Centos7搭建FastDFS，nginx负载均衡、缓存

# 概述

针对大量图片存取的需求，搭建FastDFS作为文件存储系统，由其作为文件存储系统，负载均衡、冗余备份、自动同步。并且支持主流语言客户端API的上传/下载/及删除操作。

# FastDFS架构介绍

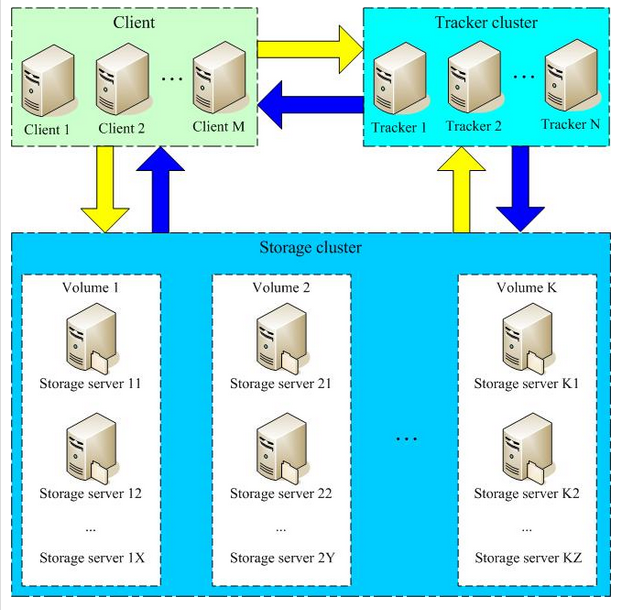
FastDFS系统包括两个角色：Tracker和Storage，Tracker主要负责任务的调度及负载均衡作用，Storage用作文件的存取、同步。

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。

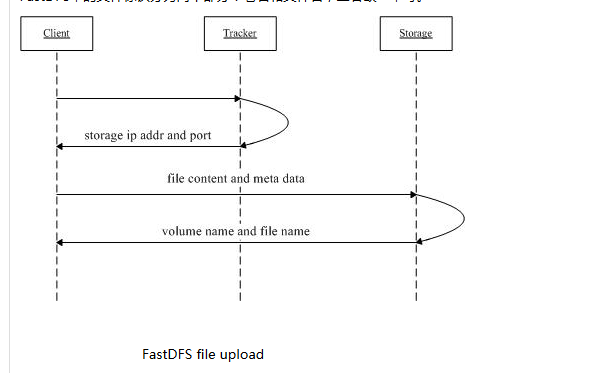
为 了支持大容量，存储节点（服务器）采用了分卷（或分组）的组织方式。存储系统由一个或多个卷组成，卷与卷之间的文件是相互独立的，所有卷 的文件容量累加就是整个存储系统中的文件容量。一个卷可以由一台或多台存储服务器组成，一个卷下的存储服务器中的文件都是相同的，卷中的多台存储服务器起 到了冗余备份和负载均衡的作用。

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。

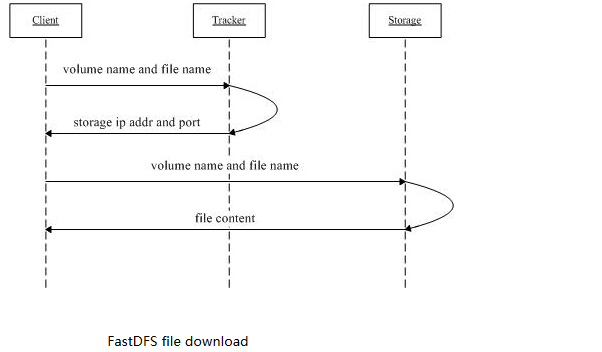
当存储空间不足或即将耗尽时，可以动态添加卷。只需要增加一台或多台服务器，并将它们配置为一个新的卷，这样就扩大了存储系统的容量。



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。



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。

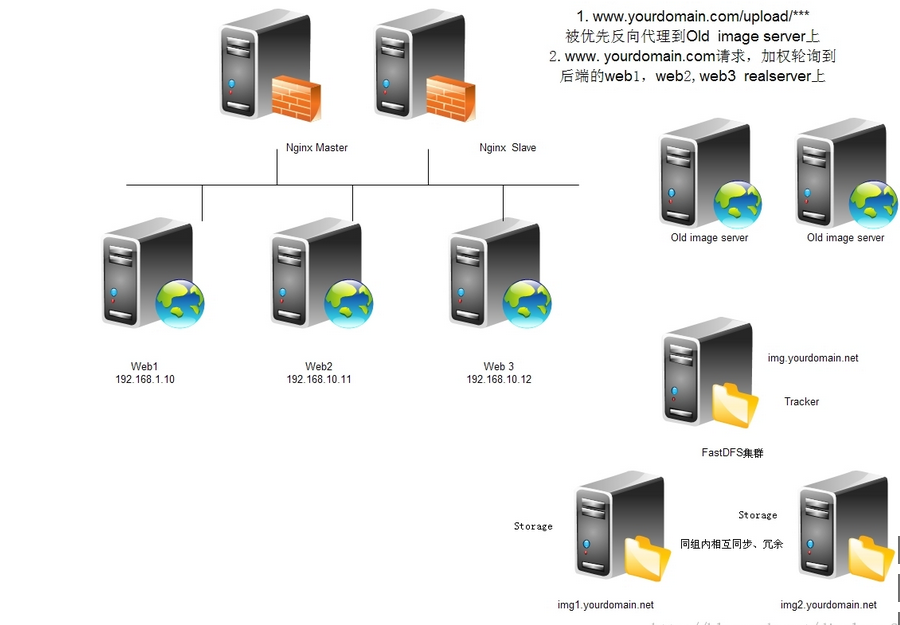


下载文件交互过程：

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。

需要说明的是，client为使用FastDFS服务的调用方，client也应该是一台服务器，它对tracker和storage的调用均为服务器间的调用。

# 文件服务器FastDFS集成架构



# FastDFS文件系统搭建

## 环境准备

Centos7.1.1503

FastDFS\_v5.07.tar.gz

fastdfs-nginx-module\_v1.16.tar.gz

ngx\_cache\_purge-2.3.tar.gz (http://labs.frickle.com/nginx\_ngx\_cache\_purge/)

nginx1-9-5(http://nginx.org/en/download.html)

FastDFS、FastDFS Nginx Module下载地址：

<http://sourceforge.net/projects/fastdfs/files/>

实验目的搭建一个FastDFS环境并且增加Nginx模块，Nginx模块的主要目的是在同组的Storage中间进行同步，防止同组Storage同步未完成时，进行访问出现错误。

准备3台centos服务器，一个tracker（192.168.128.131），两个storage互备（192.168.128.132和192.168.128.133）

## 1、首先需要安装libfastcommon（tracker+storage）

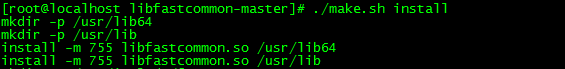
下载地址：<https://github.com/happyfish100/libfastcommon.git>

在每一台服务器上,解压libfastcommon，

unzip master.zip

./make.sh

./make.sh install



可以看到libfastcommon.so安装到了/usr/lib/ libfastcommon.so和/usr/lib64/libfastcommon.so

但是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/libfdfsclient.so /usr/local/lib/libfdfsclient.so

## 2、安装FastDFS主程序（tracker+storage）

在每台服务器,解压缩FastDFS\_v5.04.tar.gz,进入FastDFS目录  
执行  
./make.sh   
./make.sh install  
如果上步的软链接创建成功,就应该会非常顺利。

配置Tracker服务器**（192.168.128.131）**   
cp /etc/fdfs/tracker.conf.sample /etc/fdfs/tracker.conf

vim /etc/fdfs/tracker.conf文件,修改如下内容

一般只需改动以下几个参数即可：

disabled=false            #启用配置文件

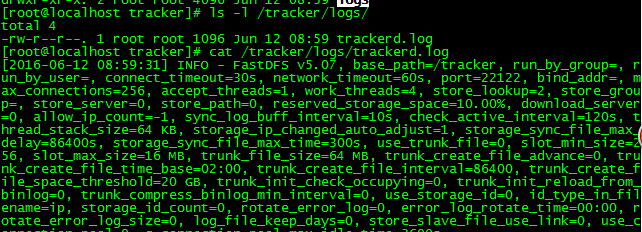
port=22122                #设置tracker的端口号

base\_path=/tracker（需要提前mkdir /tracker）  #设置tracker的数据文件和日志目录（需预先创建）

http.server\_port=8080     #设置http端口号

然后执行命令  
${FastDFS目录}/ tracker/trackerfdfs\_trackerd /etc/fdfs/tracker.conf

查看log：



配置Storage服务器**（192.168.128.132，192.168.128.133）**  
**cp /etc/fdf**s/storage.conf.sample /etc/fdfs/storage.conf

vim /etc/fdfs/storage.c**onf**

group\_name=group1

port=23000#设置storage的端口号  
base\_path=/storage #设置storage的日志目录（需预先创建）  
store\_path0=/storage #存储路径（需要提前mkdir /storage）  
tracker\_server= 192.168.128.131:22122

http.server\_port=8080     #设置http端口号

然后执行命令  
${FastDFS目录}/storage/fdfs\_storaged /etc/fdfs/storage.conf

查看log：



需要开启tracker的22122端口：

在tracker服务器上执行：

/sbin/iptables -I INPUT -p tcp --dport 22122 -j ACCEPT

/etc/rc.d/init.d/iptables save

service iptables restart

因为storage服务器之间会相互访问，tracker也会访问storage，因此需要开放storage的23000端口，同时nginx（tracker+storage）配置的8080端口，也要开放：

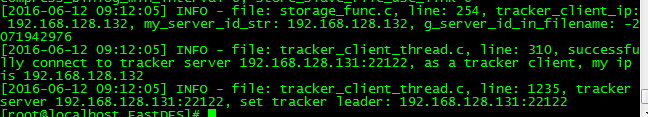
在storage服务器上执行：

/sbin/iptables -I INPUT -p tcp --dport 23000 -j ACCEPT

/etc/rc.d/init.d/iptables save

service iptables restart

在storage服务器上重新执行：${FastDFS目录}/storage/fdfs\_storaged /etc/fdfs/storage.conf，查看log：



已经连上了tracker。

### 执行测试

修改Tracker服务器192.168.128.131的配置文件/etc/fdfs/client.conf

cp /etc/fdfs/client.conf.sample /etc/fdfs/client.conf

vi /etc/fdfs/client.conf

base\_path=/tracker

tracker\_server=192.168.128.131:22122

执行命令  
[root@mysql1 fdfs]#

${FastDFS目录}./fdfs\_upload\_file /etc/fdfs/client.conf /usr/local/src/FastDFS/make.sh

返回：  
group1/M00/00/00/wKiAhFddkLaAYxScAAAWGD\_XotE0325.sh

说明上传成功！

查看storage服务器上的文件（两台服务器上都有）：

ls -l /storage/data/00/00/wKiAhFddkLaAYxScAAAWGD\_XotE0325.sh

## 3、安装fastdfs-nginx-module（storage）

FastDFS通过Tracker服务器,将文件放在Storage服务器存储,  
但是同组之间的服务器需要复制文件,有延迟的问题.  
假设Tracker服务器将文件上传到了192.168.128.132,文件ID已经返回客户端,  
这时,后台会将这个文件复制到192.168.128.133,如果复制没有完成,客户端就用这个ID在192.168.128.133取文件,肯定会出现错误....  
这个fastdfs-nginx-module可以重定向连接到源服务器取文件,避免客户端由于复制延迟的问题,出现错误。

修改fastdfs-nginx-module的config文件  
原来的内容是  
CORE\_INCS="$CORE\_INCS /usr/local/include/fastdfs /usr/local/include/fastcommon/"  
  
vim /home/nginx/fastdfs-nginx-module/src/config,修改为  
CORE\_INCS="$CORE\_INCS /usr/include/fastdfs /usr/include/fastcommon"  
  
各个版本的位置并不统一.所以需要根据自己的版本修改位置。

## 4、安装nginx（tracker+storage）

#### 在storage上安装nginx（增加fastdfs-nginx-module模块）

在storage上安装的nginx主要为了提供http的访问服务，同时解决group中storage服务器的同步延迟问题。

1. 准备编译环境：

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

1. 安装pcre和zlib库

./configure

make

make install

1. 安装nginx，需要增加fastdfs-nginx-module模块

进入nginx目录，执行

./configure

--sbin-path=/usr/local/nginx/nginx --conf-path=/usr/local/nginx/nginx.conf --pid-path=/usr/local/nginx/nginx.pid

--with-http\_ssl\_module

--with-pcre=/usr/local/src/pcre-8.37 --with-zlib=/usr/local/src/zlib-1.2.8 --add-module=/usr/local/src/fastdfs-nginx-module/src (这个是fastdfs-nginx-module模块目录)

make -j `cat /proc/cpuinfo | grep processor| wc -l` && make install

1. 复制FastDFS的配置文件到/etc/fdfs

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

1. 配置nginx

将server段中的listen端口号改为8080：

listen       8080;

在server段中添加：

location~/group[1-3]/M00{

root/fdfs/storage/data;

    ngx\_fastdfs\_module;

}

1. 配置fastdfs-nginx-module。

复制fastdfs-nginx-module源码中的配置文件到/etc/fdfs：  
cp /usr/local/src/fastdfs-nginx-module/src/mod\_fastdfs.conf /etc/fdfs/，修改该配置文件，一般只需改动以下几个参数即可：

base\_path=/storage#保存日志目录

tracker\_server=192.168.128.131:22122#tracker服务器的IP地址以及端口号

storage\_server\_port=23000#storage服务器的端口号

group\_name=group1#当前服务器的group名

url\_have\_group\_name= true        #文件url中是否有group名

store\_path\_count=1                #存储路径个数，需要和store\_path个数匹配

store\_path0=/storage         #存储路径

http.need\_find\_content\_type=true#从文件扩展名查找文件类型（nginx时为true）

group\_count= 1                   #设置组的个数（可以是多个）

在末尾增加1个组的具体信息：

[group1]

group\_name=group1

storage\_server\_port=23000

store\_path\_count=1

store\_path0=/storage

#### 在tracker上安装nginx(集成nginx cache purge模块)

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。

1. 准备编译环境+安装zlib+安装pcre（与storage相同）
2. 安装nginx cache purge模块插件

./configure

--sbin-path=/usr/local/nginx/nginx

--conf-path=/usr/local/nginx/nginx.conf

--pid-path=/usr/local/nginx/nginx.pid

--with-http\_ssl\_module

--with-pcre=/usr/local/src/pcre-8.37

--with-zlib=/usr/local/src/zlib-1.2.8

--add-module=/usr/local/src/ngx\_cache\_purge-master（路径根据实际情况配置）

make

make install

1. 配置nginx和nginx cache purge模块，修改配置文件nginx.conf

worker\_processes  4;                  #根据CPU核心数而定

events{

    worker\_connections  65535;       #最大链接数

    use epoll;#新版本的Linux可使用epoll加快处理性能

}

http{

#设置缓存参数

server\_names\_hash\_bucket\_size 128;

client\_header\_buffer\_size 32k;

large\_client\_header\_buffers 4 32k;

client\_max\_body\_size 300m;

sendfile        on;

tcp\_nopush on;

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 /var/cache/nginx/proxy\_cache levels=1:2 keys\_zone=http-cache:500m max\_size=10ginactive=30d;

proxy\_temp\_path /var/cache/nginx/proxy\_cache/tmp;

#设置group1的服务器

upstream fdfs\_group1 {

server192.168.128.132:8080 weight=1 max\_fails=2 fail\_timeout=30s;

server192.168.128.133:8080 weight=1 max\_fails=2 fail\_timeout=30s;

}

server{

#设置服务器端口

listen       8080;

#设置group1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;

}

#设置清除缓存的访问权限

location~ /purge(/.\*) {

allow 127.0.0.1;

allow 192.168.128.0/24;

denyall;

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

}

至此，nginx以及nginx cachepurge插件模块设置完成。