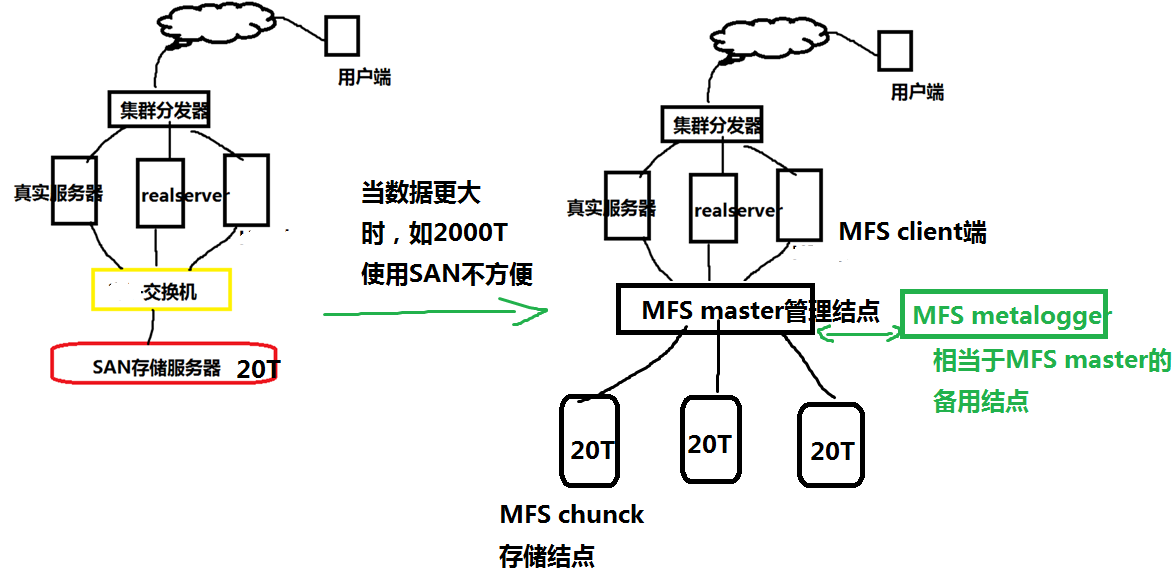
# MFS分布式文件系统

mooseFS（moose 驼鹿）是一款网络分布式文件系统。它把数据分散在多台服务器上，但对于用户来讲，看到的只是一个源。MFS也像其他类unix文件系统一样，包含了层级结构（目录树），存储着文件属性（权限，最后访问和修改时间），可以创建特殊的文件（块设备，字符设备，管道，套接字），符号链接，硬链接。

MooseFS[MFS]是一个具有容错性的网络分布式文件系统。它把数据分散存放在多个物理服务器上，而呈现给用户的则是一个统一的资源。

## SAN和MFS比较

****

当我们存储服务器的容量达到瓶颈之后，那我们就需要采用分布式的文件系统来扩大存储容量，类似与百度云的存储，也是采用分布式存储!

## MFS的特征

1：层析结构（目录树）

2：存储文件属性（权限，访问和修改时间）

3：支持特殊文件（块设备，字符设备，管道）

4：符号链接，软硬链接

5：对文件系统访问可以通过IP地址或者密码进行访问限制

6：高可靠（数据的多个拷贝存储在不同的计算机上）

7：通过附加新的计算机或者硬盘可以实现容量的动态拓展

8：删除文件可以根据一个可配置的时间周期进行保留

9：不受访问和写入影响的文件连贯快照

**应用场景**

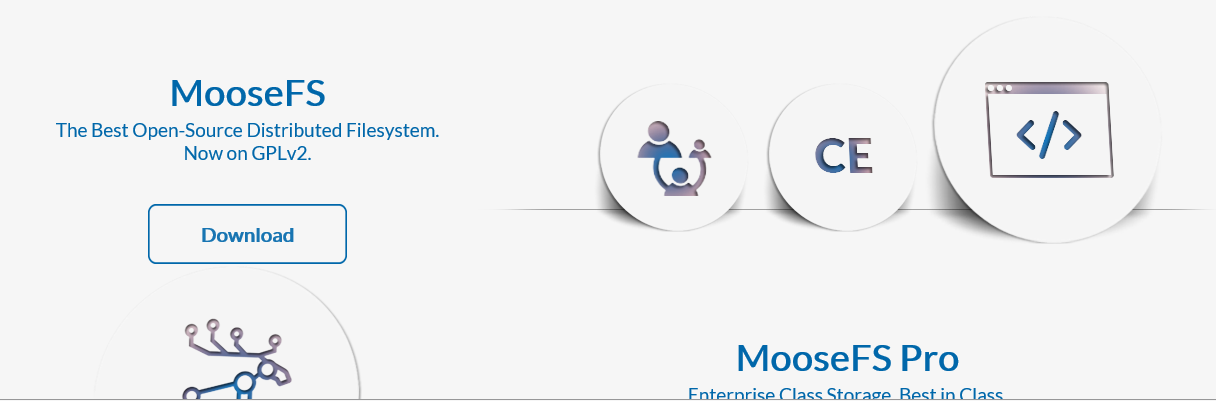
谈及MooseFS的应用场景，其实就是去谈分布式文件系统的应用场景。

1）大规模高并发的数据存储及访问（小文件、大文件），

2）大规模的数据处理，如日志分析

## MooseFS介绍

<http://www.moosefs.com/>



<http://www.moosefs.org/reference-guide.html> //安装和帮助文档

## MFS分布式文件系统部署方案

MooseFS 是一种分布式文件系统，MooseFS 文件系统结构包括以下四种角色：

1 管理服务器 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

2元数据日志服务器 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

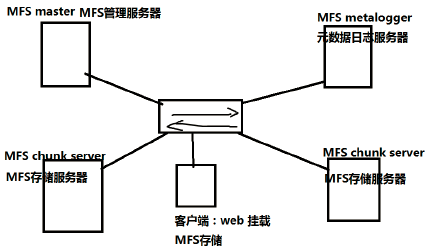
3 数据存储服务器 data servers (chunk servers ) ：

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.。 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 客户机挂载使用 client computers ：

客户端挂载远程mfs服务器共享出的存储并使用。 通过 fuse 内核接口挂载进程管理服务器上所管理的数据存储服务器共享出的硬盘。 共享的文件系统的用法和 nfs 相似。 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。

**系统部署方案**

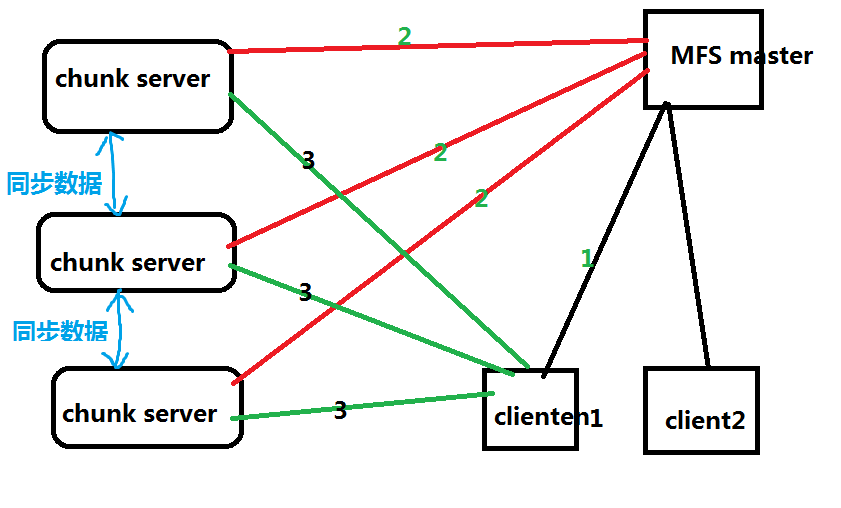


### 内部运行机制

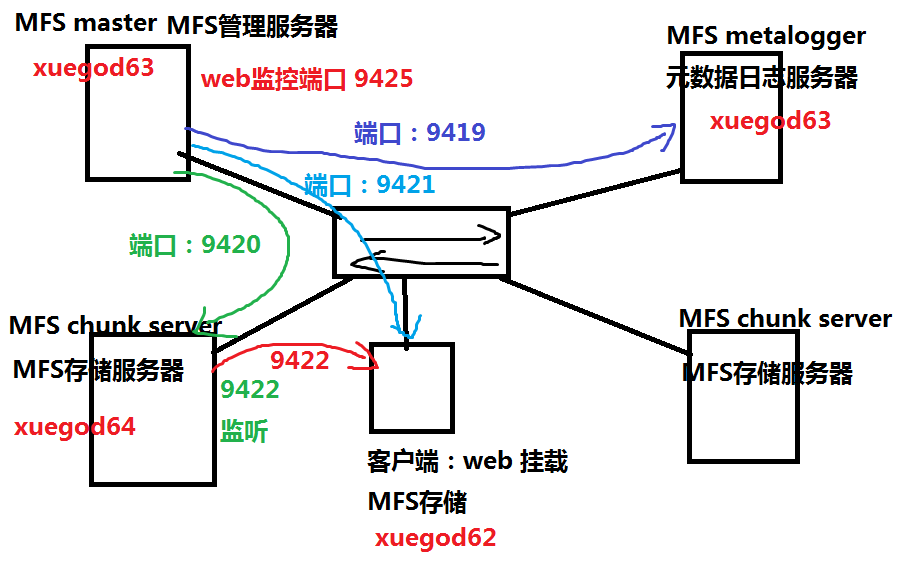
1：客户端请求访问存储，请求发送到了MFS Master

2：MFS Master根据我们的请求，查询所需要的文件分布在那些服务器上

3：客户端直接和存储服务器进行数据存储和读写

****

端口关系图

****

端口号：

9420 ： MFS master和MFS chunck通信端口

9421 ： MFS master和MFS Client端通信端口

9419 ： MFS master和MFS metalogger端通信端口

9422 ： MFS chunck 和MFS Client端通信端口

9425 ： MFS master web界面监听端口，查看整体运行状态

## MFS分布式文件实验环境

**服务器 角色**

**xuegod63 管理服务器managing server (master) 和元数据日志服务器（Metalogger）**

**xuegod64 数据存储服务器data servers (chunk servers )**

**xuegod62 客户机挂载使用client computers**

**安装环境准备：**

所有服务器端和客户端都要支持fuse内核模块。

由于mfs的客户端程序也就是加载mfs磁盘系统的命令是使用了fuse，因此只要是想挂载mfs的服务器，必要的前提条件就是先安装fuse，这样编译mfs的时候才能顺利通过。

**fuse概述：**

用户空间文件系统（Filesystem in Userspace，简称FUSE）是操作系统中的概念，指完全在用户态实现的文件系统。目前Linux通过内核模块对此进行支持。一些文件系统如NTFS-3G（作用：让Linux中对NTFS文件系统提供支持） 使用FUSE实现。

注：2.6以上内核，都自带fuse

## 部署MFS分布式文件系统

部署MFS架构，我们按照上面的拓扑，从Master开始部署

### MFS管理服务器

[root@xuegod63 ~]# yum install rpm-build gcc gcc-c++ fuse-devel zlib-devel –y

[root@xuegod63 ~]# useradd -s /sbin/nologin mfs #运行mfs的用户

[root@xuegod63 ~]# unzip moosefs-master.zip.zip && cd moosefs-master

[root@xuegod63 moosefs-master]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs

[root@xuegod63 moosefs-master]# make -j 4 && make install

[root@xuegod63 moosefs-master]# cd /usr/local/mfs/

[root@xuegod63 mfs]# ls #看下安装完成生成的目录

bin #客户端工具

etc #服务器的配置文件都放在该目录中

sbin #服务端启动程序。如：元数据服务器端程序 mfsmaster、数据存储服务器端服务程序mfschunkserver

share #文档

var #元数据目录（可在配置文件中自定义到其他目录）

生成配置文件

[root@xuegod63 etc]# cd /usr/local/mfs/etc/mfs/

[root@xuegod63 mfs]# ll

**-**rw-r--r-- 1 root root 531 Dec 24 14**:**13 mfschunkserver.cfg.sample

**-**rw-r--r-- 1 root root 4060 Dec 24 14**:**13 mfsexports.cfg.sample

**-**rw-r--r-- 1 root root 57 Dec 24 14**:**13 mfshdd.cfg.sample

**-**rw-r--r-- 1 root root 1020 Dec 24 14**:**13 mfsmaster.cfg.sample

**-**rw-r--r-- 1 root root 417 Dec 24 14**:**13 mfsmetalogger.cfg.sample

**-**rw-r--r-- 1 root root 404 Dec 24 14**:**13 mfsmount.cfg.sample

**-**rw-r--r-- 1 root root 1123 Dec 24 14**:**13 mfstopology.cfg.sample

[root@xuegod63 mfs]# cp mfsmaster.cfg.sample mfsmaster.cfg //Master配置文件

[root@xuegod63 mfs]# cp mfsexports.cfg.sample mfsexports.cfg //输出目录配置文件

[root@xuegod63 mfs]# cp mfsmetalogger.cfg.sample mfsmetalogger.cfg //元数据日志

[root@xuegod63 mfs]# cd /usr/local/mfs/var/mfs/

[root@xuegod63 mfs]# cp metadata.mfs.empty metadata.mfs //首次安装 master 时，会自动生成一个名为 metadata.mfs.empty 的元数据文件 metadata,该文件是空的。MooseFS master 运必须有文件 metadata.mfs

[root@xuegod63 mfs]# cd /usr/local/mfs/etc/mfs/

配置文件了解

[root@xuegod63 mfs]# vim mfsmaster.cfg //这个文件不需要修改，但你需要了解下面这几个字段

23 # MATOCS\_LISTEN\_HOST = \* #监听所有 IP，也直接写一个 IP 地址，如 192.168.1.63

24 # MATOCS\_LISTEN\_PORT = 9420

11 # DATA\_PATH = /usr/local/mfs/var/mfs

注：这个配置文件中所有注掉的设置都是默认的配置。

如果有需要还可以修改 DATA\_PATH 的设置将元数据目录存储到其他的分区或磁盘。其他的参数都很简单根据需要调整即可。

master 会打开 9420 端口等待 mfschunkserver 数据存储服务器连接

[root@xuegod63 mfs]# chown -R mfs:mfs /usr/local/mfs/

[root@xuegod63 mfs]# /usr/local/mfs/sbin/mfsmaster start //各个端口开始起来

working directory**:** **/**usr**/**local**/**mfs**/**var**/**mfs

lockfile created and locked

initializing mfsmaster modules **...**

loading sessions **...** file not found

**if** it is not fresh installation then you have to restart all active mounts **!!!**

exports file has been loaded

mfstopology configuration file **(/**usr**/**local**/**mfs**/**etc**/**mfstopology.cfg**)** not found **-** using defaults

loading metadata **...**

create new empty filesystemmetadata file has been loaded

no charts data file **-** initializing empty charts

master **<->** metaloggers module**:** listen on **\*:**9419

master **<->** chunkservers module**:** listen on **\*:**9420

main master server module**:** listen on **\*:**9421

mfsmaster daemon initialized properly

[root@xuegod63 mfs]# netstat -antup | grep 94\*

tcp 0 0 0**.**0**.**0**.**0**:**9419 0**.**0**.**0**.**0**:\*** LISTEN 49618**/**mfsmaster

tcp 0 0 0**.**0**.**0**.**0**:**9420 0**.**0**.**0**.**0**:\*** LISTEN 49618**/**mfsmaster

tcp 0 0 0**.**0**.**0**.**0**:**9421 0**.**0**.**0**.**0**:\*** LISTEN 49618**/**mfsmaster

[root@xuegod63 mfs]# echo "/usr/local/mfs/sbin/mfsmaster start" >> /etc/rc.local

[root@xuegod63 mfs]# chmod +x /etc/rc.local

[root@xuegod63 mfs]# /usr/local/mfs/sbin/mfsmaster stop //服务的关闭方法

[root@xuegod63 mfs]# ll /usr/local/mfs/var/mfs/ //查看生成的日志文件

**-**rw-r----- 1 mfs mfs 95 Dec 24 14**:**30 metadata.mfs

**-**rw-r----- 1 mfs mfs 95 Dec 24 14**:**28 metadata.mfs.back.1

**-**rw-r--r-- 1 mfs mfs 8 Dec 24 14**:**13 metadata.mfs.empty

**-**rw-r----- 1 mfs mfs 10 Dec 24 14**:**28 sessions.mfs

**-**rw-r----- 1 mfs mfs 762516 Dec 24 14**:**30 stats.mfs

指定需要共享的权限

[root@xuegod63 mfs]# cd **/**usr**/**local**/**mfs**/**etc**/**mfs

[root@xuegod63 mfs]# vim mfsexports.cfg

# Allow everything but "meta".

**\*** **/** rw**,**alldirs**,**maproot**=**0

# Allow "meta".

**\*** **.** rw #下面添加两行

192**.**168**.**1**.**64**/**24 **/** rw**,**alldirs**,**maproot**=**0

192**.**168**.**1**.**0**/**24 **/** rw**,**alldirs**,**mapproot**=**0

mfsexports.cfg 文件每一条由三部分组成，第一部分表示客户端的 IP 地址，第二部分表示被挂接的目录，第三部分表示客户端拥有的权限，下面对三给部分迚行解释

1**.**客户端 IP 地址

**\*** 所有 IP 地址

x.x.x.x 单个 IP 地址

x.x.x.x**/**m.m.m.m IP 网络地址**/**子网掩码

f.f.f.f-t.t.t.t IP 段

2**.**被挂载的目录

**/** 表示 MooseFS 的根

**.** 表示 MFSMETA 文件系

3**.**客户端拥有的权限

ro 只读

rw 读写

alldirs 允许挂载任何指定的子目录

maproot 映射为 root 用户还是指定的用户

password 指定客户端密码

[root@xuegod63 mfs]# sh /etc/rc.local //启动服务

### 安装MFS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。

[root@xuegod63 mfs]# cd

[root@xuegod63 ~]# rm -rf moosefs-master #元数据又是另外一台服务器，这里我们在一台机器上做，那么我们也要重新编译

[root@xuegod63 ~]# unzip moosefs-master.zip && cd moosefs-master #编译的时候，用户在若在其他设备，需要新建运行mfsmetalogger角色的用户，xuegod63上在安装master的时候，已经新建，略过，其他机器上，应该执行useradd -s /sbin/nologin mfs

[root@xuegod63 moosefs-master]# ./configure --prefix=/usr/local/mfsmeta --with-default-user=mfs --with-default-group=mfs

[root@xuegod63 moosefs-master]# make && make install

[root@xuegod63 moosefs-master]# cd /usr/local/mfsmeta/etc/mfs/

[root@xuegod63 mfs]# cp mfsmetalogger.cfg.sample mfsmetalogger.cfg

[root@xuegod63 mfs]# vim mfsmetalogger.cfg

# META\_DOWNLOAD\_FREQ = 24

#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 戒 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\_HOST **=** 192**.**168**.**1**.**63 #此行原来注释了，修改，并启用

#MASTER\_HOST，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

[root@xuegod63 mfs]# chown -R mfs:mfs /usr/local/mfsmeta/

启动元数据日志服务器

[root@xuegod63 mfs]# /usr/local/mfsmeta/sbin/mfsmetalogger start

working directory**:** **/**usr**/**local**/**mfsmeta**/**var**/**mfs

lockfile created and locked

initializing mfsmetalogger modules **...**

mfsmetalogger daemon initialized properly

[root@xuegod63 mfs]# echo "/usr/local/mfsmeta/sbin/mfsmetalogger start" >> /etc/rc.local

服务关闭方法

[root@xuegod63 mfs]# /usr/local/mfsmeta/sbin/mfsmetalogger stop

查看端口

[root@xuegod63 mfs]# /usr/local/mfsmeta/sbin/mfsmetalogger start

[root@xuegod63 mfs]# netstat -anput | grep 9419

tcp 0 0 0.0.0.0:9419 0.0.0.0: LISTEN 7266/sbin/mfsmaster

tcp 0 0 192.168.1.63:9419 192.168.1.63:49776 ESTABLISHED 7266/sbin/mfsmaster到此为止，我们Master和Metalogger服务器安装完成，且在一台机器上

### Chunk Server安装配置

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存储数据时，是在一个普通的文件系统如ext4上储存数据块或碎片(chunks/fragments)作为文件。 你在chunkserver上看不到完整的文件。

[root@xuegod64 ~]# useradd -s /sbin/nologin mfs

[root@xuegod64 ~]# unzip moosefs-master.zip

[root@xuegod64 ~]# cd moosefs-master && ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs && make && make install

配置文件修改

[root@xuegod64 mfs]# cp mfschunkserver.cfg.sample mfschunkserver.cfg#主配置文件

[root@xuegod64 mfs]# cp mfshdd.cfg.sample mfshdd.cfg #使用的磁盘空间配置文件

[root@xuegod64 mfs]# vim mfschunkserver.cfg

MASTER\_HOST **=** 192**.**168**.**1**.**63 #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 #可以启用它，也可以不启用，不启用，默认就是9420

# 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

# HDD\_CONF\_FILENAME = /usr/local/mfs/etc/mfs/mfshdd.cfg #分配给 MFS 使用的磁盘空间配置文件的位置

[root@xuegod64 mfs]# vim mfshdd.cfg

# mount points of HDD drives

#

#/mnt/hd1

#/mnt/hd2

#etc.

**/**tmp

#在这里/tmp 是一个给 mfs 的分区，但在生产环境是一个独立的磁盘的挂载目录

[root@xuegod64 mfs]# chown -R mfs:mfs /usr/local/mfs/

服务器启动关闭

[root@xuegod64 mfs]# /usr/local/mfs/sbin/mfschunkserver start

working directory**:** **/**usr**/**local**/**mfs**/**var**/**mfs

lockfile created and locked

initializing mfschunkserver modules **...**

hdd space manager**:** path to scan**:** **/**tmp**/**

hdd space manager**:** start background hdd scanning **(**searching for available chunks**)**

main server module**:** listen on **\*:**9422

no charts data file **-** initializing empty charts

mfschunkserver daemon initialized properly

[root@xuegod64 mfs]# echo "/usr/local/mfs/sbin/mfschunkserver start" >> /etc/rc.local

[root@xuegod64 mfs]# chmod +x /etc/rc.local

[root@xuegod64 mfs]# ls /tmp #分块存储，人工时无法认识的

00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

01 12 23 34 45 56 67 78 89 9A AB BC CD DE EF keyring-J8XeaN

02 13 24 35 46 57 68 79 8A 9B AC BD CE DF F0 orbit-gdm

03 14 25 36 47 58 69 7A 8B 9C AD BE CF E0 F1 orbit-root

04 15 26 37 48 59 6A 7B 8C 9D AE BF D0 E1 F2 pulse-oErxG6JvcJR1

05 16 27 38 49 5A 6B 7C 8D 9E AF C0 D1 E2 F3 pulse-ZvGMw7hIpLaC

06 17 28 39 4A 5B 6C 7D 8E 9F B0 C1 D2 E3 F4 vgauthsvclog.txt.0

07 18 29 3A 4B 5C 6D 7E 8F A0 B1 C2 D3 E4 F5 virtual-root.qMLdFF

08 19 2A 3B 4C 5D 6E 7F 90 A1 B2 C3 D4 E5 F6 VMwareDnD

09 1A 2B 3C 4D 5E 6F 80 91 A2 B3 C4 D5 E6 F7 vmware-root

0A 1B 2C 3D 4E 5F 70 81 92 A3 B4 C5 D6 E7 F8

0B 1C 2D 3E 4F 60 71 82 93 A4 B5 C6 D7 E8 F9

0C 1D 2E 3F 50 61 72 83 94 A5 B6 C7 D8 E9 FA

0D 1E 2F 40 51 62 73 84 95 A6 B7 C8 D9 EA FB

0E 1F 30 41 52 63 74 85 96 A7 B8 C9 DA EB FC

0F 20 31 42 53 64 75 86 97 A8 B9 CA DB EC FD

10 21 32 43 54 65 76 87 98 A9 BA CB DC ED FE

[root@xuegod64 mfs]# /usr/local/mfs/sbin/mfschunkserver stop #关闭方法

### 配置客户端

[root@xuegod62 ~]# yum install rpm-build gcc gcc-c++ fuse-devel zlib-devel -y

[root@xuegod62 ~]# useradd -s /sbin/nologin mfs

[root@xuegod62 ~]# unzip moosefs-master.zip && cd moosefs-master

[root@xuegod62 moosefs-master]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs --enable-mfsmount

#开启mfsmount必须有，作为客户端必须有的这一个选项。

[root@xuegod62 moosefs-master]# make -j 2 && make install

[root@xuegod62 moosefs-master]# mkdir /mfs

[root@xuegod62 moosefs-master]# lsmod | grep fuse

fuse 73530 4

[root@xuegod62 moosefs-master]# modprobe fuse #若没有，加载一下

[root@xuegod62 moosefs-master]# ln -s /usr/local/mfs/bin/mfsmount /usr/bin/mfsmount

将mfs共享出来的存储空间挂载到xuegod62的/ /mfs目录下：

[root@xuegod62 moosefs-master]# mfsmount /mfs/ -H 192.168.1.63 -p

MFS Password**:** #无密码 直接回车

mfsmaster accepted connection with parameters**:** read-write**,**restricted\_ip **;** root mapped to root**:**root

[root@xuegod62 moosefs-master]# df –h

……

192**.**168**.**1**.**63**:**9421 5**.**3G 0 5**.**3G 0**%** **/**mfs

[root@xuegod62 moosefs-master]# echo "modprobe fuse" >> /etc/rc.local

[root@xuegod62 moosefs-master]# echo "/usr/local/mfs/bin/mfsmount /mfs -H 192.168.1.63" >> /etc/rc.local

[root@xuegod62 moosefs-master]# chmod +x /etc/rc.local

### 测试

[root@xuegod64 mfs]# yum install -y tree #64上用tree监控状态

[root@xuegod64 mfs]# tree /tmp/

**/**tmp**/**

├── 00

├── 01

├── 02

├── 03

├── 04

├── 05

├── 06

├── 07

├── 08

├── 09

├── 0A

├── 0B

├── 0C

├── 0D

├── 0E

[root@xuegod62 moosefs-master]# cp -r /boot/\* /mfs/ #客户端写入数据

[root@xuegod64 mfs]# tree /tmp/ #存储服务器监控

**/**tmp**/**

├── 00

├── 01

│   └── chunk\_0000000000000001\_00000001.mfs

├── 02

│   └── chunk\_0000000000000002\_00000001.mfs

├── 03

│   └── chunk\_0000000000000003\_00000001.mfs

├── 04

│   └── chunk\_0000000000000004\_00000001.mfs

├── 05

│   └── chunk\_0000000000000005\_00000001.mfs

[root@xuegod62 moosefs-master]# ls /mfs #客户端是看得到的文件内容，而在我们的Chunk Server上看到的都是一些碎片

config-2.6.32-431.el6.x86\_64 lost+found

efi symvers-2.6.32-431.el6.x86\_64.gz

grub System.map-2.6.32-431.el6.x86\_64

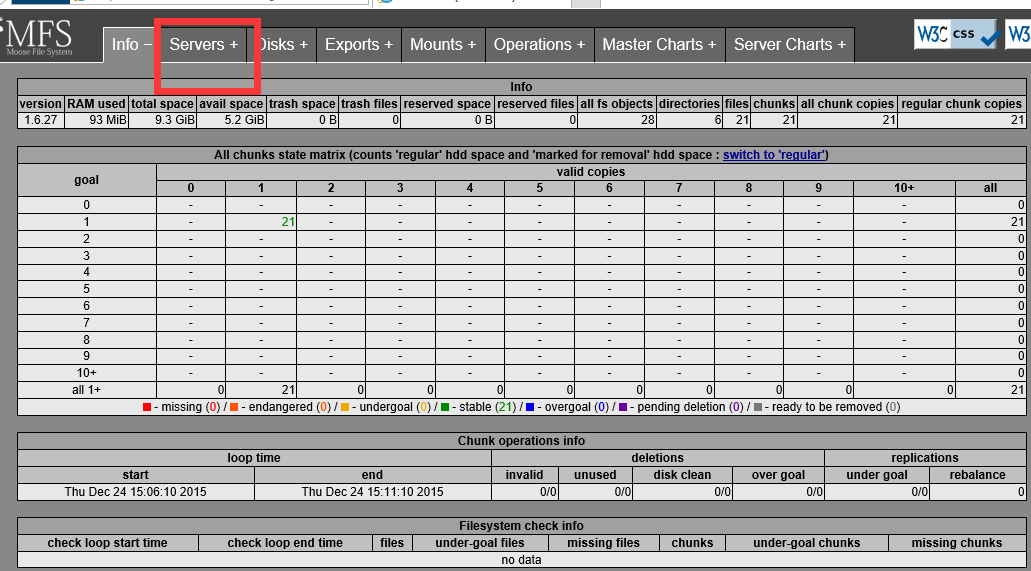
initramfs-2.6.32-431.el6.x86\_64.img vmlinuz-2.6.32-431.el6.x86\_64

### 配置web监控

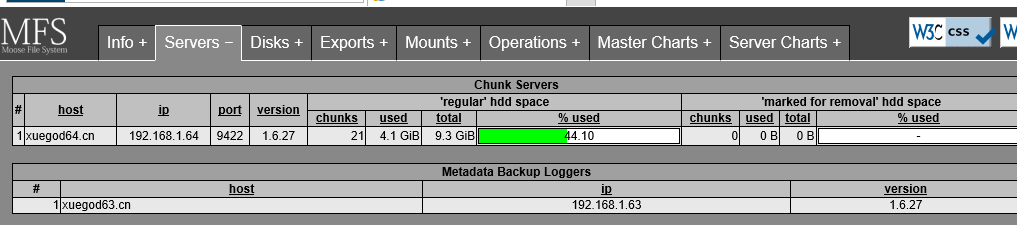
主用来监控 MFS 各节点状态信息，可部署在任一节点服务器。

[root@xuegod63 mfs]# /usr/local/mfs/sbin/mfscgiserv #启动网页监控

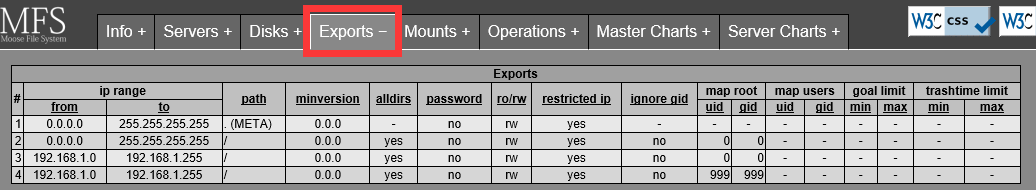
浏览器：<http://192.168.1.63:9425/>（如需指定服务器ip修改share/mfscgi/mfs.cgi）



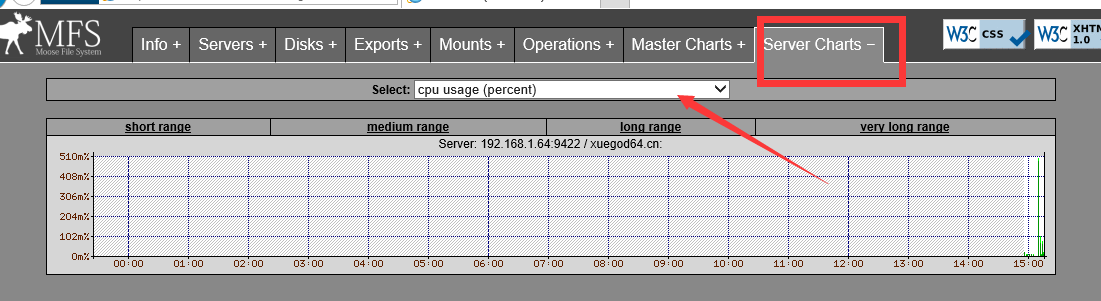
磁盘使用



权限



CPU资源



## 拓展

关于密码设置和添加一台存储设置以及复制份数

### 配置密码

[root@xuegod63 ~]# cd /usr/local/mfs/etc/mfs/

[root@xuegod63 mfs]# vim mfsexports.cfg //权限后面跟上密码即可

# Allow everything but "meta".

**\*** **/** rw**,**alldirs**,**maproot**=**0

# Allow "meta".

**\*** **.** rw

192**.**168**.**1**.**64**/**24 **/** rw**,**alldirs**,**maproot**=**0**,**password**=**123456

192**.**168**.**1**.**0**/**24 **/** rw**,**alldirs**,**mapproot**=**0

### 添加Chunk Server

开启一台新主机并添加为Chunk Server

[root@pualinux11 ~]# yum install rpm-build gcc gcc-c++ fuse-devel zlib-devel –y

[root@pualinux11 ~]# useradd -s /sbin/nologin mfs

[root@pualinux11 ~]# unzip moosefs-master.zip && cd moosefs-master

[root@pualinux11 moosefs-master]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs && make && make install

[root@pualinux11 moosefs-master]# cd /usr/local/mfs/etc/mfs/

[root@pualinux11 mfs]# cp mfschunkserver.cfg.sample mfschunkserver.cfg

[root@pualinux11 mfs]# vim mfschunkserver.cfg

MASTER\_HOST **=** 192**.**168**.**1**.**63 #修改管理IP

MASTER\_PORT **=** 9420 #去掉注释

[root@pualinux11 mfs]# cp mfshdd.cfg.sample mfshdd.cfg

[root@pualinux11 mfs]# vim mfshdd.cfg

# mount points of HDD drives

#

#/mnt/hd1

#/mnt/hd2

#etc.

**/**opt #添加一行

[root@pualinux11 mfs]# chown -R mfs:mfs /usr/local/mfs/

[root@pualinux11 mfs]# chown -R mfs:mfs /opt/

[root@pualinux11 mfs]# /usr/local/mfs/sbin/mfschunkserver start

working directory**:** **/**usr**/**local**/**mfs**/**var**/**mfs

lockfile created and locked

initializing mfschunkserver modules **...**

hdd space manager**:** path to scan**:** **/**opt**/**

hdd space manager**:** start background hdd scanning **(**searching for available chunks**)**

main server module**:** listen on **\*:**9422

no charts data file **-** initializing empty charts

mfschunkserver daemon initialized properly

[root@pualinux11 mfs]# yum install -y tree

[root@xuegod63 ~]# /usr/local/mfs/sbin/mfsmaster stop #重启管理

[root@xuegod63 ~]# /usr/local/mfs/sbin/mfsmaster start

[root@xuegod63 ~]# /usr/local/mfsmeta/sbin/mfsmetalogger stop #元数据日志

[root@xuegod63 ~]# /usr/local/mfsmeta/sbin/mfsmetalogger start

[root@xuegod62 moosefs-master]# cp /etc/passwd /mfs/ #写入数据

[root@xuegod62 moosefs-master]# cp /etc/hosts /mfs/

[root@pualinux11 mfs]# tree /opt/ #看到也有数据过来了

/tmp/

├── 00

│   ├── chunk\_0000000000000001\_00000001.mfs

│   ├── chunk\_0000000000000002\_00000001.mfs

│   ├── chunk\_0000000000000003\_00000001.mfs

│   ├── chunk\_0000000000000004\_00000001.mfs

│   ├── chunk\_0000000000000005\_00000001.mfs

│   ├── chunk\_0000000000000006\_00000001.mfs

│   ├── chunk\_0000000000000007\_00000001.mfs

│   ├── chunk\_0000000000000008\_00000001.mfs

│   ├── chunk\_0000000000000009\_00000001.mfs

├── 16

│   └── chunk\_0000000000000016\_00000001.mfs

├── 17

### 设置复制份数

[root@xuegod62 moosefs-master]# cd /usr/local/mfs/bin/ #客户端

[root@xuegod62 bin]# ./mfssetgoal -r 2 /mfs/ #设置2份

[root@xuegod62 bin]# ./mfsgetgoal /mfs # 查看份数

**/**mfs**:** 2

查看复制份数，都保存在哪儿：

[root@xuegod62 bin]# ./mfsfileinfo /mfs/initramfs-2.6.32-431.el6.x86\_64.img

**/**mfs**/**initramfs-2.6.32-431.el6.x86\_64.img**:**

chunk 0**:** 0000000000000012\_00000001 **/** **(**id**:**18 ver**:**1**)**

copy 1**:** 192**.**168**.**1**.**11**:**9422

copy 2**:** 192**.**168**.**1**.**64**:**9422

停掉XueGod64

[root@xuegod62 mfs]# cp /etc/group /mfs/

[root@xuegod62 mfs]# ls //文件仍然可写

config-2.6.32-431.el6.x86\_64 lost+found

efi passwd

group symvers-2.6.32-431.el6.x86\_64.gz

grub System.map-2.6.32-431.el6.x86\_64

hosts vmlinuz-2.6.32-431.el6.x86\_64

initramfs-2.6.32-431.el6.x86\_64.img

### 回收站清空时间

**修改MFS文件删除延迟时间（都在client端下操作）**

[root@xuegod62 ~]# cd /usr/local/mfs/bin

[root@xuegod62 bin]# ./mfssettrashtime 600 /mnt/mfs/

#回收站延迟清空时间，以秒计算，设置文件或目录的删除时间。

**查看MFS文件删除延迟时间**

[root@xuegod62 bin]# ./mfsgettrashtime /mnt/mfs

/mnt/mfs: 3600 #为什么不是600?

因为设置的时间是按照小时计算，设置的单位是秒，不满一小时就按一小时计算

### mfsmaster配置文件

# WORKING\_USER = mfs #运行master server 用户

# WORKING\_GROUP = mfs #运行master server 组

# SYSLOG\_IDENT = mfsmaster #master server 在syslog中的标识，说明是由master产生的

# LOCK\_MEMORY = 0 #是否执行mlockall（）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-19；注意：进程必须是用root启动）

# EXPORTS\_FILENAME = /usr/local/mfs/etc/mfsexports.cfg #被挂接的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，此目录分三类文件，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）

# MATOCS\_LISTEN\_HOST = \* #metalogger 监听的ip地址（默认是\*，代表任何ip）

# MATOCS\_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 = \* #用于客户端挂接的ip地址（默认\*，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9421）

# CHUNKS\_LOOP\_TIME = 300 #chunksde 回环频率（默认是：300秒）

# CHUNKS\_WRITE\_REP\_LIMIT = 1 #在一个循环里复制到一个chunkserver的最大chunk数目

# CHUNKS\_READ\_REP\_LIMIT = 5 #在一个循环里从一个chunkserver复制的最大chunk数目

#CHUNKS\_READ\_REP\_LIMIT = 0 弹出低于1.6.0的客户端挂接（0或1，默认是0

### 集群启动和关闭循序

MFS集群启动：

1 启动master server

2 启动chunk server

3 启动metalogger

4 启动客户端，使用mfsmount挂载相关目录

MFS集群停止：

1 所有客户端卸载MooseFS文件系统

2 停止chunkserver

3 停止metalogger

4 停止master server