**FastDFS 分布式文件系统（部署和运维）**

**FastDFS介绍**

**1.1   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。

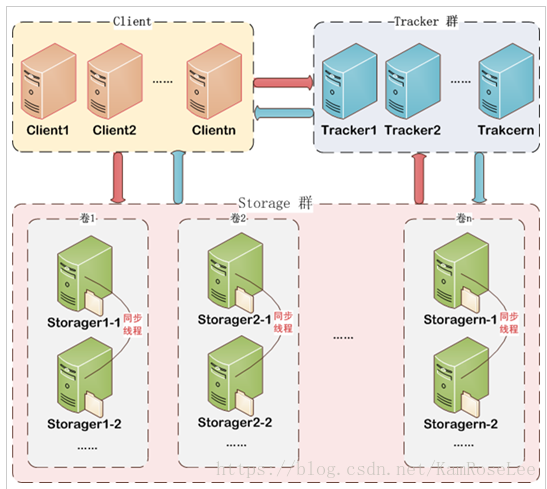
**1.2   FastDFS架构**

         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Trackerserver调度最终由Storage server完成文件上传和下载。

         Trackerserver作用是负载均衡和调度，通过Trackerserver在文件上传时可以根据一些策略找到Storageserver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。

         Storageserver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。

如下图：



**1.2.1  Tracker 集群**

         FastDFS集群中的Tracker server可以有多台，Trackerserver之间是相互平等关系同时提供服务，Trackerserver不存在单点故障。客户端请求Trackerserver采用轮询方式，如果请求的tracker无法提供服务则换另一个tracker。

**1.2.2  Storage集群**

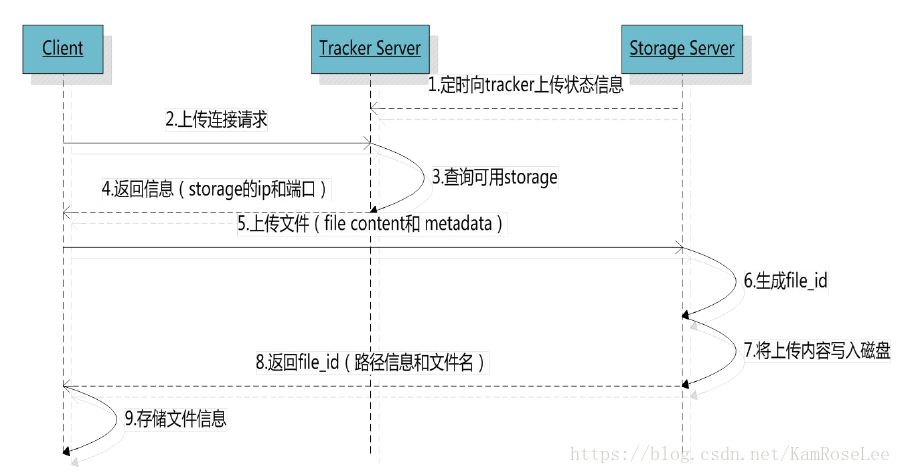
         Storage集群采用了分组存储方式。storage集群由一个或多个组构成，集群存储总容量为集群中所有组的存储容量之和。一个组由一台或多台存储服务器组成，组内的Storage server之间是平等关系，不同组的Storageserver之间不会相互通信，同组内的Storageserver之间会相互连接进行文件同步，从而保证同组内每个storage上的文件完全一致的。一个组的存储容量为该组内存储服务器容量最小的那个，由此可见组内存储服务器的软硬件配置最好是一致的。

         采用分组存储方式的好处是灵活、可控性较强。比如上传文件时，可以由客户端直接指定上传到的组也可以由tracker进行调度选择。一个分组的存储服务器访问压力较大时，可以在该组增加存储服务器来扩充服务能力（纵向扩容）。当系统容量不足时，可以增加组来扩充存储容量（横向扩容）。

**1.2.3  Storage状态收集**

         Storage server会连接集群中所有的Tracker server，定时向他们报告自己的状态，包括磁盘剩余空间、文件同步状况、文件上传下载次数等统计信息。

**1.2.4  文件上传流程**



         客户端上传文件后存储服务器将文件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。

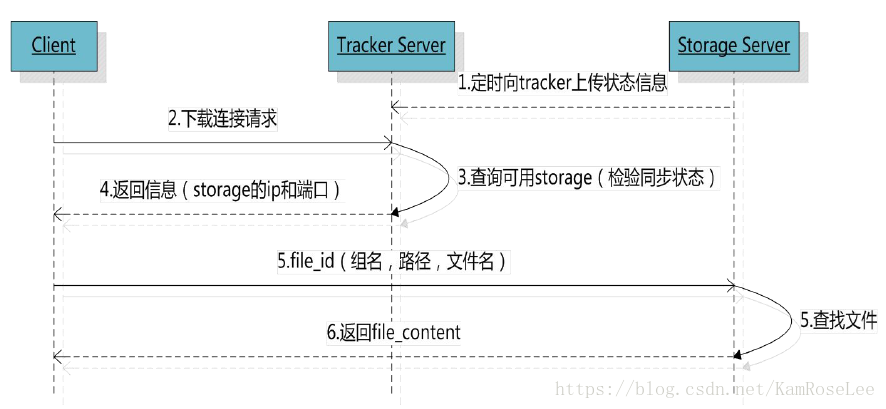
n 组名：文件上传后所在的storage组名称，在文件上传成功后有storage服务器返回，需要客户端自行保存。

n 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。

n 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。

n 文件名：与文件上传时不同。是由存储服务器根据特定信息生成，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。

**1.2.5  文件下载流程**



tracker根据请求的文件路径即文件ID 来快速定义文件。

比如请求下边的文件：

1.通过组名tracker能够很快的定位到客户端需要访问的存储服务器组是group1，并选择合适的存储服务器提供客户端访问。

2.存储服务器根据“文件存储虚拟磁盘路径”和“数据文件两级目录”可以很快定位到文件所在目录，并根据文件名找到客户端需要访问的文件。

**2      FastDFS+Nginx实现文件服务器**

**2.1   架构**

**2.1.1  架构图**

**2.2   FastDFS--tracker安装**

在192.168.101.3上安装tracker。

在192.168.101.3上安装tracker。

**2.2.1  下载**

tracker和storage使用相同的安装包，下载地址： https://github.com/happyfish100/FastDFS

本教程下载：fastdfs-5.11.zip

**2.2.2  FastDFS安装环境**

   FastDFS是C语言开发，建议在linux上运行，本教程使用Centos7作为安装环境。

   安装FastDFS需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc：yum install -y gcc-c++

**2.2.3  安装libevent**

FastDFS依赖libevent库，需要安装：

yum -y install libevent

**2.2.4  安装libfastcommon**

         libfastcommon是FastDFS官方提供的，libfastcommon包含了FastDFS运行所需要的一些基础库。

将libfastcommon-1.0.39拷贝至/op/fs/下

cd /op/fs

tar -zxvf libfastcommon-1.0.39.tar.gz

cd libfastcommon-1.0.39

./make.sh

./make.sh install

**2.2.5  tracker编译安装**

将fastdfs-5.11.zip拷贝至/op/fs/下

tar -zxvf fastdfs-5.11.zip.tar.gz

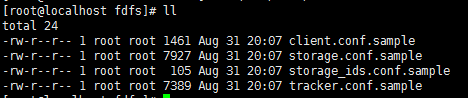
cd fastdfs-5.11

./make.sh

./make.sh install

**2.2.6  配置**

安装成功后进入/etc/fdfs目录：



拷贝一份新的tracker配置文件：

cp tracker.conf.sample tracker.conf

修改tracker.conf

vi tracker.conf

base\_path=/home/yuqing/FastDFS

改为：

base\_path=/opt/fs/data/tracker

**2.2.7  启动**

/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

 设置开机自动启动。

|  |
| --- |
| vim /etc/rc.d/rc.local |

将运行命令行添加进文件：

/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf restart

**2.3   FastDFS--storage安装**

**2.3.1  安装libevent**

同tracker安装

**2.3.2  安装libfastcommon**

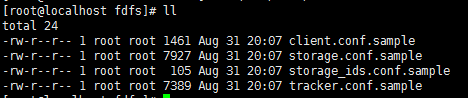
同tracker安装。

**2.3.3  storage编译安装**

同tracker编译安装。

**2.3.4  配置**

安装成功后进入/etc/fdfs目录：



拷贝一份新的storage配置文件：

cp storage.conf.sample storage.conf

修改storage.conf

vi storage.conf

group\_name=group1

base\_path=/home/yuqing/FastDFS

改为：base\_path=/opt/fs/data/store

store\_path0=/home/yuqing/FastDFS

改为：store\_path0=/opt/fs/data/file1

#如果有多个挂载磁盘则定义多个store\_path，如下

#store\_path1=.....

#store\_path2=......  
tracker\_server=192.168.1.222:22122   #配置tracker服务器:IP

#如果有多个则配置多个tracker

tracker\_server=192.168.1.223:22122

**2.3.5  启动**

/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

l 设置开机自动启动。

|  |
| --- |
| vim /etc/rc.d/rc.local |

将运行命令行添加进文件：

/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf restart

**2.4   上传图片测试**

**2.4.1  通过fdfs\_test程序**

FastDFS安装成功可通过/usr/bin/fdfs\_test测试上传、下载等操作。

修改/etc/fdfs/client.conf

base\_path=/opt/fs/data/client

tracker\_server=192.168.1.222:22122

tracker\_server=192.168.1.223:22122

使用格式：

/usr/bin/fdfs\_test 客户端配置文件地址  upload  上传文件

比如将/home下的图片上传到FastDFS中：

/usr/bin/fdfs\_test/etc/fdfs/client.conf upload /home/tomcat.png

http://192.168.101.3/group1/M00/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png就是文件的下载路径。

对应storage服务器上的

/home/fastdfs/fdfs\_storage/data/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png文件。

由于现在还没有和nginx整合无法使用http下载。

**2.5   FastDFS 和nginx整合**

**2.5.1  在tracker上安装nginx**

nginx的安装细节参考nginx文档。

在每个tracker上安装nginx，的主要目的是做负载均衡及实现高可用。如果只有一台tracker服务器可以不配置nginx。

**2.5.2  在Storage上安装nginx**

**2.5.2.1 FastDFS-nginx-module**

将fastdfs-nginx-module-1.20.tar.gz传至/opt/fs/下

cd /opt/fs

tar -zxvf fastdfs-nginx-module-1.20.tar.gz

cd fastdfs-nginx-module-1.20/src

修改config文件 --- （nginx 的版本nginx-1.16.0）

ngx\_module\_incs="/usr/local/include"

改为：

ngx\_module\_incs="/usr/include/fastdfs /usr/include/fastcommon/"

CORE\_INCS="$CORE\_INCS /usr/local/include"

改为：

CORE\_INCS="$CORE\_INCS /usr/include/fastdfs /usr/include/fastcommon/"

将fastdfs-nginx-module-1.20/src /src下的mod\_fastdfs.conf拷贝至/etc/fdfs/下

cp mod\_fastdfs.conf /etc/fdfs/

并修改mod\_fastdfs.conf的内容：

vi /etc/fdfs/mod\_fastdfs.conf

base\_path=/op/fs/data/ng

tracker\_server=192.168.1.223:22122

tracker\_server=192.168.1.222:22122

#tracker\_server=192.168.101.4:22122（多个tracker配置多行）

url\_have\_group\_name=true             #url中包含group名称

store\_path0=/opt/fs/data/file1   #指定文件存储路径

 创建目录

mkdir -p /opt/fs/data/ng

**2.5.2.2 nginx安装**

添加fastdfs-nginx-module模块

安装nginx 的依赖模块：

yum -y install pcre-devel

yum -y install zlib-devel

 ./configure --add-module=/opt/fs/fastdfs-nginx-module-1.20/src/

make

make install

**2.5.2.3 nginx配置文件**

1、在/usr/local/nginx /conf/vhost 文件下

新建一个nginx配置文件fdfs.conf.

内容如下

  server {

listen 33133;

server\_name localhost;

fastcgi\_connect\_timeout 300;

fastcgi\_send\_timeout 300;

fastcgi\_read\_timeout 300;

proxy\_set\_header Cookie $http\_cookie;

# 前端项目资源匹配

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

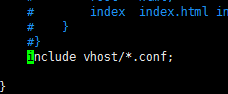
ngx\_fastdfs\_module;

}

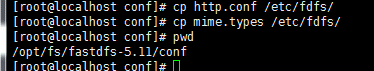
}

 2、编辑 /usr/local/nginx /conf/nginx.conf文件,

在文件最好增加include vhost/\*.conf; 如图：



3、拷贝http.conf mime.types 到 /etc/fdfs



重启nginx

**2.5.3  测试**

通过java客户端上传文件，使用浏览器访问，比如访问上传图片测试的文件：

访问storage：

http://192.168.1.222:33133/group1/M00/00/00/wKhlBVVY2M-AM\_9DAAAT7-0xdqM485\_big.png

**3          附录**

**3.1   tracker.conf**

1 基本配置

disable

#func：配置是否生效

#valu：true、false

disable=false

bind\_addr

#func：绑定IP

#valu：IP地址

bind\_addr=192.168.6.102

port

#func：服务端口

#valu：端口整数值

port=22122

connect\_timeout

#func：连接超时

#valu：秒单位正整数值

connect\_timeout=30

network\_timeout

#func：网络超时

#valu：秒单位正整数值

network\_timeout=60

base\_path

#func：Tracker数据/日志目录地址

#valu：路径

base\_path=/home/michael/fdfs/base4tracker

max\_connections

#func：最大连接数

#valu：正整数值

max\_connections=256

work\_threads

#func：线程数，通常设置CPU数

#valu：正整数值

work\_threads=4

store\_lookup

#func：上传文件的选组方式。

#valu：0、1或2。

# 0：表示轮询

# 1：表示指定组

# 2：表示存储负载均衡（选择剩余空间最大的组）

store\_lookup=2

store\_group

#func：指定上传的组，如果在应用层指定了具体的组，那么这个参数将不会起效。另外如果store\_lookup如果是0或2，则此参数无效。

#valu：group1等

store\_group=group1

store\_server

#func：上传服务器的选择方式。(一个文件被上传后，这个storageserver就相当于这个文件的storage server源，会对同组的storage server推送这个文件达到同步效果)

#valu：0、1或2

# 0: 轮询方式（默认）

# 1: 根据ip 地址进行排序选择第一个服务器（IP地址最小者）

# 2: 根据优先级进行排序（上传优先级由storage server来设置，参数名为upload\_priority），优先级值越小优先级越高。

store\_server=0

store\_path

#func：上传路径的选择方式。storage server可以有多个存放文件的basepath（可以理解为多个磁盘）。

#valu：

# 0: 轮流方式，多个目录依次存放文件

# 2: 存储负载均衡。选择剩余空间最大的目录存放文件（注意：剩余磁盘空间是动态的，因此存储到的目录或磁盘可能也是变化的）

store\_path=0

download\_server

#func：下载服务器的选择方式。

#valu：

# 0：轮询（默认）

# 1：IP最小者

# 2：优先级排序（值最小的，优先级最高。）

download\_server=0

reserved\_storage\_space

#func：保留空间值。如果某个组中的某个服务器的剩余自由空间小于设定值，则文件不会被上传到这个组。

#valu：

# G or g for gigabyte

# M or m for megabyte

# K or k for kilobyte

reserved\_storage\_space=1GB

log\_level

#func：日志级别

#valu：

# emerg for emergency

# alert

# crit for critical

# error

# warn for warning

# notice

# info for information

# debug for debugging

log\_level=info

run\_by\_group / run\_by\_user

#func：指定运行该程序的用户组

#valu：用户组名或空

run\_by\_group=

#func：

#valu：

run\_by\_user=

allow\_hosts

#func：可以连接到tracker server的ip范围。可设定多个值。

#valu

allow\_hosts=

check\_active\_interval

#func：检测 storage server 存活的时间隔，单位为秒。

#     storage server定期向trackerserver 发心跳，

#     如果tracker server在一个check\_active\_interval内还没有收到storageserver的一次心跳，

#     那边将认为该storage server已经下线。所以本参数值必须大于storage server配置的心跳时间间隔。

#     通常配置为storage server心跳时间间隔的2倍或3倍。

check\_active\_interval=120

thread\_stack\_size

#func：设定线程栈的大小。 线程栈越大，一个线程占用的系统资源就越多。

#     如果要启动更多的线程（V1.x对应的参数为max\_connections，V2.0为work\_threads），可以适当降低本参数值。

#valu：如64KB，默认值为64，tracker server线程栈不应小于64KB

thread\_stack\_size=64KB

storage\_ip\_changed\_auto\_adjust

#func：这个参数控制当storage server IP地址改变时，集群是否自动调整。注：只有在storage server进程重启时才完成自动调整。

#valu：true或false

storage\_ip\_changed\_auto\_adjust=true

2 同步

storage\_sync\_file\_max\_delay

#func：同组storage服务器之间同步的最大延迟时间。存储服务器之间同步文件的最大延迟时间，根据实际情况进行调整

#valu：秒为单位，默认值为1天（24\*3600）

#sinc：v2.0

storage\_sync\_file\_max\_delay=86400

storage\_sync\_file\_max\_time

#func：存储服务器同步一个文件需要消耗的最大时间，缺省为300s，即5分钟。

#sinc：v2.0

storage\_sync\_file\_max\_time=300

sync\_log\_buff\_interval

#func：同步或刷新日志信息到硬盘的时间间隔。注意：tracker server 的日志不是时时写硬盘的，而是先写内存。

#valu：以秒为单位

sync\_log\_buff\_interval=10

3 trunk 和 slot

#func：是否使用trunk文件来存储几个小文件

#valu：true或false

#sinc：v3.0

use\_trunk\_file=false

#func：最小slot大小

#valu：<= 4KB，默认为256字节

#sinc：v3.0

slot\_min\_size=256

#func：最大slot大小

#valu：>= slot\_min\_size，当小于这个值的时候就存储到trunkfile中。默认为16MB。

#sinc：v3.0

slot\_max\_size=16MB

#func：trunk file的size

#valu：>= 4MB，默认为64MB

#sinc：v3.0

trunk\_file\_size=64MB

4 HTTP 相关

是否启用 HTTP

#func：HTTP是否生效

#valu：true或false

http.disabled=false

HTTP 服务器端口号

#func：tracker server上的http port

#valu：

#note：只有http.disabled=false时才生效

http.server\_port=7271

检查Storage存活状态的间隔时间（心跳检测）

#func：检查storage http server存活的间隔时间

#valu：单位为秒

#note：只有http.disabled=false时才生效

http.check\_alive\_interval=30

心跳检测使用的协议方式

#func：检查storage http server存活的方式

#valu：

# tcp：连接到storage server的http端口，不进行request和response。

# http：storage check alive url must return http status 200.

#note：只有http.disabled=false时才生效

http.check\_alive\_type=tcp

检查 Storage 状态的 URI

#func：检查storage http server是否alive的uri/url

#note：只有http.disabled=false时才生效

http.check\_alive\_uri=/status.html