FastDFS 分布式存储实战

一、技术选型

普通存储方案：Rsync、DAS(IDE/SATA/SAS/SCSI 等 块)、NAS(NFS、CIFS、SAMBA 等 文件系统)、SAN(FibreChannel，iSCSI，FoE 存储网络 块)，其他 Openfiler、FreeNas(基于 ZFS 快照复制)

由于生产环境中往往由于对存储数据量很大，而 SAN 存储价格又比较昂贵，因此大多会选择分布式存储来解决一下问题：

1. 海量数据存储问题
2. 数据高可用问题(冗余备份)问题
3. 较高的读写性能和负载均衡问题
4. 支持多平台多语言问题
5. 高并发问题

几种常见分布式存储对比：

指标	FastDFS	TFS	MFS	HDFS	Ceph	MogileFS	ClusterFS
适合类型	建议 4KB~500MB	大小文件	小于 64KB 性能不高	大文件	对象、文件、块	海量小图片，效率比 MFS 高	适合大文件存储
文件分布	小文件合并存储，不分片处理	小文件合并，以 Block 组织分片	文件分片	大文件分块分片存储	OSD 一主多从		AFR(raid1) Stripe(raid0) DHT(弹性哈希)
系统性能	1.很高(未使用数据库)，文件访问点对点，不经 tracker 中转 2.无法动态调整磁盘空间负载 3.客户端需记录存储信息	1.主从 NameServer(由 HA agent 实现高可用) 2.DataServer、支持异地灾备 3. 客户端需记录存储信息	Master 相对占内存	AvatarNode/BackupNode 两种 HA 方案	通过 CRUSH 算法分布式元数据，不太成熟基于 Btrfs	高(MySQL 存储文件索引信息，文件同步由 tracker 调度中转)	无元数据服务器
复杂度	简单(tracker 和 storage 两个角色)	部署较复杂	包含 master、metalogger、chunkserver 三个角色	NameNode 和 DataNode 角色	复杂	一般(tracker、storage 和存储文件信息的 MySQL)	简单
是否支持 FUSE	不支持	不支持	支持	支持	支持	需要 mount-mogilefs 支持	支持
是否支持 POSIX	不支持	不支持	支持	支持	支持	不支持	支持
备份机制	分组组内冗余	Block 存储多份，主辅集群灾备	多点备份，动态冗余	多副本	多副本	动态冗余	镜像
通讯协议接口	原生 API、HTTP	原生 API、HTTP	使用 fuse 挂载	原生 API、使用 httpfs 提供 HTTP 接口、Thrift	原生 API	原生 API、HTTP	IP/RDMA
活跃度	国内使用较多	较少	较多	较多	较少	文档较少	使用较多

simlinux.com

文件附加属性	支持		支持	支持		不支持	支持
去重性	依赖 FastDHT	去重接口结合 Tair		结合 MapReduce		不支持	依赖 dedup xlator
语言	C 语言	C++	Perl 语言	Java 语言	C++语言	Perl 语言	C 语言
缺点	1. 服务器配置不统一无法动态调整磁盘空间负载压力 2. 客户端需记录存储信息				1.目前还不太成熟，基于 Btrfs		

注释：
关于 FUSE：https://code.csdn.net/openkb/p-FUSE
关于 GlusterFS：http://blog.csdn.net/liuaigui/article/category/373751

二、FastDFS 相关组件及原理

FastDFS 介绍

FastDFS 是一个 C 语言实现的开源轻量级分布式文件系统，作者余庆(happyfish100)，支持 Linux、FreeBSD、AID 等 Unix 系统，解决了大数据存储和读写负载均衡等问题，适合存储 4KB~500MB 之间的小文件，如图片网站、短视频网站、文档、app 下载站等，UC、京东、支付宝、迅雷、酷狗等都有使用，其中 UC 基于 FastDFS 向用户提供网盘、广告和应用下载的业务的存储服务 FastDFS 与 MogileFS、HDFS、TFS 等都不是系统级的分布式文件系统，而是应用级的分布式文件存储服务.

开源地址：https://github.com/happyfish100/fastdfs/
旧地址： https://code.google.com/p/fastdfs/
相关文档： http://www.csource.org

FastDFS 架构

FastDFS 服务有三个角色：跟踪服务器(tracker server)、存储服务器(storage server)和客户端(client)

tracker server：跟踪服务器，主要做调度工作，起到均衡的作用；负责管理所有的 storage server 和 group，每个 storage 在启动后会连接 Tracker，告知自己所属 group 等信息，并保持周期性心跳，Tracker 根据 storage 心跳信息，建立 group--->[storage server list]的映射表；tracker 管理的元数据很少，会直接存放在内存；tracker 上的元信息都是由 storage 汇报的信息生成的，本身不需要持久化任何数据，tracker 之间是对等关系，因此扩展 tracker 服务非常容易，之间增加 tracker 服务器即可，所有 tracker 都接受 stroage 心跳信息，生成元数据信息来提供读写服务(与其他 Master-Slave 架构的优势是没有单点，tracker 也不会成为瓶颈，最终数据是和一个可用的 Storage Server 进行传输的)

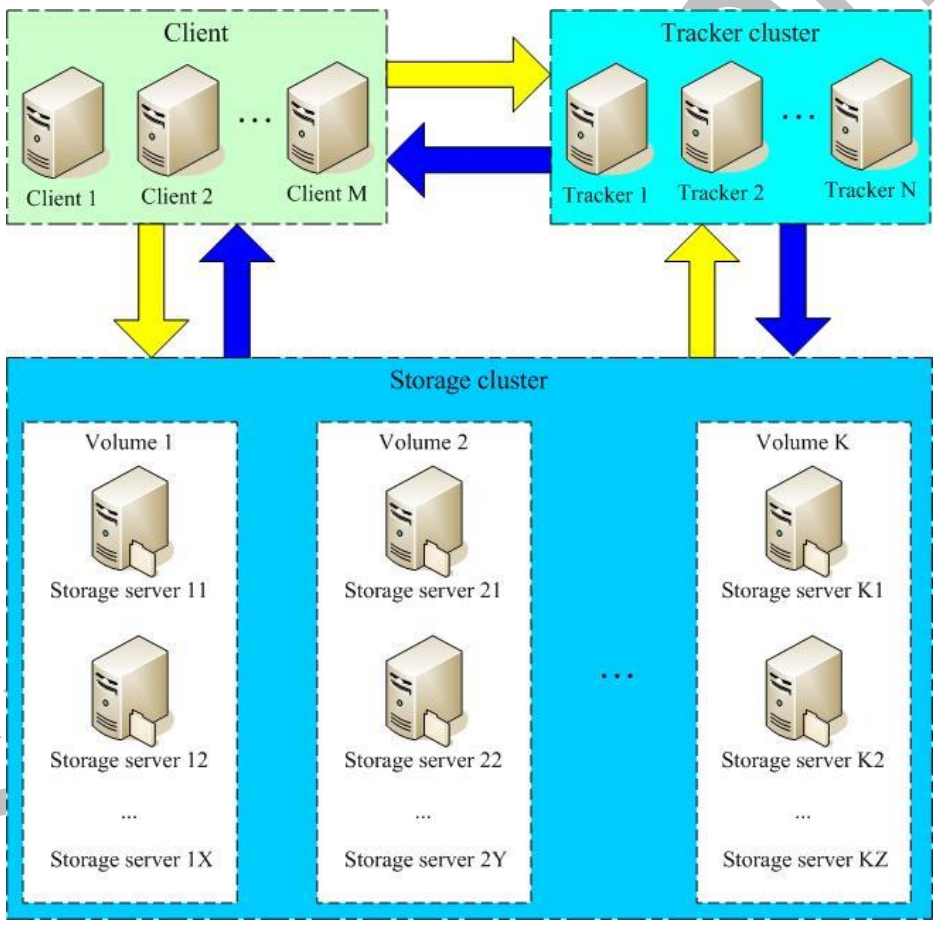
simlinux.com

storage server：存储服务器，主要提供容量和备份服务；以 group 为单位，每个 group 内可以包含多台 storage server，数据互为备份，存储容量空间以 group 内容量最小的 storage 为准；建议 group 内的 storage server 配置相同；以 group 为单位组织存储能够方便的进行应用隔离、负载均衡和副本数定制；**缺点是 group 的容量受单机存储容量的限制，同时 group 内机器坏掉，数据恢复只能依赖 group 内其他机器重新同步(坏盘替换，重新挂载重启 fdfs_storaged 即可)**

多个 group 之间的存储方式有 3 种策略：round robin(轮询)、load balance(选择最大剩余空间的组上传文件)、specify group(指定 group 上传)

group 中 storage 存储依赖本地文件系统,storage 可配置多个数据存储目录,磁盘不做 raid，直接分别挂载到多个目录，将这些目录配置为 storage 的数据目录即可

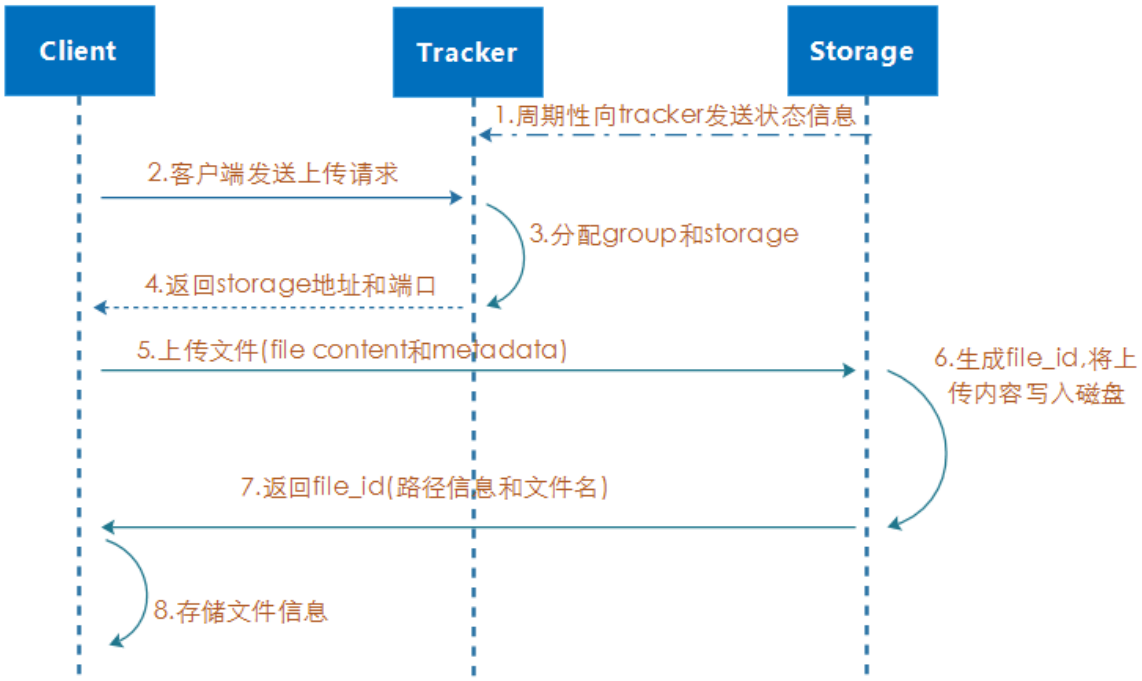
storage 接受写请求时，会根据配置好的规则，选择其中一个存储目录来存储文件；为避免单个目录下的文件过多，storage 第一次启时，会在每个数据存储目录里创建 2 级子目录，每级 256 个，总共 65536 个，新写的文件会以 hash 的方式被路由到其中某个子目录下，然后将文件数据直接作为一个本地文件存储到该目录中



总结：1.高可靠性：无单点故障 2.高吞吐性：只要 Group 足够多，数据流量是足够分散的

FastDFS 工作流程

上传



FastDFS上传文件流程

FastDFS 提供基本的文件访问接口，如 upload、download、append、delete 等

选择 tracker server

集群中 tracker 之间是对等关系，客户端在上传文件时可用任意选择一个 tracker

选择存储 group

当 tracker 接收到 upload file 的请求时，会为该文件分配一个可以存储文件的 group，目前支持选择 group 的规则为:

1. Round robin,所有 group 轮询使用
2. Specified group,指定某个确定的 group
3. Load balance,剩余存储空间较多的 group 优先

选择 storage server

当选定 group 后，tracker 会在 group 内选择一个 storage server 给客户端，目前支持选择 server 的规则为:

1. Round robin,所有 server 轮询使用(默认)
2. 根据 IP 地址进行排序选择第一个服务器(IP 地址最小者)
3. 根据优先级进行排序(上传优先级由 storage server 来设置,参数为 upload_priority)

选择 storage path(磁盘或者挂载点)

当分配好 storage server 后，客户端将向 storage 发送写文件请求,storage 会将文件分配一个数据存储目录，目前支持选择存储路径的规则为:

1. round robin,轮询(默认)
2. load balance,选择使用剩余空间最大的存储路径

选择下载服务器

目前支持的规则为：

1. 轮询方式,可以下载当前文件的任一 storage server
2. 从源 storage server 下载

生成 file_id

选择存储目录后，storage 会生成一个 file_id，采用 Base64 编码，包含字段包括：storage server ip、文件创建时间、文件大小、文件 CRC32 校验码和随机数；每个存储目录下有两个 256*256 个子目录，storage 会按文件 file_id 进行两次 hash，路由到其中一个子目录，，然后将文件已 file_id 为文件名存储到该子目录下，最后生成文件路径：group 名称、虚拟磁盘路径、数据两级目录、file_id

group1 /M00 /02/44/ wKgDrE34E8wAAAAAAAAGkEIYJK42378.sh

其中，**组名：**文件上传后所在的存储组的名称，在文件上传成功后由存储服务器返回，需要客户端自行保存

虚拟磁盘路径：存储服务器配置的虚拟路径，与磁盘选项 store_path*参数对应

数据两级目录：存储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

同步机制

1. 新增 tracker 服务器数据同步问题

   由于 storage server 上配置了所有的 tracker server，storage server 和 tracker server 之间的通信是由 storage server 主动发起的，storage server 为每个 tracker server 启动一个线程进行通信;在通信过程中，若发现该 tracker server 返回的本组 storage server 列表比本机记录少，就会将该 tracker server 上没有的 storage server 同步给该 tracker，这样的机制使得 tracker 之间是对等关系，数据保持一致

2. 新增 storage 服务器数据同步问题

   若新增 storage server 或者其状态发生变化，tracker server 都会将 storage server 列表同步给该组内所有 storage server；以新增 storage server 为例，因为新加入的 storage server 会主动连接 tracker server，tracker server 发现有新的 storage server 加入，就会将该组内所有的 storage server 返回给新加入的 storage server，并重新将该组的 storage server 列表返回给该组内的其他 storage server;

3. 组内 storage 数据同步问题

   组内 storage server 之间是对等的,文件上传、删除等操作可以在组内任意一台 storage server 上进行。文件同步只能在同组内的 storage server 之间进行，采用 push 方式，即源服务器同步到目标服务器

   A. 只在同组内的 storage server 之间进行同步
   B. 源数据才需要同步，备份数据不再同步
   C. 特例：新增 storage server 时，由其中一台将已有所有数据(包括源数据和备份数据)同步到新增服务器

   storage server 的 7 种状态：

   通过命令 fdfs_monitor /etc/fdfs/client.conf 可以查看 ip_addr 选项显示 storage

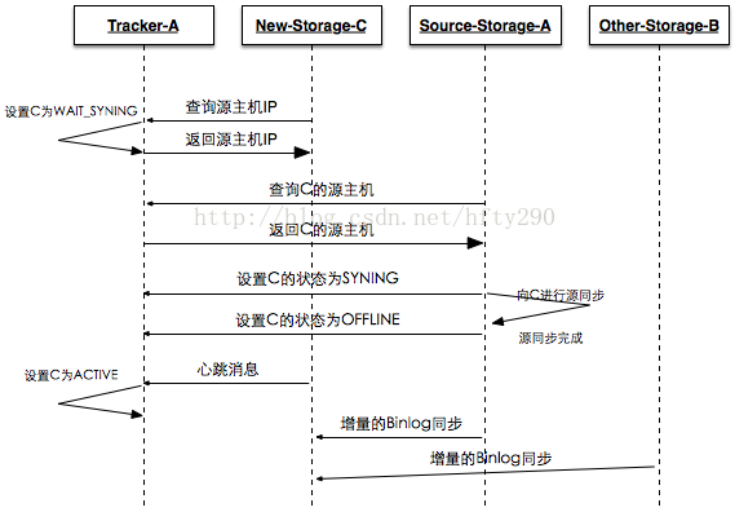
simlinux.com

server 当前状态

INIT	：	初始化，尚未得到同步已有数据的源服务器
WAIT_SYNC	：	等待同步，已得到同步已有数据的源服务器
SYNCING	：	同步中
DELETED	：	已删除，该服务器从本组中摘除
OFFLINE	：	离线
ONLINE	：	在线，尚不能提供服务
ACTIVE	：	在线，可以提供服务

组内增加 storage serverA 状态变化过程：

1. storage server A 主动连接 tracker server，此时 tracker server 将 storage serverA 状态设置为 INIT
2. storage server A 向 tracker server 询问追加同步的源服务器和追加同步截止时间点(当前时间)，若组内只有 storage server A 或者上传文件数为 0，则告诉新机器不需要数据同步，storage server A 状态设置为 ONLINE；若组内没有 active 状态机器，就返回错误给新机器，新机器睡眠尝试；否则 tracker 将其状态设置为 WAIT_SYNC
3. 假如分配了 storage server B 为同步源服务器和截至时间点，那么 storage server B 会将截至时间点之前的所有数据同步给 storage server A，并请求 tracker 设置 storage server A 状态为 SYNCING;到了截至时间点后,storage server B 向 storage server A 的同步将由追加同步切换为正常 binlog 增量同步,当取不到更多的 binlog 时，请求 tracker 将 storage server A 设置为 OFFLINE 状态，此时源同步完成
4. storage server B 向 storage server A 同步完所有数据，暂时没有数据要同步时，storage server B 请求 tracker server 将 storage server A 的状态设置为 ONLINE
5. 当 storage server A 向 tracker server 发起心跳时，tracker sercer 将其状态更改为 ACTIVE，之后就是增量同步(binlog)

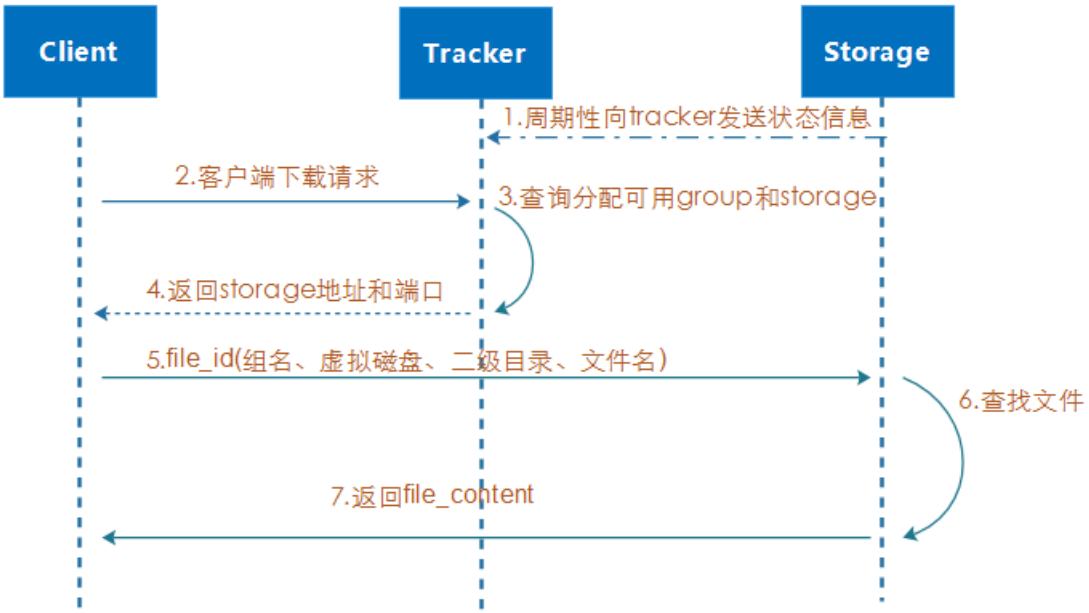


注释：1.整个源同步过程是源机器启动一个同步线程，将数据 push 到新机器，最大达到一个磁盘的 IO，不能并发

2.由于源同步截止条件是取不到 binlog，系统繁忙，不断有新数据写入的情

况，将会导致一直无法完成源同步过程

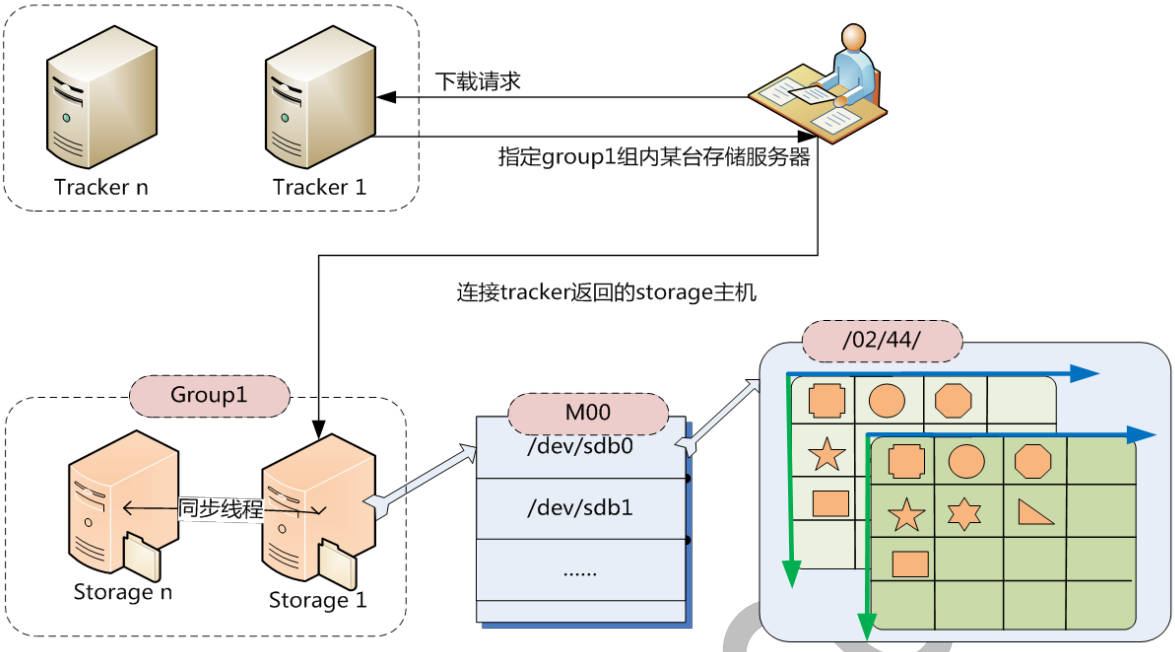
下载



FastDFS下载文件流程

client 发送下载请求给某个 tracker，必须带上文件名信息，tracker 从文件名中解析出文件的 group、大小、创建时间等信息，然后为该请求选择一个 storage 用于读请求；由于 group 内的文件同步在后台是异步进行的，可能出现文件没有同步到其他 storage server 上或者延迟的问题，后面我们在使用 nginx_fastdfs_module 模块可以很好解决这一问题

Simlinux



文件合并原理

小文件合并存储主要解决的问题：

1. 本地文件系统 inode 数量有限，存储小文件的数量受到限制
2. 多级目录+目录里很多文件，导致访问文件的开销很大(可能导致很多次 IO)
3. 按小文件存储，备份和恢复效率低

海量小文件存储问题请参考：http://blog.csdn.net/liuaigui/article/details/9981135
http://blog.csdn.net/kidd_3/article/details/6909097

FastDFS 提供合并存储功能，默认创建的大文件为 64MB，然后在该大文件中存储很多小文件；大文件中容纳一个小文件的空间称作一个 Slot，规定 Slot 最小值为 256 字节，最大为 16MB，即小于 256 字节的文件也要占用 256 字节，超过 16MB 的文件独立存储；

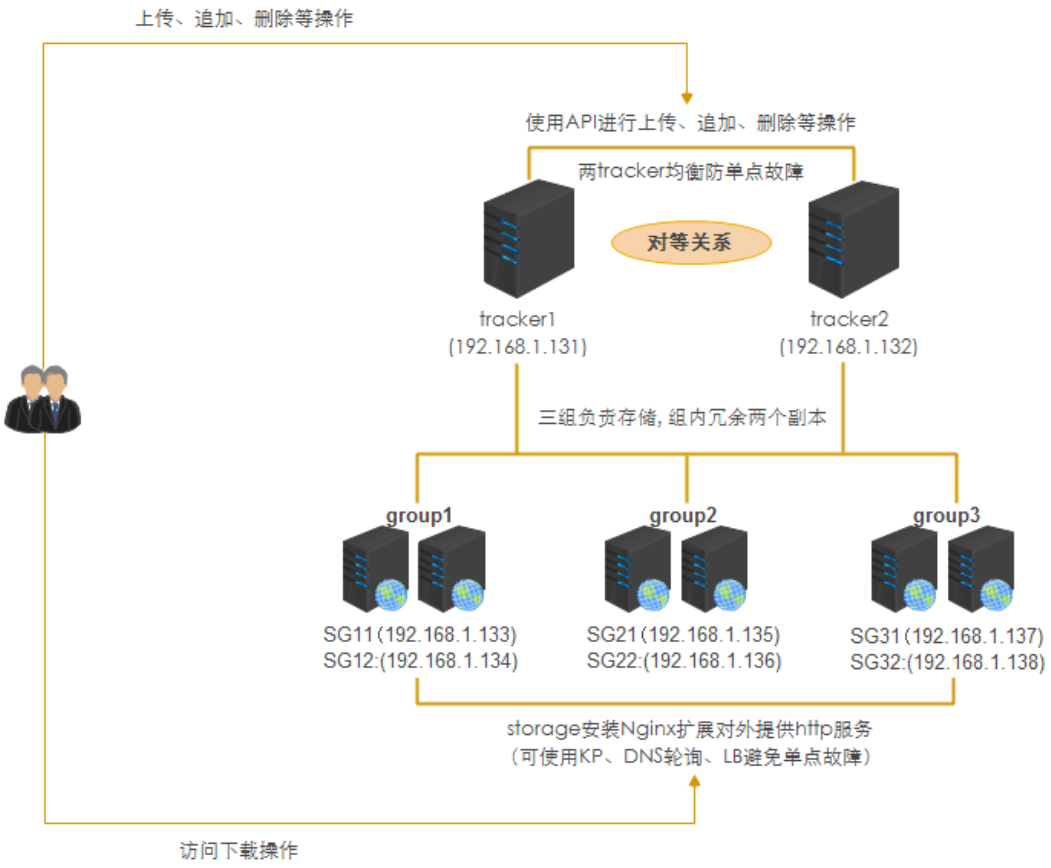
为了支持文件合并机制,FastDFS 生成的文件 file_id 需要额外增加 16 个字节;每个 trunk file 由一个 id 唯一标识，trunk file 由 group 内的 trunk server 负责创建(trunk server 是 tracker 选出来的)，并同步到 group 内其他的 storage，文件存储合并存储到 trunk file 后，根据其文件偏移量就能从 trunk file 中读取文件

FastDFS 文件合并原理细节请参考：
http://blog.csdn.net/hfty290/article/details/42026215#comments

三、实验环境说明

操作系统： Centos 6.5 x64
FastDFS 相关版本： fastdfs-5.05 fastdfs-nginx-module-v1.16 libfastcommon-v1.0.7

web 服务器软件： nginx-1.7.9
角色分配： 2 个 tracker，地址分别为:192.168.1.131,192.168.1.132
3 个 group：SG12 主机名表示 group1 的第二台主机 Storage2
G1:SG11(192.168.1.133),SG12(192.168.1.134)
G2:SG21(192.168.1.135),SG22(192.168.1.136)
G3:SG31(192.168.1.137),SG32(192.168.1.138)
目录规划：
/dev/sda2 /data1 日志元数据目录
/dev/sdb1 /data2 数据目录
注释：本实验简单期间按照 2 个 tracker 和 2 组 group 来做



FastDFS分布式存储架构方案

四、FastDFS 部署

初始化系统

系统最小化安装
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
chkconfig iptables off

echo “*/10 * * * * /usr/sbin/ntpdate asia.pool.ntp.org ”>>/etc/crontab
yum –y install wget
rm -rf /etc/yum.repos.d/*
wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-6.repo
yum makecache
yum -y install gettext gettext-devel libXft libXft-devel libXpm libXpm-devel automake autoconf libXtst-devel gtk+-devel gcc gcc-c++zlib-devel libpng-devel gtk2-devel glib-devel pcre* gcc git

安装 libfastcommon 和 fastdfs

所有机器均安装

cd /root/fastdfs/
git clone https://github.com/happyfish100/libfastcommon.git
cd libfastcommon
./make.sh
./make.sh install

wget https://github.com/happyfish100/fastdfs/archive/V5.05.tar.gz
tar xf V5.05.tar.gz
cd fastdfs-5.05
sed -i 's#/usr/local/bin/#/usr/bin/#g' init.d/fdfs_storaged
sed -i 's#/usr/local/bin/#/usr/bin/#g' init.d/fdfs_trackerd
./make.sh
./make.sh install

1. **配置两台 tracker**

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

```
disabled=false
bind_addr=
port=22122
connect_timeout=30
network_timeout=60
base_path=/data1/
max_connections=256
accept_threads=1
work_threads=4
store_lookup=2
store_group=group2
store_server=0
```

simlinux.com

```
store_path=0
download_server=0
reserved_storage_space = 10%
log_level=info
run_by_group=
run_by_user=
allow_hosts=*
sync_log_buff_interval = 10
check_active_interval = 120
thread_stack_size = 64KB
storage_ip_changed_auto_adjust = true
storage_sync_file_max_delay = 86400
storage_sync_file_max_time = 300
use_trunk_file = false
slot_min_size = 256
slot_max_size = 16MB
trunk_file_size = 64MB
trunk_create_file_advance = false
trunk_create_file_time_base = 02:00
trunk_create_file_interval = 86400
trunk_create_file_space_threshold = 20G
trunk_init_check_occupying = false
trunk_init_reload_from_binlog = false
trunk_compress_binlog_min_interval = 0
use_storage_id = false
storage_ids_filename = storage_ids.conf
id_type_in_filename = ip
store_slave_file_use_link = false
rotate_error_log = false
error_log_rotate_time=00:00
rotate_error_log_size = 0
log_file_keep_days = 0
use_connection_pool = false
connection_pool_max_idle_time = 3600
http.server_port=8080
http.check_alive_interval=30
http.check_alive_type=tcp
http.check_alive_uri=/status.html
```

启动 tracker 服务：

```
service fdfs_trackerd start
chkconfig fdfs_trackerd on
```

2. 配置 G1 和 G2

simlinux.com

组 G1 SG11(192.168.1.133)，SG12(192.168.1.134)配置相同

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

vim /etc/fdfs/storage.conf

```
disabled=false
group_name=G1
bind_addr=
client_bind=true
port=23000
connect_timeout=30
network_timeout=60
heart_beat_interval=30
stat_report_interval=60
base_path=/data1/
max_connections=256
buff_size = 256KB
accept_threads=1
work_threads=4
disk_rw_separated = true
disk_reader_threads = 1
disk_writer_threads = 1
sync_wait_msec=50
sync_interval=0
sync_start_time=00:00
sync_end_time=23:59
write_mark_file_freq=500
store_path_count=1
store_path0=/data2/
subdir_count_per_path=256
tracker_server=192.168.1.131:22122
tracker_server=192.168.1.132:22122
log_level=info
run_by_group=
run_by_user=
allow_hosts=*
file_distribute_path_mode=0
file_distribute_rotate_count=100
fsync_after_written_bytes=0
sync_log_buff_interval=10
sync_binlog_buff_interval=10
sync_stat_file_interval=300
thread_stack_size=512KB
upload_priority=10
if_alias_prefix=
check_file_duplicate=0
```

Simlinux.com

file_signature_method=hash
key_namespace=FastDFS
keep_alive=0
use_access_log = false
rotate_access_log = false
access_log_rotate_time=00:00
rotate_error_log = false
error_log_rotate_time=00:00
rotate_access_log_size = 0
rotate_error_log_size = 0
log_file_keep_days = 0
file_sync_skip_invalid_record=false
use_connection_pool = false
connection_pool_max_idle_time = 3600
http.domain_name=
http.server_port=8888

组 G2 SG11(192.168.1.135)，SG12(192.168.1.136)配置相同

disabled=false
group_name=G2
bind_addr=
client_bind=true
port=23000
connect_timeout=30
network_timeout=60
heart_beat_interval=30
stat_report_interval=60
base_path=/data1/
max_connections=256
buff_size = 256KB
accept_threads=1
work_threads=4
disk_rw_separated = true
disk_reader_threads = 1
disk_writer_threads = 1
sync_wait_msec=50
sync_interval=0
sync_start_time=00:00
sync_end_time=23:59
write_mark_file_freq=500
store_path_count=1
store_path0=/data2/
subdir_count_per_path=256
tracker_server=192.168.1.131:22122

Simlinux.com

```
tracker_server=192.168.1.132:22122
log_level=info
run_by_group=
run_by_user=
allow_hosts=*
file_distribute_path_mode=0
file_distribute_rotate_count=100
fsync_after_written_bytes=0
sync_log_buff_interval=10
sync_binlog_buff_interval=10
sync_stat_file_interval=300
thread_stack_size=512KB
upload_priority=10
if_alias_prefix=
check_file_duplicate=0
file_signature_method=hash
key_namespace=FastDFS
keep_alive=0
use_access_log = false
rotate_access_log = false
access_log_rotate_time=00:00
rotate_error_log = false
error_log_rotate_time=00:00
rotate_access_log_size = 0
rotate_error_log_size = 0
log_file_keep_days = 0
file_sync_skip_invalid_record=false
use_connection_pool = false
connection_pool_max_idle_time = 3600
http.domain_name=
http.server_port=8888
```

启动 storage 服务：

```
service    fdfs_storaged start
chkconfig fdfs_storaged on
```

storage server 安装 nginx

```
wget http://nginx.org/download/nginx-1.7.9.tar.gz
git clone https://github.com/happyfish100/fastdfs-nginx-module.git
tar xf nginx-1.7.9.tar.gz
cd nginx-1.7.9
./configure --add-module=../fastdfs-nginx-module/src/ \
```

simlinux.com

```
--prefix=/usr/local/nginx --user=nobody --group=nobody \
--with-http_gzip_static_module    --with-http_gunzip_module
make -j8
make install
```

nginx 启动脚本

```
#!/bin/bash
#
# nginx - this script starts and stops the nginx daemon
#
# chkconfig: - 85 15
# description: Nginx is an HTTP(S) server, HTTP(S) reverse
# proxy and IMAP/POP3 proxy server
# processname: nginx
# config: /usr/local/nginx/conf/nginx.conf

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ "$NETWORKING" = "no" ] && exit 0

NGINX_HOME="/usr/local/nginx/"
nginx=$NGINX_HOME"sbin/nginx"
prog=$(basename $nginx)

NGINX_CONF_FILE=$NGINX_HOME"conf/nginx.conf"

[ -f /etc/sysconfig/nginx ] && /etc/sysconfig/nginx

lockfile=/var/lock/subsys/nginx

start() {
    [ -x $nginx ] || exit 5
    [ -f $NGINX_CONF_FILE ] || exit 6
    echo -n $"Starting $prog: "
    daemon $nginx -c $NGINX_CONF_FILE
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
```

simlinux.com

```
}

stop() {
        echo -n $"Stopping $prog: "
        killproc $prog -QUIT
        retval=$?
        echo
        [ $retval -eq 0 ] && rm -f $lockfile
        return $retval
        killall -9 nginx
}

restart() {
        configtest || return $?
        stop
        sleep 1
        start
}

reload() {
        configtest || return $?
        echo -n $"Reloading $prog: "
        killproc $nginx -HUP
        RETVAL=$?
        echo
}

force_reload() {
        restart
}

configtest() {
        $nginx -t -c $NGINX_CONF_FILE
}

rh_status() {
        status $prog
}

rh_status_q() {
        rh_status >/dev/null 2>&1
}

case "$1" in
```

Simlinux.com

```
start)
        rh_status_q && exit 0
        $1
;;
stop)
        rh_status_q || exit 0
        $1
;;
restart|configtest)
        $1
;;
reload)
        rh_status_q || exit 7
        $1
;;
force-reload)
        force_reload
;;
status)
        rh_status
;;
condrestart|try-restart)
        rh_status_q || exit 0
;;
*)

        echo                                $"Usage:                                $0
{start|stop|status|restart|condrestart|try-restart|reload|force-reload|configtest}"
        exit 2
        esac
```

配置 nginx

```
cp fastdfs-nginx-module/src/mod_fastdfs.conf    /etc/fdfs/
cp fastdfs-5.05/conf/{anti-steal.jpg,http.conf,mime.types}    /etc/fdfs/
touch /var/log/mod_fastdfs.log
chown nobody.nobody /var/log/mod_fastdfs.log

server {
            listen 80;
            server_name localhost;
            location /G1/M00{
            root /data2/;
            ngx_fastdfs_module;
             }
```

simlinux.com

}

配置 mod_fastdfs.conf

G1 组配置：

vim /etc/fdfs/mod_fastdfs.conf

```
connect_timeout=2
network_timeout=30
base_path=/tmp
load_fdfs_parameters_from_tracker=true
storage_sync_file_max_delay = 86400
use_storage_id = false
storage_ids_filename = storage_ids.conf
tracker_server=192.168.1.131:22122
tracker_server=192.168.1.132:22122
storage_server_port=23000
group_name=G1
url_have_group_name = true
store_path_count=1
store_path0=/data2/
log_level=info
log_filename=/var/log/mod_fastdfs.log
response_mode=proxy
if_alias_prefix=
#include http.conf
flv_support = true
flv_extension = flv
group_count = 2
[group1]
group_name=G1
storage_server_port=23000
store_path_count=1
store_path0=/data2/
[group2]
group_name=G2
storage_server_port=23000
store_path_count=1
store_path0=/data2/
```

G2 组配置：

vim /etc/fdfs/mod_fastdfs.conf

```
connect_timeout=2
network_timeout=30
```

Simlinux.com

```
base_path=/tmp
load_fdfs_parameters_from_tracker=true
storage_sync_file_max_delay = 86400
use_storage_id = false
storage_ids_filename = storage_ids.conf
tracker_server=192.168.1.131:22122
tracker_server=192.168.1.132:22122
storage_server_port=23000
group_name=G2
url_have_group_name = true
store_path_count=1
store_path0=/data2/
log_level=info
log_filename=/var/log/mod_fastdfs.log
response_mode=proxy
if_alias_prefix=
#include http.conf
flv_support = true
flv_extension = flv
group_count = 2
[group1]
group_name=G1
storage_server_port=23000
store_path_count=1
store_path0=/data2/
[group2]
group_name=G2
storage_server_port=23000
store_path_count=1
store_path0=/data2/
```

service nginx reload

chkconfig nginx on

注释：response_mode 模式有两种

访问效果：

simlinux.com

配置下载网关

```
http {
    include       mime.types;
    default_type  application/octet-stream;
#设置主机名哈希表的表项长度默认 32|64|128
  server_names_hash_bucket_size 128;
#设置请求缓冲区大小，默认使用 client_header_buffer_size 来读取 header 值，
#如果 header 过大，会使用 large_client_header_buffers 来读取
  client_header_buffer_size 32k;
  large_client_header_buffers 4 32k;
#定义最大允许上传的文件大小
  client_max_body_size 300m;
#开启高效文件传输模式，sendfile 指令指定 nginx 是否调用 sendfile 函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘 IO 重负载应用，可设置为 off，以平衡磁盘与网络 I/O 处理速度，降低系统的负载。注意：如果图片显示不正常把这个改成 off
  sendfile on;
# 防止网络堵塞
  tcp_nopush on;
#长连接超时时间
  keepalive_timeout 120;

#gzip 模块设置
  gzip on; #开启 gzip 压缩输出
  gzip_min_length 1k; #最小压缩文件大小
  gzip_buffers 4 16k; #压缩缓冲区
  gzip_http_version 1.0; #压缩版本（默认 1.1，前端如果是 squid2.5 请使用 1.0）
  gzip_comp_level 2; #压缩等级(1~9)
  gzip_types text/plain application/x-javascript text/css application/xml;
#压缩类型，默认就已经包含 text/html
```

simlinux.com

```
        gzip_vary on; #根据 HTTP 头来判断是否需要压缩
        gzip_disable “MSIE[1-6].”#IE6 以下对 gzip 的支持不好，在 IE6 下禁用 gzip

        proxy_redirect off;
      #当后端单台 web 服务器上也配置有多个虚拟主机时，需要使用该 Header 来区分反向代理哪个主机名
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
      #后端的 Web 服务器可以通过 X-Real-IP/X-Forwarded-For 获取用户真实 IP
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      #后端服务器连接的超时时间(发起握手等候响应的超时时间)
        proxy_connect_timeout 90;
      #后端服务器数据回传时间(规定此时间内后端服务器必须传完所有数据)
        proxy_send_timeout 90;
        #等待后端服务器处理请求的时间
        proxy_read_timeout 90;
        #nginx 从被代理的服务器读取响应时，使用该缓冲区保存响应的开始部分，该值默认等于 proxy_buffers 设置的一块缓冲区大小，但可以被设置更小
        proxy_buffer_size 16k;
        #为每个连接设置缓冲区的数量为 number，每块缓冲区的大小为 size，用于保存从被
        代理服务器读取的响应
        proxy_buffers 4 64k;
        proxy_busy_buffers_size 128k;
        #在开启缓冲后端服务器响应到临时文件的功能后，设置 nginx 每次写数据到临时文件的 size(大小)限制
        proxy_temp_file_write_size 128k;

        log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

        access_log /usr/local/nginx/access.log main;

      #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;

            upstream group1 {
            server 192.168.1.133:80;
            server 192.168.1.134:80;
                }
            upstream group2 {
            server 192.168.1.135:80;
```

```
        server 192.168.1.136:80;
    }
    server {
        listen 80;
        server_name 192.168.1.138;

      location /G1/M00 {       #设置 group1 的负载均衡参数
    #如果后端的服务器返回 502、504、执行超时等错误，自动将请求转发到 upstream 负载均衡池中的另一台服务器，实现故障转移
      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://group1;
      expires 30d;
      }

      location ~* /G2/(M00|M01) { #设置 group2 的负载均衡参数
      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://group2;
      expires 30d;
      }
    }
```

```
~      #波浪线表示执行一个正则匹配，区分大小写
~*     #表示执行一个正则匹配，不区分大小写
^~     #^~表示普通字符匹配，如果该选项匹配，只匹配该选项，不匹配别的选项，一般用来匹配目录
=      #进行普通字符精确匹配
@      #"@" 定义一个命名的 location，使用在内部定向时，
       例如 error_page, try_files
```

参数可参看： http://nginx.org/cn/docs/
http://wiki.nginx.org/NginxHttpLogModule

注： 前端下载网关可以双机 Nginx 结合 Keepalived 实现 HA

tracker 和 storage 目录结构

FastDFS 服务器端运行时目录结构：

${base_path}
|__data：存放状态文件
|__logs：存放日志文件
其中，${base_path}由配置文件中的参数“base_path”设定

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）

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

simlinux.com

```
|      |__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：

(1) storage_join_time：本 storage server 创建时间；

(2) 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）；

(3) 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；

(4) 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：

(1) total_upload_count：上传文件次数

(2) success_upload_count：成功上传文件次数

(3) total_set_meta_count：更改 meta data 次数

(4) success_set_meta_count：成功更改 meta data 次数

(5) total_delete_count：删除文件次数

(6) success_delete_count：成功删除文件次数

(7) total_download_count：下载文件次数

(8) success_download_count：成功下载文件次数

(9) total_get_meta_count：获取 meta data 次数

(10) success_get_meta_count：成功获取 meta data 次数

(11) last_source_update：最近一次源头更新时间（更新操作来自客户端）

(12) 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

C 表示源创建、c 表示副本创建

A 表示源追加、a 表示副本追加

D 表示源删除、d 表示副本删除

T 表示源 Truncate、t 表示副本 Truncate

(3)filename：操作（更新）的文件名，包括相对路径，如：

simlinux.com

5A/3D/FE_93_SJZ7pAAAO_BXYD.S

${ip_addr}_${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：192.168.1.136_23000.mark。文件格式为 ini 配置文件方式，各个参数如下：

(1) binlog_index：已处理（同步）到的 binlog 索引号
(2) binlog_offset：已处理（同步）到的 binlog 文件偏移量（字节数）
(3) 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
(4) 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
(5) until_timestamp：同步已有数据截至时间点（UNIX 时间戳）
(6) scan_row_count：已扫描的 binlog 记录数
(7) sync_row_count：已同步的 binlog 记录数

测试

客户端配置/etc/fdfs/client.conf

```
connect_timeout=30
network_timeout=60
base_path=/tmp
tracker_server=192.168.1.131:22122
tracker_server=192.168.1.132:22122
log_level=info
use_connection_pool = false
connection_pool_max_idle_time = 3600
load_fdfs_parameters_from_tracker=false
use_storage_id = false
storage_ids_filename = storage_ids.conf
http.tracker_server_port=80
```

测试上传文件

```
[root@SG21 ~]# fdfs_upload_file /etc/fdfs/client.conf install.log
G1/M00/00/00/wKgBhVTJ5eOAFjBAAAAfO4TIxXY080.log
```

下载文件

```
fdfs_download_file /etc/fdfs/client.conf
G1/M00/00/00/wKgBhVTJ5eOAFjBAAAAfO4TIxXY080.log
```

查看文件属性

```
[root@SG21~]#fdfs_file_info /etc/fdfs/client.conf
G1/M00/00/00/wKgBhVTJ5eOAFjBAAAAfO4TIxXY080.log
source storage id: 0
source ip address: 192.168.1.133
file create timestamp: 2015-01-29 15:48:51
file size: 7995
file crc32: 2227750262 (0x84C8C576)
```

simlinux.com

以追加方式上传文件测试

```
[root@SG12 ~]# cat test1.log
test1
[root@SG12 ~]# cat test2.log
test2
[root@SG12 ~]# fdfs_upload_appender    /etc/fdfs/client.conf test1.log
G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
[root@SG12 ~]# fdfs_download_file /etc/fdfs/client.conf    \
G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
[root@SG12 ~]# cat wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
test1
[root@SG12 ~]# fdfs_append_file /etc/fdfs/client.conf \
G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log test2.log
[root@SG12 ~]# fdfs_download_file /etc/fdfs/client.conf    \
G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
[root@SG12 ~]# cat wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
test1
test2
```

删除文件

```
[root@SG12 ~]# fdfs_delete_file /etc/fdfs/client.conf    \
G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
```

查看集群状态

```
[root@SG12 ~]# fdfs_monitor    /etc/fdfs/client.conf
```

将 **OFFILINE** 的 **storage server** 剔除组

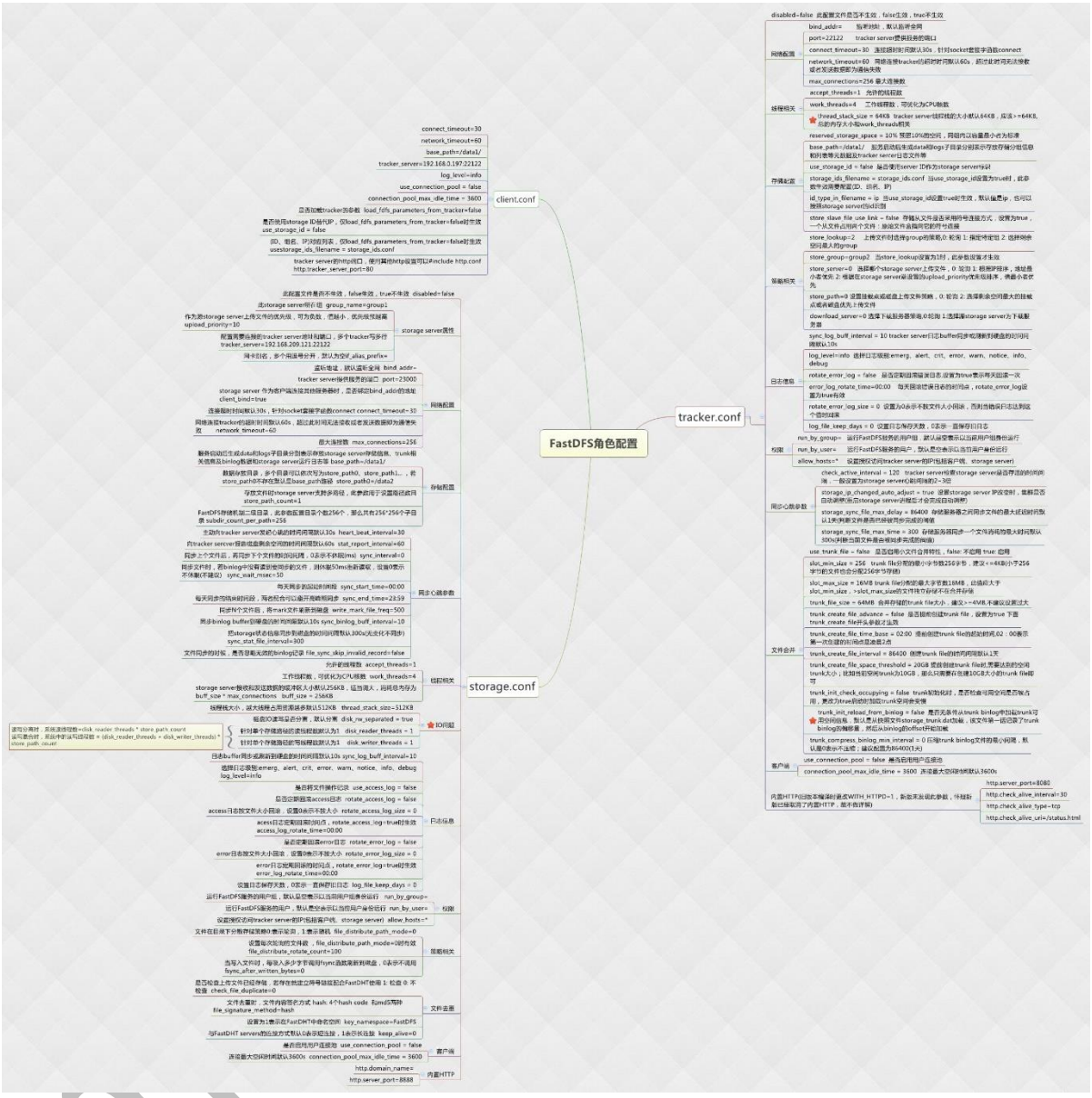
```
[root@SG12 ~]# fdfs_monitor    /etc/fdfs/client.conf    delete    G2 192.168.1.136
```

查看文件校验码

```
[root@SG12 ~]#fdfs_crc32 G1/M00/00/00/wKgBhVTJ8j2EXNDGAAAAAD2A2uQ093.log
1272276929
```

详细参数请参考：http://www.simlinux.com/archives/484.html

simlinux.com



五、高级功能

防盗链

方法一：使用 fast_nginx_module 扩展的动态生成 token 方式

开启防盗链功能

```
http.default_content_type = application/octet-stream
http.mime_types_filename=mime.types
#开启 token 防盗链功能
http.anti_steal.check_token=true
#token 过期时间
http.anti_steal.token_ttl=900
#密钥
```

http.anti_steal.secret_key=FastDFS1234567890
#token 过期后返回的内容
http.anti_steal.token_check_fail=/etc/fdfs/anti-steal.jpg
示例说明：

php 中，通过 fastdfs_http_gen_token()函数生成$token,例如：
$ts=time();//当前时间戳
$token=fastdfs_http_gen_token('M00/00/00/Ss_0rVGSABmAEZ1QAAC6WE5-JkI695.zip',$ts);
然后 url 形如：
http://localhost/M00/00/00/Ss_0rVGSABmAEZ1QAAC6WE5-JkI695.zip?token=8de0e6554be69d0b9385faad654c4364&ts=1368589809
这样服务端就可以自动根据 token ，ts ,以及设置的密钥来验证合法性。密钥过期时间在 http.conf 里设置

方法二：在 nginx 配置防盗链功能

A.使用 referer 对特定文件格式防盗链

```
location ~* \.(gif|jpg|png|swf|flv)$ {
  valid_referers none blocked server_names *.simlinux.com linuxhonker.com;
  if ($invalid_referer) {
  rewrite ^/    http://www.simlinux.com/fangdao.html;
  #return 403;
    }
}
```

注：第二行对来路域名进行判断，不是来路就 rewrite 或者返回 403 代码

B.针对图片目录防盗链

```
location /images/ {
  alias /data/images/;
  valid_referers none blocked server_names *.simlinux.com linuxhonker.com;
  if ($invalid_referer) {
     return 403;
     }
}
```

详细参考：http://nginx.org/cn/docs/http/ngx_http_referer_module.html

C.使用第三方模块 ngx_http_accesskey_module 实现 Nginx 防盗链

1.下载 NginxHttpAccessKeyModule 模块文件：

http://wiki.nginx.org/images/5/51/Nginx-accesskey-2.0.3.tar.gz

2.解压此文件后，编辑 nginx-accesskey-2.0.3 下的 config 文件，替换其中的”$HTTP_ACCESSKEY_MODULE”为”ngx_http_accesskey_module“

simlinux.com

然后编译 nginx

```
./configure --add-module=../nginx-accesskey
make && make install
```

3.修改 nginx 的 conf 文件，添加以下几行：

```
location /download {
    accesskey on;
    accesskey_hashmethod md5;
    accesskey_arg "key";
    accesskey_signature "mypass$remote_addr";
 }
```

accesskey 为模块开关；

accesskey_hashmethod 为加密方式 MD5 或者 SHA-1；

accesskey_arg 为 url 中的关键字参数；

accesskey_signature 为加密值，此处为 mypass 和访问 IP 构成的字符串。

访问测试脚本 download.php：

```
<?
$ipkey= md5("mypass".$_SERVER['REMOTE_ADDR']);
$output_add_key="<a
href=http://www.simlinux.com/download/G3200507120520LM.rar?key=".$ipkey.">download_ad
d_key</a><br />";
$output_org_url="<a
href=http://www.simlinux.com/download/G3200507120520LM.rar>download_org_path</a><br
/>";
echo $output_add_key;
echo $output_org_url;
?>
```

访问第一个 download_add_key 链接可以正常下载，第二个链接 download_org_path 会返回 403 Forbidden 错误。

详细请参考：http://wiki.nginx.org/NginxHttpAccessKeyModule

在线扩容

增加 group

目标：在线扩容组 G3
G3:SG31(192.168.1.137),SG32(192.168.1.138)，数据目录与 G1、G2 保持一致
操作步骤：
初始化系统→ 双机安装 libfastcommon 和 fasdfs → 双机安装 nginx 并配置(参考前面章节) →配置 strorage.conf 文件(tracker IP、所属组 G3、数据目录等) →启动 fdfs_storaged 和 nginx 服务→观察日志和集群中新机的状态(ACTIVE 表示可以正常提供服务)

组内增加 storage server

目标：给 G2 组增加一个副本，数据目录与同组一致(目录也可以不一致)
操作步骤：
初始化系统→ 安装 libfastcommon 和 fasdfs →安装 nginx 并配置(参考前面章节) →配置 strorage.conf 文件(tracker IP、所属组、数据目录等) →启动 fdfs_storaged 和 nginx 服务 →观察日志和集群中新机的状态(ACTIVE 表示可以正常提供服务)

故障磁盘移除

dell 用 check_openmanage HP 用 check_hpasm 也可以使用 smartmontools 去检测

文件去重存储

FastDHT 是一个基于键值对的高效分布式 hash 系统，可以存储文件名和 file_id 的映射表、session 数据、用户相关数据等等，底层采用 Berkeley DB 做数据库来持久存储数据(也可以使用 SSDB)，数据同步方式是使用自己的 binlog 复制方式

FastDHT 集群由一个或者多个组 group 组成，每个组可以有一台或者多台组成，同组服务器上存储的数据是相同的,数据同步只在同组的服务器之间进行；组内各个服务是对等的，对数据进行存取时，可以根据 key 的 hash 值来决定使用哪台机器(具体参考

https://github.com/happyfish100/fastdht)，数据同步采用推（Push）的方式，由源服务器主动将数据同步到本组的其他服务器

可以独立部署 FastDHT 集群，也可以整合到 storage server 上面，下面以整合到 G1 组为例：

1. **G1:SG11(192.168.1.133),SG12(192.168.1.134)双机安装 berkeley-db 和 fastdht**

simlinux.com

```
wget http://download.oracle.com/berkeley-db/db-6.1.19.tar.gz
tar xf db-6.1.19.tar.gz
cd db-6.1.19/build_unix
../dist/configure --prefix=/usr/local/db6.1.19
make
make install
git clone https://github.com/happyfish100/fastdht
cd fastdht
./make.sh C_INCLUDE_PATH=/usr/local/db6.1.19/include/ \
  LIBRARY_PATH=/usr/local/db6.1.19/lib/
./make.sh install
echo "/usr/local/db6.1.19/lib">>/etc/ld.so.conf.d/fastdht.conf
ldconfig
cp init.d/fdhtd    /etc/rc.d/init.d/
```

2. 配置 FastDHT

```
vim /etc/fdht/fdht_servers.conf
    group_count = 1
    group0 = 192.168.1.133:11411
    group0 = 192.168.1.134:11411

vim /etc/fdht/fdhtd.conf
    disabled=false
    bind_addr=
    port=11411
    connect_timeout=5
    network_timeout=60
    base_path=/tmp/fdht
    max_connections=256
    accept_threads=1
    max_threads=8
    max_pkg_size=64KB
    min_buff_size=64KB
    store_type = BDB
    cache_size = 64MB
    db_prefix = db
    page_size = 4096
    db_type = btree
    mpool_init_capacity = 10000
    mpool_load_factor = 0.75
    mpool_htable_lock_count = 1361
    mpool_clear_min_interval = 30
```

simlinux.com

```
log_level=info
run_by_group=
run_by_user=
allow_hosts=*
sync_log_buff_interval=10
sync_db_time_base=00:00
sync_db_interval=86400
write_to_binlog=1
sync_binlog_buff_interval=60
need_clear_expired_data=true
clear_expired_time_base=04:00
clear_expired_interval=86400
db_dead_lock_detect_interval=1000
compress_binlog_time_base=02:00
compress_binlog_interval=86400
sync_stat_file_interval=300
write_mark_file_freq=5000
thread_stack_size=1MB
if_alias_prefix=
store_sub_keys=false
#include fdht_servers.conf
```

storage server 配置增加以下参数：

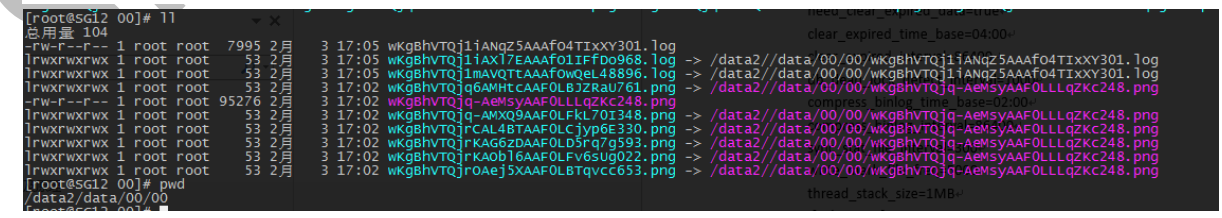
vim /etc/fdfs/storage.conf

```
check_file_duplicate=1
file_signature_method=hash
key_namespace=FastDFS
keep_alive=1
#include /etc/fdht/fdht_servers.conf
```

3. 双机启动 **fdhtd** 服务器并测试

```
service fdhtd start
chkconfig fdhtd on
```

重复文件将软链接形式存在，节省存储空间



自定义文件名

拼接 url 方式

上传文件时保存原始文件名和 file_id 的对应关系到 redis 或者 mysql 中，下载时用 Nginx 的域名和 file_id 拼出 url，url 后面增加参数，指定原始文件名

Content-disposition 可以控制用户请求所得的内容存为一个文件的时候提供一个默认的文件名，文件直接在浏览器上显示或者在访问时弹出文件下载对话框,书写格式为：

Content-Disposition "attachment;filename=$arg_attname";

我们以此方式配置 storage server 的 nginx

```
server {
        listen 80;
        server_name localhost;
        location /G1/M00{
        root /data2/;
        if ($arg_attname ~ "^(.*).(png|txt)") {
            add_header Content-Disposition "attachment;filename=$arg_attname";
        }
        ngx_fastdfs_module;
         }
      }
```

注释：

$arg_attname 变量截取 url 的 attname 参数的值，构造的 url 为：

http://192.168.1.138/G1/M00/00/00/wKgBhVTQjrCAL4BTAAF0LCjyp6E330.png?attname=test.png

访问构造好的 url 时就会下载 test.png 而不是在浏览器中直接访问

注释：

从 fastdfs_nginx_module 的 1.0.4 版本以后已经支持通过 filename 来在 url 指定原文件名，不需要额外增加上述配置

http://192.168.1.138/G1/M00/00/00/wKgBhVTQjrCAL4BTAAF0LCjyp6E330.png?filename=test.png

其他参考文档：

http://www.tuicool.com/articles/raaUNj

HTTP header 详细介绍

http://my.oschina.net/abian/blog/131548

使用 redis

可以通过 redis 或者 mysql 的方法，每次上传文件成功后将 file_id 和真实文件的映射关系记录到 redis 或者 mysql，程序使用时读取映射关系获得原文件名

simlinux.com

缩略图

下载软件包

wget http://luajit.org/download/LuaJIT-2.0.2.tar.gz (C 写的 lua 解释器)
wget https://github.com/openresty/lua-nginx-module/archive/v0.9.14.tar.gz \
-O lua-nginx-module-0.9.14.tar.gz
wget
http://downloads.sourceforge.net/project/graphicsmagick/graphicsmagick/1.3.20/GraphicsMagick-1.3.20.tar.gz?r=http%3A%2F%2Fsourceforge.net%2Fprojects%2Fgraphicsmagick%2Ffiles%2Fgraphicsmagick%2F1.3.20%2F&ts=1423041523&use_mirror=iweb –O GraphicsMagick-1.3.20.tar.gz
wget http://www.lua.org/ftp/lua-5.3.0.tar.gz
wget https://github.com/simpl/ngx_devel_kit/archive/v0.2.19.tar.gz -O ngx_devel_kit0.2.19.tar.gz(拓展 nginx 服务器核心功能的模块，第三方模块开发可以基于它来快速实现)
git clone https://github.com/agentzh/echo-nginx-module (用于测试 lua 扩展是否正常)

yum install libtermcap-devel ncurses-devel libevent-devel readline-devel pcre-*
tar xf lua-5.3.0.tar.gz
cd lua-5.3.0
make linux
make install

tar xf LuaJIT-2.0.2.tar.gz
cd LuaJIT-2.0.2
make install

tar xf GraphicsMagick-1.3.20.tar.gz
cd GraphicsMagick-1.3.20
./configure
make
make install

cd nginx-1.7.9/
./configure --prefix=/usr/local/nginx --with-pcre --add-module=../ngx_devel_kit-0.2.19/ \
--add-module=../lua-nginx-module-0.9.14/ --add-module=../echo-nginx-module \
--add-module=../fastdfs-nginx-module/src \
--with-http_gzip_static_module --with-http_stub_status_module --with-http_ssl_module

make -j8
make install

simlinux.com

```
ln -s /usr/local/lib/libluajit-5.1.so /lib64/libluajit-5.1.so.2
ldconfig
```

配置 nginx

```
server {
    listen 80;
    server_name 192.168.1.133;
        location / {
            default_type text/html;
            content_by_lua '
                ngx.say("<p>hello, world</p>")
                ';
        }

        location /hello {
            default_type text/html;
            echo "hello, world";
        }

        location /G1/M00 {
            alias /data2/data;

            set $image_root "/data2/data";
            if ($uri ~ "/([a-zA-Z0-9]+)/([a-zA-Z0-9]+)/([a-zA-Z0-9]+)/([a-zA-Z0-9]+)/(.*)") {
                set $image_dir "$image_root/$3/$4/";
                set $image_name "$5";
                set $file "$image_dir$image_name";
            }

            if (!-f $file) {
                #lua_code_cache off;
                content_by_lua_file "conf/lua/fastdfs.lua";
            }

            ngx_fastdfs_module;
        }
       expires 10d ;
    }
```

lua 处理脚本

```
vim /usr/local/nginx/conf/lua/fastdfs.lua
-- 写入文件
local function writefile(filename, info)
```

simlinux.com

```
    local wfile=io.open(filename, "w") --写入文件(w 覆盖)
    assert(wfile)    --打开时验证是否出错
    wfile:write(info)    --写入传入的内容
    wfile:close()    --调用结束后记得关闭
end

-- 检测路径是否目录
local function is_dir(sPath)
    if type(sPath) ~= "string" then return false end

    local response = os.execute( "cd " .. sPath )
    if response == 0 then
        return true
    end
    return false
end

-- 检测文件是否存在
local file_exists = function(name)
    local f=io.open(name,"r")
    if f~=nil then io.close(f) return true else return false end
end

local area = nil
local originalUri = ngx.var.uri;
local originalFile = ngx.var.file;
local index = string.find(ngx.var.uri, "([0-9]+)x([0-9]+)");
if index then
    originalUri = string.sub(ngx.var.uri, 0, index-2);
    area = string.sub(ngx.var.uri, index);
    index = string.find(area, "([.])");
    area = string.sub(area, 0, index-1);

    local index = string.find(originalFile, "([0-9]+)x([0-9]+)");
    originalFile = string.sub(originalFile, 0, index-2)
end

-- check original file
if not file_exists(originalFile) then
    local fileid = string.sub(originalUri, 2);
    -- main
    local fastdfs = require('restyfastdfs')
    local fdfs = fastdfs:new()
    fdfs:set_tracker("192.168.1.138", 22122)
```

simlinux.com

```
        fdfs:set_timeout(1000)
        fdfs:set_tracker_keepalive(0, 100)
        fdfs:set_storage_keepalive(0, 100)
        local data = fdfs:do_download(fileid)
        if data then
            -- check image dir
             if not is_dir(ngx.var.image_dir) then
                  os.execute("mkdir -p " .. ngx.var.image_dir)
             end
             writefile(originalFile, data)
        end
    end

    -- 创建缩略图
    local image_sizes = {"80x80", "800x600", "40x40", "60x60"};
    function table.contains(table, element)
        for _, value in pairs(table) do
             if value == element then
                  return true
             end
        end
        return false
    end

    if table.contains(image_sizes, area) then
        local command = "gm convert " .. originalFile   .. " -thumbnail " .. area .. " -background gray
-gravity center -extent " .. area .. " " .. ngx.var.file;
        os.execute(command);
    end;

    if file_exists(ngx.var.file) then
        --ngx.req.set_uri(ngx.var.uri, true);
        ngx.exec(ngx.var.uri)
    else
        ngx.exit(404)
    end
```

注：

1. 缩略图样式可以自定义，通过查看 gm 命令帮助
2. 拓展：缩略图增加水印 http://www.cnblogs.com/kgdxpr/archive/2014/07/11/3837264.html
3. 在 web 服务器上做缩略图处理，针对 FastDFS 来讲需要在 storage server 上配置上述模块，缩略图存储并没有通过 FastDFS，storage server 间不会同步

simlinux.com

4.其他方案：OpenResty(Nginx)+Lua+GraphicsMagick http://www.hopesoft.org/blog/?p=1188

过期缩略图清理

0 4 * * * /bin/find /data2/data -atime -7 | xargs rm –rf

清除缓存

wget http://labs.frickle.com/files/ngx_cache_purge-1.2.tar.gz
重新编译 nginx，增加如下配置

```
location ~ /purge(/.*)
{
        allow 192.168.1.134;
        deny  all;
        proxy_cache_purge http-cache $host:$server_port$1$is_args$args;
    }
```

六、FastDFS 开发 API 使用

C# API
https://code.google.com/p/fastdfs/downloads/detail?name=FastDFS_Client_Dotnet.rar&can=2&q=
https://github.com/jonnyyu/FastDFS.Net

PHP API：
http://blog.51yip.com/server/219.html
http://blog.csdn.net/john_f_lau/article/details/11020989#t7

七、性能优化及安全

系统方面

系统初始化脚本：
https://github.com/geekwolf/sa-scripts/blob/master/linux/systeminit.sh

FastDFS 方面

1、 文件上传、追加、删除等操作使用 FastDFS client API

simlinux.com

2、 文件下载采用 HTTP 方式，如使用 nginx 及 fastdfs_nginx_module 扩展模块
3、 系统做 raid1，数据盘直接挂载单盘，每个硬盘做为一个挂载点
4、 适用于小文件存储，不适合大文件存储
5、 相关参数调整（参考 FastDFS 角色配置参数说明）

安全配置

tracker server & proxy

-A INPUT -s 192.168.1.0/24 -p tcp -m multiport --dports 22122 -j ACCEPT
-A INPUT -p tcp -dports 80 -j ACCEPT

storage server

-A INPUT -s 192.168.1.0/24 -p tcp -m multiport --dports 80,23000,11411 -j ACCEPT

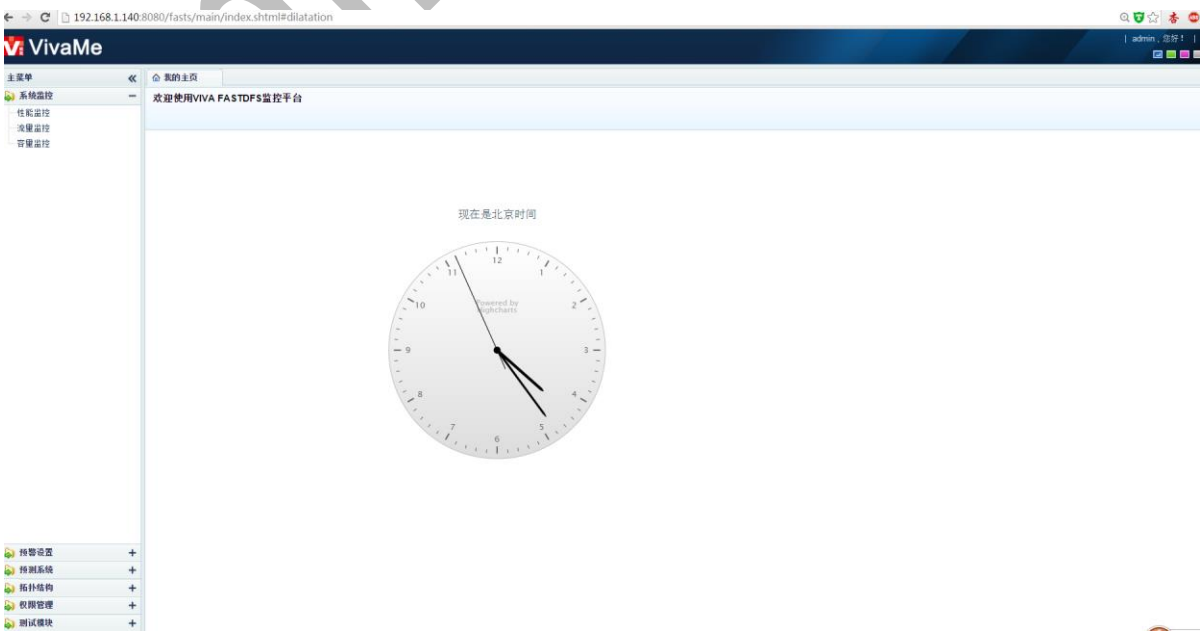
tracer server & storage server 服务访问控制

allow_hosts=

八、监控

1.Cacti 监控插件
https://github.com/zhangchu/cacti_fastdfs
http://www.percona.com/downloads/percona-monitoring-plugins/LATEST/

2.使用基于 java 开发的开源的 FastDFS 监控系统
https://code.google.com/p/fastdfs-zyc/downloads/list



simlinux.com

九、FastDFS 常见问题

1. **双 tracker 环境，其中一台宕机后，客户端依然会再次检查坏 tracker 是否可用，不可用换下一个问题？**
   A. 个人觉得这样的故障转移不彻底，可用考虑在业务层通过连接池方式解决(加 tracker 心跳)
   B. 将双 tracker 通过 Keepalive 做 HA，使用 vip 提供服务
2. **fadfs_monitor 命令查看集群状态，storage 状态始终处于 OFFLINE 状态？**
   A. 查看日志确保 fdfs_trackerd 和 fdfs_storaged 服务正常运行
   B. 如果日志正常依然 OFFLINE，可用临时关闭 fdfs_trackerd 和 fdfs_storaged 服务，删除 base_path 的元数据和日志，重新启动相关服务
3. **是否可用通过 http 方式上传文件？**
   A. 请参考 https://github.com/gzldx/lua-upload
4. **在编译 fastdfs-nginx-module 扩展时，提示编译错误**
   make[1]: *** [objs/addon/src/ngx_http_fastdfs_module.o] 错误 1
   make[1]: Leaving directory `/root/nginx-1.4.1'
   make: *** [install] 错误 2
   A. 尝试将 libfastcommon 和 fastdfs-5.05 重新编译安装即可

注释：FastDFS 客户端若不指定上传的 group 将会有 tracker 按照相应的策略分配，若指定则上传到对应的 group 组内

参考资料：

http://blog.chinaunix.net/uid/26786622/abstract/1.html

http://blog.csdn.net/john_f_lau/article/details/11020989#t14

http://blog.51yip.com/server/219.html

http://tengine.taobao.org/book/index.html

simlinux.com