

3.2.利用fastDFS存储图片

3.2.1什么是FastDFS

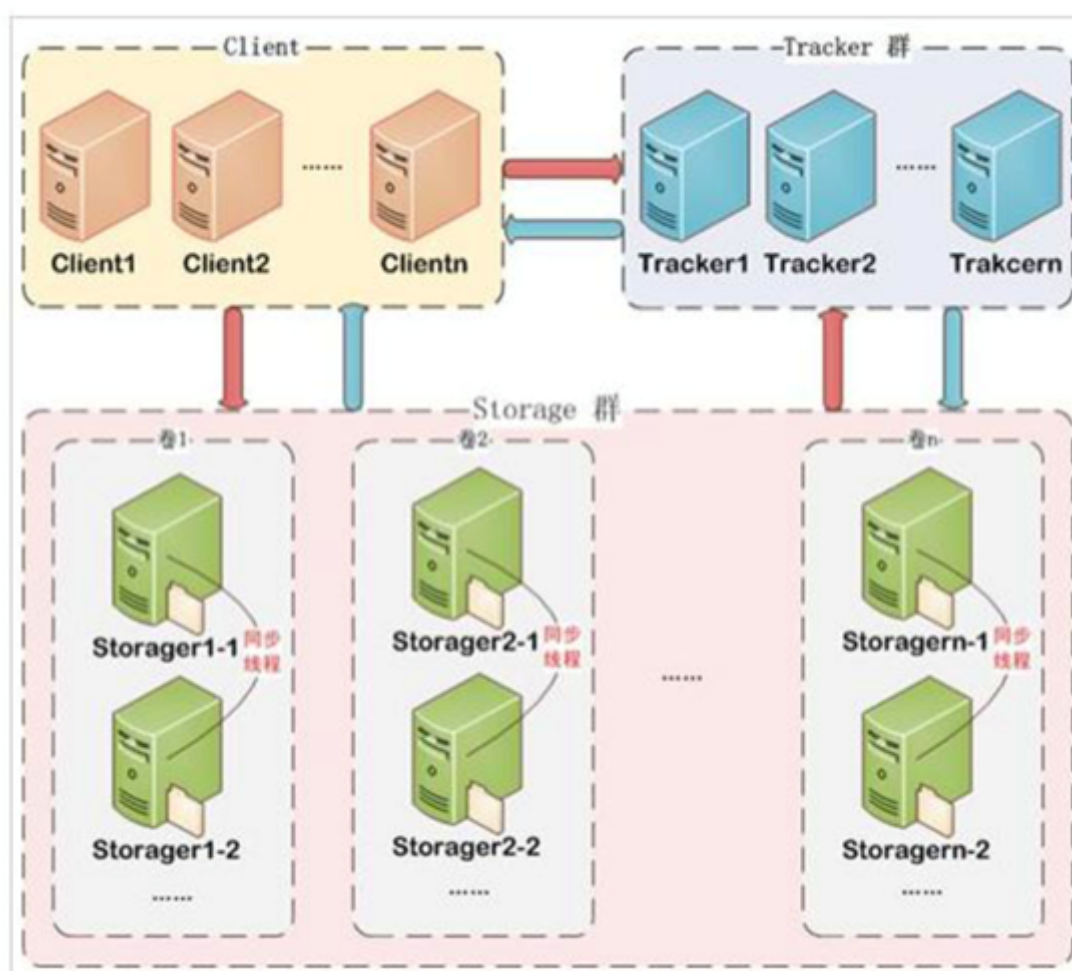
FastDFS 是用 c 语言编写的一款开源的分布式文件系统。FastDFS 为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，并注重高可用、高性能等指标，使用 FastDFS 很容易搭建一套高性能的文件服务器集群提供文件上传、下载等服务。

优点：

FastDFS 架构包括 Tracker server 和 Storage server。客户端请求 Tracker server 进行文件上传、下载，通过 Tracker server 调度最终由 Storage server 完成文件上传和下载。

Tracker server 作用是负载均衡和调度，通过 Tracker server 在文件上传时可以根据一些方法找到 Storage server 提供文件上传服务。可以将 tracker 称为追踪服务器或调度服务器。

Storage server 作用是文件存储，客户端上传的文件最终存储在 Storage 服务器上，Storage server 没有实现自己的文件系统而是利用操作系统的文件系统来管理文件。可以将 storage 称为存储服务器。



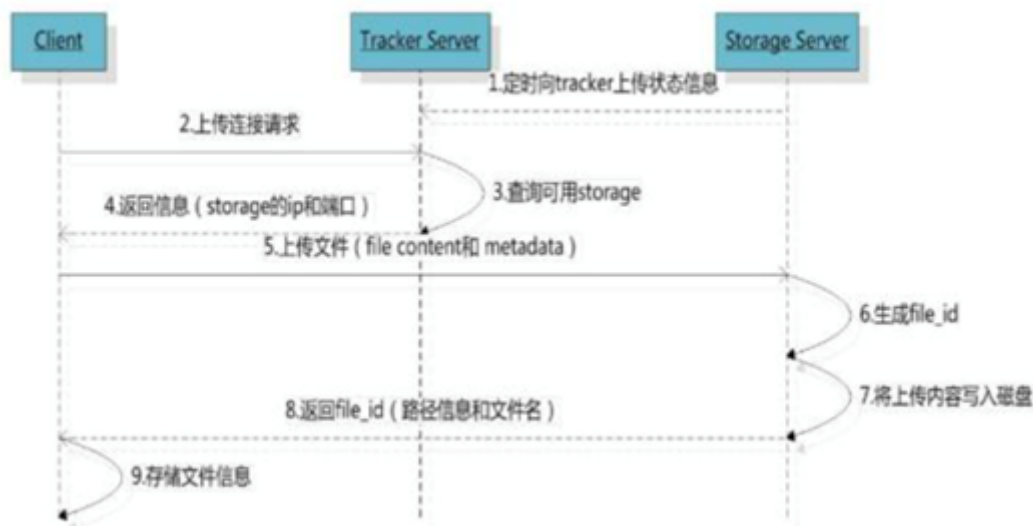
服务端两个角色：

Tracker:管理集群，tracker 也可以实现集群。每个 tracker 节点地位平等。收集 Storage 集群的状态。

Storage:实际保存文件

Storage 分为多个组，每个组之间保存的文件是不同的。每个组内部可以有多个成员，组成员内部保存的内容是一样的，组成员的地位是一致的，没有主从的概念。

3.2.2文件上传流程



客户端上传文件后存储服务将文件 ID 返回给客户端，此文件 ID 用于以后访问该文件的索引信息。文件索引信息包括:组名，虚拟磁盘路径，数据两级目录，文件名。

group1 /M00 /02/44/ wKgDrE34E8wAAAAAAAAAGkEiYJK42378.sh

组名: 文件上传后所在的 storage 组名称，在文件上传成功后有 storage 服务器返回，需要客户端自行保存。

虚拟磁盘路径: storage 配置的虚拟路径，与磁盘选项 store_path*对应。如果配置了 store_path0 则是 M00，如果配置了 store_path1 则是 M01，以此类推。

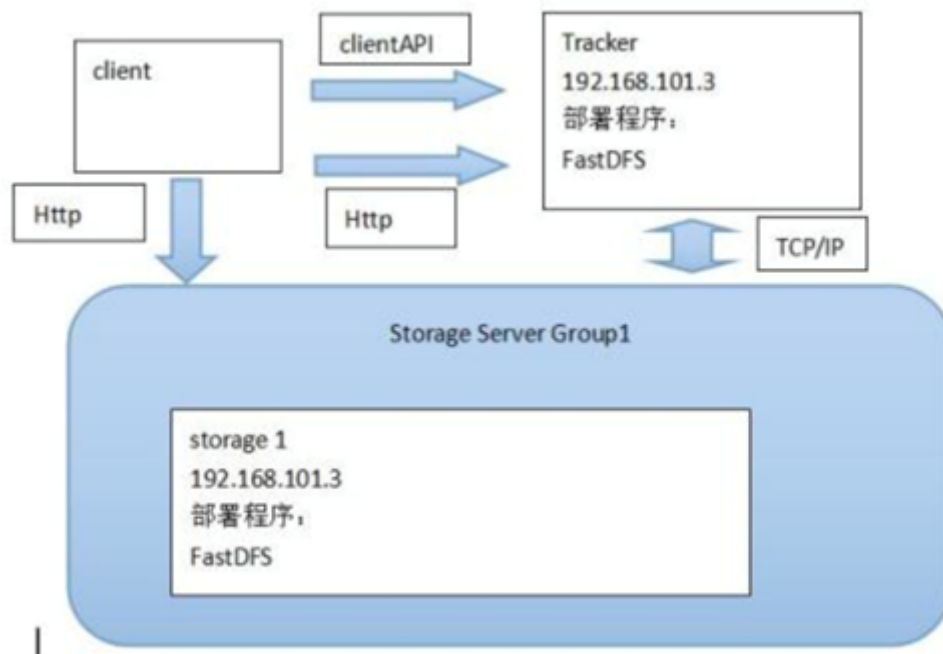
数据两级目录: storage 服务器在每个虚拟磁盘路径下创建的两级目录，用于存储数据文件。

文件名: 与文件上传时不同。是由存储服务根据特定信息生成，文件名包含:源存储服务器 IP 地址、文件创建时间戳、文件大小、随机数和文件拓展名等信息。

3.2.3文件下载流程



3.2.4简易FastDFS架构



3.2.5 FastDFS安装

3.2.5.1 安装FastDFS依赖包

1. 解压缩libfastcommon-master.zip
2. 进入到libfastcommon-master的目录中
3. 执行./make.sh
4. sudo apt-get install make
5. 执行sudo ./make.sh install

3.2.5.2 安装FastDFS

1. 解压缩fastdfs-master.zip
2. 进入到 fastdfs-master目录中
3. 执行 ./make.sh
4. 执行 sudo ./make.sh install

3.2.5.3 配置跟踪服务器tracker

```
1 | sudo cp /etc/fdfs/tracker.conf.sample /etc/fdfs/tracker.conf
```

2. 在/home/itcast/目录中创建目录 fastdfs/tracker

```
1 | mkdir -p /home/itcast/fastdfs/tracker
```

3. 编辑/etc/fdfs/tracker.conf配置文件 sudo vim /etc/fdfs/tracker.conf

修改 base_path=/home/itcast/fastdfs/tracker

3.2.5.4 配置存储服务器storage

```
1 | sudo cp /etc/fdfs/storage.conf.sample /etc/fdfs/storage.conf
```

2. 在/home/itcast/fastdfs/ 目录中创建目录 storage

```
1 | mkdir -p /home/itcast/fastdfs/storage
```

3. 编辑/etc/fdfs/storage.conf配置文件 `sudo vim /etc/fdfs/storage.conf`

修改内容:

```
1 | base_path=/home/itcast/fastdfs/storage
2 | store_path0=/home/itcast/fastdfs/storage
3 | tracker_server=自己ubuntu虚拟机的ip地址:22122
```

3.2.5.5启动tracker和storage

进入到/etc/fdfs/下面执行以下两条指令

```
1 | sudo fdfs_trackerd /etc/fdfs/tracker.conf
2 | sudo fdfs_storaged /etc/fdfs/storage.conf
```

3.2.5.6测试是否安装成功

1. `sudo cp /etc/fdfs/client.conf.sample /etc/fdfs/client.conf`
2. 编辑/etc/fdfs/client.conf配置文件 `sudo vim /etc/fdfs/client.conf`

修改内容:

```
1 | base_path=/home/itcast/fastdfs/tracker
2 | tracker_server=自己ubuntu虚拟机的ip地址:22122
```

3. 上传文件测试(fastDHT)

`sudo fdfs_upload_file /etc/fdfs/client.conf` 要上传的图片文件

如果返回类似**group1/M00/00/00/rBIK6VcaP0aARXXvAAHrUgHEviQ394.jpg** 的文件id则说明文件上传成功

3.2.5.7安装fastdfs-nginx-module

1. 解压缩 `nginx-1.8.1.tar.gz`
2. 解压缩 `fastdfs-nginx-module-master.zip`
3. 进入nginx-1.8.1目录中
4. 执行

```
1 | sudo ./configure --prefix=/usr/local/nginx/ --add-module=fastdfs-nginx-module-master的解压后的目录的绝对路径/src
```

注意: 这时候会报一个错, 说没有PCRE库

```
checking for PCRE library ... not found
checking for PCRE library in /usr/local/ ... not found
checking for PCRE library in /usr/include/pcre/ ... not found
checking for PCRE library in /usr/pkg/ ... not found
checking for PCRE library in /opt/local/ ... not found

./configure: error: the HTTP rewrite module requires the PCRE library.
You can either disable the module by using --without-http_rewrite_module
option, or install the PCRE library into the system, or build the PCRE library
statically from the source with nginx by using --with-pcre=<path> option.
```

下载缺少的库

```
1 | sudo apt-get install libpcre3 libpcre3-dev
```

- 首先你需要去更换源，因为ubuntu自带的源没有这个库
- 更换下载源为阿里的源
- 先把原来的源文件备份

```
1 | sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

- 编辑源文件

```
1 | sudo vim /etc/apt/sources.list
```

把原来的内容全部删掉，粘贴一下内容：

```
1 deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted
  universe multiverse
2 deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted
  universe multiverse
3
4 deb http://mirrors.aliyun.com/ubuntu/ bionic-security main
  restricted universe multiverse
5 deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main
  restricted universe multiverse
6
7 deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main
  restricted universe multiverse
8 deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main
  restricted universe multiverse
9
10 deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main
   restricted universe multiverse
11 deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main
   restricted universe multiverse
12
13 deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main
   restricted universe multiverse
14 deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main
   restricted universe multiverse
```

更换完源之后执行

```
1 | sudo apt-get update
2 | sudo apt-get install libpcre3 libpcre3-dev
```

然后进入nginx-1.8.1目录中，再次执行：

```
1 | sudo ./configure --prefix=/usr/local/nginx/ --add-module=fastdfs-
  nginx-module-master解压后的目录的绝对路径/src
```

然后编译：

```
1 | sudo make
```

这时候还会报一个错（错误还真多），错误原因是因为nginx编译的时候把警告当错误处理，事实上这个警告并不影响（程序员忽略警告）：

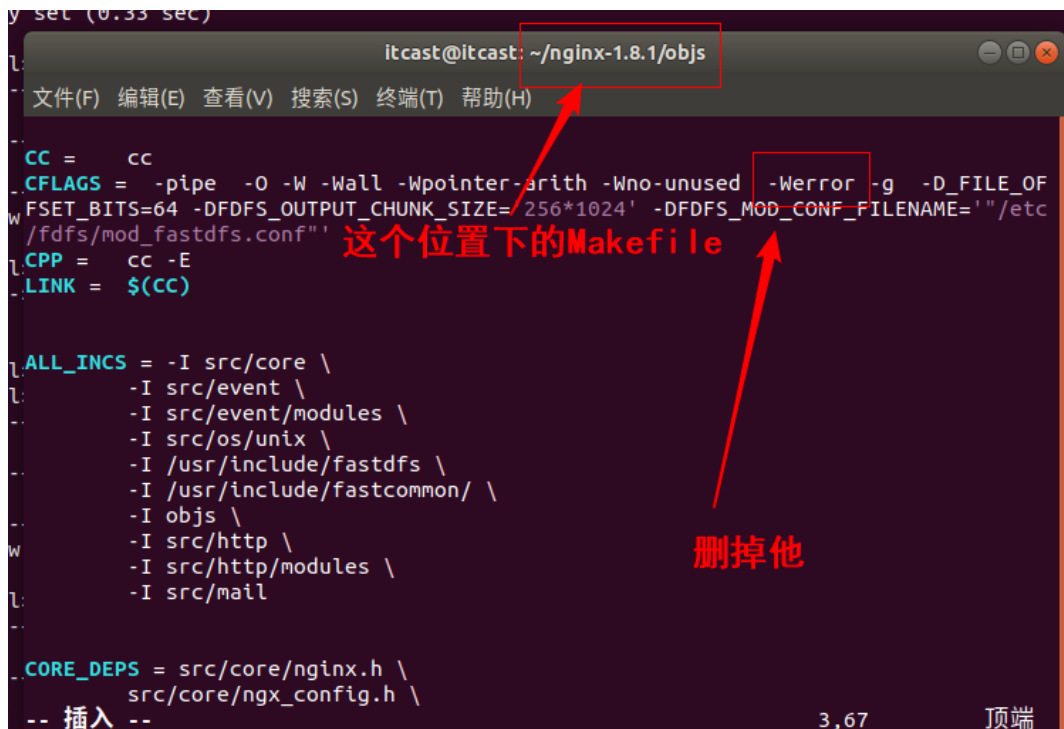
```
src/core/nginx_murmurhash.o \
src/core/nginx_murmurhash.c
src/core/nginx_murmurhash.c: In function 'ngx_murmur_hash2':
src/core/nginx_murmurhash.c:37:11: error: this statement may fall through [-Werror=implicit-fallthrough=]
    h ^= data[2] << 16;
    ~~~~~^~~~~
src/core/nginx_murmurhash.c:38:5: note: here
    case 2:
    ~~~~~
src/core/nginx_murmurhash.c:39:11: error: this statement may fall through [-Werror=implicit-fallthrough=]
    h ^= data[1] << 8;
    ~~~~~^~~~~
src/core/nginx_murmurhash.c:40:5: note: here
    case 1:
    ~~~~~
cc1: all warnings being treated as errors
objs/Makefile:440: recipe for target 'objs/src/core/nginx_murmurhash.o' failed
make[1]: *** [objs/src/core/nginx_murmurhash.o] Error 1
make[1]: 离开目录"/home/itcast/nginx-1.8.1"
Makefile:8: recipe for target 'build' failed
make: *** [build] Error 2
```

解决方法：

找到objs目录下的Makefile

vim Makefile

删掉里面的-Werror(如果没有修改权限，修改一下这个文件的权限，`chmod 777 Makefile`)



```
itcast@itcast: ~/nginx-1.8.1/objs
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

CC = cc
CFLAGS = -pipe -O -W -Wall -Wpointer-arith -Wno-unused -Werror -g -D_FILE_OFFSET_BITS=64 -DFDDFS_OUTPUT_CHUNK_SIZE='256*1024' -DFDDFS_MOD_CONF_FILENAME='/etc/fdfs/mod_fastdfs.conf'
CPP = cc -E
LINK = $(CC)

ALL_INCS = -I src/core \
-I src/event \
-I src/event/modules \
-I src/os/unix \
-I /usr/include/fastdfs \
-I /usr/include/fastcommon/ \
-I objs \
-I src/http \
-I src/http/modules \
-I src/mail

CORE_DEPS = src/core/nginx.h \
src/core/nginx_config.h \
-- 插入 --
```

然后回到nginx-1.8.1目录中

执行完成后继续执行**sudo make**

执行**sudo make install**

5. `sudo cp fastdfs-nginx-module-master`解压后的目录中src下mod_fastdfs.conf
/etc/fdfs/mod_fastdfs.conf

6. `sudo vim /etc/fdfs/mod_fastdfs.conf`

修改内容:

```
1 connect_timeout=10
2 tracker_server=自己ubuntu虚拟机的ip地址:22122
3 url_have_group_name=true
4 store_path0=/home/itcast/fastdfs/storage
```

7. `sudo cp` 解压缩的fastdfs-master目录中的conf中的http.conf /etc/fdfs/http.conf

8. `sudo cp` 解压缩的fastdfs-master目录中conf的mime.types /etc/fdfs/mime.types

9. `sudo vim /usr/local/nginx/conf/nginx.conf`

在http部分中添加配置信息如下:

```
1 server {
2     listen      8888;
3     server_name localhost;
4     location ~ /group[0-9]/ {
5         ngx_fastdfs_module;
6     }
7     error_page   500 502 503 504 /50x.html;
8     location = /50x.html {
9         root     html;
10    }
11 }
12
```

10. 启动Nginx

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

3.2.6使用go客户端上传文件测试

- 下载包

```
1 go get -u -v github.com/weilaihui/fdfs_client
```

这时候会报一个错:

```
itcast@itcast:~/fdfs$ go get github.com/weilaihui/fdfs_client
package golang.org/x/crypto/ssh/terminal: unrecognized import path "golang.org/x/crypto/ssh/terminal" (https fetch: Get https://golang.org/x/crypto/ssh/terminal?go-get=1: dial tcp 216.239.37.1:443: i/o timeout)
package golang.org/x/sys/unix: unrecognized import path "golang.org/x/sys/unix" (https fetch: Get https://golang.org/x/sys/unix?go-get=1: dial tcp 216.239.37.1:443: i/o timeout)
```

这是因为我们的网络有长城防火墙, 不能直接去google下载相应的包, 所以就失败啦

解决办法:

- 在~/workspace/go/src 目录下面创建一个golang.org/x目录

```
1 cd ~/workspace/go/src
2 mkdir -p golang.org/x
```

- 进入golang.org/x下载两个包

```
1 | cd golang.org/x
2 | git clone https://github.com/golang/crypto.git
3 | git clone https://github.com/golang/sys.git
```

- 然后再执行最初的下载命令

```
1 | go get github.com/weilaihui/fdfs_client
```

- go操作fastDFS的方法

- 先导包，把我们下载的包导入

```
1 | import "github.com/weilaihui/fdfs_client"
```

- 导包之后,我们需要指定配置文件生成客户端对象

```
1 | client,_:=fdfs_client.NewFdfsClient("/etc/fdfs/client.conf")
```

- 接着我们就可以通过client对象执行文件上传，上传有两种方法，一种是通过文件名，一种是通过字节流

- 通过文件名上传**UploadByFilename**,参数是文件名（必须通过文件名能找到要上传的文件），返回值是fastDFS定义的一个结构体，包含组名和文件ID两部分内容

```
1 | fdfsresponse,err := client.UploadByFilename("fileName")
```

- 通过字节流上传**UploadByBuffer**,参数是字节数组和文件后缀，返回值和通过文件名上传一样。

```
1 | fdfsresponse,err := client.UploadByBuffer(fileBuffer,ext)
```