# 跟冰河学习分布式存储技术

## 写在前面

> 在 **冰河技术** 微信公众号中的【分布式存储】专题，更新了不少文章，有些读者反馈说，在公众号中刷历史文章不太方便，有时会忘记自己看到哪一篇了，当打开一篇文章时，似乎之前已经看过了，但就是不知道具体该看哪一篇了。相信很多小伙伴都会有这样的问题。那怎么办呢？最好的解决方案就是我把这些文章整理成PDF电子书，免费分享给大家，这样，小伙伴们看起来就方便多了。希望这本电子书能够给大家带来实质性的帮助。后续，我也会持续在 **冰河技术** 公众号中更新【分布式存储】专题，如果这本电子书能够给你带来帮助，请关注 **冰河技术** 微信公众号，我们一起进阶，一起牛逼。
>
> 注：SpringBoot整合FastDFS实战的工程源码已提交到：https://github.com/sunshinelyz/mykit-fastdfs。

## 关于作者

大数据架构师，Mykit系列开源框架作者，多年来致力于分布式系统架构、微服务、分布式数据库、分布式事务与大数据技术的研究，曾主导过众多分布式系统、微服务及大数据项目的架构设计、研发和实施落地。在高并发、高可用、高可扩展性、高可维护性和大数据等领域拥有丰富的架构经验。对Hadoop，Storm，Spark，Flink等大数据框架源码进行过深度分析，并具有丰富的实战经验。目前在研究云原生。公众号【冰河技术】作者，《海量数据处理与大数据技术实战》、《MySQL技术大全：开发、优化与运维实战》作者，基于最终消息可靠性的开源分布式事务框架mykit-transaction-message作者。

# 基于CentOS6.X服务器搭建FastDFS环境

## CentOS6.X搭建FastDFS环境

### 前期准备

跟踪服务器： 192.168.50.131 (liuyazhuang131)
存储服务器： 192.168.50.132 (liuyazhuang132)
环境： CentOS 6.5
用户： root
数据目录： /fastdfs （注： 数据目录按你的数据盘挂载路径而定）

安装包：

- FastDFS v5.05
- libfastcommon-master.zip（是从 FastDFS 和 FastDHT 中提取出来的公共 C 函数库）
- fastdfs-nginx-module_v1.16.tar.gz
- nginx-1.6.2.tar.gz
- fastdfs_client_java._v1.25.tar.gz

源码地址： https://github.com/happyfish100/
下载地址： http://sourceforge.net/projects/fastdfs/files/
官方论坛： http://bbs.chinaunix.net/forum-240-1.html

### 所有服务器执行的操作

#### 1、编译和安装所需的依赖包

```
# yum install make cmake gcc gcc-c++
```

#### 2、 安装 libfastcommon

(1)上传或下载 libfastcommon-master.zip 到/usr/local/src 目录

(2)解压

```
# cd /usr/local/src/
# unzip libfastcommon-master.zip
# cd libfastcommon-master
```

```
[root@liuyazhuang131 libfastcommon-master]# ll
total 28
-rw-r--r-- 1 root root 2913 Feb 27  2015 HISTORY
-rw-r--r-- 1 root root  582 Feb 27  2015 INSTALL
-rw-r--r-- 1 root root 1342 Feb 27  2015 libfastcommon.spec
-rwxr-xr-x 1 root root 2151 Feb 27  2015 make.sh
drwxr-xr-x 2 root root 4096 Feb 27  2015 php-fastcommon
-rw-r--r-- 1 root root  617 Feb 27  2015 README
drwxr-xr-x 2 root root 4096 May 13 13:00 src
[root@liuyazhuang131 libfastcommon-master]#
```

(3) 编译、安装

```
# ./make.sh
# ./make.sh install
```

libfastcommon 默认安装到了如下位置。

```
/usr/lib64/libfastcommon.so
/usr/lib64/libfdfsclient.so
```

(4)因为 FastDFS 主程序设置的 lib 目录是/usr/local/lib， 所以需要创建软链接

```
# ln -s /usr/lib64/libfastcommon.so /usr/local/lib/libfastcommon.so
# ln -s /usr/lib64/libfastcommon.so /usr/lib/libfastcommon.so
# ln -s /usr/lib64/libfdfsclient.so /usr/local/lib/libfdfsclient.so
# ln -s /usr/lib64/libfdfsclient.so /usr/lib/libfdfsclient.so
```

## 3、安装 FastDFS

(1)上传或下载 FastDFS 源码包（FastDFS_v5.05.tar.gz） 到 /usr/local/src 目录

(2)解压

```
# cd /usr/local/src/
# tar -zxvf FastDFS_v5.05.tar.gz
# cd FastDFS
```

```
[root@liuyazhuang131 FastDFS]# ll
total 132
drwxr-xr-x 3 8980 users  4096 May 13 13:16 client
drwxr-xr-x 2 8980 users  4096 May 13 13:15 common
drwxr-xr-x 2 8980 users  4096 Dec  2  2014 conf
-rw-r--r-- 1 8980 users 35067 Dec  2  2014 COPYING-3_0.txt
-rw-r--r-- 1 8980 users  2802 Dec  2  2014 fastdfs.spec
-rw-r--r-- 1 8980 users 31386 Dec  2  2014 HISTORY
drwxr-xr-x 2 8980 users  4096 Dec  2  2014 init.d
-rw-r--r-- 1 8980 users  7755 Dec  2  2014 INSTALL
-rwxr-xr-x 1 8980 users  5813 Dec  2  2014 make.sh
drwxr-xr-x 2 8980 users  4096 Dec  2  2014 php_client
-rw-r--r-- 1 8980 users  2380 Dec  2  2014 README.md
-rwxr-xr-x 1 8980 users  1768 Dec  2  2014 restart.sh
-rwxr-xr-x 1 8980 users  1680 Dec  2  2014 stop.sh
drwxr-xr-x 4 8980 users  4096 May 13 13:16 storage
drwxr-xr-x 2 8980 users  4096 Dec  2  2014 test
drwxr-xr-x 2 8980 users  4096 May 13 13:16 tracker
[root@liuyazhuang131 FastDFS]#
```

(3)编译、 安装（编译前要确保已经成功安装了 libfastcommon）

```
# ./make.sh
# ./make.sh install
```

采用默认安装的方式安装,安装后的相应文件与目录：

A、 服务脚本在：

```
/etc/init.d/fdfs_storaged
/etc/init.d/fdfs_tracker
```

B、 配置文件在（样例配置文件）

```
/etc/fdfs/client.conf.sample
/etc/fdfs/storage.conf.sample
/etc/fdfs/tracker.conf.sample
```

C、 命令工具在/usr/bin/目录下的

```
fdfs_appender_test
fdfs_appender_test1
fdfs_append_file
fdfs_crc32
fdfs_delete_file
fdfs_download_file
fdfs_file_info
fdfs_monitor
fdfs_storaged
fdfs_test
fdfs_test1
fdfs_trackerd
fdfs_upload_appender
fdfs_upload_file
stop.sh
restart.sh
```

(4)因为 FastDFS 服务脚本设置的 bin 目录是/usr/local/bin， 但实际命令安装在/usr/bin， 可以进入/user/bin 目录使用以下命令查看 fdfs 的相关命令：

```
# cd /usr/bin/
# ls | grep fdfs
```

```
[root@liuyazhuang131 bin]# ls | grep fdfs
fdfs_appender_test
fdfs_appender_test1
fdfs_append_file
fdfs_crc32
fdfs_delete_file
fdfs_download_file
fdfs_file_info
fdfs_monitor
fdfs_storaged
fdfs_test
fdfs_test1
fdfs_trackerd
fdfs_upload_appender
fdfs_upload_file
[root@liuyazhuang131 bin]#
```



因此需要修改 FastDFS 服务脚本中相应的命令路径，也就是把/etc/init.d/fdfs_storaged 和/etc/init.d/fdfs_trackerd 两个脚本中的/usr/local/bin 修改成/usr/bin：

```
# vi fdfs_trackerd
```

使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin

```
# vi fdfs_storaged
```

使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin

注：使用查找替换命令为进入vi / vim编辑器，按下esc键，输入冒号(:),再输入%s+/usr/local/bin+/usr/bin即可把所有的/usr/local/bin修改为/usr/bin

# 配置 FastDFS 跟踪器(192.168.50.131)

## 1、复制 FastDFS 跟踪器样例配置文件

复制 FastDFS 跟踪器样例配置文件,并重命名:

```
# cd /etc/fdfs/
```

```
[root@liuyazhuang131 fdfs]# ll
total 32
-rw-r--r-- 1 root root 1458 May 13 15:44 client.conf
-rw-r--r-- 1 root root 1461 May 13 13:16 client.conf.sample
-rw-r--r-- 1 root root 7829 May 13 13:16 storage.conf.sample
-rw-r--r-- 1 root root 7098 May 13 15:28 tracker.conf
-rw-r--r-- 1 root root 7102 May 13 13:16 tracker.conf.sample
[root@liuyazhuang131 fdfs]#
```

```
# cp tracker.conf.sample tracker.conf
```

## 2、编辑跟踪器配置文件

```
# vi /etc/fdfs/tracker.conf
```

修改的内容如下：

```
disabled=false
port=22122
base_path=/fastdfs/tracker
```

（其它参数保留默认配置， 具体配置解释请参考官方文档说明：http://bbs.chinaunix.net/thread-1941456-1-1.html ）

### 3、创建基础数据目录（参考基础目录 base_path 配置）

```
# mkdir -p /fastdfs/tracker
```

### 4、防火墙中打开跟踪器端口（默认为 22122）

```
# vi /etc/sysconfig/iptables
```

添加如下端口行：

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22122 -j ACCEPT
```

重启防火墙

```
# service iptables restart
```

### 5、启动 Tracker

```
# /etc/init.d/fdfs_trackerd start
```

（初次成功启动，会在/fastdfs/tracker 目录下创建 data、 logs 两个目录）

```
[root@liuyazhuang131 tracker]# ll
total 8
drwxr-xr-x 2 root root 4096 May 13 15:38 data
drwxr-xr-x 2 root root 4096 May 13 15:31 logs
[root@liuyazhuang131 tracker]#
```

查看 FastDFS Tracker 是否已成功启动。

```
# ps -ef | grep fdfs
```

```
[root@liuyazhuang131 tracker]# ps -ef | grep fdfs
root      3739     1  0 15:31 ?        00:00:02 /usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
root      3798  2998  0 17:18 pts/0    00:00:00 grep fdfs
[root@liuyazhuang131 tracker]#
```

### 6、 关闭 Tracker

```
# /etc/init.d/fdfs_trackerd stop
```

### 7、设置 FastDFS 跟踪器开机启动

```
# vi /etc/rc.d/rc.local
```

添加以下内容

```
## FastDFS Tracker
/etc/init.d/fdfs_trackerd start
```

# 配置 FastDFS 存储(192.168.50.132)

## 1、 复制 FastDFS 存储器样例配置文件

复制 FastDFS 存储器样例配置文件,并重命名

```
# cd /etc/fdfs/
```

```
[root@liuyazhuang132 fdfs]# ll
total 68
-rw-r--r--. 1 root root  1461 May 13 13:16 client.conf.sample
-rw-r--r--. 1 root root   858 May 13 15:58 http.conf
-rw-r--r--. 1 root root 31172 May 13 15:58 mime.types
-rw-r--r--. 1 root root  3682 May 13 15:57 mod_fastdfs.conf
-rw-r--r--. 1 root root  7820 May 13 15:37 storage.conf
-rw-r--r--. 1 root root  7829 May 13 13:16 storage.conf.sample
-rw-r--r--. 1 root root  7102 May 13 13:16 tracker.conf.sample
[root@liuyazhuang132 fdfs]#
```



```
# cp storage.conf.sample storage.conf
```

## 2、编辑存储器样例配置文件

```
# vi /etc/fdfs/storage.conf
```

修改的内容如下:

```
disabled=false
port=23000
base_path=/fastdfs/storage
store_path0=/fastdfs/storage
tracker_server=192.168.50.131:22122
http.server_port=8888
```

（其它参数保留默认配置， 具体配置解释请参考官方文档说明：http://bbs.chinaunix.net/thread-1941456-1-1.html ）

## 3、创建基础数据目录（参考基础目录 base_path 配置）

```
# mkdir -p /fastdfs/storage
```

## 4、 防火墙中打开存储器端口（默认为 23000）

```
# vi /etc/sysconfig/iptables
```

添加如下端口行:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 23000 -j ACCEPT
```

重启防火墙：

```
# service iptables restart
```

### 5、启动 Storage

```
# /etc/init.d/fdfs_storaged start
```

（初次成功启动，会在/fastdfs/storage 目录下创建 data、 logs 两个目录）

```
[root@liuyazhuang132 fdfs]# cd /fastdfs/storage
[root@liuyazhuang132 storage]# ll
total 8
drwxr-xr-x. 259 root root 4096 May 13 16:03 data
drwxr-xr-x.   2 root root 4096 May 13 15:38 logs
[root@liuyazhuang132 storage]#
```



查看 FastDFS Storage 是否已成功启动

```
# ps -ef | grep fdfs
```

```
[root@liuyazhuang132 storage]# ps -ef | grep fdfs
root      3813     1  0 15:38 ?        00:00:02 /usr/bin/fdfs_storaged /etc/fdfs/storage.conf
root      6170  3087  4 17:26 pts/0    00:00:00 grep fdfs
[root@liuyazhuang132 storage]#
```



### 6、关闭 Storage

```
# /etc/init.d/fdfs_storaged stop
```

### 7、设置 FastDFS 存储器开机启动

```
# vi /etc/rc.d/rc.local
```

添加：

```
## FastDFS Storage
/etc/init.d/fdfs_storaged start
```

## 文件上传测试(192.168.50.131)

### 1、修改 Tracker 服务器中的客户端配置文件

```
# cp /etc/fdfs/client.conf.sample /etc/fdfs/client.conf
# vi /etc/fdfs/client.conf
base_path=/fastdfs/tracker
tracker_server=192.168.50.131:22122
```

### 2、执行如下文件上传命令

```
# /usr/bin/fdfs_upload_file /etc/fdfs/client.conf
/usr/local/src/FastDFS_v5.05.tar.gz
```

返回 ID 号： group1/M00/00/00/wKgEfVUYNYeAb7XFAAVFOL7FJU4.tar.gz（能返回以上文件 ID， 说明文件上传成功）

# 在每个存储节点上安装 nginx

## 1、 fastdfs-nginx-module 作用说明

FastDFS 通过 Tracker 服务器,将文件放在 Storage 服务器存储， 但是同组存储服务器之间需要进入文件复制， 有同步延迟的问题。假设 Tracker 服务器将文件上传到了 192.168.50.132，上传成功后文件 ID已经返回给客户端。此时 FastDFS 存储集群机制会将这个文件同步到同组存储 192.168.50.133，在文件还没有复制完成的情况下，客户端如果用这个文件 ID 在 192.168.50.133 上取文件,就会出现文件无法访问的错误。而 fastdfs-nginx-module 可以重定向文件连接到源服务器取文件,避免客户端由于复制延迟导致的文件无法访问错误。（解压后的 fastdfs-nginx-module 在 nginx 安装时使用）

## 2、上传 fastdfs-nginx-module_v1.16.tar.gz 到/usr/local/src

## 3、解压

```
# cd /usr/local/src/
# tar -zxvf fastdfs-nginx-module_v1.16.tar.gz
```

## 4、修改 fastdfs-nginx-module 的 config 配置文件

```
# cd fastdfs-nginx-module/src
# vi config
CORE_INCS="$CORE_INCS /usr/local/include/fastdfs /usr/local/include/fastcommon/"
修改为：
CORE_INCS="$CORE_INCS /usr/include/fastdfs /usr/include/fastcommon/"
```

（注意： 这个路径修改是很重要的，不然在 nginx 编译的时候会报错的）

## 5、上传当前的稳定版本 Nginx(nginx-1.13.0.tar.gz)到/usr/local/src 目录

## 6、安装编译 Nginx 所需的依赖包

```
# yum install gcc gcc-c++ make automake autoconf libtool pcre* zlib openssl
openssl-devel
```

## 7、编译安装 Nginx（添加 fastdfs-nginx-module 模块）

```
# cd /usr/local/src/
# tar -zxvf nginx-1.13.0.tar.gz
# cd nginx-1.13.0
# ./configure --add-module=/usr/local/src/fastdfs-nginx-module/src
# make && make install
```

## 8、复制 fastdfs-nginx-module

复制 fastdfs-nginx-module 源码中的配置文件到/etc/fdfs 目录， 并修改

```
# cp /usr/local/src/fastdfs-nginx-module/src/mod_fastdfs.conf /etc/fdfs/
# vi /etc/fdfs/mod_fastdfs.conf
```

修改以下配置：

```
connect_timeout=10
base_path=/tmp
tracker_server=192.168.50.131:22122
storage_server_port=23000
group_name=group1
url_have_group_name = true
store_path0=/fastdfs/storage
```

### 9、复制 FastDFS 的部分配置文件到/etc/fdfs 目录

```
# cd /usr/local/src/FastDFS/conf
# cp http.conf mime.types /etc/fdfs/
```

### 10、在/fastdfs/storage 文件存储目录下创建软连接,将其链接到实际存放数据的目录

```
# ln -s /fastdfs/storage/data/ /fastdfs/storage/data/M00
```

### 11、配置 Nginx

简洁版 nginx 配置样例：

```
user root;
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;
    server {
        listen 8888;
        server_name localhost;
        location ~/group([0-9])/M00 {
            #alias /fastdfs/storage/data;
            ngx_fastdfs_module;
        }
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root html;
        }
    }
}
```

注意、 说明：
A、 8888 端口值是要与/etc/fdfs/storage.conf 中的 http.server_port=8888 相对应，因为 http.server_port 默认为 8888,如果想改成 80，则要对应修改过来。
B、 Storage 对应有多个 group 的情况下，访问路径带 group 名，如/group1/M00/00/00/xxx，对应的 Nginx 配置为：

```
location ~/group([0-9])/M00 {
    ngx_fastdfs_module;
}
```

C、 如查下载时如发现老报 404， 将 nginx.conf 第一行 user nobody 修改为 user root 后重新启动。

## 12、防火墙中打开 Nginx 的 8888 端口

```
# vi /etc/sysconfig/iptables
```

添加：

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8888 -j ACCEPT
```

重启防火墙

```
# service iptables restart
```

## 13、启动 Nginx

```
# /usr/local/nginx/sbin/nginx
```

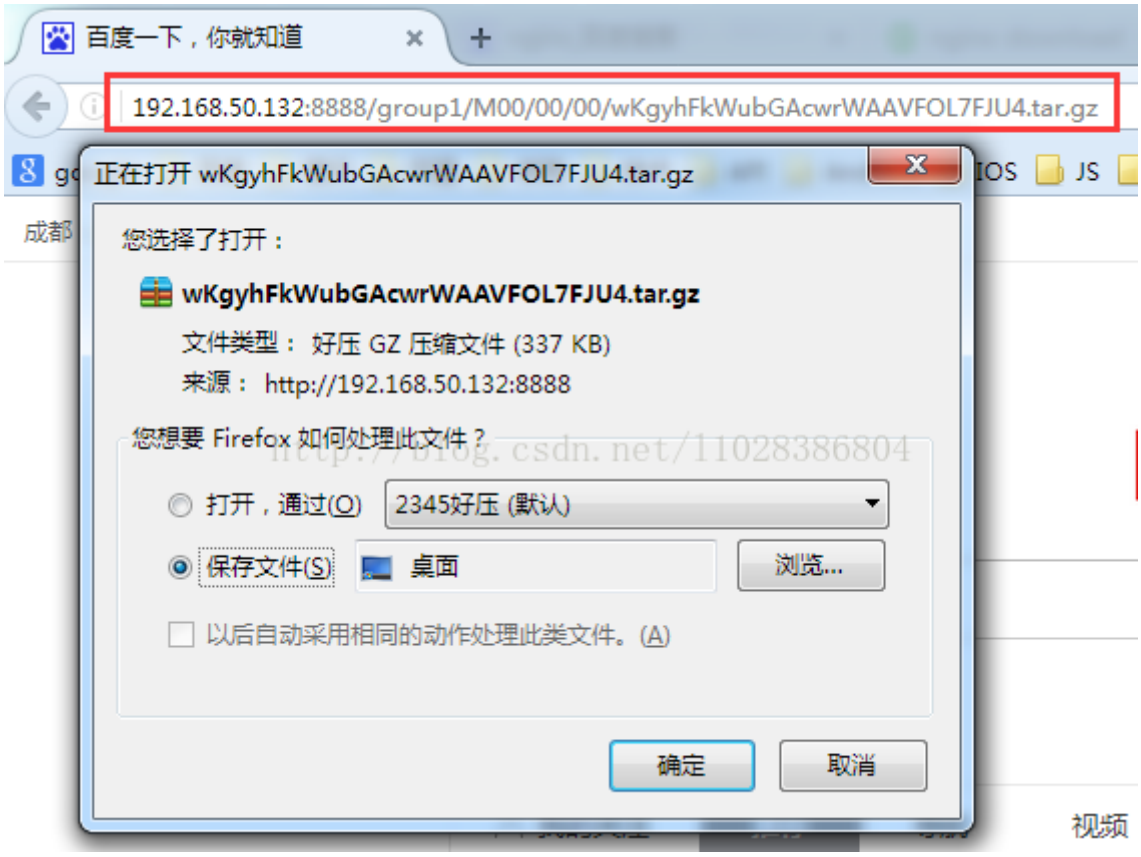
启动成功后会输入：

```
ngx_http_fastdfs_set pid=xxx
```

（重启 Nginx 的命令为： /usr/local/nginx/sbin/nginx -s reload）

## 14、通过浏览器访问测试时上传的文件

http://192.168.50.132:8888/group1/M00/00/00/wKgyhFkWubGAcwrWAAVFOL7FJU4.tar.gz



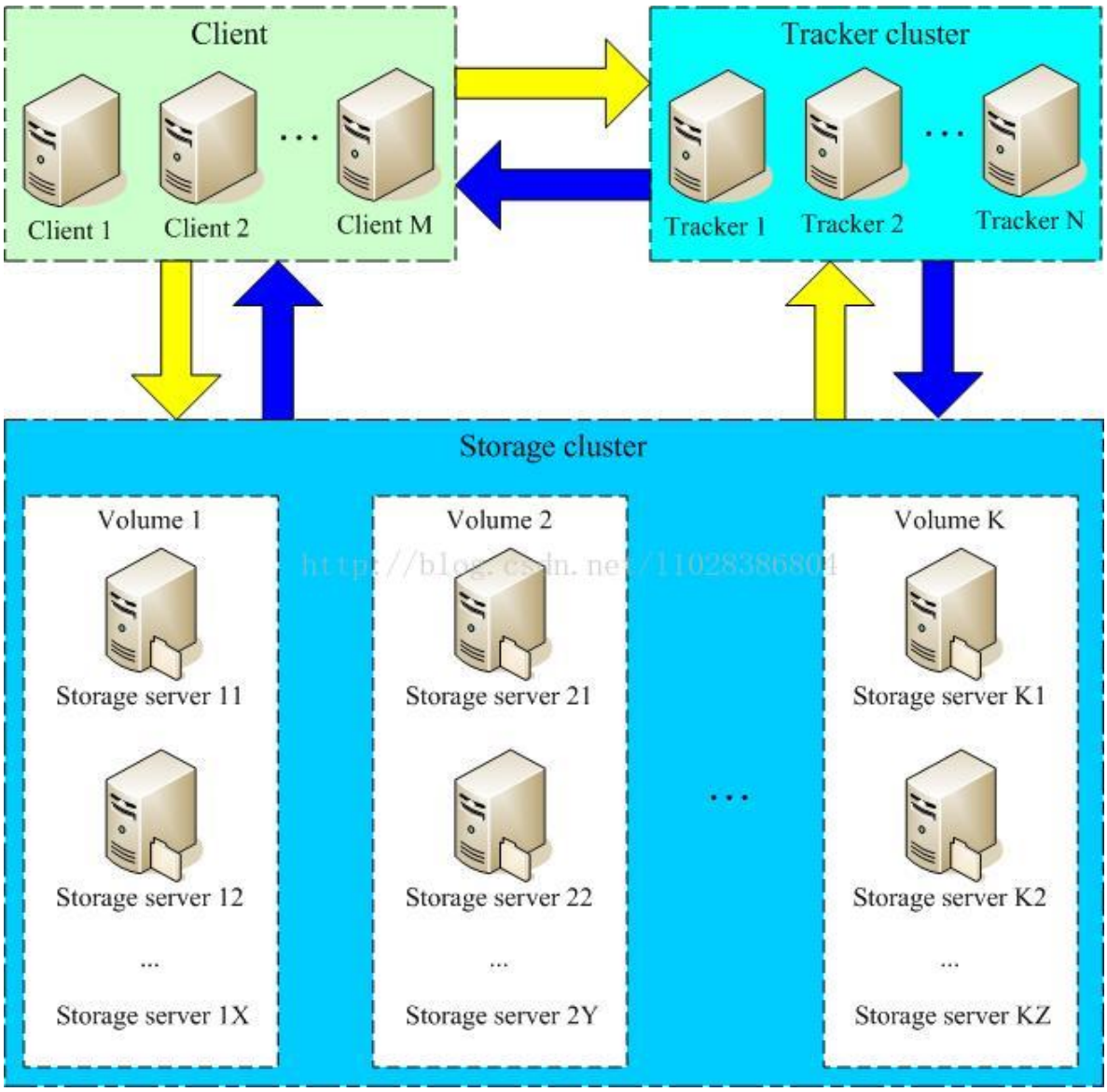

**注意： 千万不要使用 kill -9 命令强杀 FastDFS 进程，否则可能会导致 binlog 数据丢失。另外，大家可以到链接http://download.csdn.net/detail/l1028386804/9841444下载FastDFS_v5.05_安装包、工具包**

# CentOS6.X搭建FastDFS高可用环境

## FastDFS 介绍

参考： http://www.oschina.net/p/fastdfs

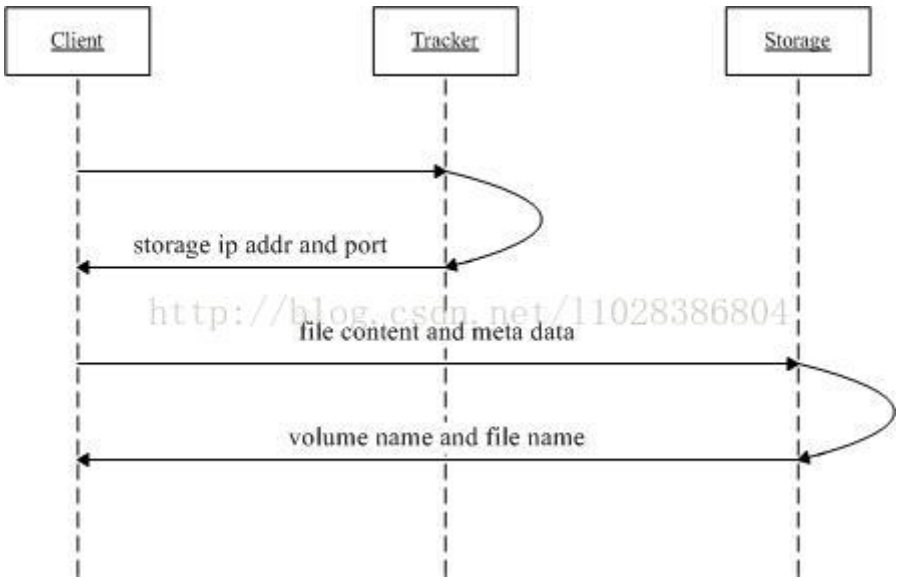
FastDFS 是一个开源的分布式文件系统，它对文件进行管理，功能包括：文件存储、文件同步、文件访问（文件上传、文件下载）等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服务，如相册网站、视频网站等等。FastDFS 服务端有两个角色：跟踪器（tracker）和存储节点（storage）。跟踪器主要做调度工作，在访问上起负载均衡的作用。存储节点存储文件，完成文件管理的所有功能：存储、同步和提供存取接口，FastDFS 同时对文件的 meta data 进行管理。所谓文件的 meta data 就是文件的相关属性，以键值对（keyvalue pair）方式表示，如： width=1024，其中的 key 为 width， value 为 1024。文件 meta data 是文件属性列表，可以包含多个键值对。 FastDFS 系统结构如下图所示：




跟踪器和存储节点都可以由一台多台服务器构成。跟踪器和存储节点中的服务器均可以随时增加或下线而不会影响线上服务。其中跟踪器中的所有服务器都是对等的，可以根据服务器的压力情况随时增加或减少。

为了支持大容量，存储节点（服务器）采用了分卷（或分组）的组织方式。存储系统由一个或多个卷组成，卷与卷之间的文件是相互独立的，所有卷的文件容量累加就是整个存储系统中的文件容量。一个卷可以由一台或多台存储服务器组成，一个卷下的存储服务器中的文件都是相同的，卷中的多台存储服务器起到了冗余备份和负载均衡的作用。在卷中增加服务器时，同步已有的文件由系统自动完成，同步完成后，系统自动将新增服务器切换到线上提供服务。当存储空间不足或即将耗尽时，可以动态添加卷。只需要增加一台或多台服务器，并将它们配置为一个新的卷，这样就扩大了存储系统的容量。 FastDFS中的文件标识分为两个部分：卷名和文件名，二者缺一不可。

**FastDFS 上传文件交互过程**



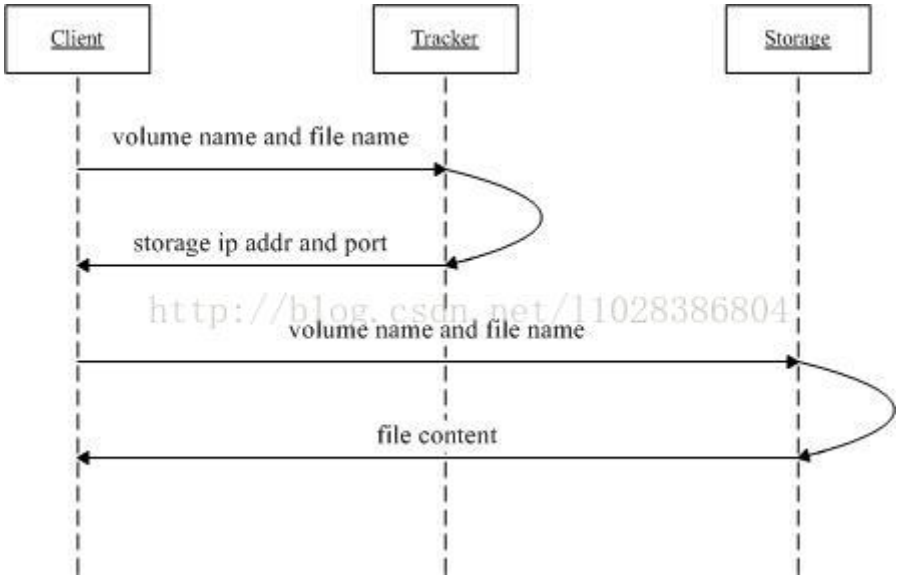

（1）client 询问 tracker 上传到的 storage，不需要附加参数；

（2）tracker 返回一台可用的 storage；

（3）client 直接和 storage 通讯完成文件上传。

客户端 client 发起对 FastDFS 的文件传输动作，是通过连接到某一台 Tracker Server 的指定端口来实现的， Tracker Server 根据目前已掌握的信息，来决定选择哪一台 Storage Server ，然后将这个 Storage Server 的地址等信息返回给 client，然后 client 再通过这些信息连接到这台 Storage Server，将要上传的文件传送到给 Storage Server 上

**FastDFS 下载文件交互过程：**




（1）client 询问 tracker 下载文件的 storage，参数为文件标识（卷名和文件名）；

（2）tracker 返回一台可用的 storage；

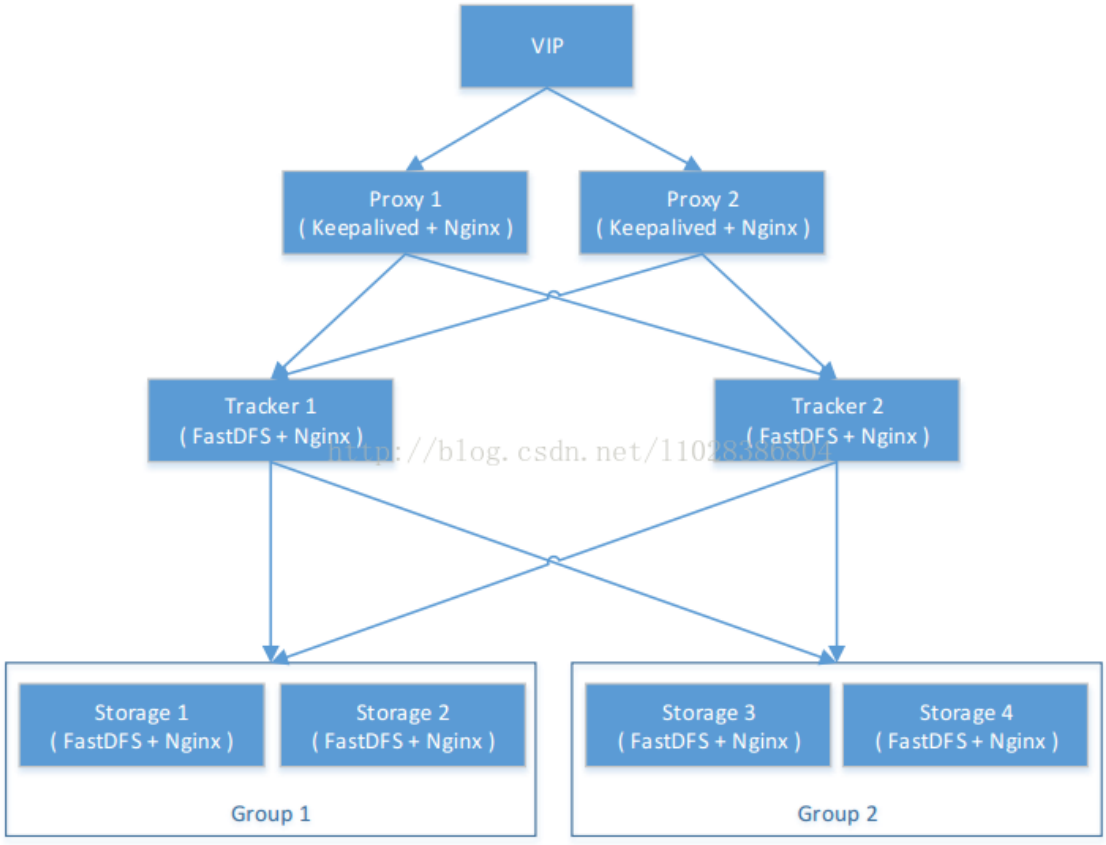
（3）client 直接和 storage 通讯完成文件下载。

## FastDFS 集群规划

跟踪服务器 1： 192.168.50.135 liuyazhuang-tracker-1
跟踪服务器 2： 192.168.50.136 liuyazhuang-tracker-2
存储服务器 1： 192.168.50.137 liuyazhuang-storage-group1-1
存储服务器 2： 192.168.50.138 liuyazhuang-storage-group1-2
存储服务器 3： 192.168.50.139 liuyazhuang-storage-group2-1

存储服务器 4： 192.168.50.140 liuyazhuang-storage-group2-2
环境： CentOS 6.5
用户： root
数据目录： /fastdfs （注： 数据目录按你的数据盘挂载路径而定）
安装包：到链接http://download.csdn.net/detail/l1028386804/9855529下载
FastDFS v5.05
libfastcommon-master.zip（是从 FastDFS 和 FastDHT 中提取出来的公共 C 函数库）
fastdfs-nginx-module_v1.16.tar.gz
nginx-1.6.2.tar.gz
fastdfs_client_java._v1.25.tar.gz

源码地址： https://github.com/happyfish100/
下载地址： http://sourceforge.net/projects/fastdfs/file
官方论坛： http://bbs.chinaunix.net/forum-240-1.html



FastDFS集群

## FastDFS 的安装（所有跟踪服务器和存储服务器均执行如下操作）

### 1、 编译和安装所需的依赖包

```
# yum install make cmake gcc gcc-c++
```

### 2、 安装 libfastcommon

https://github.com/happyfish100/libfastcommon

(1)上传或下载 libfastcommon-master.zip 到/usr/local/src 目录， 解压

```
# cd /usr/local/src/
# unzip libfastcommon-master.zip
# cd libfastcommon-master
```

(2) 编译、安装

```
# ./make.sh
# ./make.sh install
```

libfastcommon 默认安装到了

```
/usr/lib64/libfastcommon.so
/usr/lib64/libfdfsclient.so
```

(3)建立软链接

因为 FastDFS 主程序设置的 lib 目录是/usr/local/lib， 所以需要创建软链接.

```
# ln -s /usr/lib64/libfastcommon.so /usr/local/lib/libfastcommon.so
# ln -s /usr/lib64/libfastcommon.so /usr/lib/libfastcommon.so
# ln -s /usr/lib64/libfdfsclient.so /usr/local/lib/libfdfsclient.so
# ln -s /usr/lib64/libfdfsclient.so /usr/lib/libfdfsclient.so
```

## 3、安装 FastDFS

https://github.com/happyfish100/fastdfs/releases

(1)上传或下载 FastDFS 源码包（FastDFS_v5.05.tar.gz） 到 /usr/local/src 目录， 解压

```
# cd /usr/local/src/
# tar -zxvf FastDFS_v5.05.tar.gz
# cd FastDFS
```

(2)编译、 安装（编译前要确保已经成功安装了 libfastcommon）

```
# ./make.sh
# ./make.sh install
```

采用默认安装的方式安装,安装后的相应文件与目录:

A、 服务脚本在:

```
/etc/init.d/fdfs_storaged
/etc/init.d/fdfs_tracker
```

B、 配置文件在（样例配置文件）

```
/etc/fdfs/client.conf.sample
/etc/fdfs/storage.conf.sample
/etc/fdfs/tracker.conf.sample
```

C、 命令工具在/usr/bin/目录下的:

```
fdfs_appender_test
fdfs_appender_test1
fdfs_append_file
fdfs_crc32
```

```
fdfs_delete_file
fdfs_download_file
fdfs_file_info
fdfs_monitor
fdfs_storaged
fdfs_test
fdfs_test1
fdfs_trackerd
fdfs_upload_appender
fdfs_upload_file
stop.sh
restart.sh
```

(4)因为 FastDFS 服务脚本设置的 bin 目录是/usr/local/bin， 但实际命令安装在/usr/bin， 可以进入/user/bin 目录使用以下命令查看 fdfs 的相关命令：

```
# cd /usr/bin/
# ls | grep fdfs
```

因此需要修改 FastDFS 服务脚本中相应的命令路径，也就是把/etc/init.d/fdfs_storaged 和/etc/init.d/fdfs_tracker 两个脚本中的/usr/local/bin 修改成/usr/bin：

```
# vi /etc/init.d/fdfs_trackerd
使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin
# vi /etc/init.d/fdfs_storaged
使用查找替换命令进统一修改:%s+/usr/local/bin+/usr/bin
```

注意： 以上操作无论是配置 tracker 还是配置 storage 都是必须的，而 tracker 和 storage 的区别主要是在安装完 fastdfs 之后的配置过程中。

## 配置 FastDFS 跟踪器 Tracker (192.168.50.135 、 192.168.50.136)

### 1、 复制 FastDFS 跟踪器样例配置文件,并重命名

```
# cd /etc/fdfs/
# cp tracker.conf.sample tracker.conf
```

### 2、 编辑跟踪器配置文件

```
# vi /etc/fdfs/tracker.conf
```

修改的内容如下：

```
disabled=false #启用配置文件
port=22122 #tracker 的端口号，一般采用 22122 这个默认端口
base_path=/fastdfs/tracker #tracker 的数据文件和日志目录
```

（其它参数保留默认配置， 具体配置解释请参考官方文档说明： http://bbs.chinaunix.net/thread-1941456-1-1.html ）

### 3、 创建基础数据目录（参考基础目录 base_path 配置）

```
# mkdir -p /fastdfs/tracker
```

### 4、 防火墙中打开跟踪器端口（默认为 22122）

```
# vi /etc/sysconfig/iptables
```

添加如下端口行：

```
## FastDFS Tracker Port
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22122 -j ACCEPT
```

重启防火墙：

```
# service iptables restart
```

### 5、 启动 Tracker

```
# /etc/init.d/fdfs_trackerd start
```

初次成功启动，会在/fastdfs/tracker 目录下创建 data、 logs 两个目录） 可以通过以下两个方法查看 tracker 是否启动成功:

(1)查看 22122 端口监听情况

```
netstat -unltp|grep fdfs
```

(2)通过以下命令查看 tracker 的启动日志，看是否有错误

```
tail -100f /fastdfs/tracker/logs/trackerd.log
```

### 6、 关闭 Tracker

```
# /etc/init.d/fdfs_trackerd stop
```

### 7、 设置 FastDFS 跟踪器开机启动

```
# vi /etc/rc.d/rc.local
```

添加以下内容

```
## FastDFS Tracker
/etc/init.d/fdfs_trackerd start
```

## 配置 FastDFS 存储 ( 192.168.50.137 、 192.168.50.138 、 192.168.50.139 、 192.168.50.140 )

### 1、 复制 FastDFS 存储器样例配置文件,并重命名

```
# cd /etc/fdfs/
# cp storage.conf.sample storage.conf
```

### 2、 编辑存储器样例配置文件

以 group1 中的 storage 节点的 storage.conf 为例

```
# vi /etc/fdfs/storage.conf
```

修改的内容如下:

```
disabled=false #启用配置文件
group_name=group1 #组名（第一组为 group1， 第二组为 group2）
port=23000 #storage 的端口号,同一个组的 storage 端口号必须相同
base_path=/fastdfs/storage #设置 storage 的日志目录
store_path0=/fastdfs/storage #存储路径
store_path_count=1 #存储路径个数，需要和 store_path 个数匹配
tracker_server=192.168.50.135:22122 #tracker 服务器的 IP 地址和端口
tracker_server=192.168.50.136:22122 #多个 tracker 直接添加多条配置
http.server_port=8888 #设置 http 端口号
```

（其它参数保留默认配置， 具体配置解释请参考官方文档说明：[http://bbs.chinaunix.net/thread-1941456-1-1.html](http://bbs.chinaunix.net/thread-1941456-1-1.html) ）

### 3、 创建基础数据目录（参考基础目录 base_path 配置）

```
# mkdir -p /fastdfs/storage
```

### 4、 防火墙中打开存储器端口（默认为 23000）

```
# vi /etc/sysconfig/iptables
```

添加如下端口行:

```
## FastDFS Storage Port
-A INPUT -m state --state NEW -m tcp -p tcp --dport 23000 -j ACCEPT
```

重启防火墙:

```
# service iptables restart
```

### 5、 启动 Storage

```
# /etc/init.d/fdfs_storaged start
```

（初次成功启动，会在/fastdfs/storage 目录下创建数据目录 data 和日志目录 logs）

各节点启动动，使用 tail -f /fastdfs/storage/logs/storaged.log 命令监听存储节点日志， 可以看到存储节点链接到跟踪器，并提示哪一个为 leader 跟踪器。 同时也会看到同一组中的其他节点加入进来的日志信息。
查看 23000 端口监听情况

```
netstat -unltp|grep fdfs
```

所有 Storage 节点都启动之后，可以在任一 Storage 节点上使用如下命令查看集群信息:

```
# /usr/bin/fdfs_monitor /etc/fdfs/storage.conf
```

结果为：

```
server_count=2, server_index=0

tracker server is 192.168.1.131:22122

group count: 2

Group 1:
group name = group1
disk total space = 7934 MB
disk free space = 4829 MB
trunk free space = 0 MB
storage server count = 2
active server count = 2
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.1.135
                ip_addr = 192.168.1.135  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-08-31 06:29:18
                up time = 2015-08-31 06:29:18
                total storage = 7934 MB
                free storage = 4831 MB
                upload priority = 10
                store_path_count = 1
                subdir_count_per_path = 256
                storage port = 23000
```



可以看到存储节点状态为 ACTIVE 则可

### 6、 关闭 Storage

```
# /etc/init.d/fdfs_storaged stop
```

### 7、 设置 FastDFS 存储器开机启动

```
# vi /etc/rc.d/rc.local
```

添加：

```
## FastDFS Storage
/etc/init.d/fdfs_storaged start
```

## 文件上传测试（192.168.50.135）

### 1、 修改 Tracker 服务器中的客户端配置文件

```
# cp /etc/fdfs/client.conf.sample /etc/fdfs/client.conf
# vi /etc/fdfs/client.conf
base_path=/fastdfs/tracker
tracker_server=192.168.50.135:22122
tracker_server=192.168.50.136:22122
```

### 2、 执行如下文件上传命令

```
# /usr/bin/fdfs_upload_file /etc/fdfs/client.conf
/usr/local/src/FastDFS_v5.05.tar.gz
```

返回 ID 号:

```
group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz
```

（能返回以上文件 ID， 说明文件上传成功）

## 在各存储节点（192.168.50.137、 192.168.50.138、192.168.50.139、 192.168.50.140） 上安装 Nginx

### 1、 fastdfs-nginx-module 作用说明

FastDFS 通过 Tracker 服务器,将文件放在 Storage 服务器存储， 但是同组存储服务器之间需要进入文件复制， 有同步延迟的问题。假设 Tracker 服务器将文件上传到了 192.168.50.137，上传成功后文件 ID已经返回给客户端。此时 FastDFS 存储集群机制会将这个文件同步到同组存储 192.168.50.138，在文件还没有复制完成的情况下，客户端如果用这个文件 ID 在 192.168.50.138 上取文件,就会出现文件无法访问的错误。而 fastdfs-nginx-module 可以重定向文件连接到源服务器取文件,避免客户端由于复制延迟导致的文件无法访问错误。（解压后的 fastdfs-nginx-module 在 nginx 安装时使用）

### 2、上传 fastdfs-nginx-module_v1.16.tar.gz 到/usr/local/src， 解压

```
# cd /usr/local/src/
# tar -zxvf fastdfs-nginx-module_v1.16.tar.gz
```

### 3、 修改 fastdfs-nginx-module 的 config 配置文件

```
# vi /usr/local/src/fastdfs-nginx-module/src/config
CORE_INCS="$CORE_INCS /usr/local/include/fastdfs /usr/local/include/fastcommon/"
修改为:
CORE_INCS="$CORE_INCS /usr/include/fastdfs /usr/include/fastcommon/"
```

（注意： 这个路径修改是很重要的，不然在 nginx 编译的时候会报错的）

### 4、上传Nginx的安装包

上传当前的稳定版本 Nginx(nginx-1.13.0.tar.gz)到/usr/local/src 目录

### 5、 安装编译 Nginx 所需的依赖包

```
# yum install gcc gcc-c++ make automake autoconf libtool pcre pcre-devel zlib
zlib-devel openssl openssl-devel
```

### 6、 编译安装 Nginx（添加 fastdfs-nginx-module 模块）

```
# cd /usr/local/src/
# tar -zxvf nginx-1.13.0.tar.gz
# cd nginx-1.13.0
# ./configure --prefix=/usr/local/nginx --add-module=/usr/local/src/fastdfs-nginx-module/src
# make && make install
```

### 7、 复制 fastdfs-nginx-module 源码中的配置文件到/etc/fdfs 目录， 并修改

```
# cp /usr/local/src/fastdfs-nginx-module/src/mod_fastdfs.conf /etc/fdfs/
# vi /etc/fdfs/mod_fastdfs.conf
```

(1)第一组 Storage 的 mod_fastdfs.conf 配置如下：

```
connect_timeout=10
base_path=/tmp
tracker_server=192.168.1.131:22122
tracker_server=192.168.1.132:22122
storage_server_port=23000
group_name=group1
url_have_group_name = true
store_path0=/fastdfs/storage
group_count = 2
[group1]
group_name=group1
storage_server_port=23000
store_path_count=1
store_path0=/fastdfs/storage
[group2]
group_name=group2
storage_server_port=23000
store_path_count=1
store_path0=/fastdfs/storage
```

(2)第一组 Storage 的 mod_fastdfs.conf 配置与第一组配置只有 group_name 不同：

```
group_name=group2
```

### 8、 复制 FastDFS 的部分配置文件到/etc/fdfs 目录

```
# cd /usr/local/src/FastDFS/conf
# cp http.conf mime.types /etc/fdfs/
```

### 9、 在/fastdfs/storage 文件存储目录下创建软连接,将其链接到实际存放数据的目录

```
# ln -s /fastdfs/storage/data/ /fastdfs/storage/data/M00
```

### 10、 配置 Nginx， 简洁版 nginx 配置样例

```
# vi /usr/local/nginx/conf/nginx.conf
user root;
```

```
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;
    server {
        listen 8888;
        server_name localhost;
        location ~/group([0-9])/M00 {
            #alias /fastdfs/storage/data;
            ngx_fastdfs_module;
        }
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root html;
        }
    }
}
```

注意、 说明:
A、 8888 端口值是要与/etc/fdfs/storage.conf 中的 http.server_port=8888 相对应，因为 http.server_port 默认为 8888,如果想改成 80，则要对应修改过来。

B、 Storage 对应有多个 group 的情况下，访问路径带 group 名，如/group1/M00/00/00/xxx，对应的 Nginx 配置为:

```
location ~/group([0-9])/M00 {
    ngx_fastdfs_module;
}
```

C、 如查下载时如发现老报 404， 将 nginx.conf 第一行 user nobody 修改为 user root 后重新启动。

## 11、 防火墙中打开 Nginx 的 8888 端口

```
# vi /etc/sysconfig/iptables
添加:
## Nginx Port
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8888 -j ACCEPT
重启防火墙: # service iptables restart
```

## 12、 启动 Nginx

```
# /usr/local/nginx/sbin/nginx
ngx_http_fastdfs_set pid=xxx
```

（重启 Nginx 的命令为： /usr/local/nginx/sbin/nginx -s reload）

设置 Nginx 开机启动

```
# vi /etc/rc.local
```

加入：

```
/usr/local/nginx/sbin/nginx
```

### 13、 通过浏览器访问测试时上传的文件

```
http://192.168.50.137:8888/group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
http://192.168.50.139:8888/group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz
```

## 在跟踪器节点（192.168.50.135、 192.168.50.136）上安装 Nginx

### 1、 在 tracker 上安装的 nginx 主要为了提供 http 访问的反向代理、负载均衡以及缓存服务

### 2、 安装编译 Nginx 所需的依赖包

```
# yum install gcc gcc-c++ make automake autoconf libtool pcre pcre-devel zlib
zlib-devel openssl openssl-devel
```

### 3、上传 ngx_cache_purge-2.3.tar.gz 到/usr/local/src，解压

```
# cd /usr/local/src/
# tar -zxvf ngx_cache_purge-2.3.tar.gz
```

### 4、 上传当前的稳定版本 Nginx(nginx-1.13.0.tar.gz)到/usr/local/src 目录

### 5、 编译安装 Nginx（添加 ngx_cache_purge 模块）

```
# cd /usr/local/src/
# tar -zxvf nginx-1.13.0.tar.gz
# cd nginx-1.13.0
# ./configure --prefix=/usr/local/nginx --add-
module=/usr/local/src/ngx_cache_purge-2.3
# make && make install
```

### 6、配置 Nginx， 设置负载均衡以及缓存

```
# vi /usr/local/nginx/conf/nginx.conf
user root;
worker_processes 1;
#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;
#pid logs/nginx.pid;
events {
    worker_connections 1024;
    use epoll;
}
http {
    include mime.types;
    default_type application/octet-stream;
    #log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    # '$status $body_bytes_sent "$http_referer" '
```

```
    # '"$http_user_agent" "$http_x_forwarded_for"';
    #access_log logs/access.log main;
    sendfile on;
    tcp_nopush on;
    #keepalive_timeout 0;
    keepalive_timeout 65;
    #gzip on;
    #设置缓存
    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;
    client_max_body_size 300m;
    proxy_redirect off;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_connect_timeout 90;
    proxy_send_timeout 90;
    proxy_read_timeout 90;
    proxy_buffer_size 16k;
    proxy_buffers 4 64k;
    proxy_busy_buffers_size 128k;
    proxy_temp_file_write_size 128k;
    #设置缓存存储路径、存储方式、分配内存大小、磁盘最大空间、缓存期限
    proxy_cache_path /fastdfs/cache/nginx/proxy_cache levels=1:2
    keys_zone=http-cache:200m max_size=1g inactive=30d;
    proxy_temp_path /fastdfs/cache/nginx/proxy_cache/tmp;
    #设置 group1 的服务器
    upstream fdfs_group1 {
        server 192.168.50.137:8888 weight=1 max_fails=2 fail_timeout=30s;
        server 192.168.50.138:8888 weight=1 max_fails=2 fail_timeout=30s;
    } #
    设置 group2 的服务器
    upstream fdfs_group2 {
        server 192.168.50.139:8888 weight=1 max_fails=2 fail_timeout=30s;
        server 192.168.50.140:8888 weight=1 max_fails=2 fail_timeout=30s;
    }
    server {
        listen 8000;
        server_name localhost;
        #charset koi8-r;
        #access_log logs/host.access.log main;
        #设置 group 的负载均衡参数
        location /group1/M00 {
            proxy_next_upstream http_502 http_504 error timeout invalid_header;
            proxy_cache http-cache;
            proxy_cache_valid 200 304 12h;
            proxy_cache_key $uri$is_args$args;
            proxy_pass http://fdfs_group1;
            expires 30d;
        }
        location /group2/M00 {
            proxy_next_upstream http_502 http_504 error timeout invalid_header;
            proxy_cache http-cache;
            proxy_cache_valid 200 304 12h;
            proxy_cache_key $uri$is_args$args;
            proxy_pass http://fdfs_group2;
            expires 30d;
```

```
        }
        #设置清除缓存的访问权限
        location ~/purge(/.*) {
            allow 127.0.0.1;
            allow 192.168.1.0/24;
            deny all;
            proxy_cache_purge http-cache $1$is_args$args;
        }
        #error_page 404 /404.html;
        # redirect server error pages to the static page /50x.html
        #
        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
            root html;
        }
    }
}
```

按以上 nginx 配置文件的要求，创建对应的缓存目录：

```
# mkdir -p /fastdfs/cache/nginx/proxy_cache
# mkdir -p /fastdfs/cache/nginx/proxy_cache/tmp
```

## 7、系统防火墙打开对应的端口

```
# vi /etc/sysconfig/iptables
## Nginx
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8000 -j ACCEPT
# service iptables restart
```

## 8、 启动 Nginx

```
# /usr/local/nginx/sbin/nginx
重启 Nginx
# /usr/local/nginx/sbin/nginx -s reload
设置 Nginx 开机启动
# vi /etc/rc.local
加入： /usr/local/nginx/sbin/nginx
```

## 9、 文件访问测试

前面直接通过访问 Storage 节点中的 Nginx 的文件
http://192.168.50.137:8888/group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
http://192.168.50.139:8888/group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz

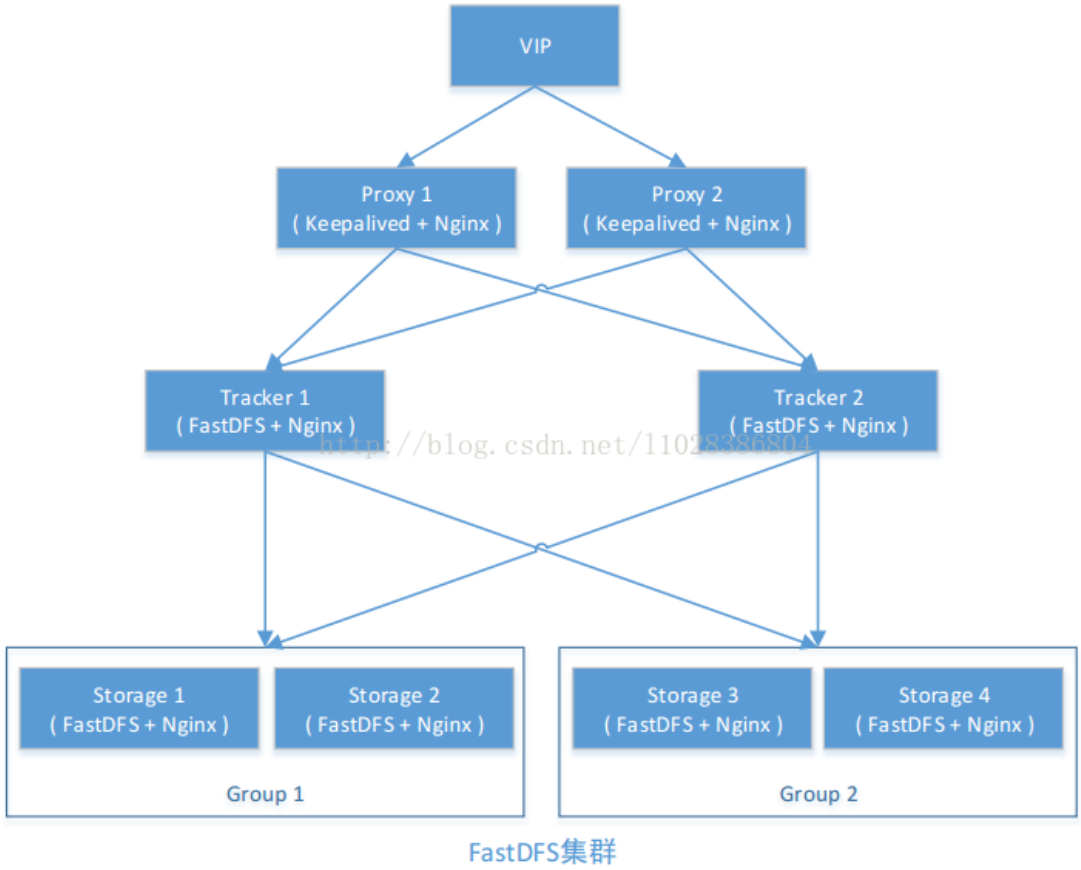
现在可以通过 Tracker 中的 Nginx 来进行访问

(1)通过 Tracker1 中的 Nginx 来访问
http://192.168.50.135:8000/group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
http://192.168.50.135:8000/group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz

(2)通过 Tracker2 中的 Nginx 来访问
http://192.168.50.136:8000/group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
http://192.168.50.136:8000/group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz

由上面的文件访问效果可以看到，每一个 Tracker 中的 Nginx 都单独对后端的 Storage 组做了负载均衡，但整套 FastDFS 集群如果想对外提供统一的文件访问地址，还需要对两个 Tracker 中的 Nginx 进行 HA 集群

## 使用 Keepalived + Nginx 组成的高可用负载均衡集群做两个 Tracker 节点中 Nginx 的负载均衡



FastDFS集群

## 1、参考博文《Keepalived之——Keepalived + Nginx 实现高可用 Web 负载均衡》

博文链接地址为：https://blog.csdn.net/l1028386804/article/details/72801492

## 2、 在 Keepalived+Nginx 实现高可用负载均衡集群中配置 Tracker 节点中 Nginx 的负载均衡反向代理

(192.168.50.133 和 192.168.50.134 中的 Nginx 执行相同的配置)

```
# vi /usr/local/nginx/conf/nginx.conf
user root;
worker_processes 1;
#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;
#pid logs/nginx.pid;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    #log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    # '$status $body_bytes_sent "$http_referer" '
```

```
    # '"$http_user_agent" "$http_x_forwarded_for"';
    #access_log logs/access.log main;
    sendfile on;
    #tcp_nopush on;
    #keepalive_timeout 0;
    keepalive_timeout 65;
    #gzip on;
    ## FastDFS Tracker Proxy
    upstream fastdfs_tracker {
        server 192.168.50.135:8000 weight=1 max_fails=2 fail_timeout=30s;
        server 192.168.50.136:8000 weight=1 max_fails=2 fail_timeout=30s;
    }
    server {
        listen 88;
        server_name localhost;
        #charset koi8-r;
        #access_log logs/host.access.log main;
        location / {
        root html;
            index index.html index.htm;
        }
        #error_page 404 /404.html;
        # redirect server error pages to the static page /50x.html
        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
            root html;
        }
        ## FastDFS Proxy
        location /dfs {
            root html;
            index index.html index.htm;
            proxy_pass http://fastdfs_tracker/;
            proxy_set_header Host $http_host;
            proxy_set_header Cookie $http_cookie;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            client_max_body_size 300m;
        }
    }
}
```

### 3、 重启 192.168.50.133 和 192.168.50.134 中的 Nginx

```
# /usr/local/nginx/sbin/nginx -s reload
```

### 4、 通过 Keepalived+Nginx 组成的高可用负载集群的 VIP(192.168.50.130)来访问 FastDFS 集群中的文件

http://192.168.50.130:88/dfs/group1/M00/00/00/wKgBh1Xtr9-AeTfWAAVFOL7FJU4.tar.gz
http://192.168.50.130:88/dfs/group2/M00/00/00/wKgBiVXtsDmAe3kjAAVFOL7FJU4.tar.gz

**注意： 千万不要使用 kill -9 命令强杀 FastDFS 进程，否则可能会导致 binlog 数据丢失。另外，大家可以到链接http://download.csdn.net/detail/l1028386804/9855529下载FastDFS集群的配置文件**

# 基于CentOS6.X服务器搭建FastDFS环境

## CentOS 8.X服务器搭建FastDFS环境

### 什么是FastDFS?

这里，我就摘录下百度百科上对于FastDFS的描述。

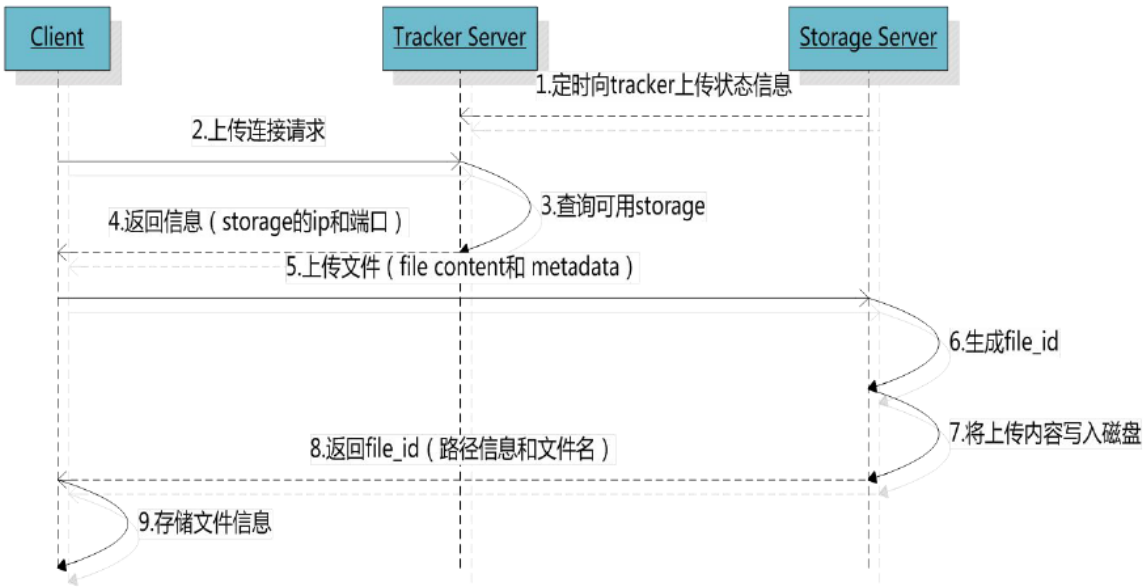
FastDFS是一个开源的轻量级分布式文件系统，它对文件进行管理，功能包括：文件存储、文件同步、文件访问（文件上传、文件下载）等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服务，如相册网站、视频网站等等。

FastDFS为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，并注重高可用、高性能等指标，使用FastDFS很容易搭建一套高性能的文件服务器集群提供文件上传、下载等服务。

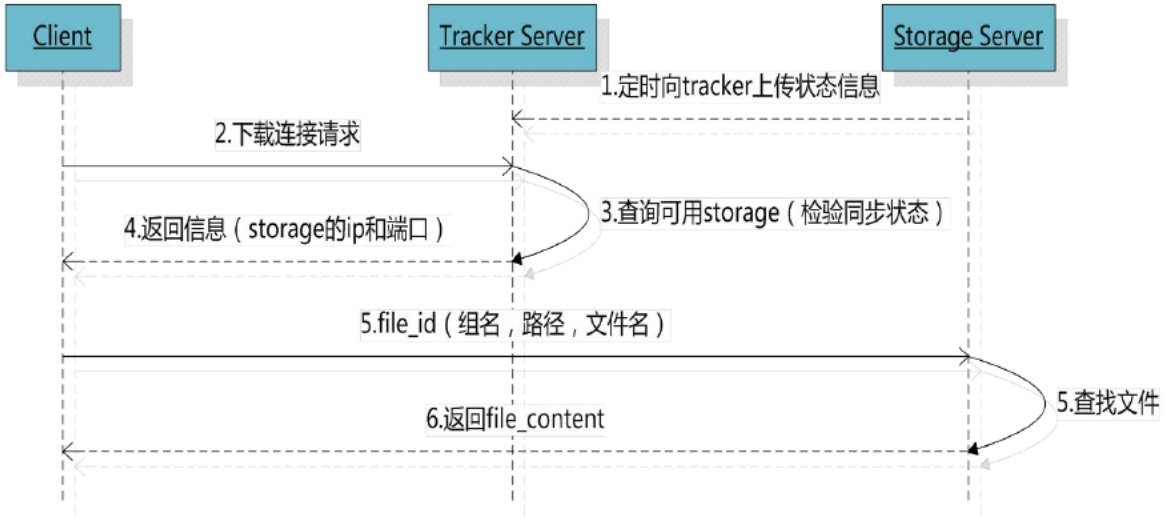
### 文件上传下载流程

这里，我们用两张图分别来说明下FastDFS文件上传和下载的过程。这样，小伙伴们也能一目了然的看到FastDFS的执行流程。

#### 文件上传




#### 文件下载




了解了FastDFS的这些基本知识之后。接下来，我们就一起来看看如何在CentOS 8服务器上搭建FastDFS环境。

## 服务器版本

在正式开始搭建FastDFS环境之前，我们先确定下服务器的版本，这里我使用的CentOS服务器的内核版本为：release 8.1.1911，如下所示。

```
[root@binghe lib]# cat /etc/redhat-release
CentOS Linux release 8.1.1911 (Core)
```

## 下载FastDFS

这里，我们使用的FastDFS版本为6.0.6，官方的地址为：https://github.com/happyfish100

在FastDFS 6.0.6中，有三个子模块，如下所示。

```
fastdfs v6.06
libfastcommon v1.0.43
fastdfs-nginx-module   v 1.22
```

我们可以在CentOS 8服务器的命令行执行如下命令来下载这些模块。

```
[root@binghe source]# wget
https://github.com/happyfish100/fastdfs/archive/V6.06.tar.gz
[root@binghe source]# wget https://github.com/happyfish100/fastdfs-nginx-
module/archive/V1.22.tar.gz
[root@binghe source]# wget
https://github.com/happyfish100/libfastcommon/archive/V1.0.43.tar.gz
```

## 下载Nginx

Nginx的官方网址为：http://nginx.org/

我们可以在CentOS 8服务器命令行输入如下命令下载Nginx。

```
[root@binghe source]# wget http://nginx.org/download/nginx-1.17.8.tar.gz
```

## 安装FastDFS依赖

```
[root@binghe dest]# yum install gcc gcc-c++
[root@binghe dest]# yum install libtool zlib zlib-devel openssl openssl-devel
[root@binghe dest]# yum -y install pcre pcre-devel libevent libevent-devel perl
unzip net-tools wget
```

# 安装libfastcommon

## 解压libfastcommon的压缩包

```
 [root@binghe source]# tar -zxvf V1.0.43.tar.gz
```

## 编译并安装

```
[root@binghe source]# cd libfastcommon-1.0.43/
[root@binghe libfastcommon-1.0.43]# ./make.sh && ./make.sh install
```

## 测试安装结果

```
[root@binghe libfastcommon-1.0.43]# ls /usr/lib64|grep libfastcommon
libfastcommon.so
[root@binghe libfastcommon-1.0.43]# ls /usr/lib|grep libfastcommon
libfastcommon.so
```

# 编译安装fastdfs

## 解压FastDFS

```
[root@binghe source]# tar -zxvf V6.06.tar.gz
```

## 安装FastDFS

```
[root@binghe source]# cd fastdfs-6.06/
[root@binghe fastdfs-6.06]# ./make.sh && ./make.sh install
```

## 查看FastDFS的安装情况

```
[root@binghe fastdfs-6.06]# ls /usr/bin|grep fdfs
fdfs_appender_test
fdfs_appender_test1
fdfs_append_file
fdfs_crc32
fdfs_delete_file
fdfs_download_file
fdfs_file_info
fdfs_monitor
fdfs_regenerate_filename
fdfs_storaged
fdfs_test
fdfs_test1
fdfs_trackerd
fdfs_upload_appender
fdfs_upload_file
```

### 修改FastDFS配置文件

```
[root@binghe fastdfs-6.06]# cd /etc/fdfs/
[root@binghe fdfs]# cp storage.conf.sample storage.conf
[root@binghe fdfs]# cp client.conf.sample client.conf
[root@binghe fdfs]# cp tracker.conf.sample tracker.conf
```

## 启动FastDFS

### 启动tracker服务

#### （1）创建tracker服务所需的目录

```
[root@binghe fdfs]# mkdir /data/fastdfs
[root@binghe fdfs]# mkdir /data/fastdfs/tracker
[root@binghe fdfs]# chmod 777 /data/fastdfs/tracker
```

#### （2）配置tracker服务

修改 tracker.conf 文件。

```
[root@binghe fdfs]# vi /etc/fdfs/tracker.conf
```

只修改base_path一项的值为我们在上面所创建的目录即可。

```
  base_path = /data/fastdfs/tracker
```

#### （3）启动 tracker 服务

```
[root@binghe fdfs]# /etc/init.d/fdfs_trackerd start
```

#### （4）检查tracker服务启动是否成功

```
[root@binghe fdfs]# ps auxfww | grep fdfs
root      15067  0.0  0.0  12320   964 pts/0    S+   01:14   0:00  |   |
  \_ grep --color=auto fdfs
root      15026  0.0  0.1  90160  5940 ?        Sl   01:13   0:00
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
```

能看到 fdfs_trackerd,表示tracker服务启动成功

#### （5）检查tracker服务是否已绑定端口 22122

```
[root@binghe dest]# netstat -anp | grep 22122
tcp        0      0 0.0.0.0:22122           0.0.0.0:*               LISTEN
 15026/fdfs_trackerd
```

说明： 22122端口是在/etc/fdfs/tracker.conf中定义的。如下所示：

```
# the tracker server port
port = 22122
```

## 启动storage服务

### （1）创建storage服务所需的目录

```
[root@binghe fdfs]# mkdir /data/fastdfs/storage
[root@binghe fdfs]# chmod 777 /data/fastdfs/storage/
```

### （2）配置storage服务

编辑storage的配置文件:

```
[root@binghe fdfs]# vi /etc/fdfs/storage.conf
```

各配置项包括：

配置base_path为上面所创建的storage目录，其中，store_path 为存储所在的目录，可以设置多个，注意从0开始。

```
base_path = /data/fastdfs/storage
store_path0 = /data/fastdfs/storage
```

配置tracker_server的ip和端口。

```
tracker_server = 192.168.175.100:22122
```

指定http服务的端口

```
http.server_port = 80
```

### （3）启动storage服务

```
[root@binghe fdfs]# /etc/init.d/fdfs_storaged start
正在启动 fdfs_storaged (via systemctl):                    [  确定  ]
```

### （4）检查storage服务启动是否成功?

```
[root@binghe fdfs]# ps auxfww | grep fdfs
root      15630  0.0  0.0  12320   972 pts/0    S+   15:46   0:00  |   |
  \_ grep --color=auto fdfs
root      15026  0.0  0.1 155696  6964 ?        Sl   15:13   0:00
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
root      15573  2.7  1.7 150736 66292 ?        Sl   15:45   0:02
/usr/bin/fdfs_storaged /etc/fdfs/storage.conf
```

说明：看到fdfs_storaged，表示storage服务启动成功

### （5）检查storage服务是否已绑定到端口:23000?

```
[root@binghe fdfs]# netstat -anp | grep 23000
tcp        0      0 0.0.0.0:23000           0.0.0.0:*               LISTEN
 15573/fdfs_storaged
```

说明：23000 端口是在配置文件 /etc/fdfs/storage.conf中定义的，如下所示。

```
# the storage server port
port = 23000
```

## 配置客户端文件

**（1）配置客户端要使用的client.conf**

```
[root@binghe fdfs]# vi /etc/fdfs/client.conf
```

以下两项配置用到的tracker目录和服务器地址端口

```
base_path = /data/fastdfs/tracker
tracker_server = 192.168.175.100:22122
```

（2）从客户端的配置可以看到：客户端只需要了解tracker_server的信息，Tracker server作用也正是负载均衡和调度

（3）Storage server作用是文件存储，客户端上传的文件最终存储在 Storage 服务器上。

## 安装nginx及fastdfs-nginx-module

### 解压nginx

```
[root@binghe source]# tar -zxvf nginx-1.17.8.tar.gz
```

### 解压fastdfs-nginx-module

```
[root@binghe source]# tar -zxvf V1.22.tar.gz
```

### 修改config文件

修改config文件，把/usr/local 替换成 /usr

```
[root@binghe source]# cd fastdfs-nginx-module-1.22/
[root@binghe fastdfs-nginx-module-1.22]# cd src
[root@binghe src]# vi config
```

### 配置Nginx

Nginx配置,添加fastdfs-nginx-module和http_stub_status_module 模块

```
[root@binghe fdfs]# cd /usr/local/source/nginx-1.17.8/
[root@binghe nginx-1.17.8]# ./configure --prefix=/usr/local/soft/nginx --with-
http_stub_status_module --add-module=/usr/local/source/fastdfs-nginx-module-
1.22/src/
```

### 编译安装Nginx

```
[root@binghe nginx-1.17.8]# make && make install
```

### 检查安装是否成功

```
[root@binghe nginx-1.17.8]# ls /usr/local/soft/ | grep nginx
nginx
```

### 验证Nginx配置

```
[root@binghe fdfs]# /usr/local/soft/nginx/sbin/nginx -V
nginx version: nginx/1.17.8
built by gcc 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)
configure arguments: --prefix=/usr/local/soft/nginx --with-
http_stub_status_module --add-module=/usr/local/source/fastdfs-nginx-module-
1.22/src/
```

## 配置fastdfs-nginx-module

### 复制配置文件

```
[root@binghe nginx-1.17.8]# cp /usr/local/source/fastdfs-nginx-module-
1.22/src/mod_fastdfs.conf /etc/fdfs/
```

### 编辑配置文件

```
[root@binghe nginx-1.17.8]# vi /etc/fdfs/mod_fastdfs.conf
```

配置以下几项

```
connect_timeout=10
tracker_server=192.168.175.100:22122
url_have_group_name = true
store_path0=/data/fastdfs/storage
```

### 复制Web配置文件

复制另两个web访问用到配置文件到fdfs配置目录下:

```
[root@binghe nginx-1.17.8]# cd /usr/local/source/fastdfs-6.06/conf/
[root@binghe conf]# cp http.conf /etc/fdfs/
[root@binghe conf]# cp mime.types /etc/fdfs/
```

## 配置nginx

编辑nginx的配置文件:

```
[root@binghe conf]# vi /usr/local/soft/nginx/conf/nginx.conf
```

在server listen 80 的这个server配置下面,

增加一个location

```
location ~/group([0-9]) {
    root  /data/fastdfs/storage/data;
    ngx_fastdfs_module;
}
```

# 启动nginx

## 启动Nginx

```
[root@binghe storage]# /usr/local/soft/nginx/sbin/nginx
```

## 检查nginx是否已成功启动

```
[root@binghe storage]# ps auxfww | grep nginx
root      24590  0.0  0.0  12320   980 pts/0    S+   16:44   0:00  |   |
  \_ grep --color=auto nginx
root      24568  0.0  0.0  41044   428 ?        Ss   16:44   0:00  \_ nginx:
master process /usr/local/soft/nginx/sbin/nginx
nobody    24569  0.0  0.1  74516  4940 ?        S    16:44   0:00      \_ nginx:
worker process
```

# 测试图片上传

### （1）命令行上传图片

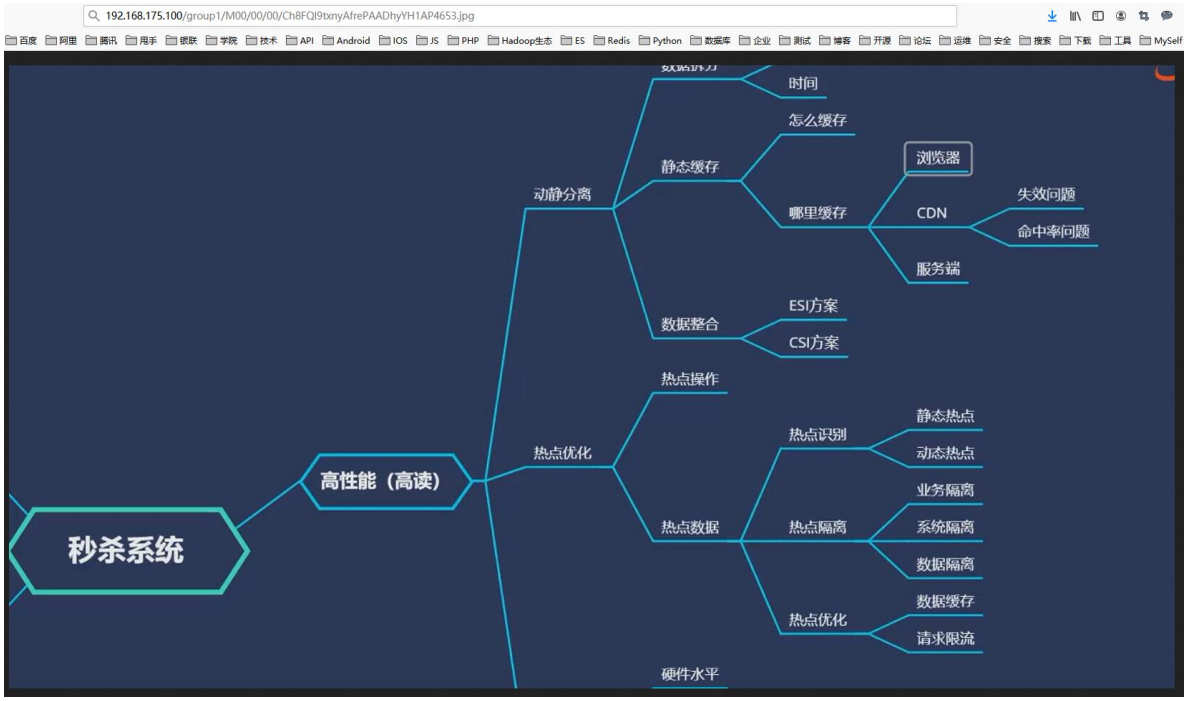
```
[root@binghe storage]# /usr/bin/fdfs_upload_file /etc/fdfs/client.conf
/home/binghe/image/test.jpg
group1/M00/00/00/Ch8FQl9txnyAfrePAADhyYH1AP4653.jpg
```

注意fdfs所返回的地址，我们需要使用这个地址进行访问

### （2）打开浏览器：访问这个地址

```
http://192.168.175.100/group1/M00/00/00/Ch8FQl9txnyAfrePAADhyYH1AP4653.jpg
```

可以看到图片




其中，192.168.175.100是拼接的本地服务器的ip

我门可以通过命令行来查看图片在服务器上的真实存储路径。

```
[root@binghe data]# pwd
/data/fastdfs/storage/data
[root@binghe data]# ls 00/00
Ch8FQl9txnyAfrePAADhyYH1AP4653.jpg
```

可见/group1/M00这两个目录是由fdfs所管理生成的目录，它们分别代表fdfs生成的组名和磁盘

## FastDFS其他命令

### 查看fdfs的版本

```
[root@binghe data]# fdfs_monitor /etc/fdfs/client.conf | grep version
[2020-09-24 01:58:01] DEBUG - base_path=/data/fastdfs/tracker,
connect_timeout=5, network_timeout=60, tracker_server_count=1,
anti_steal_token=0, anti_steal_secret_key length=0, use_connection_pool=0,
g_connection_pool_max_idle_time=3600s, use_storage_id=0, storage server id
count: 0

version = 6.06
```

### 查看fdfs的运行状态

```
[root@binghe data]# fdfs_monitor /etc/fdfs/client.conf
```

# CentOS 8.X服务器搭建FastDFS高可用环境

## 服务器版本

我们在服务器的命令行输入如下命令来查看服务器的内核版本。

```
[root@localhost lib]# cat /etc/redhat-release
CentOS Linux release 8.1.1911 (Core)
```

可以看到，集群中每台服务器的内核版本为：release 8.1.1911 (Core)。

## 服务器规划

这里，我们总共规划了6台服务器，分别为：2台tracker服务器，4台storage服务器，其中2台storage服务器为group1，两台storage服务器为group2。具体如下所示。

- tracker服务器

tranker1：192.168.175.101

tracker2：192.168.175.102

- storage服务器

storage1：192.168.175.103 group1

storage2：192.168.175.104 group1

storage3：192.168.175.105 group2

storage4：192.168.175.106 group2

# 环境准备

## 下载FastDFS

在每台服务器上执行如下命令下载FastDFS。

```
[root@localhost source]# wget
https://github.com/happyfish100/fastdfs/archive/V6.06.tar.gz
[root@localhost source]# wget https://github.com/happyfish100/fastdfs-nginx-
module/archive/V1.22.tar.gz
[root@localhost source]# wget
https://github.com/happyfish100/libfastcommon/archive/V1.0.43.tar.gz
```

## 安装环境依赖

在每台服务器上执行如下命令安装FastDFS所依赖的环境。

```
[root@localhost dest]# yum install gcc gcc-c++
[root@localhost dest]# yum install libtool zlib zlib-devel openssl openssl-devel
[root@localhost dest]# yum -y install pcre pcre-devel libevent libevent-devel
perl unzip net-tools wget
```

# 安装FastDFS

## 安装libfastcommon

在每台服务器上依次执行如下命令。

（1）解压libfastcommon的压缩包

```
[root@localhost source]# tar -zxvf V1.0.43.tar.gz
```

（2）编译并安装编译并安装

```
[root@localhost source]# cd libfastcommon-1.0.43/
[root@localhost libfastcommon-1.0.43]# ./make.sh && ./make.sh install
```

（3）检查执行的结果，看安装是否成功

```
[root@localhost libfastcommon-1.0.43]# ls /usr/lib64|grep libfastcommon
libfastcommon.so

[root@localhost libfastcommon-1.0.43]# ls /usr/lib|grep libfastcommon
libfastcommon.so
```

## 安装fastdfs

在每台服务器上依次执行如下命令。

（1）解压fastdfs

```
[root@localhost source]# tar -zxvf V6.06.tar.gz
```

（2）安装fastdfs

```
[root@localhost source]# cd fastdfs-6.06/
[root@localhost fastdfs-6.06]# ./make.sh && ./make.sh install
```

（3）检查fastdfs是否安装成功

```
[root@localhost fastdfs-6.06]# ls /usr/bin|grep fdfs
fdfs_appender_test
fdfs_appender_test1
fdfs_append_file
fdfs_crc32
fdfs_delete_file
fdfs_download_file
fdfs_file_info
fdfs_monitor
fdfs_regenerate_filename
fdfs_storaged
fdfs_test
fdfs_test1
fdfs_trackerd
fdfs_upload_appender
fdfs_upload_file
```

# 安装部署tracker服务

## 复制tracker的配置文件

在两台tracker服务器上，依次执行如下命令。

```
[root@localhost fastdfs-6.06]# cd /etc/fdfs/
[root@localhost fdfs]# cp client.conf.sample client.conf
[root@localhost fdfs]# cp tracker.conf.sample tracker.conf
```

**注意：无须生成storage.conf文件，这两台tracker不做为storage。**

## 安装Nginx

在两台tracker服务器上，依次执行如下命令。

**注意：tracker上不需要安装fastdfs-nginx-module**

（1）解压Nginx

```
[root@localhost source]# tar -zxvf nginx-1.17.8.tar.gz
```

（2）nginx配置,http_stub_status_module 模块

```
[root@localhost fdfs]# cd /usr/local/source/nginx-1.17.8/
[root@localhost nginx-1.17.8]# ./configure --prefix=/usr/local/soft/nginx --
with-http_stub_status_module
```

（3）编译安装Nginx

```
[root@localhost nginx-1.17.8]# make && make install
```

（4）检查安装是否成功

```
[root@localhost nginx-1.17.8]# ls /usr/local/soft/ | grep nginx
nginx
```

（5）查看指定的编译参数是否起作用

```
[root@localhost fdfs]# /usr/local/soft/nginx/sbin/nginx -V
nginx version: nginx/1.17.8
built by gcc 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)
configure arguments: --prefix=/usr/local/soft/nginx --with-
http_stub_status_module
```

## 配置并启动FastDFS

在两台tracker上，配置并启动FastDFS。

（1）创建tracker服务所需的目录

```
[root@localhost fdfs]# mkdir /data/fastdfs
[root@localhost fdfs]# mkdir /data/fastdfs/tracker
[root@localhost fdfs]# chmod 777 /data/fastdfs/tracker
```

（2）配置tracker服务，修改 tracker.conf 文件

```
[root@localhost fdfs]# vi /etc/fdfs/tracker.conf
```

只修改base_path一项的值为我们在上面所创建的目录即可

```
 base_path = /data/fastdfs/tracker
```

（3）启动 tracker 服务

```
[root@localhost fdfs]# /etc/init.d/fdfs_trackerd start
```

（4）检查tracker服务启动是否成功

```
[root@localhost fdfs]# ps auxfww | grep fdfs
root      15067  0.0  0.0  12320   964 pts/0    S+   15:14   0:00  |   |
  \_ grep --color=auto fdfs
root      15026  0.0  0.1  90160  5940 ?        Sl   15:13   0:00
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
```

说明：能看到 fdfs_trackerd,表示tracker服务启动成功。

（5）检查tracker服务是否已绑定端口 22122

```
[root@localhost dest]# netstat -anp | grep 22122
tcp        0      0 0.0.0.0:22122           0.0.0.0:*               LISTEN
 15026/fdfs_trackerd
```

说明： 22122端口是在/etc/fdfs/tracker.conf中定义的，如下所示：

```
# the tracker server port
port = 22122
```

## 配置client.conf

两台tracker上，配置client.conf,配置fastdfs的客户端使用的配置文件。

（1）配置client.conf

```
[root@localhost fdfs]# vi /etc/fdfs/client.conf
```

以下两项配置用到的tracker目录和服务器地址端口

```
base_path = /data/fastdfs/tracker
tracker_server = 192.168.175.101:22122
tracker_server = 192.168.175.102:22122
```

说明：两台tracker上的client.conf配置相同

（2）从客户端的配置可以看到：客户端只需要了解tracker_server的信息。tracker server作用也正是负载均衡和调度

（3）Storage server作用是文件存储，客户端上传的文件最终存储在 Storage 服务上

## 测试文件

用client.conf上传文件测试。

（1）从tacker上传一个文件

```
[root@0268c2dc2bf6 ~]# /usr/bin/fdfs_upload_file /etc/fdfs/client.conf
/root/a.txt
group1/M00/00/00/rBEABF5aTEeAXHF4AAAABHf4XZU792.txt
```

注意返回的是group1，我们可以group1下面的两台机器均找到此txt文件：

- storage1上

```
[root@d5d19e99e782 docker_tmp]# ls /data/fastdfs/storage/data/00/00
rBEABF5aTEeAXHF4AAAABHf4XZU792.txt
```

- storage2上

```
[root@f201111d0698 docker_tmp]# ls /data/fastdfs/storage/data/00/00
rBEABF5aTEeAXHF4AAAABHf4XZU792.txt
```

（2）指定group上传文件

如果想指定上传到某个group怎么办？例如：指定上传到group2

```
[root@0268c2dc2bf6 ~]# /usr/bin/fdfs_upload_file /etc/fdfs/client.conf
/root/a.txt 192.168.175.105:23000
group2/M00/00/00/rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
```

说明：指定group2中任一台的ip和端口即可。

（3）查看效果

- storage3上

```
[root@494ac47d63f8 fdfs]# ls /data/fastdfs/storage/data/00/00
rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
```

- storage4上

```
[root@59fa1efff362 fdfs]# ls /data/fastdfs/storage/data/00/00
rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
```

# 安装部署storage服务

## 生成默认配置文件

四台storage上：生成启动fastdfs默认的配置文件。

```
[root@localhost fastdfs-6.06]# cd /etc/fdfs/
[root@localhost fdfs]# cp storage.conf.sample storage.conf
[root@localhost fdfs]# cp client.conf.sample client.conf
```

说明：不需要生成tracker.conf,因为storage上不再运行tracker服务

## 安装Nginx

四台storage上：安装nginx及fastdfs-nginx-module

（1）解压nginx

```
[root@localhost source]# tar -zxvf nginx-1.17.8.tar.gz
```

（2）解压fastdfs-nginx-module

```
[root@localhost source]# tar -zxvf V1.22.tar.gz
```

（3）修改config文件，把/usr/local 替换成 /usr

```
[root@localhost source]# cd fastdfs-nginx-module-1.22/
[root@localhost fastdfs-nginx-module-1.22]# cd src
[root@localhost src]# vi config
```

（4）Nginx配置,添加fastdfs-nginx-module和http_stub_status_module 模块

```
[root@localhost fdfs]# cd /usr/local/source/nginx-1.17.8/
[root@localhost nginx-1.17.8]# ./configure --prefix=/usr/local/soft/nginx --
with-http_stub_status_module --add-module=/usr/local/source/fastdfs-nginx-
module-1.22/src/
```

（5）编译安装nginx

```
[root@localhost nginx-1.17.8]# make && make install
```

（6）检查安装是否成功

```
[root@localhost nginx-1.17.8]# ls /usr/local/soft/ | grep nginx
nginx
```

（7）查看指定的编译参数是否起作用

```
[root@localhost fdfs]# /usr/local/soft/nginx/sbin/nginx -V
nginx version: nginx/1.17.8
built by gcc 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)
configure arguments: --prefix=/usr/local/soft/nginx --with-
http_stub_status_module --add-module=/usr/local/source/fastdfs-nginx-module-
1.22/src/
```

## 配置并启动storage服务

四台storage上：配置并启动storage服务

（1）创建storage服务所需的目录

```
[root@localhost fdfs]# mkdir /data/fastdfs/storage
[root@localhost fdfs]# chmod 777 /data/fastdfs/storage/
```

（2）配置storage服务

编辑storage的配置文件:

```
[root@localhost fdfs]# vi /etc/fdfs/storage.conf
```

各配置项包括:

```
group_name = group1
#配置base_path为上面所创建的storage目录
base_path = /data/fastdfs/storage
#store_path ：存储所在的目录，可以设置多个，注意从0开始
store_path0 = /data/fastdfs/storage
#tracker_server的ip和端口
tracker_server = 192.168.175.101:22122
tracker_server = 192.168.175.102:22122
#指定http服务的端口
http.server_port = 80
```

配置的不同之处:

```
192.168.175.103   group_name = group1
192.168.175.104   group_name = group1
192.168.175.105   group_name = group2
192.168.175.106   group_name = group2
```

（3）启动storage服务

```
[root@localhost fdfs]# /etc/init.d/fdfs_storaged start
正在启动 fdfs_storaged (via systemctl):                    [  确定  ]
```

（4）检查storage服务启动是否成功

```
[root@localhost fdfs]# ps auxfww | grep fdfs
root      15630  0.0  0.0  12320   972 pts/0    S+   15:46   0:00  |   |
  \_ grep --color=auto fdfs
root      15026  0.0  0.1 155696  6964 ?        Sl   15:13   0:00
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
root      15573  2.7  1.7 150736 66292 ?        Sl   15:45   0:02
/usr/bin/fdfs_storaged /etc/fdfs/storage.conf
```

说明：看到fdfs_storaged,表示storage服务启动成功

（5）检查storage服务是否已绑定到端口：23000

```
[root@localhost fdfs]# netstat -anp | grep 23000
tcp        0      0 0.0.0.0:23000           0.0.0.0:*               LISTEN
 15573/fdfs_storaged
```

说明：23000 端口是在配置文件 /etc/fdfs/storage.conf中定义的，如下:

```
# the storage server port
port = 23000
```

## 配置fastdfs-nginx-module

四台存储服务器上：配置fastdfs-nginx-module

（1）生成配置文件

```
[root@localhost nginx-1.17.8]# cp /usr/local/source/fastdfs-nginx-module-
1.22/src/mod_fastdfs.conf /etc/fdfs/
```

（2）编辑配置文件

```
[root@localhost nginx-1.17.8]# vi /etc/fdfs/mod_fastdfs.conf
```

配置以下几项

```
group_name=group1
connect_timeout=10
tracker_server=192.168.175.101:22122
tracker_server=192.168.175.102:22122
url_have_group_name = true
```

```
store_path0=/data/fastdfs/storage
group_count = 2

[group1]

group_name=group1
storage_server_port=23000
store_path_count=1
store_path0=/data/fastdfs/storage

[group2]

group_name=group2
storage_server_port=23000
store_path_count=1
store_path0=/data/fastdfs/storage
```

说明： 最上面的group_name：当机器属于group1这组时，值为group1；当机器属于group2这组时，值为group2。

说明：url_have_group_name = true。 **注意：这一项不要漏掉，会导至nginx不正常工作**

（3）复制另两个web访问用到配置文件到fdfs配置目录下:

```
[root@d5d19e99e782 /]# cp /usr/local/source/fastdfs-6.06/conf/http.conf
/etc/fdfs/
[root@d5d19e99e782 /]# cp /usr/local/source/fastdfs-6.06/conf/mime.types
/etc/fdfs/
```

## 配置Nginx

四台存储服务器上：配置nginx

编辑nginx的配置文件:

```
[root@localhost conf]# vi /usr/local/soft/nginx/conf/nginx.conf
```

在server listen 80 的这个server配置下面，

增加一个location

```
location ~/group([0-9]) {
    root  /data/fastdfs/storage/data;
    ngx_fastdfs_module;

}
```

## 启动nginx

（1）启动Nginx

```
[root@localhost storage]# /usr/local/soft/nginx/sbin/nginx
```

（2）检查nginx是否已成功启动

```
[root@localhost storage]# ps auxfww | grep nginx
root      24590  0.0  0.0  12320   980 pts/0    S+   16:44   0:00  |   |
  \_ grep --color=auto nginx
root      24568  0.0  0.0  41044   428 ?        Ss   16:44   0:00  \_ nginx:
master process /usr/local/soft/nginx/sbin/nginx
nobody    24569  0.0  0.1  74516  4940 ?        S    16:44   0:00      \_ nginx:
worker process
```

# 配置tracker服务

## 配置tracker服务

说明：这一步等待四台storage server配置完成后再进行。使用n=Nginx做upstream负载均衡的原因：可以通过一个地址访问后端的多个group

（1）文件上传完成后，从浏览器访问各个storage的Nginx即可:

例如:

```
http://192.168.175.103/group1/M00/00/00/rBEABF5aTRiAEuHwAAAABHf4XZU322.txt
http://192.168.175.104/group1/M00/00/00/rBEABF5aTRiAEuHwAAAABHf4XZU322.txt
http://192.168.175.105/group2/M00/00/00/rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
http://192.168.175.106/group2/M00/00/00/rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
```

说明：各台storage server的ip地址后面跟着上传时所返回的地址。注意：只能访问各台机器所在的group,

- 如果想通过统一的ip地址进行访问
- 需要在Nginx中通过upstream访问到后端的机器
- 此Nginx应运行在tracker上

（2）配置nginx.conf

```
[root@0268c2dc2bf6 ~]# vi /usr/local/soft/nginx/conf/nginx.conf
```

内容:

添加 upstream到后端的storage。

```
upstream fdfs_group1 {
                server 192.168.175.103:80 weight=1  max_fails=2
fail_timeout=30s;
                server 192.168.175.104:80 weight=1  max_fails=2
fail_timeout=30s;
   }

   upstream fdfs_group2 {
                server 192.168.175.105:80 weight=1 max_fails=2 fail_timeout=30s;
                server 192.168.175.106:80 weight=1 max_fails=2 fail_timeout=30s;
   }
```

针对带有group的url进行处理

```
location /group1 {
    proxy_next_upstream http_502 http_504 error timeout invalid_header;
    proxy_pass http://fdfs_group1;
    expires 30d;
}

 location /group2 {
     proxy_next_upstream http_502 http_504 error timeout invalid_header;
     proxy_pass http://fdfs_group2;
     expires 30d;

}
```

（3）重启测试

```
[root@0268c2dc2bf6 ~]# /usr/local/soft/nginx/sbin/nginx -s stop
[root@0268c2dc2bf6 ~]# /usr/local/soft/nginx/sbin/nginx
```

在浏览器中访问:

```
http://192.168.175.101/group1/M00/00/00/rBEABF5aTRiAEuHWAAAABHf4XZU322.txt
http://192.168.175.101/group2/M00/00/00/rBEABl5aUAqAXLCZAAAABHf4XZU043.txt
```

# SpringBoot整合FastDFS实战

## 编译Java客户端

在FastDFS的官方Github上，专门有一个FastDFS Java客户端的项目，链接地址为：https://github.com/happyfish100/fastdfs-client-java。

我们将Java客户端代码下载的本地，然后进入项目的目录，使用Maven进行编译，如下所示。

```
git clone https://github.com/happyfish100/fastdfs-client-java.git
cd fastdfs-client-java
mvn clean install -Dmaven.test.skip=true
```

接下来，我们需要将FastDFS的Java客户端编译安装到本地的Maven仓库。

```
mvn install:install-file -DgroupId=com.fastdfs -DartifactId=fastdfs-client-java
-Dversion=1.29 -Dpackaging=jar -Dfile=fastdfs-client-java-1.29-SNAPSHOT.jar
```

到此，我们就在本地编译安装了FastDFS的Java客户端。

## 搭建项目

### 编辑pom.xml文件

我们在IDEA中创建一个Maven项目，并在pom.xml文件中引入SpringBoot相关依赖和我们自己编译的FastDFS的Java客户端。最终，pom.xml文件的依赖如下所示。

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <skip_maven_deploy>false</skip_maven_deploy>
    <java.version>1.8</java.version>
    <logback.version>1.1.7</logback.version>
    <slf4j.version>1.7.21</slf4j.version>
    <common.logging>1.2</common.logging>
    <fastjson.version>1.2.51</fastjson.version>
    <fastdfs.client.version>1.29</fastdfs.client.version>
</properties>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.6.RELEASE</version>
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-tomcat</artifactId>
            </exclusion>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-undertow</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-configuration-processor</artifactId>
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
        <version>${common.logging}</version>
    </dependency>

    <!-- log -->
    <dependency>
```

```
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${slf4j.version}</version>
    </dependency>

    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>${logback.version}</version>
    </dependency>

    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
        <version>${fastjson.version}</version>
    </dependency>

    <dependency>
        <groupId>com.fastdfs</groupId>
        <artifactId>fastdfs-client-java</artifactId>
        <version>${fastdfs.client.version}</version>
    </dependency>
</dependencies>
```

### 创建配置文件

（1）在项目的src/main/resources目录下创建SpringBoot的配置文件application.yml，文件内容如下所示。

```yaml
server:
  port: 9999
  servlet:
    context-path: /resource
  tomcat:
    uri-encoding: UTF-8

spring:
  servlet:
    multipart:
      max-file-size: 1024MB
      max-request-size: 1024MB
  main:
    allow-bean-definition-overriding: true
  profiles:
    include: test
    active: test
  output:
    ansi:
      enabled: detect
```

文件指定了项目启动后监听的端口，访问的根路径、项目编码、文件上传的大小，并指定了运行时的环境。

（2）在项目的src/main/resources目录下创建logback-spring.xml日志文件，具体配置见源码工程。

（3）在项目的src/main/resources目录下创建fastdfs_client.conf文件，主要用来配置与FastDFS的连接信息。

```
connect_timeout = 200
network_timeout = 3000
charset = UTF-8
http.tracker_http_port = 8080
http.anti_steal_token = no
http.secret_key = FastDFS1234567890
tracker_server = 192.168.175.100:22122
```

至此，项目搭建完成。接下来，我们就一起实现项目的功能。

# 项目开发

## 创建工具类

首先，我们在项目的io.mykit.fastdfs.utils包下创建FastDFSClientUtils工具类。这里，我给出工具类的核心实现，其他部分小伙伴们参加源码工程。

```
/**
 * @author binghe
 * @description FastDFS分布式文件系统操作客户端
 */
public class FastDFSClientUtils {

    private static Logger logger =
LoggerFactory.getLogger(FastDFSClientUtils.class);

    private static TrackerClient trackerClient;

    public static void setFile(String filePath) {
        try {
            logger.info("初始化分布式文件系统服务开始...");
            if(filePath == null || filePath.trim().isEmpty()) {
                filePath = "fastdfs_client.conf";
            }
            ClientGlobal.init(filePath);
            TrackerGroup trackerGroup = ClientGlobal.g_tracker_group;
            trackerClient = new TrackerClient(trackerGroup);
            logger.info("初始化分布式文件系统服务完成...");
        } catch (Exception e) {
            logger.error("加载文件异常：{}",e );
        }
    }

    /**
     * @param data 数据
     * @param extName 文件扩展名
     * @return 上传成功返回id，失败返回null
     */
    public static String upload(byte[] data, String extName) {
        TrackerServer trackerServer = null;
        StorageServer storageServer = null;
        StorageClient1 storageClient1 = null;
        try {
            NameValuePair[] meta_list = null; // new NameValuePair[0];

            trackerServer = trackerClient.getTrackerServer();
```

```
            if (trackerServer == null) {
                logger.error("getConnection return null");
            }
            storageServer = trackerClient.getStoreStorage(trackerServer);
            storageClient1 = new StorageClient1(trackerServer, storageServer);
            String fileid = storageClient1.upload_file1(data, extName,
meta_list);
            return fileid;
        } catch (Exception ex) {
            logger.error("上传文件异常：{}", ex);
            return null;
        }finally{
            try {
                storageClient1.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
            storageClient1 = null;
        }
    }
```

## 创建返回实体类

我们在io.mykit.fastdfs.bean包下创建ResourceBean类，用于SpringBoot接口返回结果数据，如下所示。

```
/**
 * @author binghe
 * @version 1.0.0
 * @description 上传图片后的返回数据
 */
public class ResourceBean implements Serializable {
    private static final long serialVersionUID = -2788538880352897307L;

    /**
     * 文件的访问路径
     */
    private String fileUrl;

    /**
     * 文件名称
     */
    private String fileName;

    public ResourceBean() {
        super();
    }

    public ResourceBean(String fileUrl, String fileName) {
        super();
        this.fileUrl = fileUrl;
        this.fileName = fileName;
    }

    public String getFileUrl() {
        return fileUrl;
    }
```

```
    public void setFileUrl(String fileUrl) {
        this.fileUrl = fileUrl;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

}
```

其中，定义了文件的访问路径fileUrl和文件的名称fileName。也就是说，文件上传成功后，我们会向客户端返回文件的访问路径和文件的名称信息。

### 创建常量类

在io.mykit.fastdfs.constants包下创建ResourcesConstants常量类，ResourcesConstants类中主要定义了访问文件的基础路径和获取文件完整访问路径的方法，如下所示。

```
/**
 * @author binghe
 * @version 1.0.0
 * @description 常量
 */
public class ResourcesConstants {
    private static final String BASE_RESOURCES_URL = "http://192.168.175.100/";
    public static String getResourcesUrl(String fileId) {
        return BASE_RESOURCES_URL.concat(fileId);
    }
}
```

### 创建Controller类

在项目的io.mykit.fastdfs.controller包下创建ResourceController类，用于定义文件上传的接口。这个类的功能也比较简单，就是定义一个文件上传接口，接收文件，并调用FastDFSClientUtils工具类的upload(byte[], String)方法，将文件上传到FastDFS中，如下所示。

```
/**
 * @author binghe
 * @version 1.0.0
 * @description 上传文件接口
 */
@RestController
@RequestMapping(value = "/resources/")
public class ResourceController {

    @RequestMapping(value={"/upload"}, method=RequestMethod.POST)
    @ResponseBody
    public ResourceBean upload(@RequestParam("file") MultipartFile file,
HttpServletRequest request, HttpServletResponse response){
        String extName = "";
        String fileName = "";
```

```
        String originalFilename = file.getOriginalFilename();
        if(originalFilename.contains(".")) {
            //拆分文件路径
            String[] fileArray = originalFilename.split("\\.");
            //获取文件扩展名
            extName = fileArray[1];
            //获取文件名
            fileName = fileArray[0];
        }else {
            fileName = originalFilename;
        }
        byte[] bytes = null;
        try {
            bytes = file.getBytes(); //将文件转换成字节流形式
        } catch (IOException e) {
            e.printStackTrace();
        }
        //调用上传文件的具体方法
        String fileId= FastDFSClientUtils.upload(bytes,extName);
        return new ResourceBean(ResourcesConstants.getResourcesUrl(fileId),
fileName);
    }
}
```

## 创建启动类

在项目的io.mykit.fastdfs包下，创建项目启动类ResourceStarter，如下所示。

```
/**
 * @author binghe
 * @version 1.0.0
 * @description 启动
 */
@SpringBootApplication
public class ResourceStarter {
    public static void main(String[] args) {
        try {
            String filePath = "fastdfs_client.conf";
            if(args.length > 0) {
                filePath = args[0];
            }
            FastDFSClientUtils.setFile(filePath);
            SpringApplication.run(ResourceStarter.class, args);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

从代码可以看出，ResourceStarter启动类的main方法中，为filePath变量定义了一个默认的文件路径为fastdfs_client.conf，当启动项目时，为main()方法传递了参数，则会使用第一个参数覆盖掉filePath默认的值，并调用FastDFSClientUtils类的setFile()方法将filePath传递到FastDFSClientUtils类中进行初始化操作。

## 创建html文件

最后，我们需要创建一个index.html文件，用于测试文件上传操作。index.html文件的内容也比较简单，如下所示。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>文件上传和下载</title>
</head>
<body>
<form action='http://192.168.175.100:9999/resource/resources/upload'
method='post' enctype='multipart/form-data'>
    <input type='file' name='file'>
    <button type='submit'>上传</button>
</form>
</body>
</html>
```

至此，我们整个项目就开发完成了。

## 项目测试

首先，我们在IDEA中将mykit-fastdfs项目打包成mykit-fastdfs.jar文件，然后将mykit-fastdfs.jar文件上传到服务器的/usr/local/java目录下，同时，我们将项目的src/main/resources目录下的fastdfs_client.conf文件，复制一份到服务器的/usr/local/java目录下。

在服务器命令行输入如下命令启动mykit-fastdfs.jar。

```
nohup java -jar /usr/local/java/mykit-fastdfs.jar
/usr/local/java/fastdfs_client.conf >> /dev/null &
```

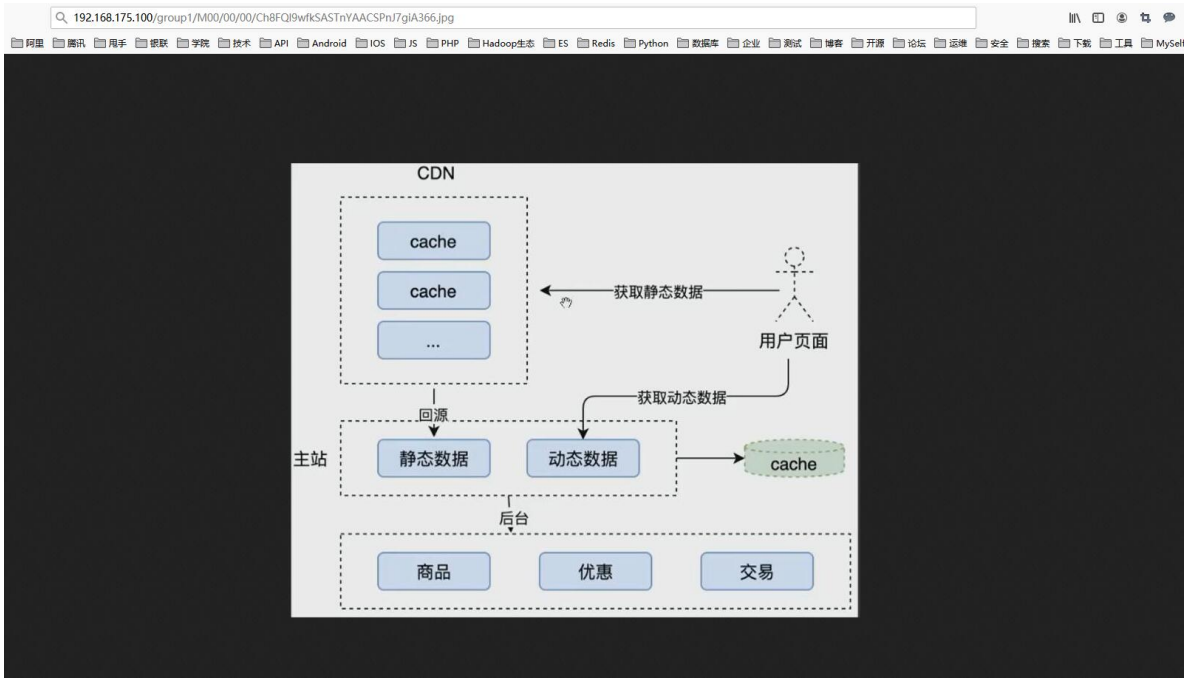
接下来，我们将index.html文件放到Nginx安装目录下的html/test目录下。此时，在浏览器地址栏中输入http://192.168.175.100/test/index.html就能够打开页面。

我们通过index.html页面将文件上传到FastDFS文件系统之后，浏览器中会显示返回的结果数据，一个是文件的访问路径fileUrl，一个是文件的名称fileName，如下所示。

```
{
    "fileUrl":
"http://192.168.175.100/group1/M00/00/00/Ch8FQl9wfkSASTnYAACSPnJ7giA366.jpg",
    "fileName": "QQ截图20200609234534"
}
```

具体如下图所示。


我们打开fileUrl标识的文件访问路径，http://192.168.175.100/group1/M00/00/00/Ch8FQl9wfkSASTnYAACSPnJ7giA366.jpg，如下所示。




可以看到，浏览器能够正确显示上传的图片，说明我们已经在项目中成功整合了FastDFS的Java客户端。

## 我的《海量数据处理与大数据技术实战》出版啦！


这本书是一本真正以实战案例为主线的大数据实战案例型书籍。这里，我给大家推荐几个购买链接。

京东购买链接：https://item.jd.com/10020420941243.html

当当购买链接：http://product.dangdang.com/29115124.html

## 重磅福利

微信搜一搜【冰河技术】微信公众号，关注这个有深度的程序员，每天阅读超硬核技术干货，公众号内回复【PDF】有我准备的一线大厂面试资料和我原创的超硬核PDF技术文档，以及我为大家精心准备的多套简历模板（不断更新中），希望大家都能找到心仪的工作，学习是一条时而郁郁寡欢，时而开怀大笑的路，加油。如果你通过努力成功进入到了心仪的公司，一定不要懈怠放松，职场成长和新技术学习一样，不进则退。如果有幸我们江湖再见！

另外，我开源的各个PDF，后续我都会持续更新和维护，感谢大家长期以来对冰河的支持！！

## 写在最后

> 如果你觉得冰河写的还不错，请微信搜索并关注「 **冰河技术** 」微信公众号，跟冰河学习高并发、分布式、微服务、大数据、互联网和云原生技术，「 **冰河技术** 」微信公众号更新了大量技术专题，每一篇技术文章干货满满！不少读者已经通过阅读「 **冰河技术** 」微信公众号文章，吊打面试官，成功跳槽到大厂；也有不少读者实现了技术上的飞跃，成为公司的技术骨干！如果你也想像他们一样提升自己的能力，实现技术能力的飞跃，进大厂，升职加薪，那就关注「 **冰河技术** 」微信公众号吧，每天更新超硬核技术干货，让你对如何提升技术能力不再迷茫！


打开“微信 / 发现 / 搜一搜”搜索