FastDFS分布式文件系统的安装与使用（单节点）

**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

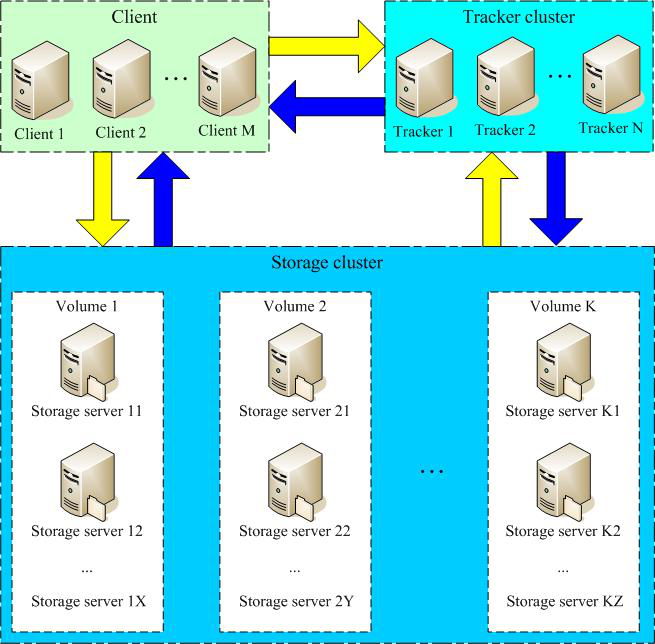
访问（文件上传、文件下载）等，解决了大容量存储和负载均衡的问题。特别适合以文件为载体的在线服

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（key

value 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。

相关的一些术语：

Tracker Server：跟踪服务器，主要做调度工作，在访问上起负载均衡的作用。记录storage server的状态，是连接Client和Storage server的枢纽。

Storage Server：存储服务器，文件和meta data都保存到存储服务器上group：组，也可称为卷。同组内服务器上的文件是完全相同的

文件标识：包括两部分：组名和文件名（包含路径）

meta data：文件相关属性，键值对（Key Value Pair）方式，如：width=1024,heigth=768

同步机制：

同一组内的storage server之间是对等的，文件上传、删除等操作可以在任意一台storage server上进行；文件同步只在同组内的storage server之间进行，采用push方式，即源服务器同步给目标服务器；源头数据才需要同步，备份数据不需要再次同步，否则就构成环路了；上述第二条规则有个例外，就是新增加一台storage server时，由已有的一台storage server将已有的所有数据（包括源头数据和备份数据）同步给该新增服务器。

运行时目录：

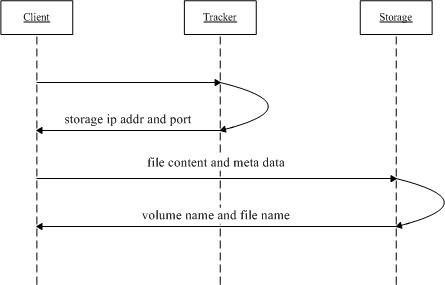
Tracker：

${base\_path}  
    |\_\_data  
    |     |\_\_storage\_groups.dat：存储分组信息  
    |     |\_\_storage\_servers.dat：存储服务器列表  
    |\_\_logs  
          |\_\_trackerd.log：tracker server日志文件

Storage：

${base\_path}  
    |\_\_data  
    |     |\_\_.data\_init\_flag：当前storage server初始化信息  
    |     |\_\_storage\_stat.dat：当前storage server统计信息  
    |     |\_\_sync：存放数据同步相关文件  
    |     |     |\_\_binlog.index：当前的binlog文件索引号  
    |     |     |\_\_binlog.###：存放更新操作记录（日志）  
    |     |     |\_\_${ip\_addr}\_${port}.mark：存放同步的完成情况  
    |     |  
    |     |\_\_一级目录：256个存放数据文件的目录，如：00, 1F  
    |           |\_\_二级目录：256个存放数据文件的目录  
    |\_\_logs  
          |\_\_storaged.log：storage server日志文件

Fastdfs上传文件交互流程：

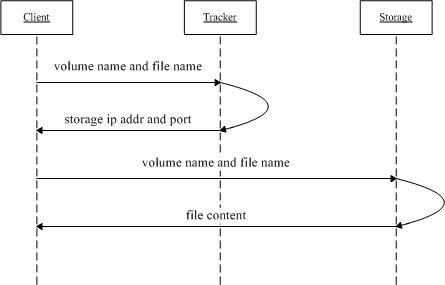


1. client询问tracker上传到的storage，不需要附加参数；

2. tracker返回一台可用的storage；

3. client直接和storage通讯完成文件上传。

Fastdfs下载文件交互流程：



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

2. tracker返回一台可用的storage；

3. client直接和storage通讯完成文件下载。

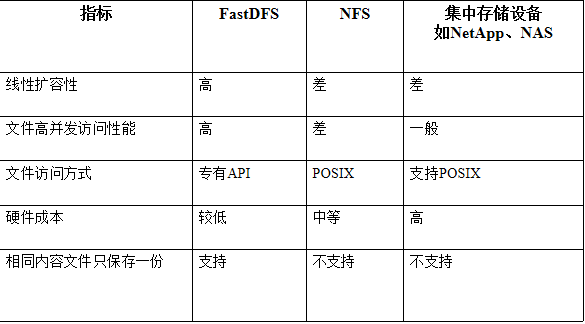
客户端 client 发起对 FastDFS 的文件传输动作，是通过连接到某一台 Tracker Server 的指定端

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

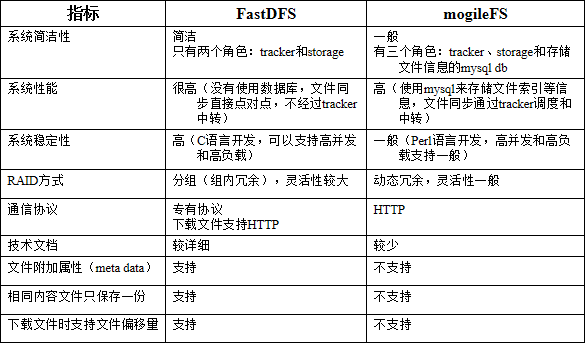
Storage Server 的地址等信息返回给 client，然后 client 再通过这些信息连接到这台 Storage Server，

将要上传的文件传送到给 Storage Server 上。

Fastdfs与集中式存储对比：



Fastdfs与mogileFS对比



**跟踪服务器 1：192.168.0.131 dfs-tracker-1**

**跟踪服务器 2：192.168.0.132 dfs-tracker-2**

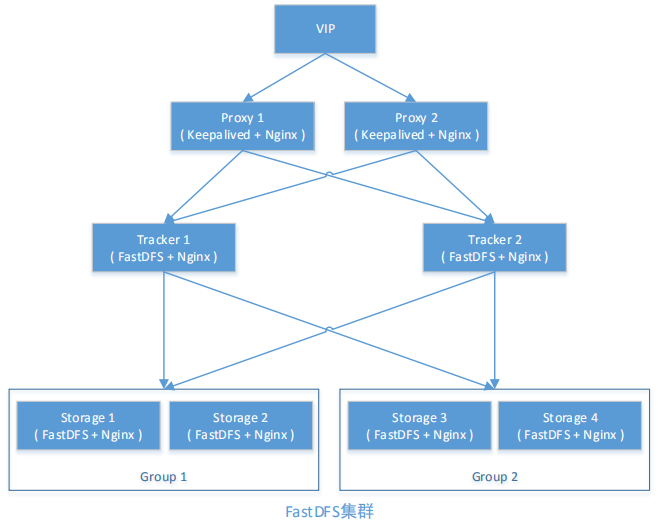
**存储服务器 1：192.168.0.135 dfs-storage-group1-1**

**存储服务器 2：192.168.0.136 dfs-storage-group1-2**

**存储服务器 3：192.168.0.137 dfs-storage-group2-1**

**存储服务器 4：192.168.0.138 dfs-storage-group2-2**

**拟搭建一个如下的集群：**

****

环境：CentOS 6.6

用户：root

数据目录：/data/fastdfs（注：数据目录按你的数据盘挂载路径而定）

安装包目录： /data/software/fastdfs-packages

安装包fastdfs最新版本6.0.3：

<https://github.com/xiongshiyan/fastdfs>

<https://github.com/xiongshiyan/libfastcommon>

（是从 FastDFS和 FastDHT中提取出来的公共 C函数库）

<https://github.com/xiongshiyan/fastdfs-nginx-module>

<https://www.nginx.com/resources/wiki/>

<https://github.com/xiongshiyan/fastdfs-client-java>

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

1. 所有跟踪服务器和存储服务器均执行如下操作
2. 编译和安装所需的依赖包:

# yum install make cmake gcc gcc-c++

1. 安装 libfastcommon:
2. 上传或者下载libfastcommon-master.zip到/data/software/fastdfs-packages目录
3. 解压

# cd /data/software/fastdfs-packages

# unzip libfastcommon-master.zip

# cd libfastcommon-master

1. 编译、安装

# ./make.sh

# ./make.sh install

libfastcommon默认安装到

/usr/lib64/libfastcommon.so

/usr/lib64/libfdfsclient.so

1. 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

1. 安装 FastDFS**（tracker、storage都需要安装，只是配置文件和启动脚本不一样）**
2. 上传或下载FastDFS源码包fastdfs-master.zip [6.0.3]到/data/software/fastdfs-packages目录
3. 解压

# cd /data/software/fastdfs-packages

# unzip fastdfs-master.zip

# cd fastdfs-master

1. 编译、安装（确保安装了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：

服务脚本在:

/etc/init.d/fdfs\_storaged

/etc/init.d/fdfs\_trackerd

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

命令工具在/usr/bin/目录下的:

/usr/bin/fdfs\_appender\_test

/usr/bin/fdfs\_appender\_test1

/usr/bin/fdfs\_append\_file

/usr/bin/fdfs\_crc32

/usr/bin/fdfs\_delete\_file

/usr/bin/fdfs\_download\_file

/usr/bin/fdfs\_file\_info

/usr/bin/fdfs\_monitor

/usr/bin/fdfs\_regenerate\_filename

/usr/bin/fdfs\_storaged

/usr/bin/fdfs\_test

/usr/bin/fdfs\_test1

/usr/bin/fdfs\_trackerd

/usr/bin/fdfs\_upload\_appender

/usr/bin/fdfs\_upload\_file

/usr/bin/restart.sh

/usr/bin/stop.sh

1. **5.0.5版本存在，6.0.3版本已修复此问题，确认即可。**~~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，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改为/usr/bin。~~

~~# vim /etc/init.d/fdfs\_storaged~~

~~使用查找替换进行统一修改： %s+/usr/local/bin+/usr/bin~~

~~# vim /etc/init.d/fdfs\_tracker~~

~~使用查找替换进行统一修改： %s+/usr/local/bin+/usr/bin~~

1. 配置fastdfs跟踪器（**192.168.0.131 、192.168.0.132**）
2. 复制fastdfs跟踪器样例配置文件，并重命名为tracker.conf。

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

1. 编辑跟踪器配置文件

# vim /etc/fdfs/tracker.conf

修改的内容如下，其他保留：

disabled=false

port=22122

base\_path=/data/fastdfs/tracker

1. 创建数据目录

# mkdir -p /data/fastdfs/tracker

1. 防火墙打开22122端口

# vim /etc/sysconfig/iptables

添加

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

重启

# service iptables restart

1. 启动tracker

# /etc/init.d/fdfs\_trackerd start

初次启动成功，会在/data/fastdfs/tracker 目录创建data、logs两个目录

tailf /fastdfs/tracker/logs/trackerd.log

1. 关闭tracker

# /etc/init.d/fdfs\_trackerd stop

1. 设置tracker开机启动

# vim /etc/rc.d/rc.local

/etc/init.d/fdfs\_trackerd start

1. 配置fastdfs存储器（192.168.0.135、192.168.0.136、192.168.0.137、192.168.0.138）
2. 复制fastdfs存储器样例配置文件重命名为storage.conf

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

1. 编辑存储器配置文件

# vim /etc/fdfs/storage.conf

修改的内容如下，其他保留

disable=false

port=23000 #同一组的端口号要一致，不同组不需要

group\_name=group1 #组名，第一组为group1，第二组为group2

base\_path=/data/fastdfs/storage

store\_path0=/data/fastdfs/storage

store\_path\_count=1 #存储路径个数，需要与store\_path个数一致

tracker\_server=192.168.0.131:22122

tracker\_server=192.168.0.131:22122

http.server\_port=8888

1. 创建数据目录

# mkdir -p /data/fastdfs/storage

1. 防火墙打开23000端口

# vim /etc/sysconfig/iptables

添加

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

重启

# service iptables restart

1. 启动storage

# /etc/init.d/fdfs\_storaged start

初次启动成功，会在/data/fastdfs/storage 目录创建data、logs两个目录

各节点启动，使用 tailf /data/fastdfs/storage/logs/storaged.log可以查看到各个节点连接到tracker的日志，并提示哪一个为leader。并且可以看到同一组的其他机器加入集群的日志。

1. 查看集群状态

可以在任意一个storage上，使用如下命令：

# /usr/bin/fdfs\_monitor /etc/fdfs/storage.conf

1. 关闭storage

# /etc/init.d/fdfs\_storaged stop

1. 设置storage开机启动

# vim /etc/rc.d/rc.local

/etc/init.d/fdfs\_storaged start

1. 文件上传测试（192.168.0.131）
2. 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

# vim /etc/fdfs/client.conf

修改

base\_path=/data/fastdfs/tracker

tracker\_server=192.168.0.131.22122

tracker\_server=192.168.0.132.22122

1. 执行文件上传命令

# /usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf XXXXXXXXXX

返回ID号则成功：

group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip

group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak199.jpg

1. 在各个storage节点上安装nginx（192.168.0.135、192.168.0.136、192.168.0.137、192.168.0.138）
2. fastdfs-nginx-module作用

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 192.168.0.101，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 192.168.0.102，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 192.168.0.102 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）

1. 上传fastdfs-nginx-module-master.zip到/data/software/fastdfs-packages
2. 解压

# cd /data/software/fastdfs-packages

# unzip fastdfs-nginx-module-master.zip

~~# cd fastdfs-nginx-module-master/src~~

~~# vim 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）~~

**5.0.5版本存在，6.0.3版本已修复此问题，确认即可**

1. 上传nginx稳定版本nginx-1.12.2.tar.gz到/data/software/fastdfs-packages
2. 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

1. 编译安装nginx，安装到/data/nginx-fastdfs,添加fastdfs-nginx-module模块

# cd /data/software/fastdfs-packages

# tar -zxvf nginx-1.12.2.tar.gz

# cd nginx-1.12.2

# ./configure --add-module=/data/software/fastdfs-packages/fastdfs-nginx-module-master/src --prefix=/data/nginx-fastdfs

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

# cp /data/software/fastdfs-packages/fastdfs-nginx-module-master/src/mod\_fastdfs.conf /etc/fdfs/

# vim /etc/fdfs/mod\_fastdfs.conf

修改如下配置，其他保留

第一组和第二组唯一的区别就是group\_name不一样：

connect\_timeout=10

base\_path=/tmp

tracker\_server=192.168.0.131:22122

tracker\_server=192.168.0.132:22122

storage\_server\_port=23000

group\_name=group1

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

1. 复制fastdfs的部分配置文件到/etc/fdfs目录

# cd /data/software/fastdfs-packages/fastdfs-master/conf

# cp http.conf mime.types /etc/fdfs/

1. 在/data/fastdfs/storage目录创建软连接，将其链接到数据的实际目录

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

1. Nginx配置

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 / {

root html;

index index.html index.htm;

}

location ~/group([0-9])/M00 {

ngx\_fastdfs\_module;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

}

1. 防火墙打开8888端口

# vi /etc/sysconfig/iptables

添加：

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

# service iptables restart

1. 启动nginx

# /data/nginx-fastdfs/sbin/nginx

ngx\_http\_fastdfs\_set pid=xxx

1. 设置nginx开机启动

# vim /etc/rc.d/rc.local

/data/nginx-fastdfs/sbin/nginx

通过浏览器访问刚刚测试上传的文件

<http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>

<http://192.168.0.136:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>

~~[http://192.168.0.137:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip](http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)~~

~~<http://192.168.0.138:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>~~

~~[http://192.168.0.135:8888/](http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg~~

~~[http://192.168.0.136:8888/](http://192.168.0.136:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg~~

[http://192.168.0.137:8888/](http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg

[http://192.168.0.138:8888/](http://192.168.0.138:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg

1. 在跟踪器节点上安装nginx（对于我们系统相当于前端nginx）
2. 在tracker上安装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。
3. 安装编译nginx锁依赖的包

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

1. 上传nginx缓存模块到软件包目录并解压

# cd /data/software/fastdfs-packages

# tar -zxvf ngx\_cache\_purge-2.3.tar.gz

1. 上传nginx 1.12.2版本包并编译安装

# cd /data/software/fastdfs-packages

# tar -zxvf nginx-1.12.2.tar.gz

# cd nginx-1.12.2

#./configure --prefix=/data/nginx-fastdfs --add-module=/data/software/fastdfs-packages/ngx\_cache\_purge-2.3

# make && make install

1. Nginx配置

user root;

worker\_processes 1;

events {

worker\_connections 1024;

}

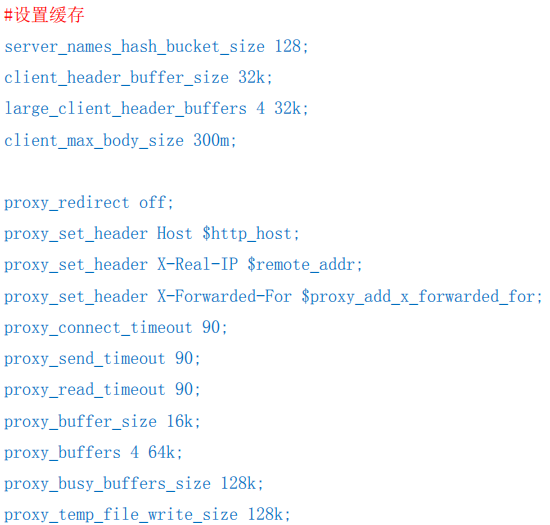
http {

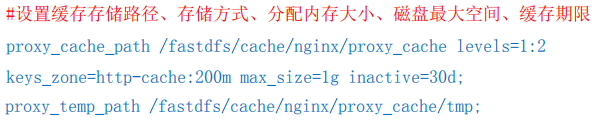
include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;





upstream fdfs\_group1{

server 192.168.0.135:8888 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.0.136:8888 weight=1 max\_fails=2 fail\_timeout=30s;

}

upstream fdfs\_group2{

server 192.168.0.137:8888 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.0.138:8888 weight=1 max\_fails=2 fail\_timeout=30s;

}

server {

listen 8000;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

location /group1/M00 {

proxy\_pass <http://fdfs_group1;>

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;

expires 30d;

}

location /group2/M00 {

proxy\_pass [http://fdfs\_group2;](http://fdfs_group1;)

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;

expires 30d;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

#设置清除缓存的

location ~/purge(/.\*){

allow 127.0.0.1;

allow 192.168.0.0/24;

deny all;

proxy\_cache\_purge http-cache $1$is\_args$args;

}

}

}

按照以上nginx的配置要求，创建相应的目录

# mkdir -p /data/fastdfs/cache/nginx/proxy\_cache

# mkdir -p /data/fastdfs/cache/nginx/proxy\_cache/tmp

1. 防火墙打开8000端口

# vi /etc/sysconfig/iptables

添加：

-A INPUT -m state --state NEW -m tcp -p tcp --dport 8000 -j ACCEPT

# service iptables restart

1. 启动nginx

# /data/nginx-fastdfs/sbin/nginx

1. 设置nginx开机启动

# vim /etc/rc.d/rc.local

/data/nginx-fastdfs/sbin/nginx

1. 浏览器测试

通过浏览器访问storage刚刚测试上传的文件

<http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>

~~<http://192.168.0.136:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>~~

[http://192.168.0.137:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip](http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)

~~<http://192.168.0.138:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip>~~

通过浏览器访问tracker刚刚测试上传的文件

[http://192.168.0.131:8000/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip](http://192.168.0.135:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)

[http://192.168.0.132:8000/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip](http://192.168.0.136:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)

http://192.168.0.131:8000/group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg

http://192.168.0.132:8000/group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg

现在的tracker上的nginx具备负载均衡能力，如果二者的nginx有keepalived保活并且有对外的VIP，那么就可以直接提供服务了。但是扩展有点儿不太方便。需要另外一套nginx集群来代理(比如VIP 192.168.0.50，端口在88)。

upstream fastdfs\_tracker{

server 192.168.0.131:8000 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.0.132:8000 weight=1 max\_fails=2 fail\_timeout=30s;

}



访问的链接为：

[http://192.168.0.50:88/dfs/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip](http://192.168.0.136:8888/group1/M00/00/00/Ci9fJF3XZ6KADez4AA28tsAei6c591.zip)

http://192.168.0.50:88/dfs/group2/M00/00/00/eBuSKl3XnbSAXWL5AAC6HG5B3ak99.jpg