第四十八章：MFS 分布式文件系统

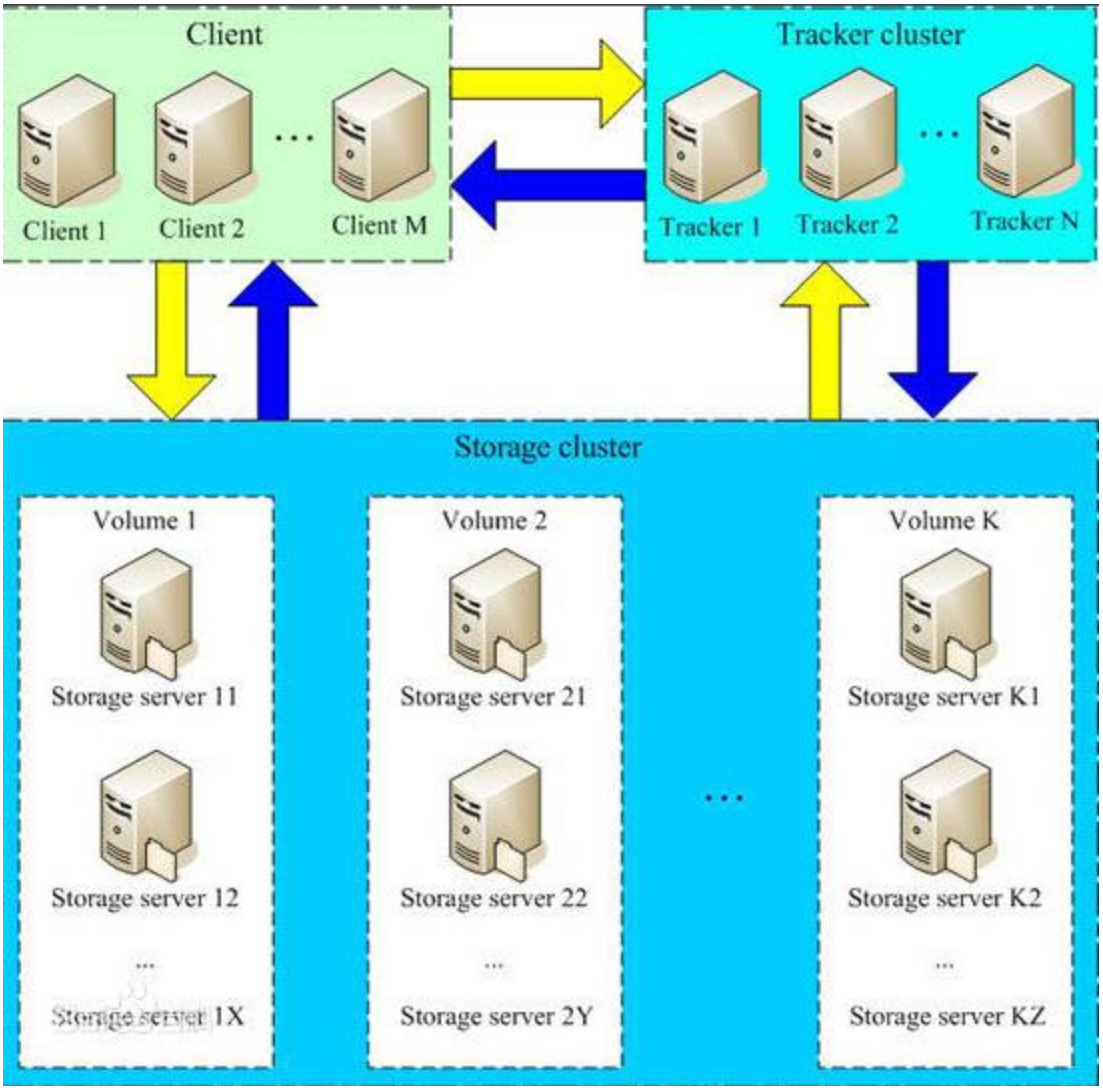
一、分布式文件系统；
二、MFS 分布式文件系统；
三、MFS 分布式文件系统存储架构；
四、案例：搭建 MFS 分布式存储；

一、分布式文件系统：

背景：计算机通过文件系统管理、存储数据，而信息爆炸时代中人们可以获取的数据成指数倍的增长，单纯通过增加硬盘个数来扩展计算机文件系统的存储容量的方式，在容量大小、容量增长速度、数据备份、数据安全等方面的表现都差强人意。分布式文件系统可以有效解决数据的存储和管理难题，人们在使用分布式文件系统时，无需关心数据是存储在哪个节点上、或者是从哪个节点从获取的，只需要像使用本地文件系统一样管理和存储文件系统中的数据。

概述：分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连。分布式文件系统的设计基于客户机/服务器模式。一个典型的网络可能包括多个供多用户访问的服务器；

目前常见的分布式文件系统有很多种，比如 Hadoop、Moosefs、HDFS、FastDFS、Lustre、TFS、GFS 等等一系列；



二、MFS 分布式文件系统：

概述：MooseFS（即 Moose File System）是一个具有容错性的网络分布式文件系统，它将数据分散存放在多个物理服务器或单独磁盘或分区上，确保一份数据有多个备份副本，对于访问 MFS 的客户端或者用户来说，整个分布式网络文件系统集群看起来就像一个资源一样，也就是说呈现给用户的是一个统一的资源。MFS 支持 FUSE（用户空间文件系统 Filesystem in Userspace），客户端挂载后可以作为一个普通的 Unix 文件系统使用 MooseFS；

优势：

1.部署简单，轻量、易配置、易维护；

2.易于扩展，支持在线扩容，不影响业务，体系架构可伸缩性极强；

3.通用文件系统，不需要修改上层应用就可以使用；

4.可设置文件备份的副本数量，一般建议 3 份，未来硬盘容量也要是存储单份的容量的三倍；

劣势：

1.master 目前是单点（虽然会把数据信息同步到备份服务器，但是恢复需要时间，因此，会影响上线，针对这个问题；可以通过 DRBD+Keeaplived 方案或者 DRBD+Inotify 方案解决）；

2.master 服务器对主机的内存要求略高；

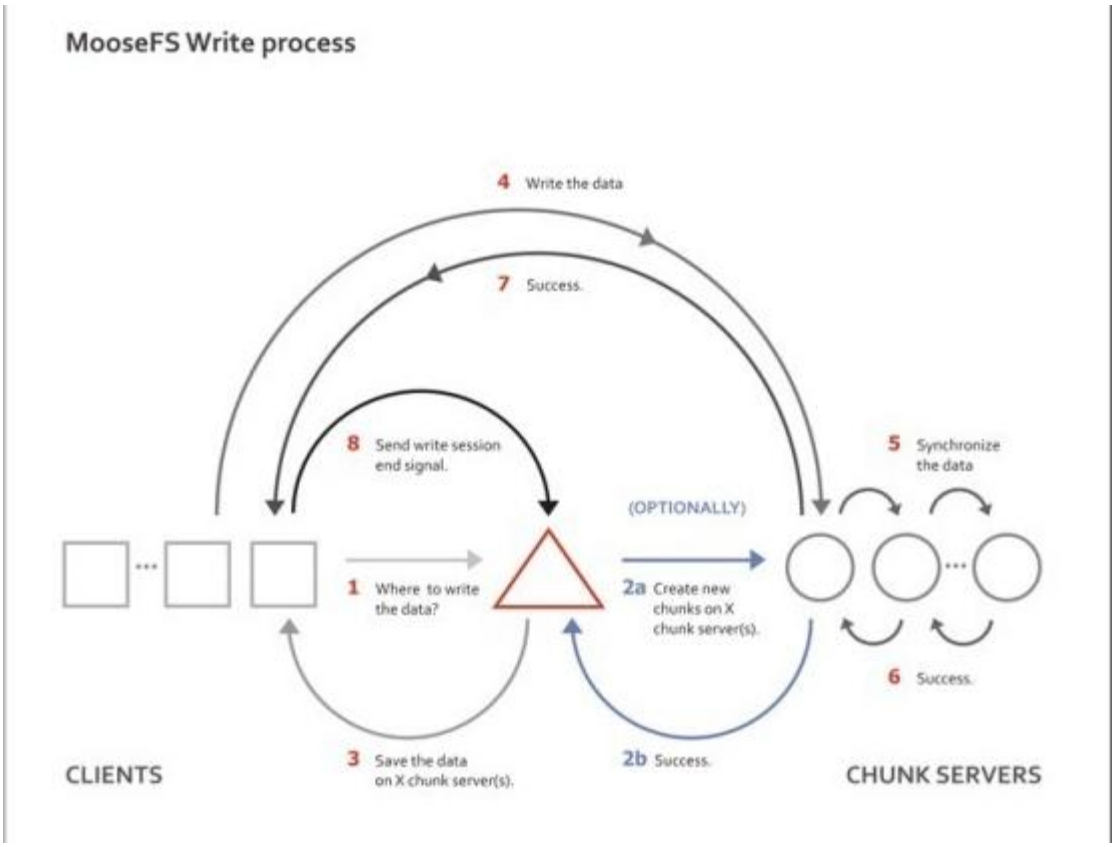
3.默认 metalogger 复制元数据时间较长；

三、MFS 分布式文件系统存储架构：

角色：

角色	作用
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
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
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。
客户机挂载使用 client computers	挂载进程 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。

MFS 存取数据的过程：



----------**MFS 存取数据详解**----------

- MFS 读取数据步骤：
  1）客户端向元数据服务器发出请求；
  2）元数据服务器把所需数据存放的位置(Chunk Server 的 IP 地址及 Chunk 编号）告知客户端；
  3）客户端向已知 Chunk Server 请求发送数据；
  4）客户端向 chunk server 请求取得所需数据；

- MFS 写入数据步骤：

  1）客户端向元数据服务器发送写入请求；

  2）元数据服务器与 Chunk Server 进行交互如下；

  1）元数据服务器指示在某些 Chunk Server 创建分块 Chunks；

  2）Chunk Server 告知元数据服务器，步骤(1)的操作成功；

  3）元数据服务器告知客户端，你可以在哪个 Chunk Server 的哪个 Chunks 写入数据；

  4）向指定的 Chunk Server 写入数据；

  5）与其他 Chunk Server 进行数据同步，同步的服务器依据设定的副本数而定，副本为 2，则需同步一个 ChunkServer；

  6）Chunk Sever 之间同步成功；

  7）Chunk Server 告知客户端数据写入成功；

  8）客户端告知元数据服务器本次写入完毕；

  9）元数据服务器将数据存放记录写入到本地日志；

  10）元数据服务器将本地日志拷贝到 logger 服务器；

- MFS 的删除文件过程：

  1）客户端有删除操作时，首先向 Master 发送删除信息；

  2）Master 定位到相应元数据信息进行删除，并将 chunk server 上块的删除操作加入队列异步清理；

  3）响应客户端删除成功的信号；

- MFS 修改文件内容的过程：

  1）客户端有修改文件内容时，首先向 Master 发送操作信息；

  2）Master 申请新的块给.swp 文件；

  3）客户端关闭文件后，会向 Master 发送关闭信息；

  4）Master 会检测内容是否有更新，若有，则申请新的块存放更改后的文件，删除原有块和.swp 文件块；

  5）若无，则直接删除.swp 文件块；

- MFS 重命名文件的过程：

  1）客户端重命名文件时，会向 Master 发送操作信息；

  2）Master 直接修改元数据信息中的文件名；返回重命名完成信息；

- MFS 遍历文件的过程：

  1）遍历文件不需要访问 chunk server，当有客户端遍历请求时，向 Master 发送操作信息；

  2）Master 返回相应元数据信息；

  3）客户端接收到信息后显示；

- 总结注意：

  1）Master 记录着管理信息，比如：文件路径|大小|存储的位置(ip,port,chunkid)|份数|时间等，元数据信息存在于内存中，会定期写入 metadata.mfs.back 文件中，定期同步到 metalogger，操作实时写入 changelog.*.mfs，实时同步到 metalogger 中。master 启动将 metadata.mfs 载入内存，重命名为 metadata.mfs.back 文件。

  2）文件以 chunk 大小存储，每 chunk 最大为 64M，小于 64M 的，该 chunk 的大小即为该文件大小（验证实际 chunk 文件略大于实际文件），超过 64M 的文件将被切分，以每一份（chunk）的大小不超过

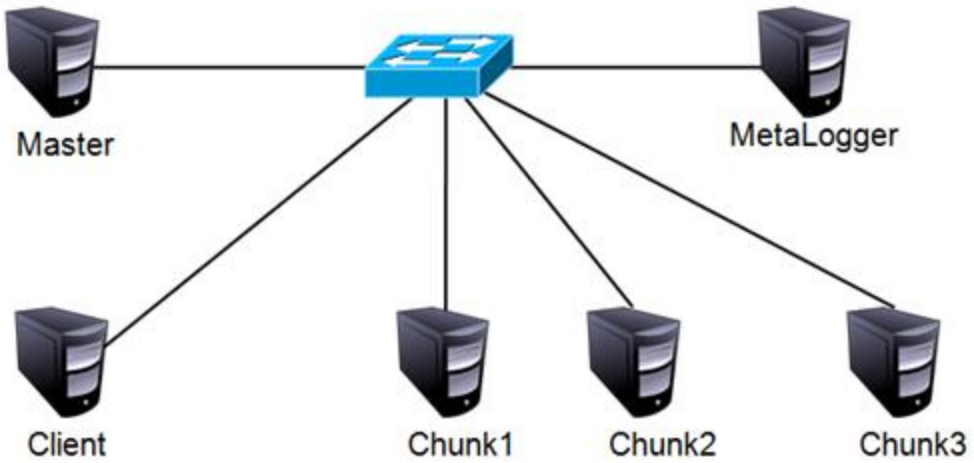
64M 为原则；块的生成遵循规则：目录循环写入(00-FF 256 个目录循环，step 为 2)、chunk 文件递增生成、大文件切分目录连续。

3）Chunkserver 上的剩余存储空间要大于 1GB（Reference Guide 有提到），新的数据才会被允许写入，否则，你会看到 No space left on device 的提示，实际中，测试发现当磁盘使用率达到 95%左右的时候，就已经不可以写入了，当时可用空间为 1.9GB。

4）文件可以有多份 copy，当 goal 为 1 时，文件会被随机存到一台 chunkserver 上，当 goal 的数大于 1 时，copy 会由 master 调度保存到不同的 chunkserver 上，goal 的大小不要超过 chunkserver 的数量，否则多出的 copy，不会有 chunkserver 去存。

四、案例：搭建 MFS 分布式存储：

案例拓扑：



案例环境：

系统类型	IP 地址	主机名	所需软件
Centos 7.4 1708 64bit	192.168.100.101	master.linuxfan.cn	mfs-1.6.27-5.tar.gz
Centos 7.4 1708 64bit	192.168.100.102	log.linuxfan.cn	mfs-1.6.27-5.tar.gz
Centos 7.4 1708 64bit	192.168.100.103	chunk1.linuxfan.cn	mfs-1.6.27-5.tar.gz
Centos 7.4 1708 64bit	192.168.100.104	chunk2.linuxfan.cn	mfs-1.6.27-5.tar.gz
Centos 7.4 1708 64bit	192.168.100.105	chunk3.linuxfan.cn	mfs-1.6.27-5.tar.gz
Centos 7.4 1708 64bit	192.168.100.106	client.linuxfan.cn	fuse-2.9.2.tar.gz mfs-1.6.27-5.tar.gz

案例步骤：

- 安装并配置 master 元数据服务器；
- 安装并配置 log 日志服务器；
- 安装并配置 chunk 节点存储服务器(在此三台 chunk 节点服务器配置相同，只列举 chunk1 节点的配置)；
- 安装并配置客户端节点；
- 配置 master 节点开启 MFS 监控；
- 客户端测试访问监控页面；
- 客户端测试写入数据；

- 验证 chunk 节点数据分配情况及 web 监控页面；
- 配置 master 节点修改数据的复制份数；
- 客户端再次写入数据测试；
- 验证 chunk 节点数据分配情况及 web 监控页面；

- **安装并配置 master 元数据服务器；**

```
[root@master ~]# yum -y install zlib-devel
[root@master ~]# useradd -s /sbin/nologin mfs
[root@master ~]# ls mfs-1.6.27-5.tar.gz
mfs-1.6.27-5.tar.gz
[root@master ~]# tar zxf mfs-1.6.27-5.tar.gz -C /usr/src
[root@master ~]# cd /usr/src/mfs-1.6.27
[root@master mfs-1.6.27 ]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs
--with-default-group=mfs --disable-mfschunkserver --disable-mfsmount
[root@master mfs-1.6.27 ]# make &&make install
[root@master mfs-1.6.27 ]# cd
[root@master ~]# cd /usr/local/mfs/etc/mfs/
[root@master mfs]# ls
mfsexports.cfg.dist mfsmaster.cfg.dist mfsmetalogger.cfg.dist mfstopology.cfg.dist
[root@master mfs]# cp mfsmaster.cfg.dist mfsmaster.cfg ##复制主配置文件
[root@master mfs]# cp mfsexports.cfg.dist mfsexports.cfg ##复制被挂载目录及权限配置文件
[root@master mfs]# cp mfstopology.cfg.dist mfstopology.cfg ##复制 ip 网络的位置配置文件
[root@master mfs]# cd /usr/local/mfs/var/mfs/
[root@master mfs]# ls
metadata.mfs.empty
[root@master mfs]# cp metadata.mfs.empty metadata.mfs ##复制日志记录的配置文件
[root@master mfs]# /usr/local/mfs/sbin/mfsmaster start ##关闭使用选项-s
[root@master mfs]# netstat -utpln |grep mfs
tcp 0 0 0.0.0.0:9419 0.0.0.0:* LISTEN 5769/mfsmaster
tcp 0 0 0.0.0.0:9420 0.0.0.0:* LISTEN 5769/mfsmaster
tcp 0 0 0.0.0.0:9421 0.0.0.0:* LISTEN 5769/mfsmaster
[root@master mfs]# cd
```

- **安装并配置 log 日志服务器；**

```
[root@log ~]# yum -y install zlib-devel
[root@log ~]# useradd -s /sbin/nologin mfs
[root@log ~]# ls mfs-1.6.27-5.tar.gz
mfs-1.6.27-5.tar.gz
[root@log ~]# tar zxf mfs-1.6.27-5.tar.gz -C /usr/src/
[root@log ~]# cd /usr/src/mfs-1.6.27
[root@log mfs-1.6.27]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
--disable-mfschunkserver --disable-mfsmount
[root@log mfs-1.6.27]# make &&make install
```

```
[root@log mfs-1.6.27]# cd
[root@log ~]# cd /usr/local/mfs/etc/mfs/
[root@log mfs]# cp mfsmetalogger.cfg.dist mfsmetalogger.cfg                    ##复制日志配置文件
[root@log mfs]# sed -i '/mfsmaster/a MASTER_HOST = 192.168.100.101' mfsmetalogger.cfg
[root@log mfs]# /usr/local/mfs/sbin/mfsmetalogger start
[root@log mfs]# ps aux |grep mfs |grep -v grep
mfs        5766  0.1  0.1  11824   832 ?        S<   05:42   0:00 /usr/local/mfs/sbin/mfsmetalogger
start
[root@log mfs]# cd
```

- **安装并配置 chunk 节点存储服务器（在此三台 chunk 节点服务器配置相同，只列举 chunk1 节点的配置）；**

```
[root@chunk1 ~]# yum -y install zlib-devel
[root@chunk1 ~]# useradd -s /sbin/nologin mfs
[root@chunk1 ~]# ls mfs-1.6.27-5.tar.gz
mfs-1.6.27-5.tar.gz
[root@chunk1 ~]# tar zxf mfs-1.6.27-5.tar.gz -C /usr/src/
[root@chunk1 ~]# cd /usr/src/mfs-1.6.27
[root@chunk1 mfs-1.6.27]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
--disable-mfsmaster --disable-mfsmount
[root@chunk1 mfs-1.6.27]# make &&make install
[root@chunk1 mfs-1.6.27]# cd /usr/local/mfs/etc/mfs/
[root@chunk1 mfs]# cp mfschunkserver.cfg.dist mfschunkserver.cfg              ##复制主配置文件
[root@chunk1 mfs]# cp mfshdd.cfg.dist mfshdd.cfg                              ##复制挂载信息的配置文件
[root@chunk1 mfs]# sed -i '/BIND_HOST/a MASTER_HOST = 192.168.100.101' mfschunkserver.cfg
[root@chunk1 mfs]# echo "/data" >>mfshdd.cfg
[root@chunk1 mfs]# mkdir /data
[root@chunk1 mfs]# chown -R mfs:mfs /data/
[root@chunk1 mfs]# /usr/local/mfs/sbin/mfschunkserver start
[root@chunk1 mfs]# ps aux |grep mfs |grep -v grep
mfs        5528  0.3  0.5 171640  2560 ?        S<l  05:44   0:00 /usr/local/mfs/sbin/mfschunkserver
start
[root@chunk1 mfs]# cd
```

- **安装并配置客户端节点；**

```
[root@client ~]# yum -y install zlib-devel
[root@client ~]# ls
fuse-2.9.2.tar.gz    mfs-1.6.27-5.tar.gz
[root@client ~]# tar zxvf fuse-2.9.2.tar.gz -C /usr/src                       ##编译安装 fuse 文件系统
[root@client ~]# cd /usr/src/fuse-2.9.2/
[root@client fuse-2.9.2]# ./configure &&make &&make install
[root@client fuse-2.9.2]# cd
[root@client ~]# echo "export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH" >>/etc/profile
[root@client ~]# source /etc/profile
```

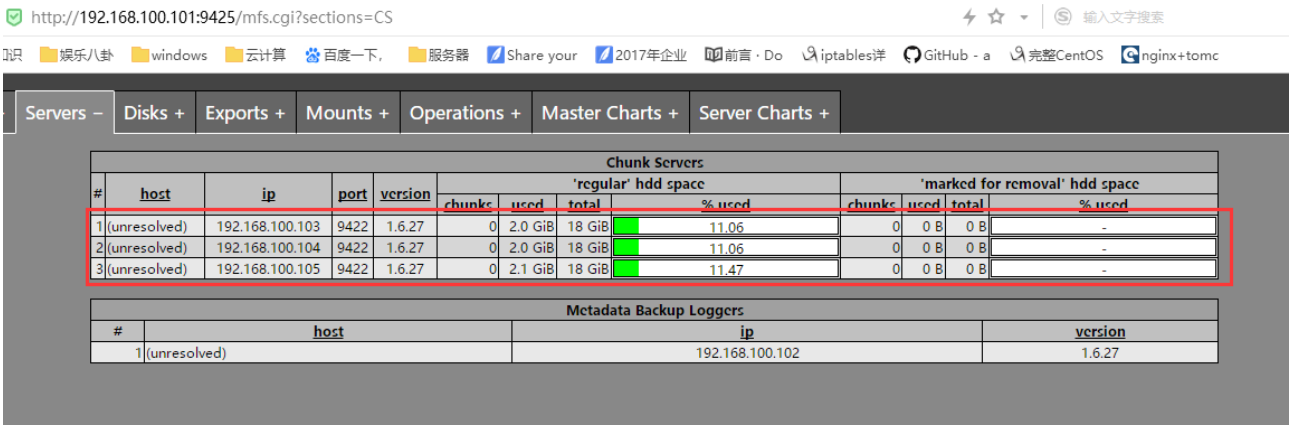
```
[root@client ~]# useradd -s /sbin/nologin mfs
[root@client ~]# tar zxvf mfs-1.6.27-5.tar.gz -C /usr/src/
[root@client ~]# cd /usr/src/mfs-1.6.27/
[root@client mfs-1.6.27]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
--disable-mfsmaster --disable-mfschunkserver --enable-mfsmount
[root@client mfs-1.6.27]# make &&make install
[root@client mfs-1.6.27]# cd
[root@client ~]# echo "export PATH=/usr/local/mfs/bin:$PATH" >>/etc/profile
[root@client ~]# source /etc/profile
[root@client ~]# modprobe fuse
[root@client ~]# lsmod |grep fuse
fuse                   91874  1
[root@client ~]# mkdir /mnt/mfs
[root@client ~]# mfsmount /mnt/mfs -H 192.168.100.101
mfsmaster accepted connection with parameters: read-write,restricted_ip ; root mapped to root:root
[root@client ~]# df -hT |grep mfs
192.168.100.101:9421   fuse.mfs   50G   0   50G   0% /mnt/mfs
```

- **配置 master 节点开启 MFS 监控；**

```
[root@master ~]# /usr/local/mfs/sbin/mfscgiserv          ##启动 mfscgiserv 是用 python 编写的 web 服务，监听端口 9425：
lockfile created and locked
starting simple cgi server (host: any , port: 9425 , rootpath: /usr/local/mfs/share/mfscgi)
[root@master ~]# netstat -utpln |grep 9425
tcp        0      0 0.0.0.0:9425            0.0.0.0:*               LISTEN      5779/python
```

- **客户端测试访问监控页面；**

http://192.168.100.101:9425



- **客户端测试写入数据；**

```
[root@client ~]# df -hT |grep mfs
192.168.100.101:9421   fuse.mfs   50G   0   50G   0% /mnt/mfs
```

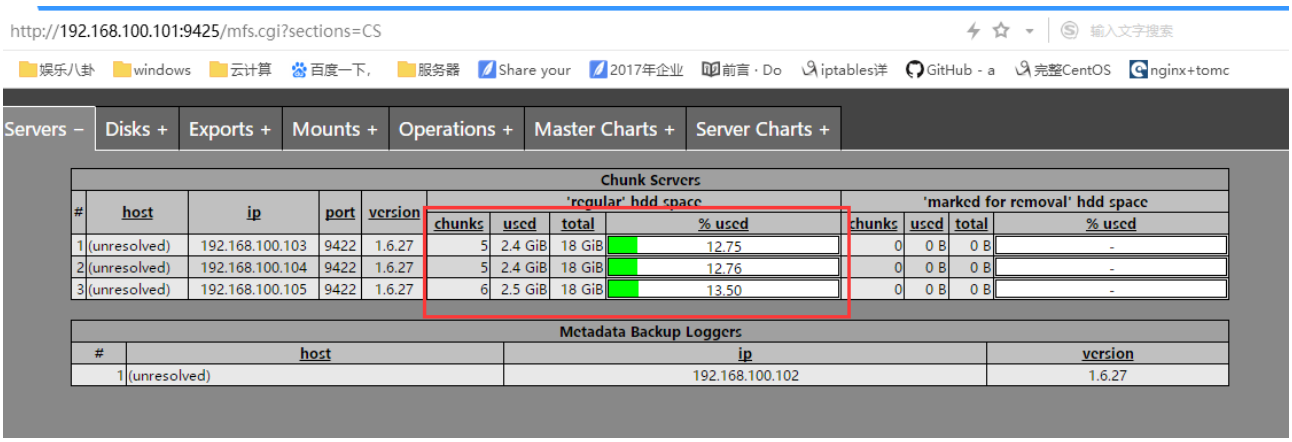
```
[root@client ~]# dd if=/dev/zero of=/mnt/mfs/1.file bs=1G count=1
记录了 1+0 的读入
记录了 1+0 的写出
1073741824 字节(1.1 GB)已复制，20.7523 秒，51.7 MB/秒
[root@client ~]# du -sh /mnt/mfs/1.file
1.0G /mnt/mfs/1.file
[root@client ~]# df -hT |grep mfs
192.168.100.101:9421          fuse.mfs     49G  1.1G   48G   3% /mnt/mfs
```

- **验证 chunk 节点数据分配情况及 web 监控页面；**

```
[root@chunk1 ~]# ls /data/
00  0B  16  21  2C  37  42  4D  58  63  6E  79  84  8F  9A  A5  B0  BB  C6  D1  DC  E7  F2  FD
01  0C  17  22  2D  38  43  4E  59  64  6F  7A  85  90  9B  A6  B1  BC  C7  D2  DD  E8  F3  FE
02  0D  18  23  2E  39  44  4F  5A  65  70  7B  86  91  9C  A7  B2  BD  C8  D3  DE  E9  F4  FF
03  0E  19  24  2F  3A  45  50  5B  66  71  7C  87  92  9D  A8  B3  BE  C9  D4  DF  EA  F5
04  0F  1A  25  30  3B  46  51  5C  67  72  7D  88  93  9E  A9  B4  BF  CA  D5  E0  EB  F6
05  10  1B  26  31  3C  47  52  5D  68  73  7E  89  94  9F  AA  B5  C0  CB  D6  E1  EC  F7
06  11  1C  27  32  3D  48  53  5E  69  74  7F  8A  95  A0  AB  B6  C1  CC  D7  E2  ED  F8
07  12  1D  28  33  3E  49  54  5F  6A  75  80  8B  96  A1  AC  B7  C2  CD  D8  E3  EE  F9
08  13  1E  29  34  3F  4A  55  60  6B  76  81  8C  97  A2  AD  B8  C3  CE  D9  E4  EF  FA
09  14  1F  2A  35  40  4B  56  61  6C  77  82  8D  98  A3  AE  B9  C4  CF  DA  E5  F0  FB
0A  15  20  2B  36  41  4C  57  62  6D  78  83  8E  99  A4  AF  BA  C5  D0  DB  E6  F1  FC
[root@chunk1 ~]# du -sh /data/
321M    /data/
```

```
[root@chunk2 ~]# ls /data/
00  0B  16  21  2C  37  42  4D  58  63  6E  79  84  8F  9A  A5  B0  BB  C6  D1  DC  E7  F2  FD
01  0C  17  22  2D  38  43  4E  59  64  6F  7A  85  90  9B  A6  B1  BC  C7  D2  DD  E8  F3  FE
02  0D  18  23  2E  39  44  4F  5A  65  70  7B  86  91  9C  A7  B2  BD  C8  D3  DE  E9  F4  FF
03  0E  19  24  2F  3A  45  50  5B  66  71  7C  87  92  9D  A8  B3  BE  C9  D4  DF  EA  F5
04  0F  1A  25  30  3B  46  51  5C  67  72  7D  88  93  9E  A9  B4  BF  CA  D5  E0  EB  F6
05  10  1B  26  31  3C  47  52  5D  68  73  7E  89  94  9F  AA  B5  C0  CB  D6  E1  EC  F7
06  11  1C  27  32  3D  48  53  5E  69  74  7F  8A  95  A0  AB  B6  C1  CC  D7  E2  ED  F8
07  12  1D  28  33  3E  49  54  5F  6A  75  80  8B  96  A1  AC  B7  C2  CD  D8  E3  EE  F9
08  13  1E  29  34  3F  4A  55  60  6B  76  81  8C  97  A2  AD  B8  C3  CE  D9  E4  EF  FA
09  14  1F  2A  35  40  4B  56  61  6C  77  82  8D  98  A3  AE  B9  C4  CF  DA  E5  F0  FB
0A  15  20  2B  36  41  4C  57  62  6D  78  83  8E  99  A4  AF  BA  C5  D0  DB  E6  F1  FC
[root@chunk2 ~]# du -sh /data/
321M    /data/
```

```
[root@chunk3 ~]# ls /data/
00  0B  16  21  2C  37  42  4D  58  63  6E  79  84  8F  9A  A5  B0  BB  C6  D1  DC  E7  F2  FD
01  0C  17  22  2D  38  43  4E  59  64  6F  7A  85  90  9B  A6  B1  BC  C7  D2  DD  E8  F3  FE
02  0D  18  23  2E  39  44  4F  5A  65  70  7B  86  91  9C  A7  B2  BD  C8  D3  DE  E9  F4  FF
03  0E  19  24  2F  3A  45  50  5B  66  71  7C  87  92  9D  A8  B3  BE  C9  D4  DF  EA  F5
04  0F  1A  25  30  3B  46  51  5C  67  72  7D  88  93  9E  A9  B4  BF  CA  D5  E0  EB  F6
05  10  1B  26  31  3C  47  52  5D  68  73  7E  89  94  9F  AA  B5  C0  CB  D6  E1  EC  F7
06  11  1C  27  32  3D  48  53  5E  69  74  7F  8A  95  A0  AB  B6  C1  CC  D7  E2  ED  F8
07  12  1D  28  33  3E  49  54  5F  6A  75  80  8B  96  A1  AC  B7  C2  CD  D8  E3  EE  F9
08  13  1E  29  34  3F  4A  55  60  6B  76  81  8C  97  A2  AD  B8  C3  CE  D9  E4  EF  FA
09  14  1F  2A  35  40  4B  56  61  6C  77  82  8D  98  A3  AE  B9  C4  CF  DA  E5  F0  FB
0A  15  20  2B  36  41  4C  57  62  6D  78  83  8E  99  A4  AF  BA  C5  D0  DB  E6  F1  FC
[root@chunk3 ~]# du -sh /data/
385M    /data/
```




```
[root@master ~]# ls /usr/local/mfs/var/mfs/
changelog.0.mfs  metadata.mfs.back  metadata.mfs.empty  sessions.mfs
```

```
[root@log mfs]# ls /usr/local/mfs/var/mfs/
changelog_ml.0.mfs       changelog_ml_back.1.mfs  metadata_ml.mfs.back     sessions_ml.mfs
changelog_ml_back.0.mfs  metadata.mfs.empty       .mfsmetalogger.lock
```

➢ **配置 master 节点修改数据的复制份数；**

a.默认每个 chunk server 中会占用一定的空间；

b.MFS 默认存放文件的份数为 1，如若存放 1G 的文件，三个 chunk 节点会将 1G 的文件分割存储（分布式存储）；

c.如若在后续将默认的 1 份改成了 2 份，那么包括以前存在的文件和改后存放的文件，都会被 chunk 节点所同步成两份，并且将两份文件的大小，分布存储在多个 chunk 节点中；

```
[root@client ~]# mfsgetgoal /mnt/mfs/
/mnt/mfs/: 1
[root@client ~]# mfssetgoal -r 2 /mnt/mfs/
/mnt/mfs/:
 inodes with goal changed:                       2
 inodes with goal not changed:                   0
 inodes with permission denied:                  0
```

➢ **客户端再次写入数据测试；**

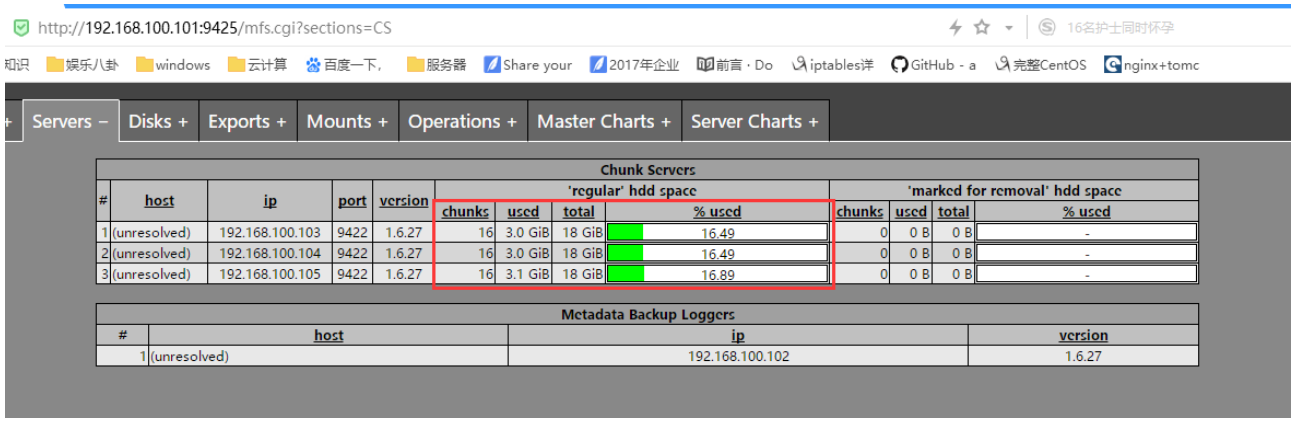
```
[root@client ~]# dd if=/dev/zero of=/mnt/mfs/2.file bs=1G count=1
记录了1+0 的读入
记录了1+0 的写出
1073741824字节(1.1 GB)已复制，48.1275 秒，22.3 MB/秒
[root@client ~]# du -sh /mnt/mfs/2.file
1.0G    /mnt/mfs/2.file
[root@client ~]# mfsgetgoal /mnt/mfs/2.file
/mnt/mfs/2.file: 2
[root@client ~]# mfsgetgoal /mnt/mfs/1.file
/mnt/mfs/1.file: 2
```

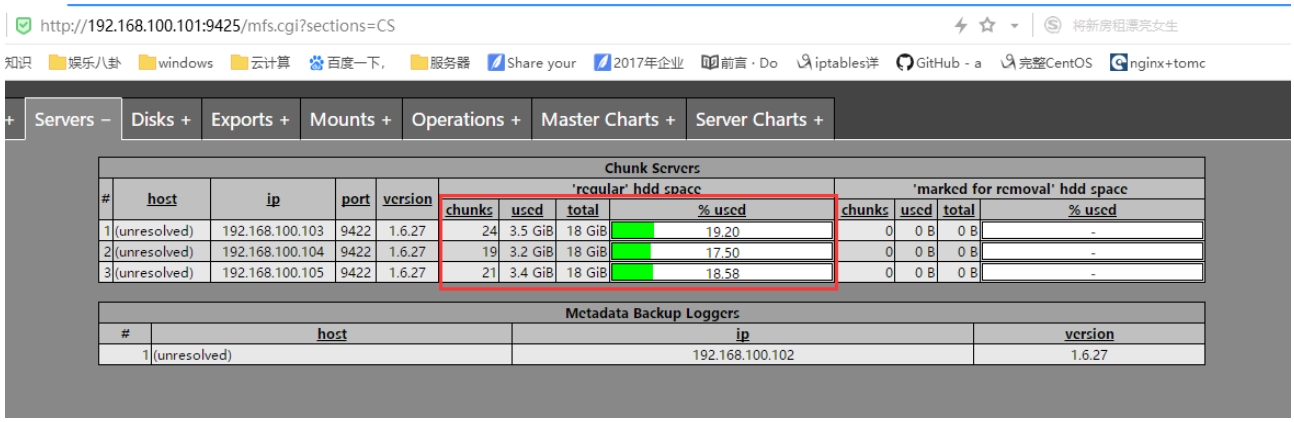
➢ **验证 chunk 节点数据分配情况及 web 监控页面；**

```
[root@chunk1 ~]# du -sh /data/
1.1G    /data/
```

```
[root@chunk2 ~]# du -sh /data/
1.1G    /data/
```

```
[root@chunk3 ~]# du -sh /data/
1.1G    /data/
```





注：MFS 集群的启动与停止：

MFS 的启动顺序：master 元数据服务器-->chunkserver 存储服务器-->metalogger 元数据日志服务器-->client 客户端

MFS 停止的顺序：client(umount)客户端-->chunkserver(-s)存储服务器-->metalogger(-s)元数据日志服务器-->master(-s)元数据服务器