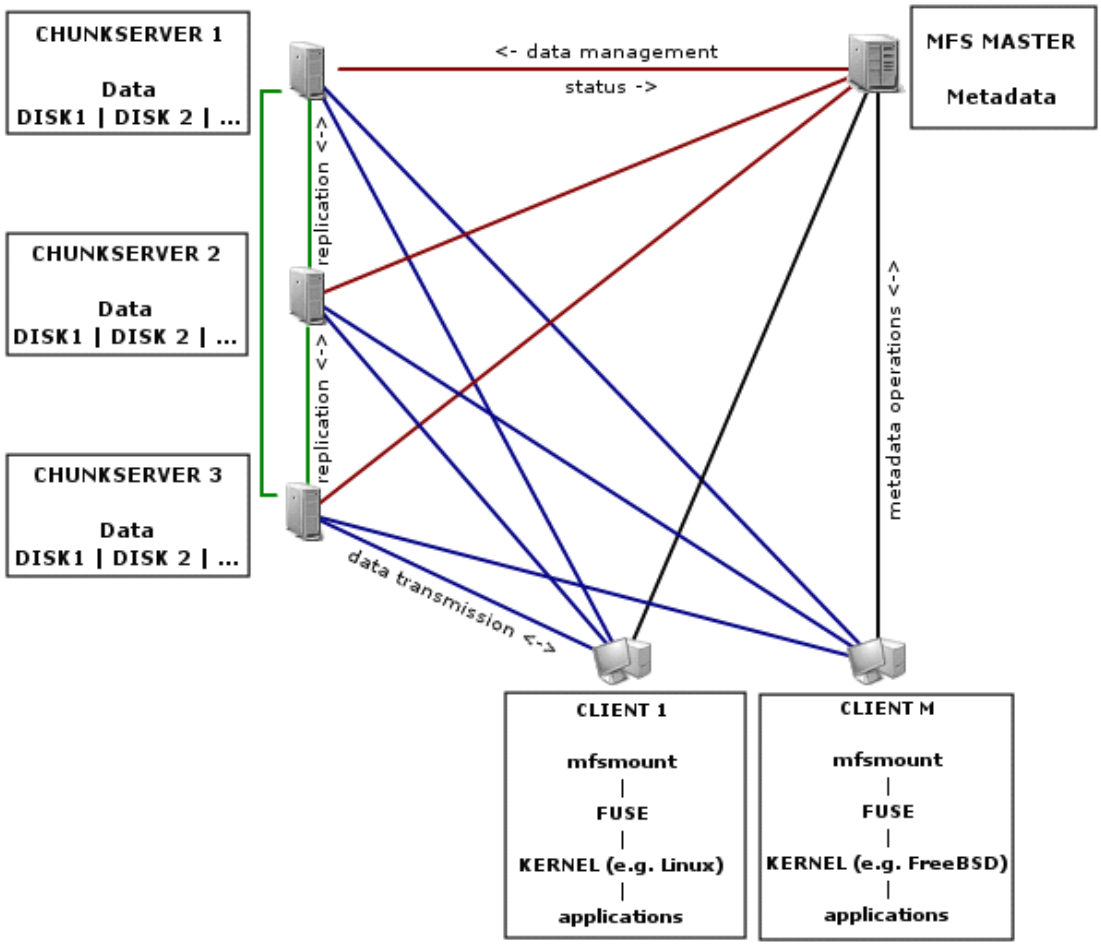
# MFS 文件系统使用手册

作者：ltgz777 整理：nonamexz

对于 mfs 文件系统也用了半年了，确实不错，最近又翻译了作者的三篇文章，再此一同发上，希望对大家有所帮助。不足之处还请指出，以便完善，谢谢！

官方的网络示意图是这样的：



# MFS 文件系统结构:

| 角色 | 角色作用 |
| --- | --- |
| 管理服务器<br>managing server (master) | 负责各个数据存储服务器的管理,文件读写调度,文件空间回收以及恢复.多节点拷贝 |
| 元数据日志服务器<br>Metalogger server（Metalogger） | 负责备份 master 服务器的变化日志文件，文件类型为 changelog_ml.*.mfs ，以便于在 master server 出问题的时候接替其进行工作 |
| 数据存储服务器<br>data servers (chunkservers) | 负责连接管理服务器,听从管理服务器调度,提供存储空间，并为客户提供数据传输. |
| 客户机挂载使用<br>client computers | 通过 fuse 内核接口挂接远程管理服务器上所管理的数据存储服务器,.看起来共享的文件系统和本地 unix 文件系统使用一样的效果. |

## 具体的实例

## 安装和配置元数据服务（master server）

### 安装元数据服务

| 创建用户 |
| --- |
| useradd mfs –s /sbin/nologin |
| 安装 |
| wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz<br>tar zxvf mfs-1.6.11.tar.gz<br>./configure –prefix=/usr/local/mfs –with-default-user=mfs –with-default-group=mfs<br>make ; make install |

## mfsmaster.cfg 的配置

配置文件位于安装目录/usr/local/mfs/etc ,需要的配置文件有两个 :mfsmaster.cfg 和 mfsexports.cfg , mfsmaster.cfg 是主配置文件，mfsexports.cfg 是被挂接目录及权限设置。

```
[root@nas etc]# cp mfsmaster.cfg.dist mfsmaster.cfg
[root@nas etc]# vi mfsmaster.cfg
# WORKING_USER = mfs              运行 master server 的用户
# WORKING_GROUP = mfs             运行 master server 的组
# SYSLOG_IDENT = mfsmaster        master server 在 syslog 中的标识，说明是由 master serve 产生的
# LOCK_MEMORY = 0                 是否执行 mlockall()以避免 mfsmaster 进程溢出（默认为 0）
# NICE_LEVEL = -19                运行的优先级(如果可以默认是 -19; 注意: 进程必须是用 root 启动)
# EXPORTS_FILENAME = /usr/local/mfs/etc/mfsexports.cfg   被挂接目录及其权限控制文件的存放位置
# DATA_PATH = /usr/local/mfs/var/mfs   数据存放路径，此目录下大致有三类文件，changelog，sessions 和 stats ;
# BACK_LOGS = 50     metadata 的改变 log 文件数目(默认是 50);
# REPLICATIONS_DELAY_INIT = 300   延迟复制的时间（默认是 300s）;
# REPLICATIONS_DELAY_DISCONNECT = 3600   chunkserver 断开的复制延迟（默认是 3600）;
# MATOML_LISTEN_HOST = *      metalogger 监听的 IP 地址(默认是*，代表任何 IP)；
# MATOML_LISTEN_PORT = 9419     metalogger 监听的端口地址(默认是 9419)；
# MATOCS_LISTEN_HOST = *     用于 chunkserver 连接的 IP 地址（默认是*，代表任何 IP）;
# MATOCS_LISTEN_PORT = 9420   用于 chunkserver 连接的端口地址（默认是 9420）;
# MATOCU_LISTEN_HOST = *     用于客户端挂接连接的 IP 地址(默认是*，代表任何 IP)；
# MATOCU_LISTEN_PORT = 9421   用于客户端挂接连接的端口地址（默认是 9421）;
# CHUNKS_LOOP_TIME = 300    chunks 的回环频率（默认是：300 秒）;
                                注：原文为 Chunks loop frequency in seconds (default is 300)
# CHUNKS_DEL_LIMIT = 100
# CHUNKS_WRITE_REP_LIMIT = 1   在一个循环里复制到一个 chunkserver 的最大 chunk 数目（默认是 1）
# CHUNKS_READ_REP_LIMIT = 5   在一个循环里从一个 chunkserver 复制的最大 chunk 数目（默认是 5）
# REJECT_OLD_CLIENTS = 0      弹出低于 1.6.0 的客户端挂接（0 或 1，默认是 0）
                                注意 mfsexports 访问控制对于那些老客户是没用的
```

需要注意的是，凡是用#注释掉的变量均使用其默认值。

以上是对 master server 的 mfsmaster.cfg 配置文件的解释，对于这个文件不需要做任何修改就可以工作。

## mfsexports.cfg 的配置

```
[root@nas etc]# vi mfsexports.cfg
#*                      /           ro
#192.168.1.0/24         /           rw
#192.168.1.0/24         /           rw,alldirs,maproot=0,password=passcode
#10.0.0.0-10.0.0.5      /test        rw,maproot=nobody,password=test
*                       .           rw
#*                      /           rw,alldirs,maproot=0

192.168.3.98            /tt         rw,alldirs,maproot=0
192.168.3.139           /           rw,alldirs,maproot=0
192.168.3.138           /           rw,alldirs,maproot=0,password=111111
```

该文件每一个条目分为三部分：

第一部分：客户端的 ip 地址

第二部分：被挂接的目录

第三部分：客户端拥有的权限

**地址可以指定的几种表现形式：**

| | |
|---|---|
| * | 所有的 ip 地址 |
| n.n.n.n | 单个 ip 地址 |
| n.n.n.n/b | IP 网络地址/位数掩码 |
| n.n.n.n/m.m.m.m | IP 网络地址/子网掩码 |
| f.f.f.f-t.t.t.t | IP 段 |

**目录部分需要注意两点：**

| | |
|---|---|
| / | 标识 MooseFS 根; |
| . | 表示 MFSMETA 文件系统 |

**权限部分：**

| | |
|---|---|
| ro | 只读模式共享 |
| rw | 读写的方式共享 |
| alldirs | 许挂载任何指定的子目录 |

| | |
|---|---|
| maproot | 映射为 root，还是指定的用户 |
| password | 指定客户端密码 |

## 启动 master server

master server 可以单独启动(所谓单独启动就是在没有数据存储服务器（ chunkserver ）的时候也可以启动，但是不能存储，chunkserver 启动后会自动的加入 )。安装配置完 MFS 后，即可启动它。

执行命令 /usr/local/mfs/sbin/mfsmaster start ，可通过检查如下：

```
[root@nas etc]# ps -ef|grep mfs
mfs       12327     1  0 08:38 ?        00:00:00 /usr/local/mfs/sbin/mfsmaster start
```

## 停止 master server

安全停止 master server 是非常必要的 ,最好不要用kill。利用 mfsmaster –s 来安全停止 master serve，一旦是用了 kill 也是有解决方法的，后文有说明。

要经常性的查看系统日志（ tail -f /var/log/messages ）

# 安装和配置元数据日志服务器（ metalogger ）

## 安装元数据日志服务

| 创建用户 |
|---|
| useradd mfs –s /sbin/nologin |
| 安装 |
| wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz<br>tar zxvf mfs-1.6.11.tar.gz<br>./configure –prefix=/usr/local/mfs –with-default-user=mfs –with-default-group=mfs<br>make ; make install |

## mfsmetalogger.cfg 的配置

该服务只有一个配置文件，那就是 mfsmetalogger.cfg。

```
[root@mail etc]# vi 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  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。
# MASTER_RECONNECTION_DELAY = 5
MASTER_HOST = 192.168.3.34
# MASTER_PORT = 9419
# MASTER_TIMEOUT = 60
# deprecated, to be removed in MooseFS 1.7
# LOCK_FILE = /var/run/mfs/mfsmetalogger.lock
```

文中的大多数变量不难理解，类似于 mfsmaster.cfg 中的变量，其中：

这个文件中需要修改的是 MASTER_HOST 变量，这个变量的值是 master server 的 IP 地址。

## 启动 metalogger 服务

```
[root@mail sbin]# ./mfsmetalogger start
working directory: /usr/local/mfs/var/mfs
lockfile created and locked
initializing mfsmetalogger modules ...
mfsmetalogger daemon initialized properly
```

这说明 metalogger 服务正常启动了。利用命令检查：

通过进程：

```
[root@mail sbin]# ps -ef |grep mfs
mfs       12254     1  0 15:25 ?        00:00:00 ./mfschunkserver start
```

通过检查端口：

```
[root@mail sbin]# lsof -i:9419
COMMAND   PID   USER   FD   TYPE  DEVICE  SIZE  NODE  NAME
mfsmetalo 12292 mfs    7u   IPv4  1395372       TCP mail.tt.com:52456->192.168.3.34:9419 (ESTABLISHED)
```

查看日志服务器的工作目录

```
[root@mail mfs]# pwd
/usr/local/mfs/var/mfs
[root@mail mfs]# ll
total 8
-rw-r----- 1 mfs mfs 249 Jan 13 15:39 changelog_ml.1.mfs
-rw-r----- 1 mfs mfs 519 Jan 13 15:40 sessions_ml.mfs
```

这是运行 18 小时后：

```
[root@mail mfs]# ll
total 1808
-rw-r----- 1 mfs mfs       0 Jan 14 08:40 changelog_ml.0.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 23:39 changelog_ml.10.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 22:39 changelog_ml.11.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 21:39 changelog_ml.12.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 20:39 changelog_ml.13.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 19:39 changelog_ml.14.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 18:39 changelog_ml.15.mfs
-rw-r----- 1 mfs mfs    4692 Jan 13 17:39 changelog_ml.16.mfs
-rw-r----- 1 mfs mfs    4722 Jan 13 16:39 changelog_ml.17.mfs
-rw-r----- 1 mfs mfs     249 Jan 13 15:39 changelog_ml.18.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 08:39 changelog_ml.1.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 07:39 changelog_ml.2.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 06:39 changelog_ml.3.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 05:39 changelog_ml.4.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 04:39 changelog_ml.5.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 03:39 changelog_ml.6.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 02:39 changelog_ml.7.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 01:39 changelog_ml.8.mfs
-rw-r----- 1 mfs mfs    4692 Jan 14 00:39 changelog_ml.9.mfs
-rw-r----- 1 mfs mfs 915016 Jan 14 09:00 csstats.mfs
-rw-r----- 1 mfs mfs 777640 Jan 14 08:10 metadata_ml.mfs.back
-rw-r----- 1 mfs mfs     519 Jan 14 09:16 sessions_ml.mfs
```

## 停止 metalogger 服务

```
[root@mail sbin]# ./mfsmetalogger -s
working directory: /usr/local/mfs/var/mfs
sending SIGTERM to lock owner (pid:12284)
waiting for termination ... terminated
```

如果没有启动 metalogger 服务，在 master server 则会有如下提示信息产生：

```
tail -f /var/log/messages
Dec 30 16:53:00 nas mfsmaster[14291]: no meta loggers connected !!!
```

# 安装配置数据存储服务器（chunkserver）

## 安装数据存储服务

| 创建用户 |
| --- |
| useradd mfs –s /sbin/nologin |
| 安装 |
| wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz<br>tar zxvf mfs-1.6.11.tar.gz<br>./configure –prefix=/usr/local/mfs –with-default-user=mfs –with-default-group=mfs<br>make ; make install |

配置文件位于安装目录 /usr/local/mfs/etc，需要的配置文件有两个：mfschunkserver.cfg 和 mfshdd.cfg，mfschunkserver.cf是主配置文件，mfshdd.cfg 是服务器用来分配给 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。

## mfschunkserver.cfg 的配置

```
[root@mail etc]# vi mfschunkserver.cfg
# 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 = 192.168.3.34              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
MASTER_PORT = 9420
# MASTER_TIMEOUT = 60
# CSSERV_LISTEN_HOST = *
# CSSERV_LISTEN_PORT = 9422             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
# CSSERV_TIMEOUT = 60
# CSTOCS_TIMEOUT = 60
# 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
```

文中的大多数变量不难理解，类似于 mfsmaster.cfg 中的变量

## mfshdd.cfg 的配置

```
[root@mail etc]# more mfshdd.cfg
/data
```

在这里/data 是一个给 mfs 的分区，但在本机上是一个独立的磁盘的挂载目录，用 chown –R mfs:mfs /data 把属主改变为 mfs。

## 启动 mfschunkserver

执行命令 /usr/local/mfs/sbin/mfschunkserver start ，如果没有意外，mfschunkserver 就应该作为一个守护进程运行起来。是否启动，检查如下：

```
[root@nas etc]# ps -ef|grep mfs
mfs         12327         1    0 08:38 ?              00:00:00 /usr/local/mfs/sbin/ mfschunkserver start
```

## 停止 mfschunkserver

停止 mfschunkserver，利用 mfschunkserver–s 来安全停止 mfschunkserver。

# MFS 客户端的安装及配置

由于 MFS 客户端依赖于 fuse，所以要先安装 fuse。

## MFS 客户端的安装

| 第一步：安装 fuse |
| --- |
| 如果所在的系统已经安装了 fuse,则跳过这个步骤，高版本的 Linux 内核已经支持了。<br>wget http://nchc.dl.sourceforge.net/project/fuse/fuse-2.X/2.8.1/fuse-2.8.1.tar.gz<br>tar zxf fuse-2.8.1.tar.gz<br>cd fuse-2.8.1<br>./configure<br>make && make install<br>echo ‘exportPKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH >>/etc/profile<br>source /etc/profile |
| 第二步：创建 mfs 用户 |
| useradd mfs –s /sbin/nologin |
| 第三步：安装 mfs |

```
wget http://ncu.dl.sourceforge.net/project/moosefs/moosefs/1.6.11/mfs-1.6.11.tar.gz
tar zxvf mfs-1.6.11.tar.gz
cd mfs-1.6.11
./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs --enable-mfsmount
make && make install
```

在这个过程中，当执行到–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，

```
.....
checking for FUSE... no
********************************** mfsmount disabled ********************************
* fuse library is too old or not installed - mfsmount needs version 2.6 or higher *
*************************************************************************************
.....
```

而不能正确安装 MFS 客户端程序，这是因为环境变量没有设置，先编辑/etc/profile 在此文件中加入如下条目：

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

然后再利用 source 命令 /etc/profile 使修改生效：source /etc/profile 即可，也可直接在命令行中直接执行：

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

## 挂接 MFS 文件系统

（1）、创建挂接点 mkdir /mnt/mfs

（2）、加载 fuse 模块到内核：modprobe fuse

（3）、挂接 MFS

```
/usr/local/mfs/bin/mfsmount /mnt/mfs -H 192.168.3.34 -p
```

然后在输入密码就可以了

特别需要注意的是，所有的 MFS 都是挂接同一个元数据服务器 master 的 IP,而不是其他数据存储服务器 chunkserver 的 IP。

## 查看挂载情况

通过 df 命令查看磁盘使用情况来检查是否被挂接成功

```
root@bzd mfs]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                           73G   25G   45G  36% /
/dev/sda1                  99M   13M   82M  13% /boot
none                      247M     0  247M   0% /dev/shm
MFS                        45G  204M   45G   1% /mnt/mfs
MFSMETA                    72K   72K     0 100% /mnt/mfsmeta
```

利用 mount 命令查看：

```
[root@www ~]# mount
mfsmeta#192.168.3.34:9421 on /mnt/mfsmeta type fuse (rw,nosuid,nodev,allow_other,default_permissions)
mfs#192.168.3.34:9421 on /mnt/mfs type fuse (rw,nosuid,nodev,allow_other,default_permissions
```

## 卸载已挂接的文件系统

利用 Linux 系统的 umount 命令就可以了，例如：

```
[root@www ~]# umount /mnt/mfs
```

如果出现下列情况：

```
[root@www ~]# umount /mnt/mfs
umount: /mnt/mfs: device is busy
umount: /mnt/mfs: device is busy
```

则说明客户端本机有正在使用此文件系统，可以查明是什么命令正在使用，然后推出就可以了，最好不要强制退出。

# mfscgiserv 的使用

Mfscgiserv 是用 python 编写的一个 web 服务器，它的监听端口是 9425，可以利用：/usr/local/mfs/sbin/mfscgiserv 来启动，用户利用浏览器就可全面监控所有客户挂接，chunkserver 及 master server，客户端的各种操作等等，绝对是个好工具。

在任何一台装有浏览器的机器上都可以查看：http://192.168.3.34：9425

# MFS 文件系统管理

## 编译和安装

MooseFS 部署的首选方法是从源代码安装,源代码包安装支持标准./configure && make && make install 的步骤，重要的配置选项有：

| | |
|---|---|
| --disable-mfsmaster | 不创建成管理服务器（用于纯节点的安装） |
| --disable-mfschunkserver | 不创建成数据存储 chunkserver 服务器 |
| --disable-mfsmount | 不创建 mfsmount 和 mfstools( 如果用开发包安装 ,他们会被默认创建的 ) |
| --enable-mfsmount | 确定安装 mfsmount 和 mfstools |
| --prefix=DIRECTORY | 锁定安装目录（默认是/usr/local） |
| --sysconfdir=DIRECTORY | 选择配置文件目录（默认是${prefix}/etc) |
| --localstatedir=DIRECTORY | 选择变量数据目录（默认是${prefix}/var，MFS 元数据被存储在 mfs 的子目录下，默认是${prefix}/var/mfs ） |
| --with-default-user | 运行守护进程的用户，如果配置文件中没有设定用户，默认为 nobody 用户 |
| --with-default-group=GROUP | 运行守护进程的用户组，如果配置文件中没有设定用户组，默认为 nogroup 用户组 |

例如用 FHS（文件系统层次标准）的兼容路径在 Linux 上的安装：

```
./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var/lib
```

编译安装遵守标准的 DESTDIR= variable ，允许安装包在临时目录(例如以创造二进制包)。已经存在的配置或这是元数据文件将会被覆盖掉。

## 管理服务器（master）

作为管理服务器(master)是 MooseFS 部署中重要的一个元素，在硬件方面，应该被安装在一台能够保证高可靠性和能胜任的整个系统存取的要求的机器上。一个明智的做法是用一个配有冗余电源、

ECC 内存、磁盘阵列，如 RAID1/RAID5/RAID10。在操作系统方面，管理服务器的操作系统应该是具有 POSIX 兼容的系统（到目前支持 Linux, FreeBSD, Mac OS X and OpenSolaris）。

**安装管理服务器（master server）的过程：**

1、安装 mfs-master

2、如果是从源码安装的话，在 configure 时不要加--disable-mfsmaster 选项。

3、创建运行 master 的用户（如果这样的用户不存在的话）

4、确定存放元数据文件的目录存在，而且能够被运行 master 的用户可写（通过 configure 的选项来设置运行 master server 的用户和元数据存储的路径，make install 命令的执行要用 root 用户）

5、配置 master server 服务器是通过配置文件 mfsmaster.cfg 来做的，要特别注意的是 TCP 端口的使用

6、添加或创建（依赖于操作系统和发布版本）一组启动 mfsmaster 进程的脚本

安装完管理服务器后，便可以用 mfsmaster 命令来启动 master server，如果用 root 用户执行 mfsmaster 命令，则在启动后转为 mfsmaster.cfg 中指定的用户来运行，否则将以执行 mfsmaster 命令的用户来运行 master server。

# 元数据日志服务器

元数据日志守护进程是在安装 master server 时一同安装的，最小的要求并不比 master 本身大，可以被运行在任何机器上（例如任一台 chunkserver），但是最好是放置在 MooseFS master 的备份机上，备份 master 服务器的变化日志文件，文件类型为 changelog_ml.*.mfs。因为主要的 master server 一旦失效，可能就会将这台 metalogger 机器取代而作为 master server。

**安装管理进程：**

1、从源代码安装 mfs-master，在执行 configure 时不要带有--disable-mfsmaste 选项

2、创建有运行 mfsmetalogger 服务权限运行的用户（如果这样的用户不存在的话）

3、确定存放元数据文件的目录存在，而且能够被运行 mfsmetalogger 服务的用户可写（通过 configure 的选项来设置运行 mfsmetalogger 服务的用户和元数据存储的路径，make install 命令的执行

要用 root 用户）

4、通过 mfsmetalogger.cfg 配置 mfsmetalogger 服务，要特别注意的是 TCP 端口，这里要使用 MASTER_PORT 要必须和 mfsmaster.cfg 文件中的 MATOML_LISTEN_PORT 一致。

5、添加或创建（依赖于操作系统和发布版本）一组启动 mfsmetalogger 进程的脚本

安装完管理服务器后，便可以用 mfsmetalogger 命令来启动 mfsmetalogger server，如果用 root 用户执行 mfsmetalogger 命令，则在启动后转为 mfsmetalogger.cfg 中指定的用户来运行，否则将以执行 mfsmetalogger 命令的用户来运行 mfsmetalogger server。

## 数据服务器

安装完管理服务器后，将安装数据服务器(chunkservers)，这些机器的磁盘上要有适当的剩余空间，而且操作系统要遵循 POSIX 标准（验证了的有这些： Linux, FreeBSD, Mac OS X and OpenSolaris）。Chunkserver 在一个普通的文件系统上储存数据块/碎片（chunks/fragments）作为文件。

**Linux:**

creating:

```
dd if=/dev/zero of=file bs=100m seek=400 count=0
mkfs -t ext3 file
```

mounting:

```
mount -o loop file mount-point
```

**FreeBSD:**

creating and mounting:

```
dd if=/dev/zero of=file bs=100m count=400
mdconfig -a -t vnode -f file -u X
newfs -m0 -O2 /dev/mdX
mount /dev/mdX mount-point
mounting a previously created file system:
mdconfig -a -t vnode -f file -u X
mount /dev/mdX mount-point
```

**Mac OS X:**

```
Start "Disk Utility" from "/Applications/Utilities"
Select from menu "Images->New->Blank Image ..."
```

注: **每一个chunkserver的磁盘都要为增长中的chunks保留些磁盘空间，从而达到创建新的chunk，只有磁盘都超过256M并且chunkservers报告自由空间超过1GB总量才可以被新的数据访问。最小的配置,应该从几个G字节的存储。**

## 安装数据服务器（chunkserver):

1、 把预先隔离的磁盘空间作为一个单独的文件系统，挂接在一个本地的目录下（如：/mnt/hd1, /mnt/hd2 等等）;

2、安装 mfs-chunkserver，在执行 configure 时要不带--disable-mfschunkserver 选项，

3、创建有运行 chunkserver 服务权限运行的用户（如果这样的用户不存在的话）

4、并给予这个户对整个 MooseFS 文件系统写的权限

5、利用 mfschunkserver.cfg 文件配置 mfschunkserver 服务，特别要注意的是 TCP 端口（MASTER_PORT 变量要和 mfsmaster.cfg 中 MATOCS_LISTEN_PORT 的值一样）。

6、在 mfshdd.conf 文件中列出要用于 MooseFS 的挂载点

7、添加或创建（依赖于操作系统和发布版本）一组启动 mfschunkserver 进程的脚本

**注：**

**Mfschunkserver 的本地 ip 很重要，Mfschunkserver 用此 ip 和 mfsmaster 进行连接，mfsmaster 通过此 ip 和 MFS 客户端连接（mfsmount），而且其它 chunkservers 之间的通讯也是通过这个 ip，因此这个 ip 必须是远程可访问的。因此 mfsmaster 的本地 ip 地址（MASTER_HOST）设置必须和 chunkserver 一样，以便于正确的连接，通常的做法是 mfsmaster，chunkservers 和 MFS 客户端在同一网段。一般的回环地址(localhost, 127.0.0.1)不能用于 MASTER_HOST，它将使 chunkserver 无法被其她主机访问(这样的配置只会是单机器的机器 mfsmaster,mfschunkserver 和 mfsmount 运行)。**

安装完 mfschunkserver 后，便可以用 mfschunkserver 命令来启动 mfschunkserver 服务器，如果用 root 用户执行 mfschunkserver 命令，则在启动后转为 mfschunkserver.cfg 中指定的用户来运行，否则将以执行 mfschunkserver 命令的用户来运行 mfschunkserver 服务。

## 客户端（mfsmount）

mfsmount 需要 FUSE 才可以正常工作，FUSE 支持多种操作系统：Linux, FreeBSD, OpenSolaris and MacOS X。

Linux 一个内核模块的 API 版本至少必需是 7.8 的，这个可以用 dmesg 命令来检测，当载入内核

模块后，应该能看到有一行 fuse init (API version 7.8)。一些可用的 fuse 版本是 2.6.0 以上，Linux kernel 2.6.20( Linux 内核从 2.6.20 后加入了 fuse )以上。由于一些小 bug,因此比较新模块被荐使用 ,如 fuse 2.7.2 及 Linux 2.6.24( 尽管 fuse 2.7.x 单独没有包含 getattr/write race condition fix )。在 FreeBSD 系统上 fusefs-kmod 版本要 0.3.9 以上的才可以，在 MacOS X Mac 上 FUSE 要 10.5 版本。

## 安装 MooseFS 客户端

1、安装 mfs-client，从源代码安装，在进行 configure 时不要加--disable-mfsmount 选项就可以了

2、建立被 MooseFS 挂接的目录，例如/mnt/mfs。

3、MooseFS 用一下的命令挂接：

**mfsmount [-h master] [-p port] [-l path] [-w mount-point]**

-H MASTER：是管理服务器（master server）的 ip 地址

-P PORT： 是管理服务器（master server）的端口号，要按照 mfsmaster.cfg 配置文件中的变量 MATOCU_LISTEN_POR 的之填写。如果 master serve 使用的是默认端口号则不用指出。

-S PATH：指出被挂接 mfs 目录的子目录，默认是/目录，就是挂载整个 mfs 目录。

Mountpoint：是指先前创建的用来挂接 mfs 的目录。

# 使用 MooseFS

## 挂载文件系统

启动管理服务器（ master server）和数据服务器(chunkservers) (chunkservers 一个是必需的,但至少两个推荐) 后，客户机便可以利用 mfsmount 挂接 mfs 文件系统。

MooseFS 文件系统利用下面的命令：

**mfsmount mountpoint [-d] [-f] [-s] [-m] [-n] [-p] [-H MASTER] [-P PORT] [-S PATH] [-o OPT[,OPT...]]**

-H MASTER：是管理服务器（master server）的 ip 地址

-P PORT： 是管理服务器（ master server）的端口号，要按照 mfsmaster.cfg 配置文件中的变量 MATOCU_LISTEN_POR 的之填写。如果 master serve 使用的是默认端口号则不用指出。

-S PATH：指出被挂接 mfs 目录的子目录，默认是/目录，就是挂载整个 mfs 目录。

Mountpoint：是指先前创建的用来挂接 mfs 的目录。

在开始 mfsmount 进程时，用一个-m 或-o mfsmeta 的选项，这样可以挂接一个辅助的文件系统 MFSMETA，这么做的目的是对于意外的从 MooseFS 卷上删除文件或者是为了释放磁盘空间而移动的文件而又此文件又过去了垃圾文件存放期的恢复，例如：

```
mfsmount -m /mnt/mfsmeta
```

需要注意的是，如果要决定挂载 mfsmeta，那么一定要在 mfsmaster 的 mfsexports.cfg 文件中加入如下条目：

```
*                               .               rw
```

原文件中有此条目，只要将其前的#去掉就可以了。

基本操作

挂载文件系统后就可以执行所有的标准的文件操作了（如创建，拷贝，删除，重命名文件，等等）。MooseFS 是一个网络文件系统，因此操作进度可能比本地系统要慢。

MooseFS 卷的剩余空间检查可以用和本地文件系统同样的方法，例如 df 命令：

```
$ df -h | grep mfs
mfsmaster:9421              85T     80T    4.9T    95% /mnt/mfs
mfsmaster:9321             394G    244G    151G    62% /mnt/mfs-test
```

重要的是每一个文件可以被储存多个副本，在这种情况下，每一个文件所占用的空间要比其文件本身大多了。此外,被删除且在有效期内（trashtime）的文件都放在一个“垃圾箱”,所以他们也占用的空间,其大小也依赖文件的份数。就像其他 Unix 的文件系统一样,以防删除一个被其它进程打开文件，数据将被一直存储,至少直到文件被关闭。

# MooseFS 的特定的操作

1、设定的目标

目标（goal），是指文件被拷贝的份数，设定了拷贝的份数后是可以通过可以 mfsgetgoal 命令来证实的，也可以通过 mfssetgoal 来改变设定。例如：

```
$ mfsgetgoal /mnt/mfs-test/test1
/mnt/mfs-test/test1: 2
```

```
$ mfssetgoal 3 /mnt/mfs-test/test1
/mnt/mfs-test/test1: 3
$ mfsgetgoal /mnt/mfs-test/test1
/mnt/mfs-test/test1: 3
```

用 mfsgetgoal –r 和 mfssetgoal –r 同样的操作可以对整个树形目录递归操作。

```
$ mfsgetgoal -r /mnt/mfs-test/test2
/mnt/mfs-test/test2:
files with goal        2 :                36
directories with goal  2 :                1
$ mfssetgoal -r 3 /mnt/mfs-test/test2
/mnt/mfs-test/test2:
inodes with goal changed:                      37
inodes with goal not changed:                  0
inodes with permission denied:                 0
$ mfsgetgoal -r /mnt/mfs-test/test2
/mnt/mfs-test/test2:
files with goal        3 :                36
directories with goal  3 :                1
```

实际的拷贝份数可以通过 mfscheckfile 和 mfsfileinfo 命令来证实，例如：

```
$ mfscheckfile /mnt/mfs-test/test1
/mnt/mfs-test/test1:
3 copies: 1 chunks
$ mfsfileinfo /mnt/mfs-test/test1
/mnt/mfs-test/test1:
        chunk 0: 00000000000520DF_00000001 / (id:336095 ver:1)
                copy 1: 192.168.0.12:9622
                copy 2: 192.168.0.52:9622
                copy 3: 192.168.0.54:9622
```

注意：一个不包含数据的零长度的文件,尽管没有设置为非零的目标（the non-zero "goal"）,但用命令查询将返回一个空的结果，例如：

```
[root@www bin]# touch       /mnt/mfs/mmm
[root@www bin]# ./mfsfileinfo    /mnt/mfs/mmm
/mnt/mfs/mmm:
```

但是如果对此文件进行编辑，如：

```
[root@www bin]# echo "1234"> /mnt/mfs/mmm
```

然后看：

```
root@www bin]# ./mfsfileinfo    /mnt/mfs/mmm
/mnt/mfs/mmm:
        chunk 0: 0000000000000040_00000001 / (id:64 ver:1)
                copy 1: 192.168.3.31:9422
                copy 2: 192.168.3.96:9422
                copy 3: 192.168.3.139:9422
```

此时在将文件清空：

```
[root@www bin]# echo ""> /mnt/mfs/mmm
```

然后在看：

```
[root@www bin]# ./mfsfileinfo    /mnt/mfs/mmm
/mnt/mfs/mmm:
        chunk 0: 0000000000000041_00000001 / (id:65 ver:1)
                copy 1: 192.168.3.31:9422
                copy 2: 192.168.3.96:9422
                copy 3: 192.168.3.139:9422
```

副本将任然存在。

假如改变一个已经存在的文件的拷贝个数，那么文件的拷贝份数将会被扩大或者被删除，这个过程会有延时。可以通过上面的命令来证实。

对一个目录设定“目标”，此目录下的新创建文件和子目录均会继承此目录的设定，但不会改变已经存在的文件及目录的拷贝份数。例如：

```
[root@bzd  f]# touch 1
[root@bzd  f]# echo “11” > 1
[root@bzd  f]# /usr/local/mfs/bin/mfsfileinfo 1
1:
        chunk 0: 0000000000000043_00000001 / (id:67 ver:1)
                copy 1: 192.168.3.31:9422
                copy 2: 192.168.3.96:9422
                copy 3: 192.168.3.139:9422
[root@bzd  f]# cd ..
[root@bzd  mfs]# /usr/local/mfs/bin/mfssetgoal 2 f
f: 2
[root@bzd  mfs]# cd f/
[root@bzd  f]# ls
1
[root@bzd  f]# touch 2
```

```
[root@bzd f]# echo “222” > 2
[root@bzd f]# /usr/local/mfs/bin/mfsfileinfo 1
1:
        chunk 0: 0000000000000043_00000001 / (id:67 ver:1)
                copy 1: 192.168.3.31:9422
                copy 2: 192.168.3.96:9422
                copy 3: 192.168.3.139:9422
[root@bzd f]# /usr/local/mfs/bin/mfsfileinfo 2
2:
        chunk 0: 0000000000000044_00000001 / (id:68 ver:1)
                copy 1: 192.168.3.31:9422
                copy 2: 192.168.3.96:9422
```

整个目录树的内容摘要可以用一个功能增强的等同于 du –s 的命令 mfsdirinfo，mfsdirinfo 为 MooseFS 列出具体的信息。

例如：

```
$ mfsdirinfo /mnt/mfs-test/test/:
inodes:                         15
  directories:                   4
  files:                         8
chunks:                          6
length:                     270604
size:                       620544
realsize:                  1170432
```

上述内容摘要显示了目录、文件及 chunks 的数目，还有整个目录占用磁盘空间的情况。

length -文件大小的总和

size –块长度总和

realsize –磁盘空间的使用包括所有的拷贝

## 垃圾箱（trash bin）设定隔离的时间（quarantine time）

一个删除文件能够存放在一个“垃圾箱”的时间就是一个隔离时间，这个时间可以用 mfsgettrashtime 命令来验证，也可以用 mfssettrashtime 命令来设置，例如：

```
$ mfsgettrashtime /mnt/mfs-test/test1
/mnt/mfs-test/test1: 604800
$ mfssettrashtime 0 /mnt/mfs-test/test1
```

```
/mnt/mfs-test/test1: 0
$ mfsgettrashtime /mnt/mfs-test/test1
/mnt/mfs-test/test1: 0
```

这些工具也有个递归选项-r，可以对整个目录树操作，例如：

```
$ mfsgettrashtime -r /mnt/mfs-test/test2
/mnt/mfs-test/test2:
files with trashtime                  0 :                      36
directories with trashtime       604800 :                      1
$ mfssettrashtime -r 1209600 /mnt/mfs-test/test2
/mnt/mfs-test/test2:
inodes with trashtime changed:                    37
inodes with trashtime not changed:                0
inodes with permission denied:                     0
$ mfsgettrashtime -r /mnt/mfs-test/test2
/mnt/mfs-test/test2:
files with trashtime                1209600 :                      36
directories with trashtime       1209600 :                      1
```

时间的单位是秒(有用的值有:1 小时是 3600 秒,24 - 86400 秒,1 - 604800 秒)。就像文件被存储的份数一样, 为一个目录 设定存放时间是要被新创建的文件和目录所继承的。数字 0 意味着一个文件被删除后, 将立即被彻底删除，在想回收是不可能的

删除文件可以通过一个单独安装 MFSMETA 文件系统。特别是它包含目录 / trash (包含任然可以被还原的被删除文件的信息)和 / trash/undel (用于获取文件)。只有管理员有权限访问 MFSMETA(用户的 uid 0，通常是 root)。

```
$ mfssettrashtime 3600 /mnt/mfs-test/test1
/mnt/mfs-test/test1: 3600
$ rm /mnt/mfs-test/test1
$ ls    /mnt/mfs-test/test1
ls: /mnt/mfs-test/test1: No such file or directory
# ls -l /mnt/mfs-test-meta/trash/*test1
-rw-r--r-- 1 user          users         1 2007-08-09 15:23 /mnt/mfs-test-meta/trash/00013BC7|test1
```

被删文件的文件名在 “垃圾箱”目录里还可见,文件名由一个八位十六进制的数 i-node 和被删文件的文件名组成，在文件名和 i-node 之间不是用“/”,而是用了“|”替代。如果一个文件名的长度超过操作系统的限制（通常是 255 个字符），那么部分将被删除。通过从挂载点起全路径的文件名被删除的文件任然可以被读写。需要注意的是被删除的文件在用全路径文件名（注意文件名是两部分）时一定要用单引号引起来。例如：

```
# cat '/mnt/mfs-test-meta/trash/00013BC7|test1'
test1
# echo 'test/test2' > '/mnt/mfs-test-meta/trash/00013BC7|test1'
# cat '/mnt/mfs-test-meta/trash/00013BC7|test1'
test/test2
```

移动这个文件到 trash/undel 子目录下，将会使原始的文件恢复到正确的 MooseFS 文件系统上 路径下（如果路径没有改变）。例如：

```
[root@www mfs]# ll dgg
-rw-r--r--  1 root root 8 Jan 13 08:45 dgg
[root@www mfs]# rm -f dgg
[root@www mfs]# ll dgg
ls: dgg: No such file or directory
[root@www trash]# ls
0000000B|dgg    00000047|f1      undel
[root@www trash]# mv '/mnt/mfsmeta/trash/0000000B|dgg' ./undel/
[root@www trash]# ls
undel 00000047|f1
[root@www mfs]# ll dgg
-rw-r--r--  1 root root 8 Jan 13 08:45 dgg
```

注意：如果在同一路径下有个新的同名文件，那么恢复不会成功

从“垃圾箱”中删除文件结果是释放之前被它站用的空间(删除有延迟,数据被异步删除)。在这种被从“垃圾箱”删除的情况下,该文件是不可能恢复了。

可以通过 mfssetgoal 工具来改变文件的拷贝数,也可以通过 mfssettrashtime 工具来改变存储在“垃圾箱”中的时间。

在 MFSMETA 的目录里，除了 trash 和 trash/undel 两个目录外，还有第三个目录 reserved，该目录内有已经删除的文件，但却有一直打开着。在用户关闭了这些被打开的文件后，reserved 目录中的文件将被删除,文件的数据也将被立即删除。在 reserved 目录中文件的命名方法同 trash 目录中的一样，但是不能有其他功能的操作。

# 快照

MooseFS 系统的另一个特征是利用 mfsmakesnapshot 工具给文件或者是目录树做快照，例如：

```
$ mfsmakesnapshot source ... destination
```

Mfsmakesnapshot 是在一次执行中整合了一个或是一组文件的拷贝，而且任何修改这些文件的源文件都不会影响到源文件的快照， 就是说任何对源文件的操作,例如写入源文件，将不会修改副本(或反之亦然)。

文件快照可以用 mfsappendchunks，就像 MooseFS1.5 中的 mfssnapshot 一样，，作为选择，二者都可以用。例如：

$ mfsappendchunks destination-file source-file ...

当有多个源文件时，它们的快照被加入到同一个目标文件中（每个 chunk 的最大量是 chunk）。

## 额外的属性

文件或目录的额外的属性(noowner, noattrcache, noentrycache)，可以被 mfsgeteattr，mfsseteattr，mfsdeleattr 工具检查，设置，删除，其行为类似 mfsgetgoal/mfssetgoal or 或者是 mfsgettrashtime/mfssettrashtime，详细可见命令手册。

# 维护 MooseFS

## 启动 MooseFS 集群

最安全的启动 MooseFS 集群（避免任何读或写的错误数据或类似的问题）的方式是按照以下命令**步骤：**

1.启动 mfsmaster 进程

2.启动所有的 mfschunkserver 进程

3.启动 mfsmetalogger 进程（如果配置了 mfsmetalogger ）

当所有的 chunkservers 连接到 MooseFS master 后，任何数目的客户端可以利用 mfsmount 去挂接被 export 的文件系统。（可以通过检查 master 的日志或是 CGI 监视器来查看是否所有的 chunkserver 被连接 )。

# 停止 MooseFS 集群

**安全的停止MooseFS 集群：**

1.在所有的客户端卸载 MooseFS 文件系统（用 umount 命令或者是其它等效的命令）

2.用 mfschunkserver –s 命令停止 chunkserver 进程

3.用 mfsmetalogger –s 命令停止 metalogger 进程

4.用 mfsmaster –s 命令停止 master 进程

# MooseFS chunkservers 的维护

假如每个文件的 goal（目标）都不小于 2，并且没有 under-goal 文件（这些可以用 mfsgetgoal –r 和 mfsdirinfo 命令来检查），那么一个单一的 chunkserver 在任何时刻都可能做停止或者是重新启动。以后每当需要做停止或者是重新启动另一个 chunkserver 的时候，要确定之前的 chunkserver 被连接，而且要没有 under-goal chunks。

# MooseFS 元数据的备份

**通常元数据有两部分的数据**

1.主要元数据文件 metadata.mfs，当 mfsmaster 运行的时候会被命名为 metadata.mfs.back

2.元数据改变日志 changelog.*.mfs ,存储了过去的 N 小时的文件改变( N 的数值是由 BACK_LOGS 参数设置的，参数的设置在 mfschunkserver.cfg 配置文件中)。

主要的元数据文件需要定期备份，备份的频率取决于取决于多少小时 changelogs 储存。元数据 changelogs 应该实时的自动复制。自从 MooseFS 1.6.5,这两项任务是由 mfsmetalogger 守护进程做的。

# MooseFS master 的恢复

一旦 mfsmaster 崩溃（例如因为主机或电源失败），需要最后一个元数据日志 changelog 并入主要的 metadata 中。这个操作时通过 mfsmetarestore 工具做的，最简单的方法是：

```
mfsmetarestore -a
```

如果 master 数据被存储在 MooseFS 编译指定地点外的路径，则要利用-d 参数指定使用，如：

```
mfsmetarestore -a -d /storage/mfsmaster
```

# 从备份恢复 MooseFS master

**为了从备份中恢复一个master，需要做：**

1、安装一个 mfsmaster

2、利用同样的配置来配置这台 mfsmaster（利用备份来找回 mfsmaster.cfg），可见配置文件也是需要备份的。

3、找回 metadata.mfs.back 文件，可以从备份中找，也可以中 metalogger 主机中找（如果启动了 metalogger 服务），然后把 metadata.mfs.back 放入 data 目录，一般为${prefix}/var/mfs。

4、从在 master 宕掉之前的任何运行 metalogger 服务的服务器上拷贝最后 metadata 文件，然后放入 mfsmaster 的数据目录。

5、利用 mfsmetarestore 命令合并元数据 changelogs ，可以用自动恢复模式 mfsmetarestore −a，也可以利用非自动化恢复模式，语法如下：

```
mfsmetarestore -m metadata.mfs.back -o metadata.mfs changelog_ml.*.mfs
```