# FastDFS安装配置手册

目录

[FastDFS安装配置手册 1](#_Toc383683153)

[一、 安装 3](#_Toc383683154)

[(一) 下载FastDFS安装包 3](#_Toc383683155)

[(二) 安装tracker 3](#_Toc383683156)

[1. 安装 3](#_Toc383683157)

[2. 配置 5](#_Toc383683158)

[3. 运行 5](#_Toc383683159)

[(三) 安装storage 6](#_Toc383683160)

[1. 安装 6](#_Toc383683161)

[2. 配置 8](#_Toc383683162)

[3. 运行 8](#_Toc383683163)

[(四) 在storage上安装nginx 10](#_Toc383683164)

[1. 安装 10](#_Toc383683165)

[2. 配置 12](#_Toc383683166)

[3. 运行 13](#_Toc383683167)

[(五) 在tracker上安装nginx 14](#_Toc383683168)

[1. 安装 15](#_Toc383683169)

[2. 配置 16](#_Toc383683170)

[3. 运行 18](#_Toc383683171)

[二、 使用 20](#_Toc383683172)

[1. 上传文件 20](#_Toc383683173)

[2. 下载文件 21](#_Toc383683174)

[3. 监视服务器资源 21](#_Toc383683175)

[4. 其他功能 27](#_Toc383683176)

[三、 附录 28](#_Toc383683177)

缺失部分开发组件吧。。  
运行以下命令试试。。。  
yum install -y libevent-devel pcre-devel zlib-devel

安装libevent

yum install libevent-devel.x86\_64

4. 启动storage server时，一直处于僵死状态。  
A：启动storage server，storage将连接tracker server，如果连不上，将一直重试。直到连接成功，启动才算真正完成。  
出现这样情况，请检查连接不上tracker server的原因。  
友情提示：从V2.03以后，多tracker server在启动时会做时间上的检测，判断是否需要从别的tracker server同步4个系统文件。  
触发时机是第一个storage server连接上tracker server后，并发起join请求。  
如果集群中有2台tracker server，而其中一台tracker没有启动，可能会导致storage server一直处于僵死状态。  
  
5. 执行fdfs\_test或fdfs\_test1上传文件时，服务器返回错误号2  
错误号表示没有ACTIVE状态的storage server。可以执行fdfs\_monitor查看服务器状态。  
  
6. 如何让server进程退出运行？  
直接kill即可让server进程正常退出，可以使用killall命令，例如：  
killall fdfs\_trackerd  
killall fdfs\_storaged  
也可以使用FastDFS自带的stop.sh脚本，如：  
/usr/local/bin/stop.sh  /usr/local/bin/fdfs\_storaged /etc/fdfs/storage.conf  
stop.sh只会停止命令行（包括参数）完全相同的进程。  
千万不要使用-9参数强杀，否则可能会导致binlog数据丢失的问题。  
  
7. 如何重启server进程？  
可以kill掉server进程后，执行启动命令行。如：  
killall fdfs\_trackerd  
/usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf  
或者直接使用FastDFS自带的restart.sh脚本，如：  
/usr/local/bin/restart.sh  /usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf  
  
8. 跨运营商通信异常问题  
比如电信和网通机房相互通信，可能会存在异常，有两种表现：  
  1）不能建立连接，这个比较直接，肯定是网络连接的问题  
  2）可以正常建立连接，但接收和发送数据失败，这个问题比较隐蔽，正常网络环境下，不应该出现此类问题。  
       还有人碰到过从一个方向建立连接可以正常通信，但从另外一个方向就不能正常通信的情况。  
解决办法：  
  尝试将服务端口改小，建议将端口修改为1024以下。比如将storage服务端口由23000修改为873等，也可以试试修改为8080  
  如果问题还不能解决，请联系你的网络（机房）服务商。  
  
9. fdfs\_test和fdfs\_test1是做什么用的？  
   这两个是FastDFS自带的测试程序，会对一个文件上传两次，分别作为主文件和从文件。返回的文件ID也是两个。  
   并且会上传文件附加属性，storage server上会生成4个文件。  
   这两个程序仅用于测试目的，请不要用作实际用途。  
   V2.05提供了比较正式的三个小工具：  
      上传文件：/usr/local/bin/fdfs\_upload\_file  <config\_file> <local\_filename>  
      下载文件：/usr/local/bin/fdfs\_download\_file <config\_file> <file\_id> [local\_filename]  
       删除文件：/usr/local/bin/fdfs\_delete\_file <config\_file> <file\_id>  
  
10. 什么是主从文件？  
主从文件是指文件ID有关联的文件，一个主文件可以对应多个从文件。  
    主文件ID = 主文件名 + 主文件扩展名  
    从文件ID = 主文件名 + 从文件后缀名 + 从文件扩展名  
使用主从文件的一个典型例子：以图片为例，主文件为原始图片，从文件为该图片的一张或多张缩略图。  
FastDFS中的主从文件只是在文件ID上有联系。FastDFS server端没有记录主从文件对应关系，因此删除主文件，FastDFS不会自动删除从文件。  
删除主文件后，从文件的级联删除，需要由应用端来实现。  
主文件及其从文件均存放到同一个group中。  
主从文件的生成顺序：  
  1）先上传主文件（如原文件），得到主文件ID  
  2）然后上传从文件（如缩略图），指定主文件ID和从文件后缀名（当然还可以同时指定从文件扩展名），得到从文件ID。  
  
  
11. 如何删除无效的storage server？  
可以使用fdfs\_monitor来删除。命令行如下：  
/usr/local/bin/fdfs\_monitor <config\_filename> delete <group\_name> <storage\_ip>  
例如：  
/usr/local/bin/fdfs\_monitor /etc/fdfs/client.conf delete group1 192.168.0.100  
注意：如果被删除的storage server的状态是ACTIVE，也就是该storage server还在线上服务的情况下，是无法删除掉的。  
  
12. FastDFS扩展模块升级到V1.06及以上版本的注意事项  
apache和nginx扩展模块版本v1.06及以上版本，需要在配置文件/etc/fdfs/fastdfs\_mod.conf中设置storage server的存储路径信息。  
一个示例如下所示：  
store\_path\_count=1  
store\_path0=/home/yuqing/fastdfs  
store\_path\_count和store\_path#均需要正确设置，必须和storage.conf中的相应配置完全一致，否则将导致文件不能正确下载！  
  
13. nginx和apache扩展模块与FastDFS server版本对应关系  
    扩展模块1.05：  针对FastDFs server v2.x，要求server版本大于等于v2.09  
    扩展模块1.07及以上版本：  针对FastDFs server v3.x  
  
14. FastDFS有QQ技术交流群吗？  
   有的。群号：164684842，欢迎大家加入交流。  
  
15. 上传文件失败，返回错误码28，这是怎么回事？  
  返回错误码28，表示磁盘空间不足。注意FastDFS中有预留空间的概念，在tracker.conf中设置，配置项为：reserved\_storage\_space，缺省值为4GB，即预留4GB的空间。  
  请酌情设置reserved\_storage\_space这个参数，比如可以设置为磁盘总空间的20%左右。  
  
16. fdfs\_trackerd或者fdfs\_storaged的日志中出现：malloc task buff failed字样的错误，这是怎么回事？  
  出现此类信息表示已经达到最大连接数。server端支持的最大连接数可以通过max\_connections这个参数来设置。  
  出现这样的问题，需要排查一下是否客户端使用不当导致的，比如客户端没有及时关闭无用的连接。  
  
17. FastDFS的文件ID中可以反解出哪些字段？  
文件ID中除了包含group name和存储路径外，文件名中可以反解出如下几个字段：  
  1）文件创建时间（unix时间戳，32位整数）  
  2）文件大小  
  3）上传到的源storage server IP地址（32位整数）  
  4）文件crc32校验码  
  5）随机数（这个字段用来避免文件重名）  
  
18. 为什么生成的token验证无法通过？  
  出现这样的问题，请进行如下两项检查：  
  1）确认调用token生成函数，传递的文件ID中没有包含group name。传递的文件ID格式形如：M00/00/1B/wKgnVE84utyOG9hEAAATz5-S0SI99.java  
  2）确认服务器时间基本是一致的，注意服务器时间不能相差太多，不要相差到分钟级别。  
  
19. 最新程序包的下载地址是什么？  
   因google code不支持上传程序包，最新的程序包可以在sourceforge上下载，下载地址：<https://sourceforge.net/projects/fastdfs/files/>

#### 机器硬盘定时忙

Iotop发现是 updatedb -f sysfs

* 在我机器上引起硬盘狂转的是 updatedb.mlocate, 为硬盘文件建立索引，以便于使用locate命令能找到文件。

## 2. 解决

* 由于我几乎从来不用locate命令，我决定降低updatedb的频率。
* 通过下面的查找得知,ubuntu中updatedb是由mlocate来定时启动的。
* 在/etc搜索updatedb，在/etc/cron.daily下找到mlocate
* 只要把这个脚本删了或移到其它位置就可以了。
* 我是将它变成每月运行一次。
  + $ sudo mv /etc/cron.daily/mlocate /etc/cron.monthly/
* 问题解决啦！

是否可以认为，只要知道tracker的服务器IP和端口，任何都可以使用api上传文件，这样是否会有恶意上传的问题？？

可以指定访问限制  
tracker.conf  
storage.conf  
添加访问IP限制：（例）  
  
# allow\_hosts can ocur more than once, host can be hostname or ip address,  
# "\*" means match all ip addresses, can use range like this: 10.0.1.[1-15,20] or  
# host[01-08,20-25].domain.com, for example:  
# allow\_hosts=10.0.1.[1-15,20]  
# allow\_hosts=host[01-08,20-25].domain.com  
#allow\_hosts=\*  
allow\_hosts=222.222.222.[152-154]  
allow\_hosts=111.111.111.111

## 安装

### 下载FastDFS安装包

FastDFS官方论坛：<http://www.csource.org>

下载1：<http://sourceforge.net/projects/fastdfs/files/>

下载2：<https://code.google.com/p/fastdfs/downloads/list>

本手册使用CentOS 6.5 x86\_64版操作系统，按照以下网络结构进行部署：



所需要下载的压缩包有：

FastDFS源代码：FastDFS\_v5.01.tar.gz

nginx模块源代码：fastdfs-nginx-module\_v1.15.tar.gz

nginx服务器源代码：nginx-1.4.7.tar.gz

nginx cache purge插件源代码：ngx\_cache\_purge-2.1.tar.gz

nginx依赖的pcre库源代码：pcre-8.34.tar.gz

nginx依赖的zlib库源代码：zlib-1.2.8.tar.gz

### 安装tracker

### 安装

首先在172.16.1.202上安装FastDFS tracker，使用FastDFS\_v5.01.tar.gz源代码包。

可以直接从Linux系统下载源代码包并解压安装。如果是在Windows系统下载，可以在Linux系统中挂载Windows的共享目录并且将源代码包复制进Linux系统内。挂载命令如下

|  |
| --- |
| [root@centos-db01 ~]# mount -o ro //Windows主机的IP地址/共享目录名 /挂载目标目录 |

* 首先将代码包复制到系统的/usr/local/src内（这一步可选），然后使用tar命令解压

|  |
| --- |
| [root@tracker opt]# cp FastDFS\_v5.01.tar.gz /usr/local/src/  [root@tracker opt]# cd /usr/local/src/  [root@tracker src]# tar zxf FastDFS\_v5.01.tar.gz  [root@tracker src]# cd FastDFS  [root@tracker FastDFS]# ll  总用量 128  drwxrwxr-x. 3 500 500 4096 2月 6 18:07 client  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 common  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 conf  -rw-rw-r--. 1 500 500 35067 7月 28 2008 COPYING-3\_0.txt  -rw-rw-r--. 1 500 500 29691 2月 2 13:17 HISTORY  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 init.d  -rw-rw-r--. 1 500 500 7639 1月 5 14:08 INSTALL  -rwxrwxr-x. 1 500 500 5531 12月 7 15:19 make.sh  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 php\_client  -rw-rw-r--. 1 500 500 2380 7月 28 2008 README  -rwxrwxr-x. 1 500 500 1768 4月 12 2010 restart.sh  -rwxrwxr-x. 1 500 500 1680 4月 10 2010 stop.sh  drwxrwxr-x. 4 500 500 4096 2月 6 18:07 storage  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 test  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 tracker |

* 运行make.sh，确认make成功。期间如果有错误，可能会是缺少依赖的软件包，需安装后再次make。

|  |
| --- |
| [root@tracker FastDFS]# ./make.sh |

* 运行make.sh install，确认install成功。

|  |
| --- |
| [root@tracker FastDFS]# ./make.sh install |

安装完成后，所有可执行文件在/usr/local/bin下，以fdfs开头：

|  |
| --- |
| [root@tracker FastDFS]# ll /usr/local/bin/fdfs\*  -rwxr-xr-x. 1 root root 522918 3月 25 14:57 /usr/local/bin/fdfs\_appender\_test  -rwxr-xr-x. 1 root root 522871 3月 25 14:57 /usr/local/bin/fdfs\_appender\_test1  -rwxr-xr-x. 1 root root 514023 3月 25 14:57 /usr/local/bin/fdfs\_append\_file  -rwxr-xr-x. 1 root root 513433 3月 25 14:57 /usr/local/bin/fdfs\_crc32  -rwxr-xr-x. 1 root root 513967 3月 25 14:57 /usr/local/bin/fdfs\_delete\_file  -rwxr-xr-x. 1 root root 514377 3月 25 14:57 /usr/local/bin/fdfs\_download\_file  -rwxr-xr-x. 1 root root 514133 3月 25 14:57 /usr/local/bin/fdfs\_file\_info  -rwxr-xr-x. 1 root root 525064 3月 25 14:57 /usr/local/bin/fdfs\_monitor  -rwxr-xr-x. 1 root root 1179682 3月 25 14:57 /usr/local/bin/fdfs\_storaged  -rwxr-xr-x. 1 root root 529845 3月 25 14:57 /usr/local/bin/fdfs\_test  -rwxr-xr-x. 1 root root 527774 3月 25 14:57 /usr/local/bin/fdfs\_test1  -rwxr-xr-x. 1 root root 655809 3月 25 14:57 /usr/local/bin/fdfs\_trackerd  -rwxr-xr-x. 1 root root 514213 3月 25 14:57 /usr/local/bin/fdfs\_upload\_appender  -rwxr-xr-x. 1 root root 514999 3月 25 14:57 /usr/local/bin/fdfs\_upload\_file |

所有配置文件在/etc/fdfs下：

|  |
| --- |
| [root@tracker FastDFS]# ll /etc/fdfs/  总用量 60  -rw-r--r--. 1 root root 1461 3月 13 15:15 client.conf  -rw-r--r--. 1 root root 858 3月 13 15:15 http.conf  -rw-r--r--. 1 root root 31172 3月 13 15:15 mime.types  -rw-r--r--. 1 root root 3837 3月 25 10:03 mod\_fastdfs.conf  -rw-r--r--. 1 root root 7515 3月 24 10:36 storage.conf  -rw-r--r--. 1 root root 6989 3月 13 15:15 tracker.conf |

* 至此tracker安装完成。

### 配置

* 编辑配置文件目录下的tracker.conf，设置相关信息并保存。

|  |
| --- |
| [root@tracker FastDFS]# vim /etc/fdfs/tracker.conf |

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=/fdfs/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

如需要进行性能调优，可以参照附录的配置文件的详细说明。

### 运行

* 运行tracker之前，先要把防火墙中对应的端口打开（本例中为22122）。

|  |
| --- |
| [root@tracker FastDFS]# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 22122 -j ACCEPT  [root@tracker FastDFS]# /etc/init.d/iptables save  iptables：将防火墙规则保存到 /etc/sysconfig/iptables：[确定] |

* 启动tracker，确认启动是否成功。（查看是否对应端口22122是否开始监听）

|  |
| --- |
| [root@tracker FastDFS]# /usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart  [root@tracker FastDFS]# netstat -unltp | grep fdfs  tcp 0 0.0.0.0:22122 0.0.0.0:\* LISTEN 1766/fdfs\_trackerd |

也可查看tracker的日志是否启动成功或是否有错误。

|  |
| --- |
| [root@tracker FastDFS]# cat /fdfs/tracker/logs/trackerd.log |

* 设置开机自动启动。

|  |
| --- |
| [root@tracker FastDFS]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

### 安装storage

### 安装

首先在172.16.1.203上安装FastDFS storage，使用FastDFS\_v5.01.tar.gz源代码包。

* 首先将代码包复制到系统的/usr/local/src内（这一步可选），然后使用tar命令解压

|  |
| --- |
| [root@storage1 opt]# cp FastDFS\_v5.01.tar.gz /usr/local/src/  [root@storage1 opt]# cd /usr/local/src/  [root@storage1 src]# tar zxf FastDFS\_v5.01.tar.gz  [root@storage1 src]# cd FastDFS  [root@storage1 FastDFS]# ll  总用量 128  drwxrwxr-x. 3 500 500 4096 2月 6 18:07 client  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 common  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 conf  -rw-rw-r--. 1 500 500 35067 7月 28 2008 COPYING-3\_0.txt  -rw-rw-r--. 1 500 500 29691 2月 2 13:17 HISTORY  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 init.d  -rw-rw-r--. 1 500 500 7639 1月 5 14:08 INSTALL  -rwxrwxr-x. 1 500 500 5531 12月 7 15:19 make.sh  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 php\_client  -rw-rw-r--. 1 500 500 2380 7月 28 2008 README  -rwxrwxr-x. 1 500 500 1768 4月 12 2010 restart.sh  -rwxrwxr-x. 1 500 500 1680 4月 10 2010 stop.sh  drwxrwxr-x. 4 500 500 4096 2月 6 18:07 storage  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 test  drwxrwxr-x. 2 500 500 4096 2月 6 18:07 tracker |

* 运行make.sh，确认make成功。期间如果有错误，可能会是缺少依赖的软件包，需安装后再次make。

|  |
| --- |
| [root@storage1 FastDFS]# ./make.sh |

* 运行make.sh install，确认install成功。

|  |
| --- |
| [root@storage1 FastDFS]# ./make.sh install |

安装完成后，所有可执行文件在/usr/local/bin下，以fdfs开头：

|  |
| --- |
| [root@storage1 FastDFS]# ll /usr/local/bin/fdfs\*  -rwxr-xr-x. 1 root root 522918 3月 25 14:57 /usr/local/bin/fdfs\_appender\_test  -rwxr-xr-x. 1 root root 522871 3月 25 14:57 /usr/local/bin/fdfs\_appender\_test1  -rwxr-xr-x. 1 root root 514023 3月 25 14:57 /usr/local/bin/fdfs\_append\_file  -rwxr-xr-x. 1 root root 513433 3月 25 14:57 /usr/local/bin/fdfs\_crc32  -rwxr-xr-x. 1 root root 513967 3月 25 14:57 /usr/local/bin/fdfs\_delete\_file  -rwxr-xr-x. 1 root root 514377 3月 25 14:57 /usr/local/bin/fdfs\_download\_file  -rwxr-xr-x. 1 root root 514133 3月 25 14:57 /usr/local/bin/fdfs\_file\_info  -rwxr-xr-x. 1 root root 525064 3月 25 14:57 /usr/local/bin/fdfs\_monitor  -rwxr-xr-x. 1 root root 1179682 3月 25 14:57 /usr/local/bin/fdfs\_storaged  -rwxr-xr-x. 1 root root 529845 3月 25 14:57 /usr/local/bin/fdfs\_test  -rwxr-xr-x. 1 root root 527774 3月 25 14:57 /usr/local/bin/fdfs\_test1  -rwxr-xr-x. 1 root root 655809 3月 25 14:57 /usr/local/bin/fdfs\_trackerd  -rwxr-xr-x. 1 root root 514213 3月 25 14:57 /usr/local/bin/fdfs\_upload\_appender  -rwxr-xr-x. 1 root root 514999 3月 25 14:57 /usr/local/bin/fdfs\_upload\_file |

所有配置文件在/etc/fdfs下：

|  |
| --- |
| [root@storage1 FastDFS]# ll /etc/fdfs/  总用量 60  -rw-r--r--. 1 root root 1461 3月 13 15:15 client.conf  -rw-r--r--. 1 root root 858 3月 13 15:15 http.conf  -rw-r--r--. 1 root root 31172 3月 13 15:15 mime.types  -rw-r--r--. 1 root root 3837 3月 25 10:03 mod\_fastdfs.conf  -rw-r--r--. 1 root root 7515 3月 24 10:36 storage.conf  -rw-r--r--. 1 root root 6989 3月 13 15:15 tracker.conf |

* 至此storage安装完成。

### 配置

* 编辑配置文件目录下的storage.conf，设置相关信息并保存。

|  |
| --- |
| [root@storage1 FastDFS]# vim /etc/fdfs/storage.conf |

一般只需改动以下几个参数即可：

disabled=false #启用配置文件

group\_name=group1 #组名，根据实际情况修改

port=23000 #设置storage的端口号

base\_path=/fdfs/storage #设置storage的日志目录（需预先创建）

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

store\_path0=/fdfs/storage #存储路径

tracker\_server=172.16.1.202:22122 #tracker服务器的IP地址和端口号

http.server\_port=8080 #设置http端口号

如需要进行性能调优，可以参照附录的配置文件的详细说明。

### 运行

* 运行storage之前，先要把防火墙中对应的端口打开（本例中为23000）。

|  |
| --- |
| [root@storage1 FastDFS]# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 23000 -j ACCEPT  [root@storage1 FastDFS]# /etc/init.d/iptables save  iptables：将防火墙规则保存到 /etc/sysconfig/iptables：[确定] |

* 启动storage，会根据配置文件的设置自动创建多级存储目录，确认启动是否成功。（查看是否对应端口23000是否开始监听）

|  |
| --- |
| [root@storage1 FastDFS]# /usr/local/bin/fdfs\_storaged /etc/fdfs/storage.conf restart  data path: /fdfs/storage/data, mkdir sub dir...  mkdir data path: 00 ...  mkdir data path: 01 ...  mkdir data path: 02 ...  mkdir data path: 03 ...  ...  data path: /fdfs/storage/data, mkdir sub dir done.  [root@storage1 FastDFS]# netstat -unltp | grep fdfs  tcp 0 0.0.0.0:23000 0.0.0.0:\* LISTEN 1766/fdfs\_storaged |

也可查看storage的日志是否启动成功或是否有错误。

|  |
| --- |
| [root@storage1 FastDFS]# cat /fdfs/storage/logs/storaged.log |

确认启动成功后，可以运行fdfs\_monitor查看storage服务器是否已经登记到tracker服务器。

|  |
| --- |
| [root@storage1 FastDFS]# /usr/local/bin/fdfs\_monitor /etc/fdfs/storage.conf  [2014-03-26 01:51:20] DEBUG - base\_path=/fdfs/storage, connect\_timeout=30, 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  server\_count=1, server\_index=0  tracker server is 172.16.1.202:22122  group count: 1  Group 1:  group name = group1  disk total space = 27789 MB  disk free space = 23920 MB  trunk free space = 0 MB  storage server count = 1  active server count = 1  storage server port = 23000  storage HTTP port = 8080  store path count = 1  subdir count per path = 256  current write server index = 0  current trunk file id = 0  Storage 1:  id = 172.16.1.203  ip\_addr = 172.16.1.203 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 01:48:19  up time = 2014-03-26 01:48:19  total storage = 27789 MB  free storage = 23920 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id=  if\_trunk\_server= 0  total\_upload\_count = 0  success\_upload\_count = 0  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 0  success\_upload\_bytes = 0  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 0  success\_sync\_in\_bytes = 0  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 0  success\_file\_open\_count = 0  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 0  success\_file\_write\_count = 0  last\_heart\_beat\_time = 2014-03-26 01:51:03  last\_source\_update = 1970-01-01 08:00:00  last\_sync\_update = 1970-01-01 08:00:00  last\_synced\_timestamp = 1970-01-01 08:00:00 |

看到“172.16.1.203 ACTIVE”即可确认storage运行正常。

* 设置开机自动启动。

|  |
| --- |
| [root@storage1 FastDFS]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/local/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

* 之后依次在172.16.1.204~208上全部安装上storage并确认运行正常。注意配置文件中group名参数需要根据实际情况调整，本例中group是这样分配的：

group1：172.16.1.203，172.16.1.204

group2：172.16.1.205，172.16.1.206

group3：172.16.1.207，172.16.1.208

另外每个group中所有storage的端口号必须一致。

### 在storage上安装nginx

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。

### 安装

首先在172.16.1.203上安装nginx，使用nginx-1.4.7.tar.gz源代码包以及FastDFS的nginx插件fastdfs-nginx-module\_v1.15.tar.gz。

* 首先将代码包和插件复制到系统的/usr/local/src内（可选），然后使用tar命令解压

|  |
| --- |
| [root@storage1 opt]# cp nginx-1.4.7.tar.gz /usr/local/src  [root@storage1 opt]# cp fastdfs-nginx-module\_v1.15.tar.gz /usr/local/src  [root@storage1 opt]# cp pcre-8.34.tar.gz /usr/local/src  [root@storage1 opt]# cp zlib-1.2.8.tar.gz /usr/local/src  [root@storage1 opt]# cd /usr/local/src/  [root@storage1 src]# tar zxf nginx-1.4.7.tar.gz  [root@storage1 src]# tar zxf fastdfs-nginx-module\_v1.15.tar.gz  [root@storage1 src]# tar zxf pcre-8.34.tar.gz  [root@storage1 src]# tar zxf zlib-1.2.8.tar.gz  [root@storage1 src]# cd nginx-1.4.7  [root@storage1 nginx-1.4.7]# ll  总用量 592  drwxr-xr-x. 6 1001 1001 4096 3月 26 02:34 auto  -rw-r--r--. 1 1001 1001 225213 3月 18 21:17 CHANGES  -rw-r--r--. 1 1001 1001 343040 3月 18 21:17 CHANGES.ru  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 conf  -rwxr-xr-x. 1 1001 1001 2369 3月 18 21:17 configure  drwxr-xr-x. 3 1001 1001 4096 3月 26 02:34 contrib  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 html  -rw-r--r--. 1 1001 1001 1397 3月 18 21:17 LICENSE  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 man  -rw-r--r--. 1 1001 1001 49 3月 18 21:17 README  drwxr-xr-x. 8 1001 1001 4096 3月 26 02:34 src |

* 运行./configure进行安装前的设置，主要设置安装路径、FastDFS插件模块目录、pcre库目录、zlib库目录。

如果提示错误，可能缺少依赖的软件包，需先安装依赖包，再次运行./configure

|  |
| --- |
| [root@storage1 nginx-1.4.7]# ./configure --prefix=/usr/local/nginx --add-module=/usr/local/src/fastdfs-nginx-module/src --with-pcre=/usr/local/src/pcre-8.34/ --with-zlib=/usr/local/src/zlib-1.2.8 |

* 运行make进行编译，确保编译成功。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# make |

* 运行make install进行安装。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# make install |

* 将FastDFS的nginx插件模块的配置文件copy到FastDFS配置文件目录。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# cp /usr/local/src/fastdfs-nginx-module/src/mod\_fastdfs.conf /etc/fdfs/ |

安装完成后，nginx所有文件在/usr/local/nginx下：

|  |
| --- |
| [root@storage1 nginx-1.4.7]# ll /usr/local/nginx/  总用量 16  drwxr-xr-x. 2 root root 4096 3月 26 03:11 conf  drwxr-xr-x. 2 root root 4096 3月 26 03:11 html  drwxr-xr-x. 2 root root 4096 3月 26 03:11 logs  drwxr-xr-x. 2 root root 4096 3月 26 03:11 sbin |

* 至此nginx以及FastDFS的nginx插件模块安装完成。

### 配置

* 编辑/usr/local/nginx/conf配置文件目录下的nginx.conf，设置添加storage信息并保存。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# vim /usr/local/nginx/conf/nginx.conf |

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;

}

yyy注：有多个data store，需要添加多次

location ~/group[1-2]/M00 {

root /fdfs/storage/data;

ngx\_fastdfs\_module;

}

location ~/group[1-2]/M01 {

root /fdfs/storage/data;

ngx\_fastdfs\_module;

}

location ~/group[1-2]/M02 {

root /fdfs/storage/data;

ngx\_fastdfs\_module;

}

location ~/group[1-2]/M03 {

root /fdfs/storage/data;

ngx\_fastdfs\_module;

}

* 编辑/etc/fdfs配置文件目录下的mod\_fastdfs.conf，设置storage信息并保存。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# vim /etc/fdfs/mod\_fastdfs.conf |

一般只需改动以下几个参数即可：

base\_path=/fdfs/storage #保存日志目录

tracker\_server=172.16.1.202:22122 #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=/fdfs/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 = 3 #设置组的个数

在末尾增加3个组的具体信息：

[group1]

group\_name=group1

storage\_server\_port=23000

store\_path\_count=1

store\_path0=/fdfs/storage

[group2]

group\_name=group2

storage\_server\_port=23000

store\_path\_count=1

store\_path0=/fdfs/storage

[group3]

group\_name=group3

storage\_server\_port=23000

store\_path\_count=1

store\_path0=/fdfs/storage

* 建立M00至存储目录的符号连接。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# ln -s /fdfs/storage/data /fdfs/storage/data/M00  [root@storage1 nginx-1.4.7]# ll /fdfs/storage/data/M00  lrwxrwxrwx. 1 root root 19 3月 26 03:44 /fdfs/storage/data/M00 -> /fdfs/storage/data/ |

* 至此，nginx以及FastDFS插件模块设置完成。

### 运行

* 运行nginx之前，先要把防火墙中对应的端口打开（本例中为8080）。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT  [root@storage1 nginx-1.4.7]# /etc/init.d/iptables save  iptables：将防火墙规则保存到 /etc/sysconfig/iptables：[确定] |

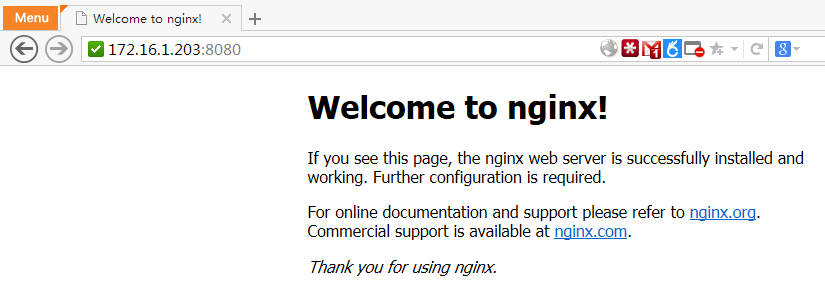
* 启动nginx，确认启动是否成功。（查看是否对应端口8080是否开始监听）

|  |
| --- |
| [root@storage1 nginx-1.4.7]# /usr/local/nginx/sbin/nginx  ngx\_http\_fastdfs\_set pid=40638  [root@storage1 nginx-1.4.7]# netstat -unltp | grep nginx  tcp 0 0.0.0.0:8080 0.0.0.0:\* LISTEN 40639/nginx |

也可查看nginx的日志是否启动成功或是否有错误。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# cat /usr/local/nginx/logs/error.log  ngx\_http\_fastdfs\_process\_init pid=40640  [2014-03-26 03:47:17] INFO - local\_host\_ip\_count: 2, 127.0.0.1 172.16.1.203  [2014-03-26 03:47:17] INFO - fastdfs apache / nginx module v1.15, response\_mode=proxy, base\_path=/tmp, url\_have\_group\_name=1, group\_count=3, connect\_timeout=2, network\_timeout=30, tracker\_server\_count=1, if\_alias\_prefix=, local\_host\_ip\_count=2, need\_find\_content\_type=1, default\_content\_type=application/octet-stream, anti\_steal\_token=0, token\_ttl=0s, anti\_steal\_secret\_key length=0, token\_check\_fail content\_type=, token\_check\_fail buff length=0, load\_fdfs\_parameters\_from\_tracker=1, storage\_sync\_file\_max\_delay=86400s, use\_storage\_id=0, storage server id count=0, flv\_support=1, flv\_extension=flv  [2014-03-26 03:47:17] INFO - group 1. group\_name=group1, storage\_server\_port=23000, path\_count=1, store\_path0=/fdfs/storage  [2014-03-26 03:47:17] INFO - group 2. group\_name=group2, storage\_server\_port=23000, path\_count=1, store\_path0=/fdfs/storage  [2014-03-26 03:47:17] INFO - group 3. group\_name=group3, storage\_server\_port=23000, path\_count=1, store\_path0=/fdfs/storage |

在error.log中没有错误，既启动成功。可以打开浏览器，直接访问<http://172.16.1.203:8080>，查看是否弹出nginx欢迎页面。



之后依次在172.16.1.204~208上全部安装上nginx并确认运行正常。

* 设置开机自动启动。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/local/nginx/sbin/nginx

### 在tracker上安装nginx

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。

### 安装

* 首先将代码包和插件复制到系统的/usr/local/src内（可选），然后使用tar命令解压

|  |
| --- |
| [root@tracker opt]# cp nginx-1.4.7.tar.gz /usr/local/src  [root@tracker opt]# cp ngx\_cache\_purge-2.1.tar.gz /usr/local/src  [root@tracker opt]# cp pcre-8.34.tar.gz /usr/local/src  [root@tracker opt]# cp zlib-1.2.8.tar.gz /usr/local/src  [root@tracker opt]# cd /usr/local/src/  [root@tracker src]# tar zxf nginx-1.4.7.tar.gz  [root@tracker src]# tar zxf ngx\_cache\_purge-2.1.tar.gz  [root@tracker src]# tar zxf pcre-8.34.tar.gz  [root@tracker src]# tar zxf zlib-1.2.8.tar.gz  [root@tracker src]# cd nginx-1.4.7  [root@tracker nginx-1.4.7]# ll  总用量 592  drwxr-xr-x. 6 1001 1001 4096 3月 26 02:34 auto  -rw-r--r--. 1 1001 1001 225213 3月 18 21:17 CHANGES  -rw-r--r--. 1 1001 1001 343040 3月 18 21:17 CHANGES.ru  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 conf  -rwxr-xr-x. 1 1001 1001 2369 3月 18 21:17 configure  drwxr-xr-x. 3 1001 1001 4096 3月 26 02:34 contrib  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 html  -rw-r--r--. 1 1001 1001 1397 3月 18 21:17 LICENSE  drwxr-xr-x. 2 1001 1001 4096 3月 26 02:34 man  -rw-r--r--. 1 1001 1001 49 3月 18 21:17 README  drwxr-xr-x. 8 1001 1001 4096 3月 26 02:34 src |

* 运行./configure进行安装前的设置，主要设置安装路径、nginx cache purge插件模块目录、pcre库目录、zlib库目录。

如果提示错误，可能缺少依赖的软件包，需先安装依赖包，再次运行./configure

|  |
| --- |
| [root@storage1 nginx-1.4.7]# ./configure --prefix=/usr/local/nginx --add-module=/usr/local/src/ngx\_cache\_purge-2.1 --with-pcre=/usr/local/src/pcre-8.34/ --with-zlib=/usr/local/src/zlib-1.2.8 |

* 运行make进行编译，确保编译成功。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# make |

* 运行make install进行安装。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# make install |

* 至此nginx以及nginx cache purge插件模块安装完成。

### 配置

* 编辑/usr/local/nginx/conf配置文件目录下的nginx.conf，设置负载均衡以及缓存。

|  |
| --- |
| [root@storage1 nginx-1.4.7]# vim /usr/local/nginx/conf/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=10g inactive=30d;

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

#设置group1的服务器

upstream fdfs\_group1 {

server 172.16.1.203:8080 weight=1 max\_fails=2 fail\_timeout=30s;

server 172.16.1.204:8080 weight=1 max\_fails=2 fail\_timeout=30s;

}

#设置group2的服务器

upstream fdfs\_group2 {

server 172.16.1.205:8080 weight=1 max\_fails=2 fail\_timeout=30s;

server 172.16.1.206:8080 weight=1 max\_fails=2 fail\_timeout=30s;

}

#设置group3的服务器

upstream fdfs\_group3 {

server 172.16.1.207:8080 weight=1 max\_fails=2 fail\_timeout=30s;

server 172.16.1.208:8080 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group2的负载均衡参数

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group3的负载均衡参数

location /group3/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\_group3;

expires 30d;

}

#设置清除缓存的访问权限

location ~ /purge(/.\*) {

allow 127.0.0.1;

allow 172.16.1.0/24;

deny all;

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

}

* 至此，nginx以及nginx cache purge插件模块设置完成。

Yyy注：多store挂载的需要依次添加

location /group1/M01 {

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/M01 {

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;

}

location /group1/M02 {

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/M02 {

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;

}

location /group1/M03 {

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/M03 {

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;

}

### 运行

运行nginx之前，先要把防火墙中对应的端口打开（本例中为8080）。

|  |
| --- |
| [root@tracker nginx-1.4.7]# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT  [root@tracker nginx-1.4.7]# /etc/init.d/iptables save  iptables：将防火墙规则保存到 /etc/sysconfig/iptables：[确定] |

启动nginx，确认启动是否成功。（查看是否对应端口8080是否开始监听）

|  |
| --- |
| [root@tracker nginx-1.4.7]# /usr/local/nginx/sbin/nginx  ngx\_http\_fastdfs\_set pid=40638  [root@tracker nginx-1.4.7]# netstat -unltp | grep nginx  tcp 0 0.0.0.0:8080 0.0.0.0:\* LISTEN 40639/nginx |

* 尝试上传一个文件到FastDFS，然后访问试试。先配置client.conf文件。

|  |
| --- |
| [root@tracker nginx-1.4.7]# vim /etc/fdfs/client.conf |

修改以下参数：

base\_path=/fdfs/tracker #日志存放路径

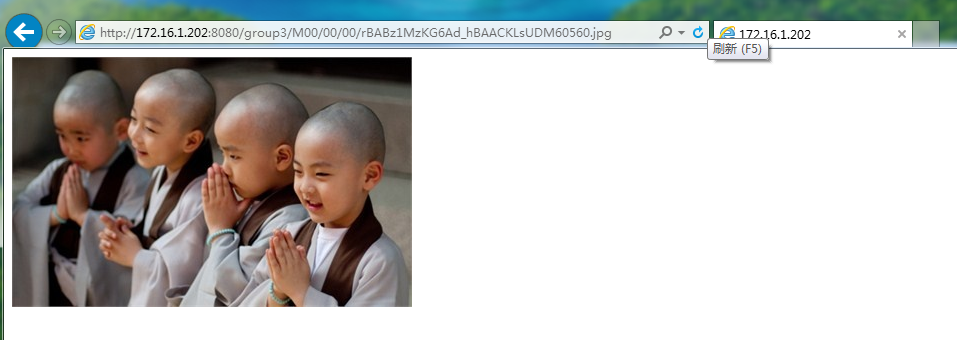
tracker\_server=172.16.1.202:22122 #tracker服务器IP地址和端口号

http.tracker\_server\_port=8080 #tracker服务器的http端口号

使用/usr/local/bin/fdfs\_upload\_file上传一个文件，程序会自动返回文件的URL。

|  |
| --- |
| [root@tracker nginx-1.4.7]# /usr/local/bin/fdfs\_upload\_file /etc/fdfs/client.conf /mnt/monk.jpg  group3/M00/00/00/rBABz1MzKG6Ad\_hBAACKLsUDM60560.jpg |

然后使用浏览器访问：



可以看到文件被正确读取出来了。

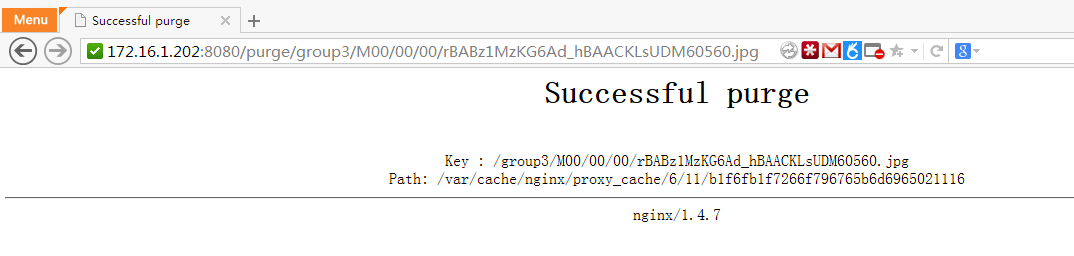
查看nginx的access.log日志，可以看到访问返回200成功。

|  |
| --- |
| [root@tracker nginx-1.4.7]# tail -n 10 -f /usr/local/nginx/logs/access.log  172.16.1.201 - - [26/Mar/2014:13:15:00 +0800] "GET /group3/M00/00/00/rBABz1MzKG6Ad\_hBAACKLsUDM60560.jpg HTTP/1.1" 200 35374 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko" |

查看nginx的cache目录，可以看到已经生成了缓存文件。

|  |
| --- |
| [root@tracker nginx-1.4.7]# ll /var/cache/nginx/proxy\_cache/ -R  /var/cache/nginx/proxy\_cache/:  总用量 8  drwx------. 3 nobody nobody 4096 3月 26 12:57 6  drwxr-xr-x. 2 nobody root 4096 3月 26 13:14 tmp  /var/cache/nginx/proxy\_cache/6:  总用量 4  drwx------. 2 nobody nobody 4096 3月 26 13:08 11  /var/cache/nginx/proxy\_cache/6/11:  总用量 36  -rw-------. 1 nobody nobody 35686 3月 26 13:08 b1f6fb1f7266f796765b6d6965021116 |

如果要手动清除缓存，可以在文件URL之前加上purge：



* 设置开机自动启动。

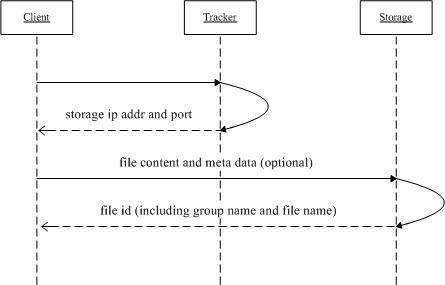
|  |
| --- |
| [root@tracker nginx-1.4.7]# vim /etc/rc.d/rc.local |

将运行命令行添加进文件：/usr/local/nginx/sbin/nginx

至此，tracker服务器上的http反向代理+负载均衡+缓存已经安装完成。

## 使用

### 上传文件

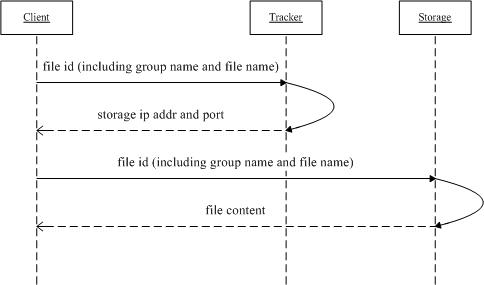


FastDFS提供了多种方式上传文件：

* 使用fdfs\_upload\_file上传（之前已经有演示）
* 使用C语言客户端接口上传
* 使用PHP客户端接口上传
* 使用Java客户端接口上传
* 使用Python客户端接口上传
* 使用.NET客户端接口上传

根据具体情况选择使用。上传均支持断点续传。

### 下载文件



* 使用fdfs\_download\_file通过tracker服务器下载
* 使用nginx通过http方式直接从storage下载（支持断点续传，作者推荐）

### 监视服务器资源

* 使用fdfs\_monitor查看tracker和所有group的运行情况。

|  |
| --- |
| [root@tracker tmp]# /usr/local/bin/fdfs\_monitor /etc/fdfs/client.conf  2014-03-26 15:01:07] DEBUG - base\_path=/fdfs/tracker, connect\_timeout=30, 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  server\_count=1, server\_index=0  tracker server is 172.16.1.202:22122  group count: 3  Group 1:  group name = group1  disk total space = 27789 MB  disk free space = 23825 MB  trunk free space = 0 MB  storage server count = 2  active server count = 2  storage server port = 23000  storage HTTP port = 8080  store path count = 1  subdir count per path = 256  current write server index = 1  current trunk file id = 0  Storage 1:  id = 172.16.1.203  ip\_addr = 172.16.1.203 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 01:48:19  up time =  total storage = 27789 MB  free storage = 23844 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id=  if\_trunk\_server= 0  total\_upload\_count = 1  success\_upload\_count = 1  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 24  success\_upload\_bytes = 24  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 0  success\_sync\_in\_bytes = 0  total\_sync\_out\_bytes = 24  success\_sync\_out\_bytes = 24  total\_file\_open\_count = 1  success\_file\_open\_count = 1  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 1  success\_file\_write\_count = 1  last\_heart\_beat\_time = 2014-03-26 15:00:47  last\_source\_update = 2014-03-26 13:13:55  last\_sync\_update = 1970-01-01 08:00:00  last\_synced\_timestamp = 1970-01-01 08:00:00  Storage 2:  id = 172.16.1.204  ip\_addr = 172.16.1.204 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 18:53:19  up time = 2014-03-26 18:53:19  total storage = 27789 MB  free storage = 23825 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id= 172.16.1.203  if\_trunk\_server= 0  total\_upload\_count = 0  success\_upload\_count = 0  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 0  success\_upload\_bytes = 0  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 24  success\_sync\_in\_bytes = 24  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 1  success\_file\_open\_count = 1  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 1  success\_file\_write\_count = 1  last\_heart\_beat\_time = 2014-03-26 15:00:40  last\_source\_update = 1970-01-01 08:00:00  last\_sync\_update = 2014-03-27 03:38:23  last\_synced\_timestamp = 2014-03-26 13:13:56 (-1s delay)  Group 2:  group name = group2  disk total space = 27789 MB  disk free space = 23825 MB  trunk free space = 0 MB  storage server count = 2  active server count = 2  storage server port = 23000  storage HTTP port = 8080  store path count = 1  subdir count per path = 256  current write server index = 0  current trunk file id = 0  Storage 1:  id = 172.16.1.205  ip\_addr = 172.16.1.205 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 19:37:36  up time = 2014-03-26 19:37:36  total storage = 27789 MB  free storage = 23825 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id=  if\_trunk\_server= 0  total\_upload\_count = 0  success\_upload\_count = 0  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 0  success\_upload\_bytes = 0  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 0  success\_sync\_in\_bytes = 0  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 0  success\_file\_open\_count = 0  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 0  success\_file\_write\_count = 0  last\_heart\_beat\_time = 2014-03-26 15:01:02  last\_source\_update = 1970-01-01 08:00:00  last\_sync\_update = 1970-01-01 08:00:00  last\_synced\_timestamp = 1970-01-01 08:00:00  Storage 2:  id = 172.16.1.206  ip\_addr = 172.16.1.206 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 22:38:04  up time = 2014-03-26 22:38:04  total storage = 27789 MB  free storage = 23825 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id= 172.16.1.205  if\_trunk\_server= 0  total\_upload\_count = 0  success\_upload\_count = 0  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 0  success\_upload\_bytes = 0  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 0  success\_sync\_in\_bytes = 0  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 0  success\_file\_open\_count = 0  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 0  success\_file\_write\_count = 0  last\_heart\_beat\_time = 2014-03-26 15:01:02  last\_source\_update = 1970-01-01 08:00:00  last\_sync\_update = 1970-01-01 08:00:00  last\_synced\_timestamp = 1970-01-01 08:00:00  Group 3:  group name = group3  disk total space = 27789 MB  disk free space = 23825 MB  trunk free space = 0 MB  storage server count = 2  active server count = 2  storage server port = 23000  storage HTTP port = 8080  store path count = 1  subdir count per path = 256  current write server index = 1  current trunk file id = 0  Storage 1:  id = 172.16.1.207  ip\_addr = 172.16.1.207 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 22:50:28  up time = 2014-03-26 22:50:28  total storage = 27789 MB  free storage = 23825 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id= 172.16.1.208  if\_trunk\_server= 0  total\_upload\_count = 1  success\_upload\_count = 1  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 35374  success\_upload\_bytes = 35374  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 0  success\_sync\_in\_bytes = 0  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 1  success\_file\_open\_count = 1  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 1  success\_file\_write\_count = 1  last\_heart\_beat\_time = 2014-03-26 15:01:20  last\_source\_update = 2014-03-27 03:20:13  last\_sync\_update = 1970-01-01 08:00:00  last\_synced\_timestamp = 1970-01-01 08:00:00  Storage 2:  id = 172.16.1.208  ip\_addr = 172.16.1.208 ACTIVE  http domain =  version = 5.01  join time = 2014-03-26 22:49:37  up time = 2014-03-26 22:49:37  total storage = 27789 MB  free storage = 23825 MB  upload priority = 10  store\_path\_count = 1  subdir\_count\_per\_path = 256  storage\_port = 23000  storage\_http\_port = 8080  current\_write\_path = 0  source storage id=  if\_trunk\_server= 0  total\_upload\_count = 0  success\_upload\_count = 0  total\_append\_count = 0  success\_append\_count = 0  total\_modify\_count = 0  success\_modify\_count = 0  total\_truncate\_count = 0  success\_truncate\_count = 0  total\_set\_meta\_count = 0  success\_set\_meta\_count = 0  total\_delete\_count = 0  success\_delete\_count = 0  total\_download\_count = 0  success\_download\_count = 0  total\_get\_meta\_count = 0  success\_get\_meta\_count = 0  total\_create\_link\_count = 0  success\_create\_link\_count = 0  total\_delete\_link\_count = 0  success\_delete\_link\_count = 0  total\_upload\_bytes = 0  success\_upload\_bytes = 0  total\_append\_bytes = 0  success\_append\_bytes = 0  total\_modify\_bytes = 0  success\_modify\_bytes = 0  stotal\_download\_bytes = 0  success\_download\_bytes = 0  total\_sync\_in\_bytes = 35374  success\_sync\_in\_bytes = 35374  total\_sync\_out\_bytes = 0  success\_sync\_out\_bytes = 0  total\_file\_open\_count = 1  success\_file\_open\_count = 1  total\_file\_read\_count = 0  success\_file\_read\_count = 0  total\_file\_write\_count = 1  success\_file\_write\_count = 1  last\_heart\_beat\_time = 2014-03-26 15:01:32  last\_source\_update = 1970-01-01 08:00:00  last\_sync\_update = 2014-03-27 03:20:19  last\_synced\_timestamp = 2014-03-27 03:20:14 (-1s delay) |

### 其他功能

* 使用fdfs\_crc32获取文件CRC
* 使用fdfs\_delete\_file删除文件
* 使用fdfs\_file\_info查看文件属性信息

以上所有功能均可使用接口在代码中调用。

## 测试&问题

停掉一台tracker，upload文件有台报错，但仍成功。

[root@localhost bin]# ./fdfs\_upload\_file /etc/fdfs/client.conf yyy.jpg

[2014-06-23 18:13:10] ERROR - file: ../common/connection\_pool.c, line: 84, connect to 10.4.65.24:22122 fail, errno: 113, error info: No route to host

group2/M03/00/00/CgRBF1On\_j2AUK38AAEGMkytgds556.jpg

1、停掉group中的一台store，会有约30秒，文件会间歇上传失败（轮询）。（heart beat调到5秒仍有约30秒的失败）

[root@centos6 nginx]# fdfs\_upload\_file /etc/fdfs/client.conf /tmp/yyy.jpg

group1/M01/00/00/CgcB5VOhWriATn1\_AADyDEqA1vY223.jpg

[root@centos6 nginx]# fdfs\_upload\_file /etc/fdfs/client.conf /tmp/yyy.jpg

[2014-06-18 17:24:12] ERROR - file: ../common/connection\_pool.c, line: 84, connect to 10.7.1.228:23000 fail, errno: 113, error info: **No route to host**

upload file fail, error no: 113, error info: No route to host

[root@centos6 nginx]# fdfs\_upload\_file /etc/fdfs/client.conf /tmp/yyy.jpg

group1/M00/00/00/CgcB5VOhWsqAHOXvAADyDEqA1vY575.jpg

[root@centos6 nginx]# fdfs\_upload\_file /etc/fdfs/client.conf /tmp/yyy.jpg

[2014-06-18 17:24:30] ERROR - file: ../common/connection\_pool.c, line: 84, connect to 10.7.1.228:23000 fail, errno: 113, error info: No route to host

upload file fail, error no: 113, error info: No route to host

[root@centos6 nginx]#

## 报警监控

[root@VMS06271 ~]# /usr/local/bin/fdfs\_monitor /etc/fdfs/client.conf

[2014-09-17 15:59:49] DEBUG - base\_path=/fdfs/tracker, connect\_timeout=30, network\_timeout=60, tracker\_server\_count=2, 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

server\_count=2, server\_index=1

tracker server is 10.8.115.131:22122

group count: 2

Group 1:

group name = g1

disk total space = 33800196 MB

disk free space = 32077721 MB

trunk free space = 0 MB

**storage server count = 3**

**active server count = 2 小于“storage server count”报警**

storage server port = 23000

storage HTTP port = 8080

store path count = 12

subdir count per path = 256

current write server index = 0

current trunk file id = 0

Storage 1:

id = 10.8.115.124

**ip\_addr = 10.8.115.124 ACTIVE 非“ACTIVE”报警**

http domain =

version = 5.01

join time = 2014-09-11 10:03:51

up time = 2014-09-12 15:29:18

total storage = 33800196 MB

free storage = 32077721 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id=

if\_trunk\_server= 0

**total\_upload\_count = 93**

**success\_upload\_count = 93 小于“total\_upload\_count”报警，值可继承，可调整调整**

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

**total\_delete\_count = 91**

**success\_delete\_count = 91小于“total\_delete\_count”报警，值可继承，可调整调整**

**total\_download\_count = 0**

**success\_download\_count = 0 小于“total\_download\_count”报警，值可继承，可调整调整**

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 330579958

success\_upload\_bytes = 330579958

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 0

success\_download\_bytes = 0

total\_sync\_in\_bytes = 648540228

success\_sync\_in\_bytes = 648540228

total\_sync\_out\_bytes = 661159916

success\_sync\_out\_bytes = 661159916

total\_file\_open\_count = 275

success\_file\_open\_count = 275

total\_file\_read\_count = 0

success\_file\_read\_count = 0

total\_file\_write\_count = 3773

success\_file\_write\_count = 3773

**last\_heart\_beat\_time = 2014-09-17 15:59:30 比当前事件小于1分钟**

last\_source\_update = 2014-09-12 15:30:38

last\_sync\_update = 2014-09-12 15:30:07

last\_synced\_timestamp = 2014-09-12 15:30:05 (0s delay)

Storage 2:

id = 10.8.115.125

ip\_addr = 10.8.115.125 OFFLINE

http domain =

version = 5.01

join time = 2014-09-11 10:09:59

up time =

total storage = 33800196 MB

free storage = 32077721 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id= 10.8.115.124

if\_trunk\_server= 0

total\_upload\_count = 91

success\_upload\_count = 91

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

total\_delete\_count = 89

success\_delete\_count = 89

total\_download\_count = 1

success\_download\_count = 1

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 323304602

success\_upload\_bytes = 323304602

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 3637678

success\_download\_bytes = 3637678

total\_sync\_in\_bytes = 655815584

success\_sync\_in\_bytes = 655815584

total\_sync\_out\_bytes = 646609204

success\_sync\_out\_bytes = 646609204

total\_file\_open\_count = 276

success\_file\_open\_count = 276

total\_file\_read\_count = 14

success\_file\_read\_count = 14

total\_file\_write\_count = 3773

success\_file\_write\_count = 3773

last\_heart\_beat\_time = 2014-09-17 15:57:24

last\_source\_update = 2014-09-12 14:54:05

last\_sync\_update = 2014-09-12 15:30:46

last\_synced\_timestamp = 2014-09-12 15:30:05 (33s delay)

Storage 3:

id = 10.8.115.126

ip\_addr = 10.8.115.126 ACTIVE

http domain =

version = 5.01

join time = 2014-09-11 10:10:03

up time = 2014-09-11 10:10:03

total storage = 33800196 MB

free storage = 32077721 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id= 10.8.115.124

if\_trunk\_server= 0

total\_upload\_count = 91

success\_upload\_count = 91

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

total\_delete\_count = 89

success\_delete\_count = 89

total\_download\_count = 0

success\_download\_count = 0

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 325235626

success\_upload\_bytes = 325235626

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 0

success\_download\_bytes = 0

total\_sync\_in\_bytes = 653884560

success\_sync\_in\_bytes = 653884560

total\_sync\_out\_bytes = 650471252

success\_sync\_out\_bytes = 650471252

total\_file\_open\_count = 275

success\_file\_open\_count = 275

total\_file\_read\_count = 0

success\_file\_read\_count = 0

total\_file\_write\_count = 3773

success\_file\_write\_count = 3773

last\_heart\_beat\_time = 2014-09-17 15:59:22

last\_source\_update = 2014-09-12 15:30:05

last\_sync\_update = 2014-09-12 15:30:46

last\_synced\_timestamp = 2014-09-12 14:54:05 (36m:33s delay)

Group 2:

group name = g2

disk total space = 31255627 MB

disk free space = 29660052 MB

trunk free space = 0 MB

storage server count = 3

active server count = 0

storage server port = 23000

storage HTTP port = 8080

store path count = 12

subdir count per path = 256

current write server index = 0

current trunk file id = 0

Storage 1:

id = 10.8.115.127

ip\_addr = 10.8.115.127 OFFLINE

http domain =

version = 5.01

join time = 2014-09-11 10:10:07

up time =

total storage = 33800196 MB

free storage = 32077734 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id= 10.8.115.129

if\_trunk\_server= 0

total\_upload\_count = 26

success\_upload\_count = 26

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

total\_delete\_count = 24

success\_delete\_count = 24

total\_download\_count = 0

success\_download\_count = 0

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 94579628

success\_upload\_bytes = 94579628

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 0

success\_download\_bytes = 0

total\_sync\_in\_bytes = 185521578

success\_sync\_in\_bytes = 185521578

total\_sync\_out\_bytes = 189159256

success\_sync\_out\_bytes = 189159256

total\_file\_open\_count = 77

success\_file\_open\_count = 77

total\_file\_read\_count = 0

success\_file\_read\_count = 0

total\_file\_write\_count = 1078

success\_file\_write\_count = 1078

last\_heart\_beat\_time = 2014-09-17 15:59:09

last\_source\_update = 2014-09-12 14:43:41

last\_sync\_update = 2014-09-12 15:30:29

last\_synced\_timestamp = 2014-09-12 14:43:41 (46m:47s delay)

Storage 2:

id = 10.8.115.128

ip\_addr = 10.8.115.128 OFFLINE

http domain =

version = 5.01

join time = 2014-09-11 10:10:10

up time =

total storage = 33800196 MB

free storage = 32077734 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id= 10.8.115.129

if\_trunk\_server= 0

total\_upload\_count = 25

success\_upload\_count = 25

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

total\_delete\_count = 25

success\_delete\_count = 25

total\_download\_count = 0

success\_download\_count = 0

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 90941950

success\_upload\_bytes = 90941950

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 0

success\_download\_bytes = 0

total\_sync\_in\_bytes = 189159256

success\_sync\_in\_bytes = 189159256

total\_sync\_out\_bytes = 181883900

success\_sync\_out\_bytes = 181883900

total\_file\_open\_count = 77

success\_file\_open\_count = 77

total\_file\_read\_count = 0

success\_file\_read\_count = 0

total\_file\_write\_count = 1078

success\_file\_write\_count = 1078

last\_heart\_beat\_time = 2014-09-17 15:59:13

last\_source\_update = 2014-09-12 14:43:41

last\_sync\_update = 2014-09-12 15:30:30

last\_synced\_timestamp = 2014-09-12 14:43:41 (46m:47s delay)

Storage 3:

id = 10.8.115.129

ip\_addr = 10.8.115.129 OFFLINE

http domain =

version = 5.01

join time = 2014-09-11 10:10:13

up time =

total storage = 31255627 MB

free storage = 29660052 MB

upload priority = 10

store\_path\_count = 12

subdir\_count\_per\_path = 256

storage\_port = 23000

storage\_http\_port = 8080

current\_write\_path = 0

source storage id=

if\_trunk\_server= 0

total\_upload\_count = 26

success\_upload\_count = 26

total\_append\_count = 0

success\_append\_count = 0

total\_modify\_count = 0

success\_modify\_count = 0

total\_truncate\_count = 0

success\_truncate\_count = 0

total\_set\_meta\_count = 0

success\_set\_meta\_count = 0

total\_delete\_count = 26

success\_delete\_count = 26

total\_download\_count = 0

success\_download\_count = 0

total\_get\_meta\_count = 0

success\_get\_meta\_count = 0

total\_create\_link\_count = 0

success\_create\_link\_count = 0

total\_delete\_link\_count = 0

success\_delete\_link\_count = 0

total\_upload\_bytes = 94579628

success\_upload\_bytes = 94579628

total\_append\_bytes = 0

success\_append\_bytes = 0

total\_modify\_bytes = 0

success\_modify\_bytes = 0

stotal\_download\_bytes = 0

success\_download\_bytes = 0

total\_sync\_in\_bytes = 185521578

success\_sync\_in\_bytes = 185521578

total\_sync\_out\_bytes = 189159256

success\_sync\_out\_bytes = 189159256

total\_file\_open\_count = 77

success\_file\_open\_count = 77

total\_file\_read\_count = 0

success\_file\_read\_count = 0

total\_file\_write\_count = 1078

success\_file\_write\_count = 1078

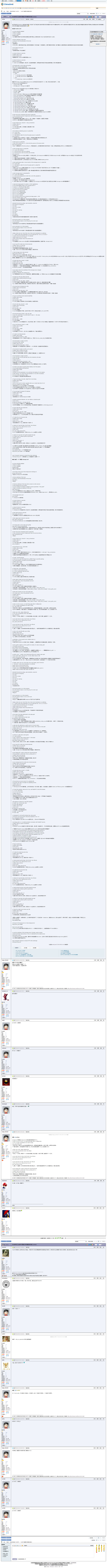
last\_heart\_beat\_time = 2014-09-17 15:59:34

last\_source\_update = 2014-09-12 15:30:28

last\_sync\_update = 2014-09-12 14:43:50

last\_synced\_timestamp = 2014-09-12 14:43:41 (0s delay)

[root@VMS06271 ~]#



# Protocol

The communication protocol of FastDFS is TCP/IP, the package composes of header and body which may be empty.  
  
header format:  
  @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes package length  
  @ 1 byte command  
  @ 1 byte status  
  
note:   
   # TRACKER\_PROTO\_PKG\_LEN\_SIZE (8) bytes number buff is Big-Endian bytes  
  
  
body format:  
1. common command  
\* FDFS\_PROTO\_CMD\_QUIT  
  # function: notify server connection will be closed  
  # request body: none (no body part)  
  # response: none (no header and no body)  
  
\* FDFS\_PROTO\_CMD\_ACTIVE\_TEST  
  # function: active test  
  # request body: none   
  # response body: none  
  
2. storage server to tracker server command  
  
\* the reponse command is TRACKER\_PROTO\_CMD\_STORAGE\_RESP  
  
\* TRACKER\_PROTO\_CMD\_STORAGE\_JOIN  
  # function: storage join to tracker  
  # request body:   
      @ FDFS\_GROUP\_NAME\_MAX\_LEN + 1 bytes: group name  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage http server port  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: path count  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: subdir count per path  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: upload priority  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: join time (join timestamp)  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: up time (start timestamp)  
      @ FDFS\_VERSION\_SIZE bytes: storage server version  
      @ FDFS\_DOMAIN\_NAME\_MAX\_SIZE bytes: domain name of the web server on the storage server  
      @ 1 byte: init flag ( 1 for init done)  
      @ 1 byte: storage server status  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: tracker server count excluding current tracker  
  # response body:   
      @ FDFS\_IPADDR\_SIZE bytes: sync source storage server ip address  
  # memo: return all storage servers in the group only when storage servers changed or return none  
  
    
\* TRACKER\_PROTO\_CMD\_STORAGE\_BEAT  
  # function: heart beat  
  # request body: none or storage stat info  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total upload count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success upload count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total set metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success set metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total delete count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success delete count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total download count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success download count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total get metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success get metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total create link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success create link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total delete link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success delete link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: last source update timestamp  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: last sync update timestamp  
       @TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes:  last synced timestamp  
       @TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes:  last heart beat timestamp  
  # response body: n \* (1 + FDFS\_IPADDR\_SIZE) bytes, n >= 0. One storage entry format:  
      @ 1 byte: storage server status  
      @ FDFS\_IPADDR\_SIZE bytes: storage server ip address  
  # memo: storage server sync it's stat info to tracker server only when storage stat info changed  
  
  
\* TRACKER\_PROTO\_CMD\_STORAGE\_REPORT  
  # function: report disk usage  
  # request body 1 or more than 1 following entries:  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total space in MB  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: free space in MB  
  # response body: same to command TRACKER\_PROTO\_CMD\_STORAGE\_BEAT  
  
  
\* TRACKER\_PROTO\_CMD\_STORAGE\_REPLICA\_CHG  
  # function: replica new storage servers which maybe not exist in the tracker server  
  # request body: n \* (1 + FDFS\_IPADDR\_SIZE) bytes, n >= 1. One storage entry format:  
      @ 1 byte: storage server status  
      @ FDFS\_IPADDR\_SIZE bytes: storage server ip address  
  # response body: none  
  
  
\* TRACKER\_PROTO\_CMD\_STORAGE\_SYNC\_SRC\_REQ  
  # function: source storage require sync. when add a new storage server, the existed storage servers in the same group will ask the tracker server to tell the source storage server which will sync old data to it  
  # request body:  
      @ FDFS\_GROUP\_NAME\_MAX\_LEN: group name  
      @ FDFS\_IPADDR\_SIZE bytes: dest storage server (new storage server) ip address  
  # response body: none or  
     @ FDFS\_IPADDR\_SIZE bytes: source storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: sync until timestamp  
  # memo: if the dest storage server not do need sync from one of storage servers in the group, the response body is emtpy  
  
    
\* TRACKER\_PROTO\_CMD\_STORAGE\_SYNC\_DEST\_REQ  
  # function: dest storage server (new storage server) require sync  
  # request body: none  
  # response body: none or  
     @ FDFS\_IPADDR\_SIZE bytes: source storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: sync until timestamp  
  # memo: if the dest storage server not do need sync from one of storage servers in the group, the response body is emtpy  
  
  
\* TRACKER\_PROTO\_CMD\_STORAGE\_SYNC\_NOTIFY  
  # function: new storage server sync notify  
  # request body:  
     @ FDFS\_IPADDR\_SIZE bytes: source storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: sync until timestamp  
  # response body: same to command TRACKER\_PROTO\_CMD\_STORAGE\_BEAT  
  
  
3. client to tracker server command  
  
\* the reponse command of following 2 commands is TRACKER\_PROTO\_CMD\_SERVER\_RESP  
  
\* TRACKER\_PROTO\_CMD\_SERVER\_LIST\_GROUP  
  # function: list all groups  
  # request body: none  
  # response body: n group entries, n >= 0, the format of each entry:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN+1 bytes: group name  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: free disk storage in MB  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server count  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server http port  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: active server count  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: current write server index  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: store path count on storage server  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: subdir count per path on storage server  
  
  
\* TRACKER\_PROTO\_CMD\_SERVER\_LIST\_STORAGE  
  # function: list storage servers of a group  
  # request body:   
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: the group name to query  
  # response body: n storage entries, n >= 0, the format of each entry:  
       @ 1 byte: status  
       @ FDFS\_IPADDR\_SIZE bytes: ip address  
       @ FDFS\_DOMAIN\_NAME\_MAX\_SIZE  bytes : domain name of the web server  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: source storage server ip address  
       @ FDFS\_VERSION\_SIZE bytes: storage server version  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: join time (join in timestamp)  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: up time (start timestamp)  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total space in MB  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: free space in MB  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: upload priority  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: store path count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: subdir count per path  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: current write path[  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage http port  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total upload count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success upload count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total set metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success set metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total delete count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success delete count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total download count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success download count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total get metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success get metadata count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total create link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success create link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: total delete link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: success delete link count  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: last source update timestamp  
       @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: last sync update timestamp  
       @TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes:  last synced timestamp  
       @TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes:  last heart beat timestamp  
  
\* the reponse command of following 2 commands is TRACKER\_PROTO\_CMD\_SERVICE\_RESP  
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_STORE\_WITHOUT\_GROUP\_ONE  
  # function: query which storage server to store file  
  # request body: none  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port   
     @1 byte: store path index on the storage server   
  
    
  \* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_STORE\_WITHOUT\_GROUP\_ALL  
  # function: query which storage server to store file  
  # request body: none  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address (\* multi)  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port (\*multi)  
     @1 byte: store path index on the storage server  
  
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_STORE\_WITH\_GROUP\_ONE  
  # function: query which storage server to store file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port   
     @1 byte: store path index on the storage server   
  
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_STORE\_WITH\_GROUP\_ALL  
  # function: query which storage server to store file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  (\* multi)  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port   (\* multi)  
     @1 byte: store path index on the storage server   
  
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_FETCH  
  # function: query which storage server to download the file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port   
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_FETCH\_ALL  
  # function: query all storage servers to download the file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port  
     @ n \* (FDFS\_IPADDR\_SIZE - 1) bytes:  storage server ip addresses, n can be 0    
  
\* TRACKER\_PROTO\_CMD\_SERVICE\_QUERY\_UPDATE  
  # function: query which storage server to download the file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ FDFS\_IPADDR\_SIZE - 1 bytes: storage server ip address  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: storage server port  
  
  
4. storage server to storage server command  
\* the reponse command is STORAGE\_PROTO\_CMD\_RESP  
  
\* STORAGE\_PROTO\_CMD\_SYNC\_CREATE\_FILE  
  # function: sync new created file  
  # request body:  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: filename bytes  
     @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file size/bytes  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes : filename  
     @ file size bytes: file content  
  # response body: none  
    
    
\* STORAGE\_PROTO\_CMD\_SYNC\_DELETE\_FILE  
  # function: sync deleted file  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  # response body: none    
  
  
\* STORAGE\_PROTO\_CMD\_SYNC\_UPDATE\_FILE  
# function: sync updated file  
# request body: same to command STORAGE\_PROTO\_CMD\_SYNC\_CREATE\_FILE  
# respose body: none  
  
  
5. client to storage server command  
\* the reponse command is STORAGE\_PROTO\_CMD\_RESP  
  
\* STORAGE\_PROTO\_CMD\_UPLOAD\_FILE  
  # function: upload file to storage server  
  # request body:  
      @ 1 byte: store path index on the storage server  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file size  
      @ FDFS\_FILE\_EXT\_NAME\_MAX\_LEN bytes: file ext name, do not include dot (.)  
      @ file size bytes: file content  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  
  \* STORAGE\_PROTO\_CMD\_UPLOAD\_SLAVE\_FILE  
  # function: upload slave file to storage server  
  # request body:  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: master filename length  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file size  
      @ FDFS\_FILE\_PREFIX\_MAX\_LEN bytes: filename prefix  
      @ FDFS\_FILE\_EXT\_NAME\_MAX\_LEN bytes: file ext name, do not include dot (.)  
      @ master filename bytes: master filename  
      @ file size bytes: file content  
  # response body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  
\* STORAGE\_PROTO\_CMD\_DELETE\_FILE  
  # function: delete file from storage server  
  # request body:  
     @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name  
     @ filename bytes: filename  
  # response body: none  
  
    
\* STORAGE\_PROTO\_CMD\_SET\_METADATA  
  # function: delete file from storage server  
  # request body:  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: filename length   
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: meta data size   
      @ 1 bytes: operation flag,    
           'O' for overwrite all old metadata   
           'M' for merge, insert when the meta item not exist, otherwise update it   
      @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name   
      @ filename bytes: filename  
      @ meta data bytes: each meta data seperated by \x01,  
                         name and value seperated by \x02  
  # response body: none  
  
  
\* STORAGE\_PROTO\_CMD\_DOWNLOAD\_FILE  
  # function: download/fetch file from storage server  
  # request body:  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file offset  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: download file bytes  
      @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name   
      @ filename bytes: filename  
  # response body:  
      @ file content  
    
  
\* STORAGE\_PROTO\_CMD\_GET\_METADATA  
  # function: get metat data from storage server  
  # request body:  
      @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name   
      @ filename bytes: filename  
  # response body  
      @ meta data buff, each meta data seperated by \x01, name and value seperated by \x02  
  
  
\* STORAGE\_PROTO\_CMD\_QUERY\_FILE\_INFO  
  # function: query file info from storage server  
  # request body:  
      @ FDFS\_GROUP\_NAME\_MAX\_LEN bytes: group name   
      @ filename bytes: filename  
  # response body:  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file size  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file create timestamp  
      @ TRACKER\_PROTO\_PKG\_LEN\_SIZE bytes: file CRC32 signature

FastDFS服务器端运行时目录结构如下：  
  ${base\_path}  
    |\_\_data：存放数据文件  
    |\_\_logs：存放日志文件  
  
其中，${base\_path}由配置文件中的参数“base\_path”设定。  
  
一、tracker server  
  
tracker server目录及文件结构：  
  ${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  
  
数据文件storage\_groups.dat和storage\_servers.dat中的记录之间以换行符（\n）分隔，字段之间以西文逗号（,）分隔。  
storage\_groups.dat中的字段依次为：  
  1. group\_name：组名  
  2. storage\_port：storage server端口号  
  
storage\_servers.dat中记录storage server相关信息，字段依次为：  
  1. group\_name：所属组名  
  2. ip\_addr：ip地址  
  3. status：状态  
  4. sync\_src\_ip\_addr：向该storage server同步已有数据文件的源服务器  
  5. sync\_until\_timestamp：同步已有数据文件的截至时间（UNIX时间戳）  
  6. stat.total\_upload\_count：上传文件次数  
  7. stat.success\_upload\_count：成功上传文件次数  
  8. stat.total\_set\_meta\_count：更改meta data次数  
  9. stat.success\_set\_meta\_count：成功更改meta data次数  
  10. stat.total\_delete\_count：删除文件次数  
  11. stat.success\_delete\_count：成功删除文件次数  
  12. stat.total\_download\_count：下载文件次数  
  13. stat.success\_download\_count：成功下载文件次数  
  14. stat.total\_get\_meta\_count：获取meta data次数  
  15. stat.success\_get\_meta\_count：成功获取meta data次数  
  16. stat.last\_source\_update：最近一次源头更新时间（更新操作来自客户端）  
  17. stat.last\_sync\_update：最近一次同步更新时间（更新操作来自其他storage server的同步）  
  
二、storage server  
  
storage server目录及文件结构：  
  ${base\_path}  
    |\_\_data  
    |     |\_\_.data\_init\_flag：当前storage server初始化信息  
    |     |\_\_storage\_stat.dat：当前storage server统计信息  
    |     |\_\_sync：存放数据同步相关文件  
    |     |     |\_\_binlog.index：当前的binlog（更新操作日志）文件索引号  
    |     |     |\_\_binlog.###：存放更新操作记录（日志）  
    |     |     |\_\_${ip\_addr}\_${port}.mark：存放向目标服务器同步的完成情况  
    |     |  
    |     |\_\_一级目录：256个存放数据文件的目录，目录名为十六进制字符，如：00, 1F  
    |           |\_\_二级目录：256个存放数据文件的目录，目录名为十六进制字符，如：0A, CF  
    |\_\_logs  
          |\_\_storaged.log：storage server日志文件  
  
.data\_init\_flag文件格式为ini配置文件方式，各个参数如下：  
  # storage\_join\_time：本storage server创建时间  
  # sync\_old\_done：本storage server是否已完成同步的标志（源服务器向本服务器同步已有数据）  
  # sync\_src\_server：向本服务器同步已有数据的源服务器IP地址，没有则为空  
  # sync\_until\_timestamp：同步已有数据文件截至时间（UNIX时间戳）  
  
storage\_stat.dat文件格式为ini配置文件方式，各个参数如下：  
  # total\_upload\_count：上传文件次数  
  # success\_upload\_count：成功上传文件次数  
  # total\_set\_meta\_count：更改meta data次数  
  # success\_set\_meta\_count：成功更改meta data次数  
  # total\_delete\_count：删除文件次数  
  # success\_delete\_count：成功删除文件次数  
  # total\_download\_count：下载文件次数  
  # success\_download\_count：成功下载文件次数  
  # total\_get\_meta\_count：获取meta data次数  
  # success\_get\_meta\_count：成功获取meta data次数  
  # last\_source\_update：最近一次源头更新时间（更新操作来自客户端）  
  # last\_sync\_update：最近一次同步更新时间（更新操作来自其他storage server）  
  
binlog.index中只有一个数据项：当前binlog的文件索引号  
  
binlog.###，###为索引号对应的3位十进制字符，不足三位，前面补0。索引号基于0，最大为999。一个binlog文件最大为1GB。记录之间以换行符（\n）分隔，字段之间以西文空格分隔。字段依次为：  
  1. timestamp：更新发生时间（Unix时间戳）  
  2. op\_type：操作类型，一个字符  
  3. filename：操作（更新）的文件名，包括相对路径，如：5A/3D/VKQ-CkpWmo0AAAAAAKqTJj0eiic6891.a  
  
${ip\_addr}\_${port}.mark：ip\_addr为同步的目标服务器IP地址，port为本组storage server端口。例如：10.0.0.1\_23000.mark。文件格式为ini配置文件方式，各个参数如下：  
  # binlog\_index：已处理（同步）到的binlog索引号  
  # binlog\_offset：已处理（同步）到的binlog文件偏移量（字节数）  
  # need\_sync\_old：同步已有数据文件标记，0表示没有数据文件需要同步  
  # sync\_old\_done：同步已有数据文件是否完成标记，0表示未完成，1表示已完成  
  # until\_timestamp：同步已有数据截至时间点（UNIX时间戳）  
  # scan\_row\_count：已扫描的binlog记录数  
  # sync\_row\_count：已同步的binlog记录数  
  
数据文件名由系统自动生成，包括5部分：存储服务器IP地址、当前时间（Unix时间戳）、文件大小（字节数）、随机数和文件后缀。文件名长度为33字节。文件可以按目录顺序存放，也可以按照PJW Hash算法hash到65536（256\*256）个目录中分散存储，通过配置文件控制。  
  
[ *本帖最后由 happy\_fish100 于 2009-8-5 18:39 编辑* ]

# FastDFS FAQ

**分享到**

* [一键分享](http://bbs.chinaunix.net/)
* [QQ空间](http://bbs.chinaunix.net/)
* [新浪微博](http://bbs.chinaunix.net/)
* [百度云收藏](http://bbs.chinaunix.net/)
* [人人网](http://bbs.chinaunix.net/)
* [腾讯微博](http://bbs.chinaunix.net/)
* [百度相册](http://bbs.chinaunix.net/)
* [开心网](http://bbs.chinaunix.net/)
* [腾讯朋友](http://bbs.chinaunix.net/)
* [百度贴吧](http://bbs.chinaunix.net/)
* [豆瓣网](http://bbs.chinaunix.net/)
* [搜狐微博](http://bbs.chinaunix.net/)
* [百度新首页](http://bbs.chinaunix.net/)
* [QQ好友](http://bbs.chinaunix.net/)
* [和讯微博](http://bbs.chinaunix.net/)
* [更多...](http://bbs.chinaunix.net/)

[百度分享](http://bbs.chinaunix.net/)

|  |  |
| --- | --- |
| [http://www.chinaunix.net/images_new/login.gif](http://bbs.chinaunix.net/member.php?mod=logging&action=login&logsubmit=yes)  [忘记密码](http://u.it168.com/findPassword?returnUrl=http://bbs.chinaunix.net/home.php?mod=spacecp)   [免费注册](http://bbs.chinaunix.net/member.php?mod=register.php) | [查看新帖](http://bbs.chinaunix.net/listthread.php) | [论坛精华区](http://www.chinaunix.net/jh/) |
|  | |

|  |  |
| --- | --- |
| [ChinaUnix.net](http://bbs.chinaunix.net/) | [http://168.it168.com/newImage/20140604/20140604061257837.jpg](http://bbs.chinaunix.net/thread-4140768-1-1.html) |

|  |  |
| --- | --- |
|  | |
| **平台**： [论坛](http://bbs.chinaunix.net/) [博客](http://blog.chinaunix.net/) [问答](http://ask.chinaunix.net/) [文库](http://wenku.it168.com/) [沙龙](http://www.it168.com/itclub/) [自测](http://selftest.chinaunix.net/) [下载](http://download.chinaunix.net/) [订阅](http://edm.it168.com/) [搜索](http://search.chinaunix.net/) **频道**： [自动化运维](http://www.chinaunix.net/ops/) [虚拟化](http://www.chinaunix.net/virtual/) [服务器](http://www.chinaunix.net/server/) [储存备份](http://www.chinaunix.net/stor/) [C/C++](http://www.chinaunix.net/cpp/) [PHP](http://www.chinaunix.net/php/) [MySQL](http://www.chinaunix.net/mysql/) [嵌入式](http://www.chinaunix.net/embedded/) [Linux系统](http://www.chinaunix.net/linux/) | 窗体顶端    窗体底端 |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| ·[cn域名35元，时代云主机52元](http://www.now.cn/?now_qing_it168wenzi) | ·[初创企业IT系统选购与运维大吐槽](http://bbs.chinaunix.net/thread-4140768-1-1.html) | ·[IBM POWER知识自测 赢三重大奖](http://selftest.chinaunix.net/Start_a416aec4-edfb-477f-94b5-832b433a1bce.html) | ·[Mac OS X和iOS内核初探有奖交流](http://bbs.chinaunix.net/thread-4138999-1-1.html) |

› [论坛](http://bbs.chinaunix.net/forum.php) › [IT运维](http://bbs.chinaunix.net/forum.php?gid=277) › [分布式文件系统（FastDFS）](http://bbs.chinaunix.net/forum.php?mod=forumdisplay&fid=240&page=1) › [FastDFS FAQ （欢迎反馈，我将及时整理）](http://bbs.chinaunix.net/thread-1920470-1-1.html)

**1**[2](http://bbs.chinaunix.net/thread-1920470-2-1.html)[3](http://bbs.chinaunix.net/thread-1920470-3-1.html)[4](http://bbs.chinaunix.net/thread-1920470-4-1.html)[5](http://bbs.chinaunix.net/thread-1920470-5-1.html)[6](http://bbs.chinaunix.net/thread-1920470-6-1.html)[7](http://bbs.chinaunix.net/thread-1920470-7-1.html)[8](http://bbs.chinaunix.net/thread-1920470-8-1.html)[下一页](http://bbs.chinaunix.net/thread-1920470-2-1.html)

[最近访问板块](http://bbs.chinaunix.net/forum.php?mod=forumdisplay&fid=240&page=1) [发新帖](javascript:;)

|  |
| --- |
| [打印](http://bbs.chinaunix.net/forum.php?mod=viewthread&action=printable&tid=1920470)[上一主题](http://bbs.chinaunix.net/forum.php?mod=redirect&goto=nextoldset&tid=1920470)[下一主题](http://bbs.chinaunix.net/forum.php?mod=redirect&goto=nextnewset&tid=1920470) [[FastDFS]](http://bbs.chinaunix.net/forum.php?mod=forumdisplay&fid=240&filter=typeid&typeid=424) [FastDFS FAQ （欢迎反馈，我将及时整理）](http://bbs.chinaunix.net/thread-1920470-1-1.html) [[复制链接]](http://bbs.chinaunix.net/thread-1920470-1-1.html)[11](http://bbs.chinaunix.net/) |



|  |  |
| --- | --- |
| 电梯直达 [跳转到指定楼层](javascript:;)  [1**楼**](http://bbs.chinaunix.net/thread-1920470-1-1.html)[**[收藏(0)]**](http://bbs.chinaunix.net/home.php?mod=spacecp&ac=favorite&type=thread&id=1920470)[**[报告]**](javascript:;)  http://bbs.chinaunix.net/static/image/common/online_moderator.gif发表于 2010-11-22 21:28:17 |[只看该作者](http://bbs.chinaunix.net/forum.php?mod=viewthread&tid=1920470&page=1&authorid=20315669) |[倒序浏览](http://bbs.chinaunix.net/forum.php?mod=viewthread&tid=1920470&extra=page%3D1&ordertype=1)   |  | | --- | | 本帖最后由 happy\_fish100 于 2014-03-20 11:12 编辑   定位问题首先要看日志文件。出现问题时，先检查返回的错误号和错误信息。然后查看服务器端日志，相信可以定位到问题所在。  1. FastDFS适用的场景以及不适用的场景？ FastDFS是为互联网应用量身定做的一套分布式文件存储系统，非常适合用来存储用户图片、视频、文档等文件。对于互联网应用，和其他分布式文件系统相比，优势非常明显。具体情况大家可以看相关的介绍文档，包括FastDFS介绍PPT等等。 出于简洁考虑，FastDFS没有对文件做分块存储，因此不太适合分布式计算场景。  2. FastDFS需要的编译和运行环境是怎样的？ FastDFS Server仅支持unix系统，在Linux和FreeBSD测试通过。在Solaris系统下网络通信方面有些问题。 编译需要的其他库文件有pthread，V5.0以前的版本还依赖libevent；V5.0以后，不再依赖libevent。 pthread使用系统自带的即可。 对libevent的版本要求为1.4.x，建议使用最新的stable版本，如1.4.14b。 注意，千万不要使用libevent 2.0非stable版本。 测试了一下，libevent 2.0.10是可以正常工作的。 在64位系统下，可能需要自己在/usr/lib64下创建libevent.so的符号链接。比如： ln -s /usr/lib/libevent.so /usr/lib64/libevent.so 在ubuntu 11及后续版本，可能会出现找不到动态库pthread库，解决方法参见：<http://bbs.chinaunix.net/thread-2324388-1-2.html>  3. 有人在生产环境中使用FastDFS吗？ 答案是肯定的。据我所知，至少有25家公司在使用FastDFS，其中有好几家是做网盘的公司。 其中存储量最大的一家，集群中存储group数有400个，存储服务器超过800台，存储容量达到6PB，文件数超过1亿，Group持续增长中。。。 以下是使用FastDFS的用户列表：   某大型网盘（因对方要求对公司名保密，就不提供名字了。有400个group，存储容量达到了6PB，文件数超过1亿）     UC （<http://www.uc.cn/>，存储容量超过10TB）   支付宝（<http://www.alipay.com/>）   京东商城（<http://www.360buy.com/>）   淘淘搜（<http://www.taotaosou.com/>）   飞信（[http://feixin.1008\*\*/](http://feixin.1008**/)）   赶集网（<http://www.ganji.com/>）   淘米网（<http://www.61.com/>）   迅雷（<http://www.xunlei.com/>）   蚂蜂窝（<http://www.mafengwo.cn/>）   丫丫网（<http://www.iyaya.com/>）   虹网（[http://3g.ahong.com](http://3g.ahong.com/)）   5173（<http://www.5173.com/>）   华夏原创网（<http://www.yuanchuang.com/>）   华师京城教育云平台（<http://www.hsjdy.com.cn/>）   视友网（<http://www.cuctv.com/>）   搜道网（<http://www.sodao.com/>）   58同城（<http://www.58.com/>）   商务联盟网（<http://www.biz72.com/>）   中青网（<http://www.youth.cn/>）   缤丽网 （<http://www.binliy.com/>）   飞视云视频（<http://www.freeovp.com/>）   梦芭莎（<http://www.moonbasa.com/>）   活动帮（[www.eventsboom.com](http://www.eventsboom.com/)）   51CTO（<http://www.51cto.com/>）   搜房网（<http://www.soufun.com/>）  4. 启动storage server时，一直处于僵死状态。 A：启动storage server，storage将连接tracker server，如果连不上，将一直重试。直到连接成功，启动才算真正完成。 出现这样情况，请检查连接不上tracker server的原因。 友情提示：从V2.03以后，多tracker server在启动时会做时间上的检测，判断是否需要从别的tracker server同步4个系统文件。 触发时机是第一个storage server连接上tracker server后，并发起join请求。 如果集群中有2台tracker server，而其中一台tracker没有启动，可能会导致storage server一直处于僵死状态。  5. 执行fdfs\_test或fdfs\_test1上传文件时，服务器返回错误号2 错误号表示没有ACTIVE状态的storage server。可以执行fdfs\_monitor查看服务器状态。  6. 如何让server进程退出运行？ 直接kill即可让server进程正常退出，可以使用killall命令，例如： killall fdfs\_trackerd killall fdfs\_storaged 也可以使用FastDFS自带的stop.sh脚本，如： /usr/local/bin/stop.sh  /usr/local/bin/fdfs\_storaged /etc/fdfs/storage.conf stop.sh只会停止命令行（包括参数）完全相同的进程。 千万不要使用-9参数强杀，否则可能会导致binlog数据丢失的问题。  7. 如何重启server进程？ 可以kill掉server进程后，执行启动命令行。如： killall fdfs\_trackerd /usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf 或者直接使用FastDFS自带的restart.sh脚本，如： /usr/local/bin/restart.sh  /usr/local/bin/fdfs\_trackerd /etc/fdfs/tracker.conf  8. 跨运营商通信异常问题 比如电信和网通机房相互通信，可能会存在异常，有两种表现：   1）不能建立连接，这个比较直接，肯定是网络连接的问题   2）可以正常建立连接，但接收和发送数据失败，这个问题比较隐蔽，正常网络环境下，不应该出现此类问题。        还有人碰到过从一个方向建立连接可以正常通信，但从另外一个方向就不能正常通信的情况。 解决办法：   尝试将服务端口改小，建议将端口修改为1024以下。比如将storage服务端口由23000修改为873等，也可以试试修改为8080   如果问题还不能解决，请联系你的网络（机房）服务商。  9. fdfs\_test和fdfs\_test1是做什么用的？    这两个是FastDFS自带的测试程序，会对一个文件上传两次，分别作为主文件和从文件。返回的文件ID也是两个。    并且会上传文件附加属性，storage server上会生成4个文件。    这两个程序仅用于测试目的，请不要用作实际用途。    V2.05提供了比较正式的三个小工具：       上传文件：/usr/local/bin/fdfs\_upload\_file  <config\_file> <local\_filename>       下载文件：/usr/local/bin/fdfs\_download\_file <config\_file> <file\_id> [local\_filename]        删除文件：/usr/local/bin/fdfs\_delete\_file <config\_file> <file\_id>  10. 什么是主从文件？ 主从文件是指文件ID有关联的文件，一个主文件可以对应多个从文件。     主文件ID = 主文件名 + 主文件扩展名     从文件ID = 主文件名 + 从文件后缀名 + 从文件扩展名 使用主从文件的一个典型例子：以图片为例，主文件为原始图片，从文件为该图片的一张或多张缩略图。 FastDFS中的主从文件只是在文件ID上有联系。FastDFS server端没有记录主从文件对应关系，因此删除主文件，FastDFS不会自动删除从文件。 删除主文件后，从文件的级联删除，需要由应用端来实现。 主文件及其从文件均存放到同一个group中。 主从文件的生成顺序：   1）先上传主文件（如原文件），得到主文件ID   2）然后上传从文件（如缩略图），指定主文件ID和从文件后缀名（当然还可以同时指定从文件扩展名），得到从文件ID。   11. 如何删除无效的storage server？ 可以使用fdfs\_monitor来删除。命令行如下： /usr/local/bin/fdfs\_monitor <config\_filename> delete <group\_name> <storage\_ip> 例如： /usr/local/bin/fdfs\_monitor /etc/fdfs/client.conf delete group1 192.168.0.100 注意：如果被删除的storage server的状态是ACTIVE，也就是该storage server还在线上服务的情况下，是无法删除掉的。  12. FastDFS扩展模块升级到V1.06及以上版本的注意事项 apache和nginx扩展模块版本v1.06及以上版本，需要在配置文件/etc/fdfs/fastdfs\_mod.conf中设置storage server的存储路径信息。 一个示例如下所示： store\_path\_count=1 store\_path0=/home/yuqing/fastdfs store\_path\_count和store\_path#均需要正确设置，必须和storage.conf中的相应配置完全一致，否则将导致文件不能正确下载！  13. nginx和apache扩展模块与FastDFS server版本对应关系     扩展模块1.05：  针对FastDFs server v2.x，要求server版本大于等于v2.09     扩展模块1.07及以上版本：  针对FastDFs server v3.x  14. FastDFS有QQ技术交流群吗？    有的。群号：164684842，欢迎大家加入交流。  15. 上传文件失败，返回错误码28，这是怎么回事？   返回错误码28，表示磁盘空间不足。注意FastDFS中有预留空间的概念，在tracker.conf中设置，配置项为：reserved\_storage\_space，缺省值为4GB，即预留4GB的空间。   请酌情设置reserved\_storage\_space这个参数，比如可以设置为磁盘总空间的20%左右。  16. fdfs\_trackerd或者fdfs\_storaged的日志中出现：malloc task buff failed字样的错误，这是怎么回事？   出现此类信息表示已经达到最大连接数。server端支持的最大连接数可以通过max\_connections这个参数来设置。   出现这样的问题，需要排查一下是否客户端使用不当导致的，比如客户端没有及时关闭无用的连接。  17. FastDFS的文件ID中可以反解出哪些字段？ 文件ID中除了包含group name和存储路径外，文件名中可以反解出如下几个字段：   1）文件创建时间（unix时间戳，32位整数）   2）文件大小   3）上传到的源storage server IP地址（32位整数）   4）文件crc32校验码   5）随机数（这个字段用来避免文件重名）  18. 为什么生成的token验证无法通过？   出现这样的问题，请进行如下两项检查：   1）确认调用token生成函数，传递的文件ID中没有包含group name。传递的文件ID格式形如：M00/00/1B/wKgnVE84utyOG9hEAAATz5-S0SI99.java   2）确认服务器时间基本是一致的，注意服务器时间不能相差太多，不要相差到分钟级别。  19. 最新程序包的下载地址是什么？    因google code不支持上传程序包，最新的程序包可以在sourceforge上下载，下载地址：<https://sourceforge.net/projects/fastdfs/files/> | |
| * 本版精华 * 热门专题 * 文库|博客 * [mfs（mooseFS）文件系统](http://bbs.chinaunix.net/thread-1643863-1-1.html) * [Lustre 主要的测试工具](http://bbs.chinaunix.net/thread-2092122-1-1.html) * [Lustre with the IP-SAN](http://bbs.chinaunix.net/thread-2009317-1-1.html) * [Lustre文件系统安装配置](http://bbs.chinaunix.net/thread-2152718-1-1.html) * [freebsd-MOOSEFS-MFS实施笔记！！](http://bbs.chinaunix.net/thread-1457559-1-1.html) * [很久之前写的Lustre测试报告](http://bbs.chinaunix.net/thread-2034099-1-1.html) * [lustre（CFS）的安装配置过程，大家给个建议](http://bbs.chinaunix.net/thread-2138984-1-1.html) * [用VMWare 实现4节点的Lustre 并行文件系统集群](http://bbs.chinaunix.net/thread-783629-1-1.html) * [用VMWare 实现4节点的Lustre 并行文件系统集群](http://bbs.chinaunix.net/thread-2172161-1-1.html) * [创建hacmp基本步骤](http://bbs.chinaunix.net/zhuanti/1027/hacmpceshibuzhou_1027994.shtml) * [chroot命令的用法](http://bbs.chinaunix.net/zhuanti/1045/chrootminglingxiangjie_1045228.shtml) * [java解析xml文件](http://bbs.chinaunix.net/zhuanti/1027/jiexixmlwenjian_1027830.shtml) * [Unix 命令普及：如何用tar命令备份300G分区到20G的磁带机上~~](http://bbs.chinaunix.net/zhuanti/1045/unixtarcidaiji_1045649.shtml) * [通过ssh远程执行命令问题](http://bbs.chinaunix.net/zhuanti/1044/sshyuanchengzhixingmingling_1044939.shtml) * [debian 无线网络配置？](http://bbs.chinaunix.net/zhuanti/1021/debianwuxianweituoguan_1021117.shtml) * [关于linux模块编程?](http://bbs.chinaunix.net/zhuanti/1015/linuxmokuai_1015275.shtml) * [关于arm-linux-gcc](http://bbs.chinaunix.net/zhuanti/1002/armlinuxgcccygwin_1002540.shtml) * [Flex](http://bbs.chinaunix.net/zhuanti/1020/flex35localzhcn_1020548.shtml) * [如何用VASP计算晶格常数](http://bbs.chinaunix.net/zhuanti/1042/vaspruhechakanzhendongdejieguo_1042257.shtml) * [Microsoft Azure 负载平衡服务](http://blog.chinaunix.net/uid-7528962-id-4298853.html) * [Windows Azure 虚拟网络中虚拟机的网络隔离选项](http://blog.chinaunix.net/uid-7528962-id-4298824.html) * [设置 Windows Azure Active Directory ACS，向 Windows Azure Pack 提供身份](http://blog.chinaunix.net/uid-7528962-id-4297610.html) * [Windows Azure公有云新增VPN及虚拟网络联接功能](http://blog.chinaunix.net/uid-7528962-id-4297556.html) * [python webkit 模拟网页点击事件](http://blog.chinaunix.net/uid-29461699-id-4287053.html)   本人开发了一款开源的分布式文件系统FastDFS，纯C编写，非常轻量级。它解决大容量存储和大并发访问的问题，当前版本为V5.01，请各位大虾多多支持。详情参阅：http://bbs.chinaunix.net/forum-240-1.html  [**系统提供的库函数存在内存泄漏问题**](http://ask.chinaunix.net/question/785566)  |  [**初学UNIX环境高级编程的，关于cat的疑问**](http://ask.chinaunix.net/question/785562)  |  [**chinaunix博客什么时候可以设置背景啊，感觉 ...**](http://ask.chinaunix.net/question/785550)  |  [**一个访问量较大网站的服务器。配置那些文件 ...**](http://ask.chinaunix.net/question/785535)  | |
|  |
|  |

fastDFS 多个tracker如何配置？ nginx又该如何配置？ http访问IP地址如何确定

tracker支持配置多个的，写多行就ok了  
nginx 有个mod\_client的配置文件 里边也可以配置多个tracker Server地址的

# Nginx+FastDFS的配置

      这几天在玩FastDFS，在惊叹余大之牛的同时也由衷地感谢鱼大的共享。针对Nginx+FastDFS的配置，做了些简单记录，希望能对后面了解这方面的朋友有所帮助。也希望余大及其他对这块有研究的朋友多多指正。  
      具体步骤如下：  
1、安装FastDFS（我装的是最新的版本FastDFS\_v4.06），具体安装步骤这里就不重复了。  
  
2、安装Nginx（我装的版本是：nginx-1.4.2），可直接以root用户按默认安装，我是非root用户指定目录编译安装：  
   1）下载nginx安装包及依赖包（nginx-1.4.2.tar.gz、pcre-8.33.tar.gz、zlib-1.2.8.tar.gz）  
   2）分别解压：  
      # tar zxvf nginx-1.4.2.tar.gz   
      # tar zxvf pcre-8.33.tar.gz  
      # tar zxvf zlib-1.2.8.tar.gz  
   3）编译安装  
      # cd nginx-1.4.2  
      #./configure --prefix=/webapp/nginx --with-pcre=/webapp/tools/pcre-8.33 --with-zlib=/webapp/tools/zlib-1.2.8  
      # make  
      # make install  
   4）修改nginx.conf配置文件，启动nginx验证是否安装成功  
  
3、Nginx添加模块  
   1）下载安装包（我用的是fastdfs-nginx-module\_v1.15）  
   2）解压：tar xzf fastdfs-nginx-module\_v1.15.tar.gz  
   3）编译安装（Nginx的安装源目录，即nginx-1.4.2解压目录）：  
   # ./configure --prefix=/webapp/nginx --add-module=/webapp/tools/fastdfs-nginx-module/src --with-pcre=/webapp/tools/pcre-8.33 --with-zlib=/webapp/tools/zlib-1.2.8  
   # make  
   # make install  
   说明：如果需要修改mod\_fastdfs.conf文件的路径，需要在安装之前修改conf配置文件中的：-DFDFS\_MOD\_CONF\_FILENAME='\"/etc/fdfs/mod\_fastdfs.conf\"'"  
  
4、修改配置文件  
   1）nginx.conf  
     在server中增加配置：  
     访问路径不带group名（storage只有一个group的情况），如/M00/00/00/xxx：   
      location /M00 {  
            ngx\_fastdfs\_module;  
        }  
     访问路径带group名（storage对应有多个group的情况），如/group1/M00/00/00/xxx:  
      location ~ /group([0-9])/M00 {  
            ngx\_fastdfs\_module;  
        }  
   2）mod\_fastdfs.conf  
        将fastdfs-nginx-module\_v1.15解压目录fastdfs-nginx-module/src下的mod\_fastdfs.conf复制到-DFDFS\_MOD\_CONF\_FILENAME配置的目录下，如/etc/fdfs/。  
    修改配置：  
    tracker\_server=192.168.1.241:22122        -- tracker server的ip和端口，此处可以写多个tracker server，每行一个  
    url\_have\_group\_name = true              --访问路径带group时为true否则为false，多个group的情况下，必须为true，为false时，group\_count必须为0  
  
storage只有一个group的情况：  
   group=group1                           --storage对应的group  
   store\_path0=/webapp/fastdfs/storage        --storage存储路径  
   group\_count = 0  
  
storage对应有多个group的情况：  
   url\_have\_group\_name = true      --必须设置为true才能支持多个group  
   group\_count = 2                 --storage的group个数  
  
   [group1]  
   group\_name=group1  
   storage\_server\_port=23000  
   store\_path\_count=1  
   store\_path0=/webapp/fastdfs/storage  
     
   [group2]  
   group\_name=group2  
   storage\_server\_port=23000  
   store\_path\_count=1  
   store\_path0=/webapp/fastdfs/storage2  
  
5、重启Nginx，验证。

# Nginx

两个server 对应两个storage  
每个storage 对应一个nginx  
location /group1/M00 {  
           # internal;  
            proxy\_next\_upstream http\_502 http\_504 error timeout i            nvalid\_header;  
            proxy\_pass http://fdfs\_group1;  
            expires 30d;  
        }  
    upstream fdfs\_group1 {  
        server 192.168.248.91:8990 weight=1 max\_fails=2 fail\_timeout=30s;  
        server 192.168.248.90:8993 weight=1 max\_fails=2 fail\_timeout=30s;  
  
}  
通过nginx 访问storage 怎么防止没同步完全的storage不能被访问

# **nginx+tracker server**

的配置是怎样的，请给出具体的配置  
    A: 下面是我的配置, 能用.

upstream group1 {

server 192.168.92.23:8888;

server 192.168.92.33:8888;

}

upstream group2 {

server 192.168.92.21:8888;

server 192.168.92.31:8888;

}

upstream group3 {

server 192.168.92.22:8888;

server 192.168.92.32:8888;

}

upstream group4 {

server 192.168.92.20:8888;

server 192.168.92.30:8888;

}

server {

listen 8887;

server\_name 192.168.92.11;

location /group1/ {

proxy\_pass http://group1/;

proxy\_redirect default ;

}

location /group2/ {

proxy\_pass http://group2/;

proxy\_redirect default ;

}

location /group3/ {

proxy\_pass http://group3/;

proxy\_redirect default ;

}

location /group4/ {

proxy\_pass http://group4/;

proxy\_redirect default ;

}

location ~ \.php$ {

proxy\_pass http://192.168.92.11;

}

}

# api使用手册

为了避免大家有一样的烦恼，写了个api使用手册，提取出精华部分分享，请自己对齐缩进；  
本api使用手册是基于循环处理在线请求的应用场景考虑的，所以有continue或break用来表示本个请求处理结束，继续处理后面的请求；  
个人可以根据情况可以设计成tracker和storage短连接的简单方式，也可设计为storage连接池的方式，步骤都一样：  
  
FastDFS api使用手册之文件上传，其余部分流程类似：  
  
7.1  配置client.conf  
  
7.2  初始化并传入配置文件路径  
这一步在程序start时完成，参见附件fdfs\_load\_test.cpp最后的StartApp().  
if ((result=fdfs\_client\_init(conf\_filename)) != 0)  
{  
    return result;  
}  
  
7.3  获取tracker连接  
获取跟tracker的连接，可以容灾，只要一个tracker可用，即能获取到连接。  
自从FastDFSv1.22开始支持api多线程安全，多线程环境使用api主要是获取tracker连接时不同。多线程的客户端使用这个函数tracker\_get\_connection\_r()，避免各个并发之间互相影响,其余使用tracker\_get\_connection()。  
  
   //为了避免频繁重连，只要不出错，就只连tracker和storage一次          
pTrackerServer = tracker\_get\_connection();  
if (pTrackerServer == NULL)  
   //tracker\_get\_connection\_r(pTrackerServer);   //多线程安全函数  
   //if (pTrackerServer->sock <= 0)  
   {  
//连不上tracker主备机，本次请求处理失败，给用户出错信息，然后继续处理以后的请求，处理以后的请求前调用continue去重连tracker，等tracker只要有一台恢复正常后恢复业务。  
       sleep(1);                //可以间隔多少毫秒再试  
       continue;  
  }  
7.4  询问tracker可用的storage  
获取tracker连接后向该tracker查询当前哪个storage可用，使用tracker\_query\_storage\_store函数。  
参数store\_path\_index用来支持1个storage支持多个磁盘分区，返回的值用在下面的函数调用中，storageServer变量返回可用storage的group、ip和port：  
                store\_path\_index = 0;  
                if ((result=tracker\_query\_storage\_store(pTrackerServer, \  
                                &storageServer, &store\_path\_index)) != 0)  
                {  
                        //表示本tracker异常或没有可用storage，本次请求处理失败，给用户出错信息，但继续处理以后的请求，然后返回到step3去重新获取tracker:  
                                      fdfs\_quit(pTrackerServer);  
                                      tracker\_disconnect\_server(pTrackerServer);  
                                      sleep(1);                //可以间隔多少毫秒再试  
                         continue;  
                }  
7.5  连接该storage  
根据返回的storage ip等信息连接该storage，并发的程序，每个cgi进程或每个线程可以重复使用storage连接直到连接异常：  
            if ((result=tracker\_connect\_server(&storageServer)) != 0)  
            {  
                //tracker给了一个连不上的storage，本次请求处理失败，给用户出错信息,重连tracker。  
                fdfs\_quit(pTrackerServer);  
                tracker\_disconnect\_server(pTrackerServer);  
                sleep(1);  
                continue;  
            }  
  
Fdfs\_connpool\_test.cpp示例了用连接池，适合后台批量导入文件的程序，用数组即可实现连接池，因为每个线程或进程使用的storage连接数最大为所有group的storage总的部署数目，数组的size即等于这个数接口；  
查找时根据返回的storage ip等信息遍历这个小数组，比较ip和端口等是否相同，没有找到即连接该storage，连接好后放到连接池中。  
7.6  上传文件  
上传文件可分3种情况，多线程的例子只示例了1种情况的上传，更全的请参见作者自带的例子或fdfs\_simple\_test.c。  
出错处理在3种情况调用的后面。  
  
因为需要专门的db来保存返回的文件索引和应用的信息，可以不用Metadata，下面两个参数meta\_list, meta\_count填NULL和0即可。  
  
        //文件后缀名,FDFS\_UPLOAD\_BY\_FILE分支时这样使用，FDFS\_UPLOAD\_BY\_BUFF或另一种类时直接给file\_ext\_name赋值，例如赋值”jpeg”  
                file\_ext\_name = strrchr(local\_filename, '.');  
                if (file\_ext\_name != NULL)  
                {  
                        file\_ext\_name++;  
                }  
  
                //这个参数填””  
                strcpy(group\_name, "");  
                //这里是分类上传，具体应用时选择一类即可，第一类是文件已经存放在客户端本次磁盘了，指定文件名（含路径）上传。  
                if (upload\_type == FDFS\_UPLOAD\_BY\_FILE)  
                {  
                        result = storage\_upload\_by\_filename(pTrackerServer, \  
                                &storageServer, store\_path\_index, \  
                                local\_filename, file\_ext\_name, \  
                                meta\_list, meta\_count, \  
                                group\_name, remote\_filename);  
  
                        printf("storage\_upload\_by\_filename\n");  
                }  
  
                //第二类是完整的文件已由用户上传到web程序内存中的file\_content这个buffer内，file\_size参数指明buffer内收到的文件长度。  
                else if (upload\_type == FDFS\_UPLOAD\_BY\_BUFF)  
                {  
                        char \*file\_content;  
                        if ((result=getFileContent(local\_filename, \  
                                        &file\_content, &file\_size)) == 0)  
                        {  
                        result = storage\_upload\_by\_filebuff(pTrackerServer, \  
                                &storageServer, store\_path\_index, \  
                                file\_content, file\_size, file\_ext\_name, \  
                                meta\_list, meta\_count, \  
                                group\_name, remote\_filename);  
                        free(file\_content);  
                        }  
  
                        printf("storage\_upload\_by\_filebuff\n");  
                }  
  
                //第三类跟第一类类似，也需要本地磁盘有文件，只是上传完成后可以执行一个业务定义的回调函数。  
                else  
                {  
                        struct stat stat\_buf;  
  
                        if (stat(local\_filename, &stat\_buf) == 0 && \  
                                S\_ISREG(stat\_buf.st\_mode))  
                        {  
                        file\_size = stat\_buf.st\_size;  
                        result = storage\_upload\_by\_callback(pTrackerServer, \  
                                &storageServer, store\_path\_index, \  
                                uploadFileCallback, local\_filename, \  
                                file\_size, file\_ext\_name, \  
                                meta\_list, meta\_count, \  
                                group\_name, remote\_filename);  
                        }  
  
                        printf("storage\_upload\_by\_callback\n");  
                }  
  
                if (result != 0)  
                {  
                        //上传失败，本次请求处理失败，给用户出错信息，清理该storage的连接，可以返回循环体开始处重连tracker和storage。  
                         fdfs\_quit(&storageServer);  
                                      tracker\_disconnect\_server(&storageServer);  
                                      fdfs\_quit(pTrackerServer);  
                                      tracker\_disconnect\_server(pTrackerServer);  
                                      break;  
                }  
  
上传成功则保存文件索引信息group\_name和 remote\_filename，后面的timestamp和file size可以不要。  
sprintf(file\_id, "%s/%s", group\_name, remote\_filename);  
url\_len = sprintf(file\_url, "http://%s:%d/%s", \  
pTrackerServer->ip\_addr, \  
g\_tracker\_server\_http\_port, file\_id);  
  
                memset(buff, 0, sizeof(buff));  
                base64\_decode\_auto(&context, remote\_filename + FDFS\_FILE\_PATH\_LEN, \  
                        strlen(remote\_filename) - FDFS\_FILE\_PATH\_LEN \  
                        - (FDFS\_FILE\_EXT\_NAME\_MAX\_LEN + 1), buff, &len);  
                printf("group\_name=%s, remote\_filename=%s\n", \  
                        group\_name, remote\_filename);  
                printf("file timestamp=%d\n", buff2int(buff+sizeof(int)));  
                printf("file size="INT64\_PRINTF\_FORMAT"\n", \  
                        buff2long(buff+sizeof(int)\*2));  
                printf("file url: %s\n", file\_url);  
                        //成功则不要关闭tracker和storage连接，下次处理请求时继续使用  
  
7.7  关闭连接，清理客户端  
客户端退出时调用一次下面函数即可，注意在上面的循环处理出错时不要调用fdfs\_client\_destroy();清理整个客户端，注意下面函数的说明和作用：  
  
//循环处理请求时没关闭则关闭storage连接  
fdfs\_quit(&storageServer);  
tracker\_disconnect\_server(&storageServer);  
  
//循环处理请求时没关闭则要关闭tracker连接  
fdfs\_quit(pTrackerServer);  
tracker\_close\_all\_connections();  
  
//清理客户端  
fdfs\_client\_destroy();  
  
[ *本帖最后由 happy\_fastdfs 于 2009-11-9 14:27 编辑* ]

给一个多线程的例子，默认是长连接的，可以改为短连接：  
  
void \* MultiThreadTest( void \*pParam )  
{  
    int                                 MessageSum=0;  
    memcpy(&MessageSum, pParam,4);  
      
    int nStartTime=time(NULL);  
    int nReqStartTime=0,nReqEndTime=0,nTimeout=0;  
  
    TrackerServerInfo theTrackerServer;  
    memset(&theTrackerServer,0,sizeof(theTrackerServer));  
    TrackerServerInfo \*pTrackerServer = &theTrackerServer;  
      
    int result=0;  
    TrackerServerInfo storageServer;  
    char group\_name[FDFS\_GROUP\_NAME\_MAX\_LEN + 1];  
    char remote\_filename[256];  
    char buff[32];  
    char token[32 + 1];  
    char file\_id[128];  
    char file\_url[256];  
    int len;  
    int url\_len;  
    time\_t ts;  
    char \*file\_buff;  
    int64\_t file\_size;  
    char \*meta\_buff;  
    int store\_path\_index;  
    struct base64\_context context;  
    nReqStartTime=time(NULL);  
  
    base64\_init\_ex(&context, 0, '-', '\_', '.');//为了最后解析出文件大小需要初始化这个变量  
     
    int i=0;//发送数目  
      
    //处理在线请求时可以改成while(Running)  
    while(i<MessageSum)  
    {  
        //pTrackerServer = tracker\_get\_connection();  
        tracker\_get\_connection\_r(pTrackerServer);   //多线程安全函数  
        if (pTrackerServer->sock <= 0)  
        {  
            sleep(1);  
            continue;  
        }  
  
        store\_path\_index = 0;  
        if ((result=tracker\_query\_storage\_store(pTrackerServer, \  
                        &storageServer, &store\_path\_index)) != 0)  
        {  
            //查询tracker获取storage失败,重连tracker  
            printf("tracker\_query\_storage fail, " \  
                    "error no: %d, error info: %s\n", \  
                    result, strerror(result));  
            fdfs\_quit(pTrackerServer);  
            tracker\_disconnect\_server(pTrackerServer);  
            sleep(1);  
            continue;  
        }  
  
        printf("group\_name=%s, ip\_addr=%s, port=%d\n", \  
                storageServer.group\_name, \  
                storageServer.ip\_addr, \  
                storageServer.port);  
  
        if ((result=tracker\_connect\_server(&storageServer)) != 0)  
        {  
            //tracker给了一个连不上的storage,重连tracker  
            fdfs\_quit(pTrackerServer);  
            tracker\_disconnect\_server(pTrackerServer);  
            sleep(1);  
            continue;  
        }  
          
        //1下面保持tracker和storage的各1个长连接，本线程固定跟一个storage交互，出错后才返回重连进行容灾  
        //2实际中如果是大文件没必要保持长连接，则下面这个for循环去掉成顺序执行，出错时continue回while循环去重连tracker和storage  
        for(;i<MessageSum;i++)  
        {  
            //以上传为例,上传目前有3类，根据应用自己选一类即可,参见版本自带的fdfs\_test.c  
            //if (strcmp(operation, "upload") == 0)  
            {  
                strcpy(group\_name, "");  
                result = storage\_upload\_by\_filename(pTrackerServer, \  
                                &storageServer, store\_path\_index, \  
                                local\_filename, NULL, \  
                                NULL, 0, \  
                                group\_name, remote\_filename);       
  
                //下面的出错或成功处理，跟上述上传种类无关,通用的  
                if (result != 0)  
                {  
                    printf("storage\_upload\_by\_filename fail, " \  
                                "error no: %d, error info: %s\n", \  
                                result, strerror(result));  
                    fdfs\_quit(&storageServer);  
                    tracker\_disconnect\_server(&storageServer);  
                    fdfs\_quit(pTrackerServer);  
                    tracker\_disconnect\_server(pTrackerServer);  
  
                    //上传次数的统计，测试用  
                    pthread\_mutex\_lock(&theSumMutex);  
                    nTradeFailed ++;  
                    nSum++;  
                    pthread\_mutex\_unlock(&theSumMutex);          
                    break;  
                }  
  
                sprintf(file\_id, "%s/%s", group\_name, remote\_filename);  
                url\_len = sprintf(file\_url, "http://%s:%d/%s", \  
                                pTrackerServer->ip\_addr, \  
                                g\_tracker\_server\_http\_port, file\_id);  
                /\*  
                if (g\_anti\_steal\_token)  
                {  
                        ts = time(NULL);  
                        fdfs\_http\_gen\_token(&g\_anti\_steal\_secret\_key, file\_id, \  
                                        ts, token);  
                        sprintf(file\_url + url\_len, "?token=%s&ts=%d", \  
                                token, (int)ts);  
                }\*/  
  
                memset(buff, 0, sizeof(buff));  
                base64\_decode\_auto(&context, remote\_filename + FDFS\_FILE\_PATH\_LEN, \  
                        strlen(remote\_filename) - FDFS\_FILE\_PATH\_LEN \  
                         - (FDFS\_FILE\_EXT\_NAME\_MAX\_LEN + 1), buff, &len);  
                printf("group\_name=%s, remote\_filename=%s\n", \  
                        group\_name, remote\_filename);  
                printf("file timestamp=%d\n", buff2int(buff+sizeof(int)));  
                printf("file size="INT64\_PRINTF\_FORMAT"\n", \  
                        buff2long(buff+sizeof(int)\*2));  
                printf("file url: %s\n", file\_url);  
              
                //上传次数的统计，测试用  
                pthread\_mutex\_lock(&theSumMutex);  
                nTradeOK ++;  
                nSum++;  
                pthread\_mutex\_unlock(&theSumMutex);  
            }                    
        }  
  
        fdfs\_quit(&storageServer);  
        tracker\_disconnect\_server(&storageServer);  
        fdfs\_quit(pTrackerServer);  
        tracker\_disconnect\_server(pTrackerServer);  
    }  
    //上传次数的统计，测试用  
    int nEndTime=time(NULL);  
    pthread\_mutex\_lock(&theSumMutex);  
    OverThread ++;  
    nTimeAllUsed += (nEndTime - nStartTime);  
    pthread\_mutex\_unlock(&theSumMutex);          
  
    return NULL;  
}