，可以看见新增的数据存储服务器 chunkserver 被加入。

```
tail -f /var/log/messages
Mar 27 14:28:00 mfs-ctrl mfsmaster[29647]: server 3 (192.168.0.71): usedspace: 65827913728 (61 GB), totalspace: 879283101696 (818 GB), usage: 7.49%
```

（四） 关闭数据存储服务器

跟元数据服务器 master 相似，执行命令 /usr/local/mfs/sbin/mfschunkserver –s , chunkserver 服务就停下来了。为了使系统重启过程能自动启动 chunkserver 服务，可以通过在/etc/rc.local 文件追加行 /usr/local/mfs/sbin/mfschunkserver start 来达到这个目的（master 的自动重启处理也可同样处理）。

MFS 客户端的安装及配置

我的生产环境，只有 centos 和 freebsd 两种环境，因此下面的描述，只有 centos 及 freebsd 挂接 MFS 文件系统的情形，其他类型的 unix 系统，待日后尝试。对比前面的操作过程，客户端挂接后使用 MFS 集群文件系统才是最费时的事情。

一、centos 作为 MFS 的客户端。

（一） 安装 MFS 客户端

◆Mfsmount 需要依赖 FUSE,因此需要先安装好 fuse，这里我选用 fuse-2.7.4.tar.gz。

1、解包 tar zxvf fuse-2.7.4.tar.gz

2、切换目录 cd fuse-2.7.4.

3、配置 ./configure

4、编译安装 make；make install

如果系统已经安装了 fuse,则跳过这个步骤。

◆安装 MFS 客户端程序

1、修改环境变量文件/etc/profile ,追加下面的行，然后再执行命令 source /etc/profile 使修改生效。

```
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
```

如果不执行这个操作，在后面安装 MFS 的过程中，执行命令 ./configure --enable-mfsmount 时可能出现"checking for FUSE... no configure: error: mfsmount build was forced, but fuse development package is not installed"这样的错误，而不能正确安装 MFS 客户端程序。

2、解包 tar zxvf mfs-1.6.11.tar.gz

3、切换目录 cd mfs-1.6.11

4、创建用户 useradd mfs –s /sbin/nologin

5、配置 ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs --enable-mfsmount

6、编译安装 make ; make install

◆检查 MFS 客户端安装的结果。通过查看目录/usr/local/mfs/bin 目录的文件，应该发现如下文件：

```
[root@mysql-bk ~]# ll /usr/local/mfs/bin/
total 292
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfscheckfile -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsdirinfo -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsfileinfo -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsgetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsgettrashtime -> mfstools
-rwxr-xr-x 1 root root 185509 Mar  6 11:43 mfsmount
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrgetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrgettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrsetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrsettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssnapshot -> mfstools
-rwxr-xr-x 1 root root  49121 Mar  6 11:43 mfstools
```

（二）挂接和使用 MFS 文件系统

1、创建挂接点 mkdir /mnt/mfs

2、保证 fuse 模块被加载内核。

3、挂接 MFS /usr/local/mfs/bin/mfsmount /mnt/mfs –H 192.168.0.19 .注意，所有的 MFS 都是挂接同一个元数据服务器 master,而不是其他数据存储服务器 chunkserver !

4、通过查看磁盘使用情况来检查是否被挂接成功。

```
[root@mysql-bk ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1              19G  2.7G   16G  15% /
/dev/hda7              51G  180M   48G   1% /backup
/dev/hdc1             145G  6.4G  131G   5% /data
/dev/hda5              19G  173M   18G   1% /home
/dev/hda3              24G  217M   23G   1% /var
/dev/hda2              29G  1.6G   26G   6% /usr
tmpfs                 1.7G     0  1.7G   0% /dev/shm
mfs#192.168.0.19:9421                 2.5T  256G  2.2T  11% /mnt/mfs
```

5、 进入目录/mnt/mfs，上传一个文件，看是否正常？接着在手动用 touch 创建一个文件，然后再删除它们，看是否可以正常操作。

6、 设置文件副本数量，建议以 3 份为佳。

```
设置副本数目
mfsrsetgoal 3 /mnt/mfs
查看设置是否如我所愿
mfsgetgoal /mnt/mfs/serydir/bind-9.4.0.tar.gz
/mnt/mfs/serydir/bind-9.4.0.tar.gz: 3
```

7、 设置删除文件后空间回收时间。默认的回收时间为 7 天（604800 秒）

```
修改回收时间为 10 分钟
mfsrsettrashtime 600 /mnt/mfs
```

8、 把挂接命令追加到文件/etc/rc.local，可实现开机自动启动挂接 MFS.

二、freebsd 作为 MFS 客户端

Freebsd 安装和挂接 MFS 集群文件系统,比 centos 操作起来要复杂一些.mfsmount 需要依赖 fuse,并且需要在内核中加载 fusefs 模块。

（一）安装 fuse

1、解包 tar zxvf fuse-2.7.4.tar.gz

2、切换目录 cd fuse-2.7.4.

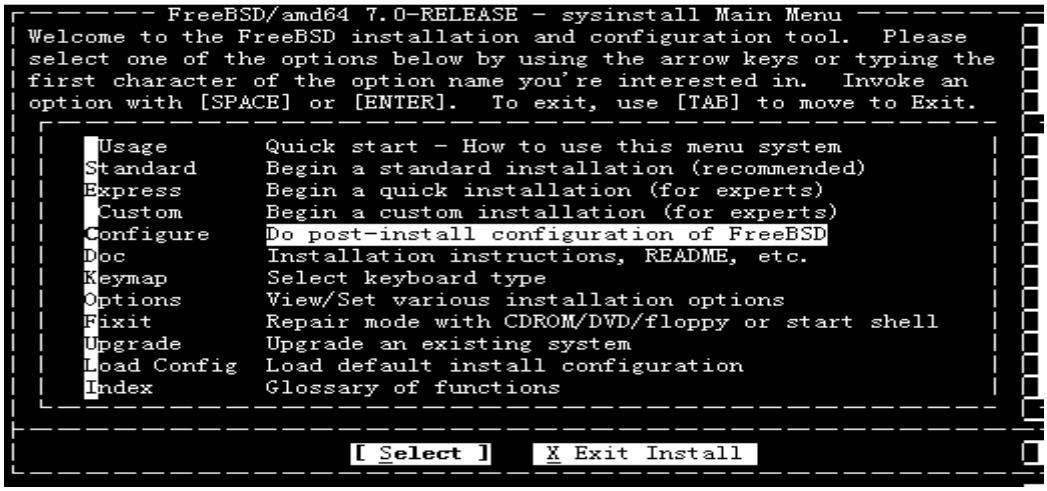
3、配置 ./configure

4、编译安装 make；make install

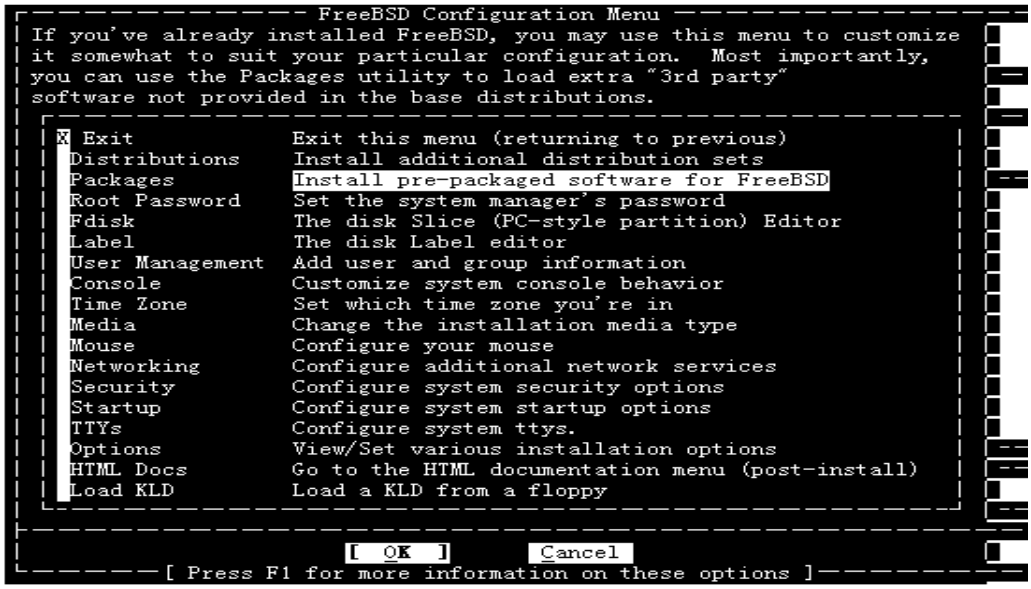
如果系统已经安装了 fuse,则跳过这个步骤。

（二） 安装内核模块 fusefs-kmod

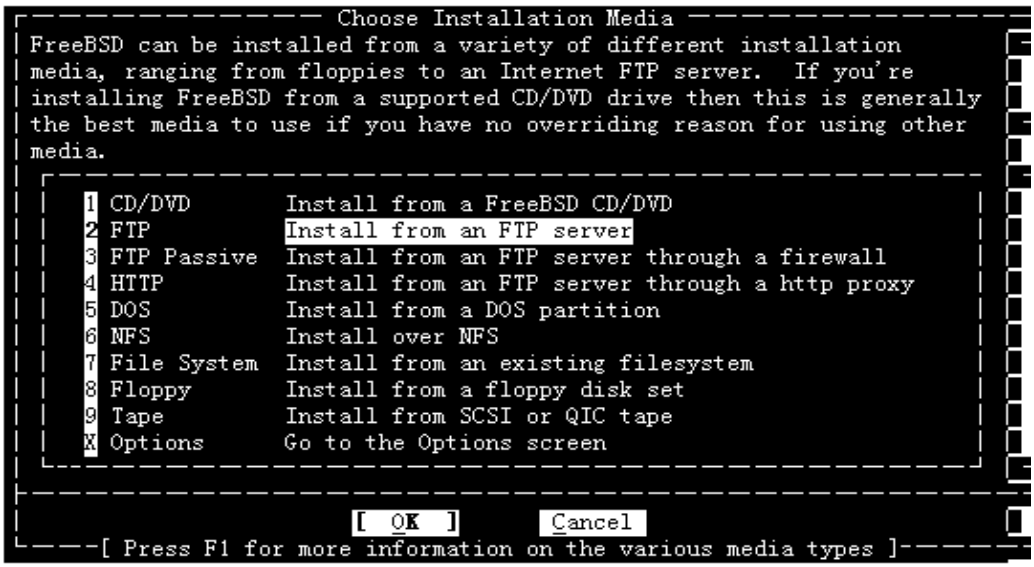
1、执行系统命令 sysinstall



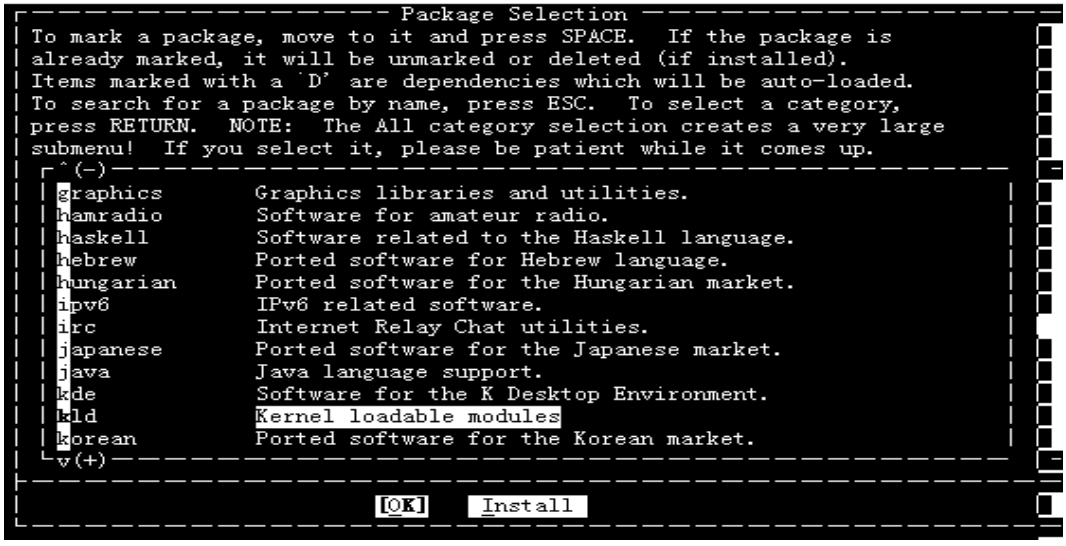
2、光标选定 Configure,进入下一步。



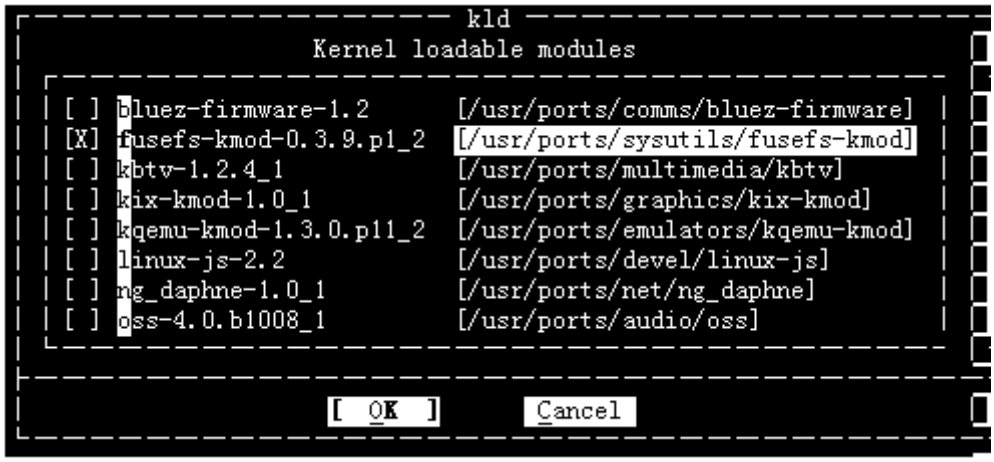
3、选择“Packages”,进入下一步。



4、选择“FTP”作为安装源，进入下一步。



5、选择“kld”后，回车执行默认动作“[OK]”，进入下一步选软件包。



6、选择“fusefs-kmod-0.3.9.p1_2”,按[OK]返回到第“4”步出现的那个操作界面。这时我们用“Tab”键选中底部右边的“Install”，完成安装后，会出现一个安装成功的提示，然后瞬

间消失。

◆ 加载 fusefs 模块 kldload /usr/local/modules/fuse.ko .如果加载不成功，请检查是否存在模块文件 fuse.ko.

◆ 检查 fusefs 模块是否被加载到内核：

```
fav0# kldstat
Id Refs Address            Size     Name
 1    2 0xffffffff80100000 ac6c08   kernel
 2    1 0xffffffffb08e0000 a232     fuse.ko
```

如果没有类似上面馆的输出，就表明 fusefs 模块没有加载成功。

（三）安装包 pkg-config：

1、cd /usr/ports/devel/pkg-config

2、make install clean

（四）安装 MFS 客户端

1、解包 tar zxvf mfs-1.6.11.tar.gz

2、切换目录 cd mfs-1.6.11

3、创建用户 pw useradd mfs –s /sbin/nologin

4、配置 ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs --enable-mfsmount

5、编译安装 make ; make install

◆检查 MFS 客户端安装的结果。通过查看目录/usr/local/mfs/bin 目录的文件，应该发现如下文件：

```
[root@mysql-bk ~]# ll /usr/local/mfs/bin/
total 292
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfscheckfile -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsdirinfo -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsfileinfo -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsgetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsgettrashtime -> mfstools
-rwxr-xr-x 1 root root 185509 Mar  6 11:43 mfsmount
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrgetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrgettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrsetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfsrsettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssetgoal -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssettrashtime -> mfstools
lrwxrwxrwx 1 root root      8 Mar  6 11:43 mfssnapshot -> mfstools
-rwxr-xr-x 1 root root  49121 Mar  6 11:43 mfstools
```

（五）挂接和使用 MFS 文件系统

1、创建挂接点 mkdir /mnt/mfs

2、挂接 MFS /usr/local/mfs/bin/mfsmount /mnt/mfs –H 192.168.0.19 .注意，所有的 MFS 都是挂接同一个元数据服务器 master,而不是其他数据存储服务器 chunkserver !

3、通过查看磁盘使用情况来检查是否被挂接成功。

```
[root@mysql-bk ~]# df -h
Filesystem              Size    Used   Avail Capacity  Mounted on
/dev/ad4s1a              26G    570M    24G     2%     /
```

devfs	1.0K	1.0K	0B	100%	/dev
/dev/ad4s1g	356G	157G	170G	48%	/data
/dev/ad4s1f	17G	215M	15G	1%	/home
/dev/ad4s1d	28G	1.1G	25G	4%	/usr
/dev/ad4s1e	24G	362M	21G	2%	/var
/dev/fuse0	2.5T	256G	2.2T	11%	/mnt/mfs

7、 进入目录/mnt/mfs，我们可以看见前面步骤用 centos 上传到分布式文件系统 MFS 里的文件。

8、 设置文件副本数量，建议以 3 份为佳。

```
设置副本数目
mfsrsetgoal 3 /mnt/mfs
查看设置是否如我所愿
mfsgetgoal /mnt/mfs/serydir/bind-9.4.0.tar.gz
/mnt/mfs/serydir/bind-9.4.0.tar.gz: 3
```

6、设置删除文件后空间回收时间。默认的回收时间为 7 天（604800 秒）

```
修改回收时间为 10 分钟
mfsrsettrashtime 600 /mnt/mfs
```

（六）自动挂接 MFS

创建文件/etc/rc.local,加入如下内容：.

```
#!/bin/sh
/sbin/kldload /usr/local/modules/fuse.ko
/usr/local/mfs/bin/mfsmount -h 192.168.0.19
```

就能实现开机或重启系统自动挂接 MFS 文件系统。

破坏性测试

一、测试数据存储服务器

我用 5 个服务器组成了 MFS 的存储平台，其中一个是 master,其余四个服务器是 chunkserver. 先停止一个 chunkserver 服务，然后在某个 MFS 客户端往挂接点的目录（/mnt/mfs）里复制数据或者创建目录/文件、或者读取文件、或者删除文件，观察操作是否能正常进行。再停止第 2 个 chunkserver，重复执行上述操作；然后再停止第 3 个服务器，执行类似的文件读些操作。减少 chunkserver 试验后，我们再来逐步增加 chunkserver 服务器,然后对 MFS 执行读写等相关访问操作，检验其正确性。

通过增减 chunkserver 服务器的测试，服务的可靠性确实不错，哪怕只剩下最后一个服务器，也能正常提供存储访问服务。

二、测试元数据服务器

元数据服务器最重要的文件在目录 /usr/local/mfs/var/mfs ,MFS 每一个数据的变化,都被记录在这个目录的文件里,我们可以通过备份这个目录的全部文件,来保障整个 MFS 文件系统的可靠性.在正常情况下,元数据服务器的改变日志文件(changelogs) 实时地、自动地复制到所有的数据存储服务器，并且以 changelog_csback.*.mfs 的形式命名。换句换说，即使元数据服务器报废了，也能再部署一个元数据服务器，然后从数据存储服务器 chunkserver 取得恢复所需要的文件。

（一）本地测试

1、停止元数据服务 /usr/local/mfs/sbin/mfsmaster

2、备份元数据服务器数据 cd /usr/local/mfs/var; tar czvf mfs.tgz mfs

3、删除目录 mv mfs mfs.bk 或 rm –rf mfs

4、启动元数据服务 ../sbin/mfsmaster start 启动失败，提示不能初始化数据。

5、解包 tar zxvf mfs.tgz

6、执行恢复操作 .. /sbin/mfsmetarestore －a

7、启动元数据服务 ../sbin/mfsmaster start

8、在 MFS 客户端检查 MFS 存储的数据是否跟恢复前一致？能否正常访问等等。

（二） 迁移测试

1、 安装新的 MFS 元数据服务器。

2、 从当前的元数据服器（ｍａｓｔｅｒ）或日志备份服务器(mfsmetalogger)复制备份文件 metadata.mfs.back/metadate_ml.mfs.back 到新的元服务器目录(metadata.mfs.back 需要定时用 crontab 备份).

3、 从当前的元数据服器（ｍａｓｔｅｒ）或日志备份服务器(mfsmetalogger)复制元数据服务器数据目录（/usr/local/mfs/var/mfs）到这个新的元数据服务器。

4、 停止原先的那个元数据服务器（关闭计算机或停止它的网络服务）。

5、 更改新的元数据服务器的 ip 为原来那个服务器的 ip.

6、 执行数据恢复操作，其命令为：mfsmetarestore -m metadata.mfs.back -o metadata.mfs changelog_ml.*.mfs;mfsmetarestore -a 恢复成功后再执行启动新的元数据服务操作。

7、 启动新的元数据服务 /usr/local/mfs/sbin/mfsmaster start

8、 在 MFS 客户端检查 MFS 存储的数据是否跟恢复前一致？能否正常访问等等。

注意:如果元数据服务不是以命令 mfsmastar stop 方式停止的话,再次重启会出现意外,因此需要先执行 mfsmetarestore －a 然后再启动.

感谢 Pawel Kalinowski （mfs 作者）提供帮助!

```
[root@hynfs-4 mfs]# time dd if=/dev/zero of=/root/test.dbf bs=64k count=300000
300000+0 records in
300000+0 records out
19660800000 bytes (20 GB) copied, 223.724 seconds, 87.9 MB/s

real    3m44.437s
user    0m0.155s
sys     0m49.681s
[root@hynfs-4 mfs]# time dd if=/dev/zero of=/mnt/mfs/test.dbf bs=64k count=300000
300000+0 records in
300000+0 records out
19660800000 bytes (20 GB) copied, 552.832 seconds, 35.6 MB/s

real    9m12.877s
```

```
user    0m0.205s
sys     0m32.115s
[root@hynfs-4 mfs]# time dd if=/mnt/mfs/test.dbf of=/dev/null bs=8k
2400000+0 records in
2400000+0 records out
19660800000 bytes (20 GB) copied, 381.884 seconds, 51.5 MB/s

real    6m21.932s
user    0m0.678s
sys     0m15.046s
[root@hynfs-4 mfs]# time dd if=/root/test.dbf of=/dev/null bs=8k
2400000+0 records in
2400000+0 records out
19660800000 bytes (20 GB) copied, 181.854 seconds, 108 MB/s

real    3m1.958s
user    0m0.700s
sys     0m15.962s
```

2010/1/11