FastDFS

目　　录

1 FastDFS 简介

FastDFS(distributed file system)是一款开源的分布式文件系统，用来对文件进行管理，功能包括：文件存储、文件同步、文件访问（上传、下载），解决了大容量存储的问题，并支持负载均衡。（FastDFS 开源社区论坛地址：http://bbs.chinaunix.net/forum-240-1.html，此资料中的绝大部分内容来源于开源社区）。

2 FastDFS 应用场景

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，不适合分布式计算场景。

FastDFS Server 仅支持 unix 系统，在 Linux 和 FreeBSD 测试通过。

V5.0 以前的版本还依赖 libevent；V5.0 以后不再依赖 libevent。V5.04 开始依赖 libfastcommon。V5 版本从 V5.05 开始才是稳定版本，截止目前（2015-11-25）为止最新版是 V5.07。

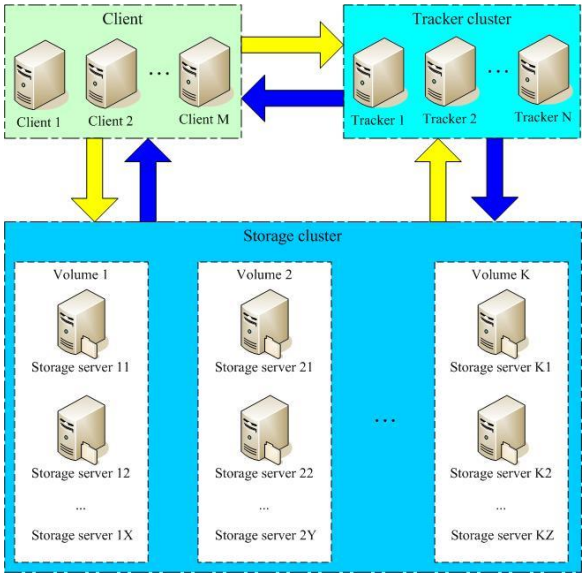
3 FastDFS 的组成

FastDFS 服务端包含两个角色：

Tracker（跟踪器）：负责访问调度工作，实现负载均衡。

Storage（存储节点）：完成管理文件的所有功能，包括：文件的存储、同步、访问；同时对文件的 metadata（元数据）进行管理，metadata 是文件属性的键值对。

4 FastDFS 的系统架构



FastDFS 中的两个角色 Tracker 和 Storage 都支持集群部署，即 Tracker 和 Storage 都可以由一个或多个服务器构成，Tracker 和 Storage 中的节点可以随时添加或删除，而且不会影响其他线上节点的使用。

Tracker 中的所有服务器都是对等的，可以根据服务器的压力情况随时增加或减少。

Storage 采用来了分卷（组）的管理方式，存储系统由一个或多个卷组成。

卷之间彼此独立，不进行数据同步。

一个卷可以由一台或多台存储服务器组成，一个卷下的服务器之间的文件都是相同的，卷中的多台服务器起到了冗余备份和负载均衡的作用。

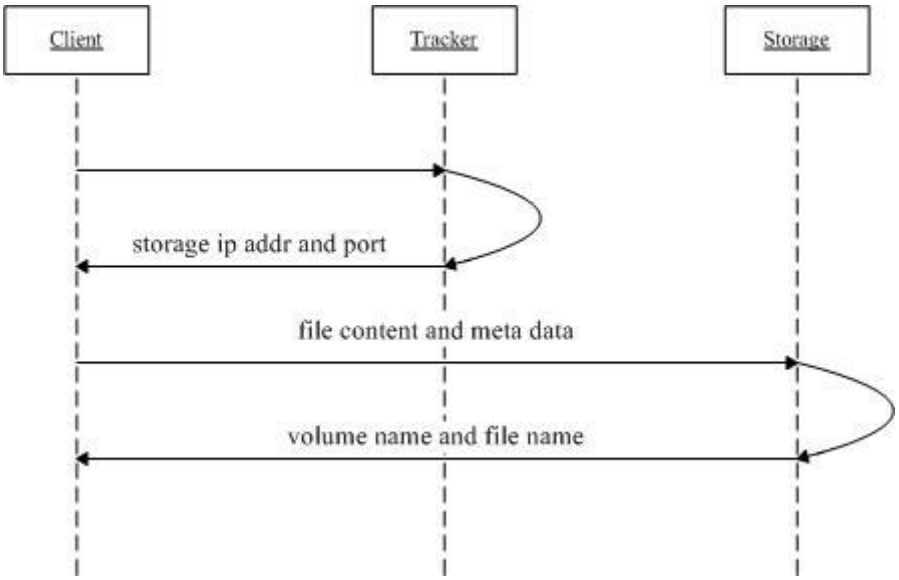
在卷中增加服务器时，文件同步由系统自动完成，同步完成后，系统自动将新服务器切换到线

上提供服务。

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

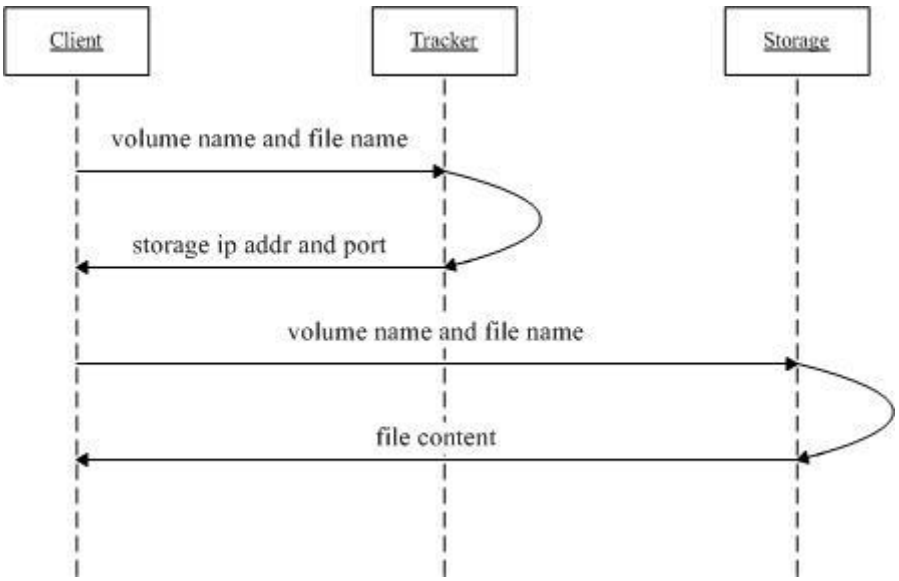
一台 storage 服务器可以管理多块硬盘啊，一台 storage 服务器上只需要启动一个 storage server 进程即可，即一台 storage server 只属于一个 group。

5 FastDFS 的文件上传过程



1. Client 询问 Tracker 上传到的 Storage，不需要附加参数。
2. Tracker 返回一台可用的 Storage。
3. Client 直接和 Storage 通讯完成文件上传。

6 FastDFS 的文件下载过程



1. Client 询问 Tracker 下载文件的 Storage，参数为文件标识（组名和文件名）
2. Tracker 返回一台可用的 Storage

3. Client 直接和 Storage 通讯完成文件下载

7 Storage 文件源同步过程

每次部署集群，源同步就会发生，首次启动第一台机器时是直接添加到组内，而后面的机器都要进行源同步。添加 Storage 需要做源同步，也就是从组内现有的一台机器上推送历史数据到新机器。源同步的目的就是为了保持新添加的机器数据与旧机器一致。

源同步性能：

整个源同步过程中都是源机器使用一个同步线程，将数据推到新机器，因此和机器的磁盘个数没有关系。也就是说最多只能达到一个磁盘的 IO，而不能并发。

由于源同步的截止条件为取不到 binlog 的时刻，因此若系统很繁忙，不断有新数据写入，就将会导致一致无法完成源同步这个过程。所以不建议在使用高峰期进行数据同步。

概念解释：

源机器：表示被 Tracker 选择出来，源机器需要将持有的数据推送给新机器。

源同步：就是源机器推送旧数据给新机器的过程。

增量同步：也就是 Binlog 同步，此时是组内的机器进行互相同步。

涉及到的文件：

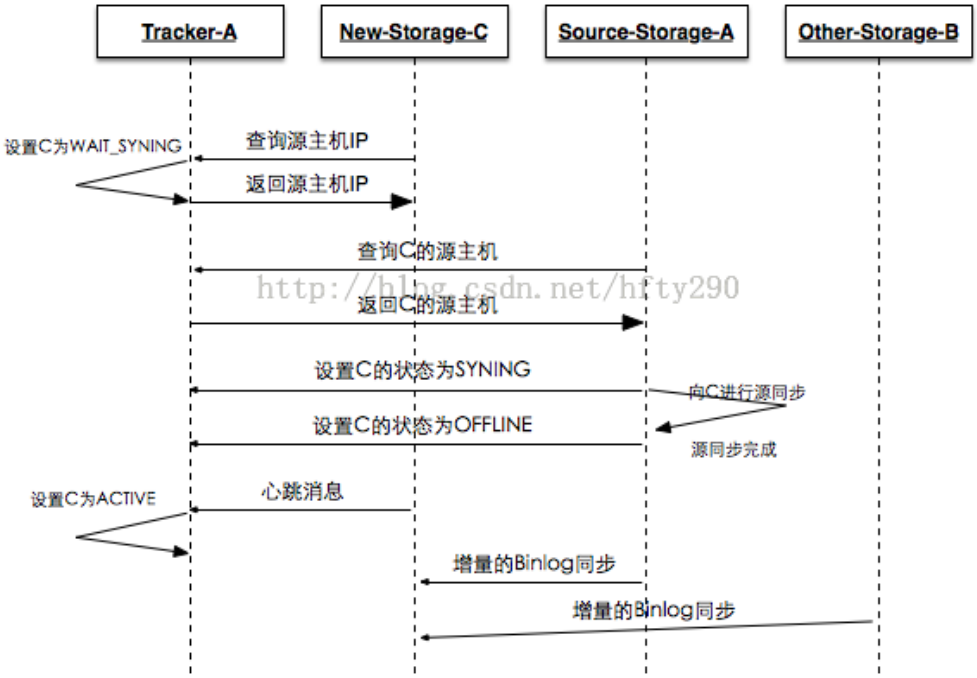
$base_path/data/.data_init_flag，与源同步相关的内容：

```
sync_old_done=1              ##是否成功获取到源机器 IP
sync_src_server=192.168.126.131    ##源主机的 IP 地址
sync_until_timestamp=1418285426      ##源主机负责的旧数据截止时间
```

$base_path/data/sync/192.168.126.131_23000.mark，与源同步相关的内容：

```
need_sync_old=0      ##是否需要同步旧文件给对方
sync_old_done=0      ##若需要同步旧文件，那么旧文件是否同步完成
until_timestamp=0    ##源主机负责的旧数据截止时间
```

源同步过程：



源同步过程

Storage 在启动时会为每个 Tracker 开启一个线程，用于维持与其的通讯。当 Storage 是首次加入集群时，会向 Tracker 发送一个请求，向 Tracker 咨询由哪台机器作为源。

Tracker 收到请求后的处理如下：

若组内没有机器，则告诉新机器不需要进行源同步；

若组内有机器但是没有状态为 Active 的机器，那么返回一个错误，新机器将睡眠后重试；

若组内有机器并且有状态为 Active 的机器，那么选择一台作为其源，返回两个值：**当前时间作为源同步截止时间**与**源机器的 IP**，并且在本地记录该信息，同时将新机器状态设置成**WAIT_SYNC**。

由于可能配置多个 Tracker，因此该请求在 Storage 中是采用互斥量同步的，因此一次只会有一个线程能够发起该请求，若请求成功则会设置标志位，第二个线程就不会再作该请求。

当新机器收到 Tracker 返回的源机器 IP 和源同步截止时间后，会将信息被写入到 .data_init_flag 的 **sync_src_server** 和 **sync_until_timestamp** 两个字段，并且将 **sync_old_done** 字段更新成 1，表示已完成获取源主机的操作。

例如：当前 FastDFS 集群的配置如下，两个 Tracker 为 Tracker-A，Tracker-B；两台已存在的 Storage-A、Storage-B，现在添加 Storage-C。当添加 Storage-C 时查询到的源机器为 Storage-A。

新机器 Storage-C：

在从一台 Tracker 获取到源机器的 IP 和截止时间后，马上将该信息通知其他的 Tracker。然后进入循环定期向 Tracker 发送心跳信息。当源同步完成后，源会将 Storage-C 状态设置成 **OFFLINE**，在下一个心跳中，Tracker 会将 Storage-C 状态设置成 **ACTIVE**。此时新机器加入完毕。

源机器 Storage-A：

在与 Trakcer 的心跳中，得到新机器 Storage-C 的信息，启动一个线程负责对 Storage-C 的同步。下面的操作都在该同步线程之中：

1. 由于本地并没有 Storage-C 的信息（没有对应的 mark 文件），因此就会向 Tracker 查询

Storage-C 的源机器 IP 与源同步截止时间，同时判断这个源是否为自己。

2. 发现自己就是新机器 Storage-C 的源，则将信息写入到 **Storage-C_23000.mark** 文件之中，其中 **need_sync_old=1; sync_old_done=0; util_timestamp = 截止时间**。
3. 请求 Tracker 将 Storage-C 的状态设置成 **SYNING**。
4. 从头开始遍历本地的 binlog 文件，对于每一条记录执行如下判断：

   若为**源操作**、或者**非源操作并且操作时间小于源同步截止时间**，则将该条记录对应操作同步给 Storage-C;（同步过程中也会记录最后同步的 offset，假如此时宕机了，下次启动也可以读到 Storage-C_23000.mark 文件的 need_sync_old 与 binlog_offset 值继续源同步过程）。
5. 当在某一个时刻，取不到更多地 binlog 时，请求 Tracker 将 Storage-C 状态设置成 OFFLINE。此时源同步完成。

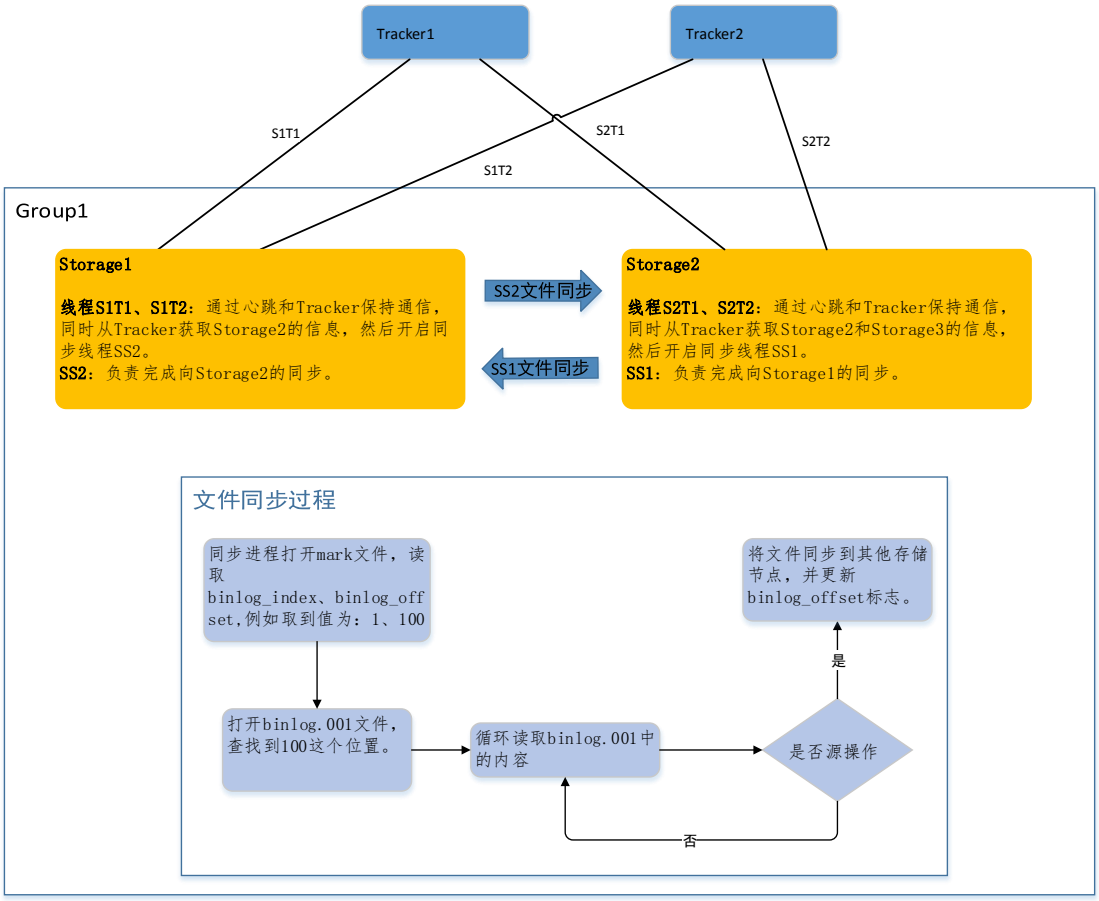
非源机器 Storage-B：

在与 Trakcer 的心跳中，得到新机器 Storage-C 的信息，启动一个线程负责对 Storage-C 的同步。下面的操作都在该同步线程之中，Storage-B 的同步属于增量同步过程：

1. 由于本地并没有 Storage-C 的信息（没有对应的 mark 文件），因此就会向 Tracker 查询 Storage-C 的源机器 IP 与源同步截止时间，同时判断这个源是否为自己。
2. 新机器 Storage-C 的源是 Storage-A 不是自己，将信息写入到 Storage-C_23000.mark 文件之中，其中 **need_sync_old=0; sync_old_done=0; util_timestamp = 截止时间**。
3. 进入睡眠等待，知道 Storage-C 的状态变成 ACTIVE。
4. 打开 binlog，跳过操作时间小于源同步截止时间的记录，对之后的每一条为元操作的 Binlog，都同步给 Storage-C。

8 Storage 文件增量同步过程

Storage 节点增量同步过程如下：



文件同步过程

同步过程描述：

1. 获取组内的其他 Storage 信息，并启动同步线程

在 Storage.conf 配置文件中,只配置了 Tracker 的 IP 地址,并没有配置组内其他的 Storage。因此同组的其他 Storage 必须从 Tracker 获取。具体过程如下：

1）Storage启动时为每一个配置的Tracker启动一个线程负责与该Tracker的通讯。

2）默认每间隔30秒，与Tracker发送一次心跳包，**在心跳包的回复中，将会有该组内的其他Storage信息**。

3）Storage获取到同组的其他Storage信息之后，为组内的每个其他Storage开启一个线程负责同步。

2. 同步线程执行同步过程

每个同步线程负责一台 Storage 的同步，以阻塞方式进行。

1）打开对应 Storage 的 mark 文件，如负责到 192.168.182.132 的同步则打开 192.168.182.132_23000.mark 文件，从中读取 binlog_index、binlog_offset 两个字段值，如取到值为：1、100，那么就打开 binlog.001 文件，seek 到 100 这个位置。

2）进入一个 while 循环，尝试着读取一行，若读取不到则睡眠等待。若读取到一行，并且该行的操作方式为源操作，如 C、A、D、T（大写的都是），则将该行指定的操作同步给对方（非源操作不需要同步），同步成功后更新 binlog_offset 标志，该值会定期写入到 192.168.182.132_23000.mark 文件之中。

同步信息记录位置：base_path/data/sync，同步涉及到的文件：

IP_XXX.mark：同步状态文件，记录本机到IP地址机器的同步状态。

binlog.index：记录当前使用的binlog文件序号，如果为1，则使用binlog.001文件。

binlog.000,binlog001...binlogXXX：真实的 binlog 文件。

Mark文件内容说明

Mark文件名的格式：192.168.182.132_23000.mark

Mark文件中同步相关内容描述：

binlog_index=0 表示上次同步给192.168.182.132机器的最后一条binlog文件索引。

binlog_offset=116 表示上次同步给 192.168.182.132 机器的最后一条 binlog 偏移量，若程序重启了，只要从这个位置开始向后同步即可。

Binlog文件内容说明

binlog.000文件是由一条条binlog日志组成的,内容样例：

1418285342 c M01/00/00/CgAHl1SJUR6AZqSHAAAF1vgN0rw59.conf

1418285734 C M00/00/00/CgAFklSJUqaAL7hBAAAF1vgN0rw38.conf

1418285809 C M01/00/00/CgAFklSJUvGAL96CAAAF1vgN0rw96.conf

每一条记录都是使用空格符分成三个字段：

1. 第一个字1418285342表示文件上传的时间戳。
2. 第二个字段表示文件创建方式：

   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. 第三个字段M01/00/00/CgAHl1SJUR6AZqSHAAAF1vgN0rw59.conf

   M01为storepath索引，紧接着00/00/为路径，后面CgAHl1SJUR6AZqSHAAAF1vgN0rw59.conf 为文件系统中实际的文件名。

9 Storage 文件最后最早被同步时间

一组内有Storage-A、Storage-B、Storage-C三台机器。对于A这台机器来说，B与C机器都会同步Binlog（包括文件）给他，A在接受同步时会记录每台机器同步给他的最后时间（也就是Binlog中的第一个字段timpstamp）。比如B最后同步给A的Binlog-timestamp为100，C最后同步给A的Binlog-timestamp为200，那么A机器的最后最早被同步时间就为100.

最早被同步时间判断一个文件是否存在某个Storage上。比如这里A机器的最后最早被同步时间为100，那么如果一个文件的创建时间为99，就可以肯定这个文件在A上肯定有。为什呢？一个文件会Upload到组内三台机器的任何一台上：

1）若这个文件是直接Upload到A上，那么A肯定有。

2）若这个文件是Upload到B上，由于B同步给A的最后时间为100，也就是说在100之前的文件都已经同步A了，那么A肯定有。

3）同理C也一样。

Storage会定期将每台机器同步给他的最后时间告诉给Tracker，Tracker在客户端要下载一个文件时，需要判断一个Storage是否有该文件，只要解析文件的创建时间，然后与该值作比较，若该值大于创建创建时间，说明该Storage存在这个文件，可以从其下载。

Tracker也会定期将该值写入到一个文件之中，Storage_sync_timestamp.dat，内容如下：

group1,10.0.0.1,0,1408524351,1408524352

group1,10.0.0.2,1408524353,0,1408524354

group1,10.0.0.3,1408524355,1408524356,0

每一行记录了，对应 Storage 同步给其他 Storage 的最后时间，如第一行表示 10.0.0.1 同步给 10.0.0.2 的最后时间为 1408524351，同步给 10.0.0.3 的最后时间为 1408524352；同理可以知道后面两行的含义了。每行中间都有一个 0，这个零表示自己同步给自己，没有记录。按照纵列看，取非零最小值就是最后最早同步时间了，如第三纵列为 10.0.0.1 被同步的时间，其中 1408524353 就是其最后最早被同步时间。

10 FastDFS 配置文件说明

10.1 跟踪器配置文件 Tracker.conf

10.1.1 基础配置参数

disabled=false

定义 tracker.conf 配置文件是否生效，false 生效，true 不生效，需要注意。

bind_addr=

设置 tracker 服务绑定的 IP，如果服务器有多个 IP 地址，但只想用一个 IP 提供服务可以使用此项，默认为空，即所有 IP 都提供服务。

port=22122

tracker 服务使用的端口号。

download_server=0

执行文件下载的服务器位置，取值范围：0，1。

0：轮询方式，在存储文件的卷中的节点上轮流下载。

1：从源存储节点下载文件（就是文件一开始上传到的存储节点）。

reserved_storage_space = 10%

指定存储节点保留的空闲空间，可以是百分比或绝对值大小，可以指定 K,M,G 大小，百分比支持 xx.xx%的格式。如果一个卷中的存储节点磁盘大小不一样，如果有一个节点的剩余空间<= reserved_storage_space 的值，则该组中的所有节点都不能再上传文件。

run_by_group=

运行 FastDFS 软件的操作系统用户组，不填默认为启动 FastDFS 的用户所在用户组。

run_by_user=

运行 FastDFS 软件的操作系统用户，不填默认为启动 FastDFS 的用户。

store_slave_file_use_link = false

存储从文件是否采用 symbol link（符号链接）方式。如果设置为 true，一个从文件将占用两个文件：原始文件及指向它的符号链接。

base_path=/data/fastdfs/tracker

存储数据和日志的根目录，base_path 的值必须设置，子目录不用设置，会自动创建。

10.1.2 网络参数

allow_hosts=*

允许连接到当前 tracker 服务器的主机，可以是主机名和 IP 地址，默认为*即所有主机都可以连接到此 tracker 服务器，此限制对客户端、存储节点都有效。

配置文件中给出的示例：

allow_hosts=10.0.1.[1-15,20]：主机 IP 地址在 10.0.1.1 到 10.0.1.15 之间的机器和 IP 地址为 10.0.1.20 的机器都能连接当前 tracker 服务器，一共 16 台机器，要连接当前 tracker 服务器的客户端和存储节 IP 地址必须是这 16 个 IP 中的一个。

allow_hosts=host[01-08,20-25].domain.com：主机名为 host01.domain.com 到 host08.domain.com、host020.domain.com 到 host25.domain.com 之间的机器都能连接当前 tracker 服务器，一共 14 台机器，要连接当前 tracker 服务器的客户端和存储节点的主机名必须是这 14 个主机名中的一个。

10.1.3 连接、线程、内存参数

use_connection_pool = false

是否使用连接池。

connection_pool_max_idle_time = 3600

连接池中连接的最大空闲时间，单位秒，默认 3600 秒，一个连接的空闲时间超过此值后将被关闭。

connect_timeout=30

连接超时时间，单位为秒，默认 30 秒。

network_timeout=60

tracker 服务的网络超时时间，单位为秒，默认 60 秒。发送或接收数据时，如果在超时时间后还不能发送或接收数据，则本次网络通信失败。

max_connections=256

服务能提供的最大连接数，默认值 256。

work_threads=4

工作线程数，通常设置为 CPU 数。

thread_stack_size = 64KB

线程栈的大小，默认是 64KB，不能小于 64KB，线程栈越大，一个线程占用的系统内存资源就越多。如果线程分配不够用，并且一个线程栈分配过大时，可以适当加大线程数，并降低线程栈的大小，但是不能小于 64KB。

10.1.4 storage 和 tracker 通信参数

store_server=0

指定上传到存储节点的方式，取值范围 0、1、2。

0：轮询方式。

1：根据 IP 地址排序，选择 IP 最小的一个服务器。

2：根据优先级来设置，每个存储节点的优先级在存储节点的 storage.conf 文件中配置，由

upload_priority 参数决定。

store_server 相当于指定了一个卷中文件存放的源节点，源节点会将该文件同步到一个卷中的其他存储节点上。

store_path=0

指定文件上传时选择目录的方式，store_path 取值范围：0，2。

0：轮询方式，多个磁盘依次存放。

2：选择剩余空间最大的目录存放。

store_path 指定将文件上传到存储节点的哪个 base_path 目录下，因为在一个存储节点中可以指定多个 base_path，一个 base_path 可以设置为一个磁盘挂载点。

store_lookup=2

文件上传到卷的方式，取值范围：0、1、2。

0：轮询方式，假如有 2 个存储卷，第 1 次文件上传到卷 1，第二次文件上传到卷 2，第 3 次文件再上传到卷 1，第 4 次文件上传到卷 2，如此循环。

1：指定卷，指定上传的卷。

2：平衡负载，选择空闲空间最大的卷存放文件。

如果在应用中指定了上传到哪个组，那么这个参数被绕过。

store_group=group2

当 store_lookup=1 时和 store_lookup 配合使用，指定文件上传的卷。如果 store_lookup 的值为 0 或 2，此参数即使有值也失效。

check_active_interval = 120

检查存储节点是否活跃的时间间隔，单位秒，默认 120 秒检查一次。

storage server 定期向 tracker server 发心跳，如果 tracker server 在一个 check_active_interval 内还没有收到 storage server 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。**

storage_ip_changed_auto_adjust = true

控制当 storage server IP 地址改变时，集群是否自动调整，只有在 storage server 进程重启时才完成自动调整。

storage_sync_file_max_delay = 86400

存储服务器之间同步文件的最大延迟时间，单位秒，默认为 1 天（86400 秒）。根据实际情况进行调整。

storage_sync_file_max_time = 300

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。

use_storage_id = false

是否使用 server ID 作为 storage server 标识。

storage_ids_filename = storage_ids.conf

use_storage_id 设置为 true，才需要设置本参数，可以使用相对路径或绝对路径。在文件中设置组名、server ID 和对应的 IP 地址，参见源码目录下的配置示例：conf/storage_ids.conf

storage_id.conf 文件示例：

```
# <id>  <group_name>  <ip_or_hostname>
# 100001   group1  192.168.0.196
# 100002   group1  192.168.0.116
```

id_type_in_filename = ip

设置存储节点服务器在文件中的 id 类型，可以为 ip 或 id，默认为 ip。

ip：存储节点服务器的 ip 地址。

id：存储节点服务器的 id。

只有 use_storage_id=true 时，此参数才生效。

10.1.5 日志文件参数

log_level=info

定义日志级别，区分大小写，日志级别由高到低有：

emerg：紧急，alert：警报，crit：严重，error：错误，warn：警告，notice：注意事项，info：信息，debug：调试。

log_file_keep_days = 0

日志文件保存的天数，默认值 0 表示不删除日志文件一直保存。

######################### error log 错误日志#########################

rotate_error_log = false

是否定期轮转 error log，目前仅支持一天轮转一次。

error_log_rotate_time=00:00

error log 定期轮转的时间点，只有当 rotate_error_log 设置为 true 时有效。

rotate_error_log_size = 0

error log 按大小轮转

设置为 0 表示不按文件大小轮转，否则当 error log 达到该大小，就会轮转到新文件中。

10.1.6 文件由内存到磁盘的同步参数

sync_log_buff_interval = 10

将内存中的日志信息写入到磁盘的时间间隔，单位秒，默认 10 秒写入一次。

10.1.7 小文件合并存储特性参数

use_trunk_file = false

是否使用小文件合并存储特性，缺省是关闭的。

slot_min_size = 256

trunk file 分配的最小字节数。比如文件只有 16 个字节，系统也会分配 slot_min_size 个字节。

slot_max_size = 16MB

定义小文件的界限值，当文件<=slot_max_size 时，才会合并存储。如果一个文件的大小大于这个参数值，将直接保存到一个文件中（即不采用合并存储方式）。

slot_max_size 必须大约 slot_min_size。

trunk_file_size = 64MB

合并存储的 trunk file 大小，至少 4MB，默认值 64MB。不建议设置的过大。

trunk_create_file_advance = false

是否提前创建 trunk file。只有当这个参数为 true，下面 3 个以 trunk_create_file_打头的参数才有效。

trunk_create_file_time_base = 02:00

提前创建 trunk file 的起始时间点，02:00 表示第一次创建的时间点是凌晨 2 点。

trunk_create_file_interval = 86400

创建 trunk file 的时间间隔，单位为秒。如果每天只提前创建一次，则设置为 86400。

trunk_create_file_space_threshold = 20G

提前创建 trunk file 时，需要达到的空闲 trunk 大小，比如本参数为 20G，而当前空闲 trunk 为 4GB，那么只需要创建 16GB 的 trunk file 即可。

trunk_init_check_occupying = false

trunk 初始化时，是否检查可用空间是否被占用。

trunk_init_reload_from_binlog = false

是否无条件从 trunk binlog 中加载 trunk 可用空间信息

FastDFS 缺省是从快照文件 storage_trunk.dat 中加载 trunk 可用空间，

该文件的第一行记录的是 trunk binlog 的 offset，然后从 binlog 的 offset 开始加载

trunk_compress_binlog_min_interval = 0

压缩 binlog 文件的最小时间间隔，默认值为 0，不压缩。

FastDFS 在 trunk 文件初始化和销毁时压缩 binlog，推荐设置为 86400 秒。

10.1.8 其他参数

http.server_port=8080

tracker 服务的端口

http.check_alive_interval=30

每隔多长时间检查 storage 服务器的活动状态，单位秒，默认每隔 30 秒检查一次

http.check_alive_type=tcp

设定检查 storage 服务器活动的类型，取值：tcp 或 http，

tcp：表示只连接到 storage 服务器的 http 端口，不请求和获取回复。

http：检查 storage 服务器活动的 url，必须返回 http 的状态为 200。

http.check_alive_uri=/status.html

检查 storage 的 http 服务活动的 url

10.2 存储节点配置文件 Storage.conf

10.2.1 基础配置参数

disabled=false

定义 storage.conf 配置文件是否生效，false 生效，true 不生效，需要注意。

group_name=group1

指定存储节点所属的卷（组），如果不配置或注释掉这个属性，需要将 tracker.conf 中的 use_storage_id 设置为 true，并配置 storage_ids.conf 配置，才能使用这个存储节点。

base_path=/home/yuqing/fastdfs

数据文件和日志文件的存放位置。

store_path_count=1

存放文件时 storage server 支持多个路径（例如磁盘）。这里配置存放文件的基路径数目，通常只配一个目录。

store_path0=/home/yuqing/fastdfs

#store_path1=/home/yuqing/fastdfs2

逐一配置 store_path 的路径，索引号基于 0。注意配置方法后面有 0,1,2 .....，需要配置 0 到 store_path - 1。如果不配置 base_path0，那它就和 base_path 对应的路径一样。

subdir_count_per_path=256

FastDFS 存储文件时，采用了两级目录。这里配置存放文件的目录个数，取值范围 1-256。如果本参数只为 N（如：256），那么 storage server 在初次运行时，会自动创建 N * N 个存放文件的子目录。

run_by_group=

运行 FastDFS 软件的操作系统用户组，不填默认为启动 FastDFS 的用户所在用户组。

run_by_user=

运行 FastDFS 软件的操作系统用户，不填默认为启动 FastDFS 的用户所在用户。

file_distribute_path_mode=0

文件在 data 目录下分散存储策略。

0: 轮流存放，在一个目录下存储设置的文件数后（参数 file_distribute_rotate_count 中设置文件数），使用下一个目录进行存储。

1: 随机存储，根据文件名对应的 hash code 来分散存储。

file_distribute_rotate_count=100

当 file_distribute_path_mode 配置为 0（轮流存放方式）时，本参数有效。

当一个目录下的文件存放的文件数达到本参数值时，后续上传的文件存储到下一个目录中。

upload_priority=10

本 storage server 作为源服务器，上传文件的优先级，可以为负数。值越小，优先级越高，默认值 10。这里和 tracker.conf 中 store_server= 2 时的配置相对应。

10.2.2 网络参数

bind_addr=

设置 storage 服务绑定的 IP，如果服务器有多个 IP 地址，但只想用一个 IP 提供服务可以使用此项，默认为空，即所有 IP 都提供服务。

client_bind=true

本 storage server 作为 client 连接其他服务器（如 tracker server、其他 storage server）时是否绑定 IP 地址，true 绑定和 bing_addr 一样的地址，false 表示可以绑定任意一个本机的 IP 地址。

port=23000

storage server 服务端口。

connect_timeout=30

连接超时时间，单位秒，针对 socket 套接字函数 connect。

network_timeout=60

storage 服务器网络超时时间，单位秒。发送或接收数据时，如果在该值后还不能发送或接收数据，则本次网络通信失败。

allow_hosts=*

允许连接到当前 storage 服务器的主机，可以是主机名和 IP 地址，默认为*即所有主机都可以连接到此 storage 服务器。

配置文件中给出的示例：

allow_hosts=10.0.1.[1-15,20]：主机 IP 地址在 10.0.1.1 到 10.0.1.15 之间的机器和 IP 地址为 10.0.1.20 的机器都能连接当前 storage 服务器，一共 16 台机器，要连接当前 storage 服务器的客户端和存储节 IP 地址必须是这 16 个 IP 中的一个。

allow_hosts=host[01-08,20-25].domain.com：主机名为 host01.domain.com 到 host08.domain.com、host020.domain.com 到 host25.domain.com 之间的机器都能连接当前 storage 服务器，一共 14 台机器，要连接当前 storage 服务器的客户端和存储节点的主机名必须是这 14 个主机名中的一个。

if_alias_prefix=

网卡接口的前缀，例如 linux 中的 eth，多个前缀用逗号分隔，默认为空，意思为自动使用操作系统的设置。

10.2.3 连接、线程、内存参数

use_connection_pool = false

是否使用连接池，默认 false，不使用。

connection_pool_max_idle_time = 3600

连接池中连接的最大空闲时间，单位秒，默认值 3600，空闲时间超过此值的连接将被关闭。

max_connections=256

服务器支持的最大连接数

buff_size = 256KB

设置队列结点的 buffer 大小。工作队列消耗的内存大小 = buff_size * max_connections。设置得大一些，系统整体性能会有所提升。对于 32 位系统，请注意使用到的内存不要超过 3GB。

work_threads=4

工作线程数，通常设置为 CPU 数，配置的是 NIO 线程，也就是网络线程数。

accept_threads=1

配置 accept 线程数，专门处理客户端的连接操作，并为每个 Socket 连接创建一个任务，将该任务提交到 NIO 线程，如果配置了多个 Accept 线程，那么会开启 N-1 个线程执行同样地操作，因为主线程也执行该操作，如果为 1 则直接在主线程上进行。

Storage 之中有两种处理网络的线程，分别为 Accept 线程与 NIO 线程，前者专门处理客户端的连接操作，并为每个 Socket 连接创建一个任务，将该任务提交到 NIO 线程，处理网络数据的读写。这两种线程的个数都可以独立配置。

thread_stack_size=512KB

线程栈的大小。FastDFS server 端采用了线程方式。 线程栈越大，一个线程占用的系统资源就越多。torage server 线程栈不应不小于 512KB

10.2.4 I/O 参数

disk_rw_separated = true

磁盘读写是否分离，默认磁盘读写分离。

disk_reader_threads = 1

针对**单个存储路径**的读线程数，缺省值为 1。

读写分离时，系统中的读线程数 = disk_reader_threads * store_path_count。

读写混合时，系统中的读写线程数 = (disk_reader_threads + disk_writer_threads) *

disk_writer_threads = 1

针对**单个存储路径**的写线程数，缺省值为 1。

读写分离时，系统中的写线程数 = disk_writer_threads * store_path_count。

读写混合时，系统中的读写线程数 = (disk_reader_threads + disk_writer_threads) * store_path_count。

10.2.5 storage 和 tracker 通信参数

heart_beat_interval=30

心跳间隔时间，单位秒，这里是指主动向 tracker server 发送心跳。

stat_report_interval=60

向 tracker server 报告磁盘剩余空间的时间间隔，单位为秒。

tracker_server=192.168.209.121:22122

tracker 可以配置多个，格式：tracker_server=host:port，host 可以是主机名或 IP 地址，端口号必

填，如果有多个 tracker_server，可以配置多行 tracker_server=host:port。

10.2.6 storage 服务之间的同步参数

sync_wait_msec=50

同步等待时间，单位毫秒。同步文件时，如果从 binlog 中没有读到要同步的文件，休眠 N 毫秒后重新读取。0 表示不休眠，立即再次尝试读取。

出于 CPU 消耗考虑，不建议设置为 0。如果希望同步尽可能快一些，可以将本参数设置得小一些，比如设置为 10ms。

sync_interval=0

同步上一个文件后，再同步下一个文件的时间间隔，单位为毫秒，0 表示不休眠，直接同步下一个文件。

sync_start_time=00:00

同步开始时间，格式[小时:分钟]，小时取值：0-23，分钟取值：0-59。

sync_end_time=23:59

同步结束时间，格式[小时:分钟]，小时取值：0-23，分钟取值：0-59。sync_start_time 和 sync_end_time 共同设置允许系统同步的时间段 (默认是全天) 。一般用于避免高峰同步产生一些问题而设定。

file_sync_skip_invalid_record=false

文件同步的时候，是否忽略无效的 binlog 记录。

10.2.7 文件由内存到磁盘的同步参数

sync_binlog_buff_interval=10

同步 **binglog（更新操作日志）**到硬盘的时间间隔，单位为秒，默认值 60 秒。本参数会影响新上传文件同步延迟时间。【7.2.6】中的参数配合此参数一块使用。

write_mark_file_freq=500

同步完 N 个文件后，把 storage 的 **mark 文件**同步到磁盘，默认值 500。

sync_log_buff_interval=10

同步或刷新**日志信息**到硬盘的时间间隔，单位为秒，默认值 10 秒。storage server 的日志信息不是时时写硬盘的，而是先写内存。

sync_stat_file_interval=300

把 storage 的 **stat 文件**同步到磁盘的时间间隔，单位为秒，如果 stat 文件内容没有变化，不会进行同步，单位秒，默认值 300。

fsync_after_written_bytes=0

当写入**大文件**时，每写入 N 个字节，调用一次系统函数 fsync 将内容强行同步到硬盘。默认值为 0，表示从不调用 fsync。

10.2.8 日志文件参数

log_level=info

定义日志级别，区分大小写，日志级别由高到低有：

emerg：紧急，alert：警报，crit：严重，error：错误，warn：警告，notice：注意事项，info：信息，debug：调试。

log_file_keep_days = 0

日志文件保存的天数，默认为 0，表示不删除任何旧的日志。

########################## **access log 访问日志**##########################

use_access_log = false

是否将文件操作记录到 access log。

rotate_access_log = false

是否定期轮转 access log，目前仅支持一天轮转一次。

access_log_rotate_time=00:00

access log 定期轮转的时间点，只有当 rotate_access_log 设置为 true 时有效。

rotate_access_log_size = 0

access log 按文件大小轮转。

设置为 0 表示不按文件大小轮转，否则当 access log 达到该大小，就会轮转到新文件中。

########################## **error log 错误日志**##########################

rotate_error_log = false

是否定期轮转 error log，目前仅支持一天轮转一次。

error_log_rotate_time=00:00

error log 定期轮转的时间点，只有当 rotate_error_log 设置为 true 时有效。

rotate_error_log_size = 0

error log 按文件大小轮转。

设置为 0 表示不按文件大小轮转，否则当 error log 达到该大小，就会轮转到新文件中

10.2.9 其他参数

check_file_duplicate=0

是否检测上传文件已经存在。如果已经存在，则不存在文件内容，建立一个符号链接以节省磁盘空间。

这个应用要配合 FastDHT 使用，所以打开前要先安装 FastDHT

1 或 yes 是检测，0 或 no 是不检测。

file_signature_method=hash

文件去重时，文件内容的签名方式：hash 或 md5，默认 hash。

key_namespace=FastDFS

当 check_file_duplicate 的值为 1 或 yes 时，在 FastDHT 中的命名空间，此时 key_namespace 必须设置。

keep_alive=0

与 FastDHT servers 的连接方式 (是否为持久连接) ，默认是 0（短连接方式）。可以考虑使用长连接，这要看 FastDHT server 的连接数是否够用。

##include /home/yuqing/fastdht/conf/fdht_servers.conf

可以通过 #include filename 方式来加载 FastDHT servers 的配置。

同样要求 check_file_duplicate=1 时才有用，不然系统会忽略。

fdht_servers.conf 记载的是 FastDHT servers 列表。

########################## HTTP 服务配置部分##########################

http.domain_name=

storage server 上 web server 域名，通常仅针对单独部署的 web server。这样 URL 中就可以通过域名方式来访问 storage server 上的文件了，这个参数为空就是 IP 地址的方式。

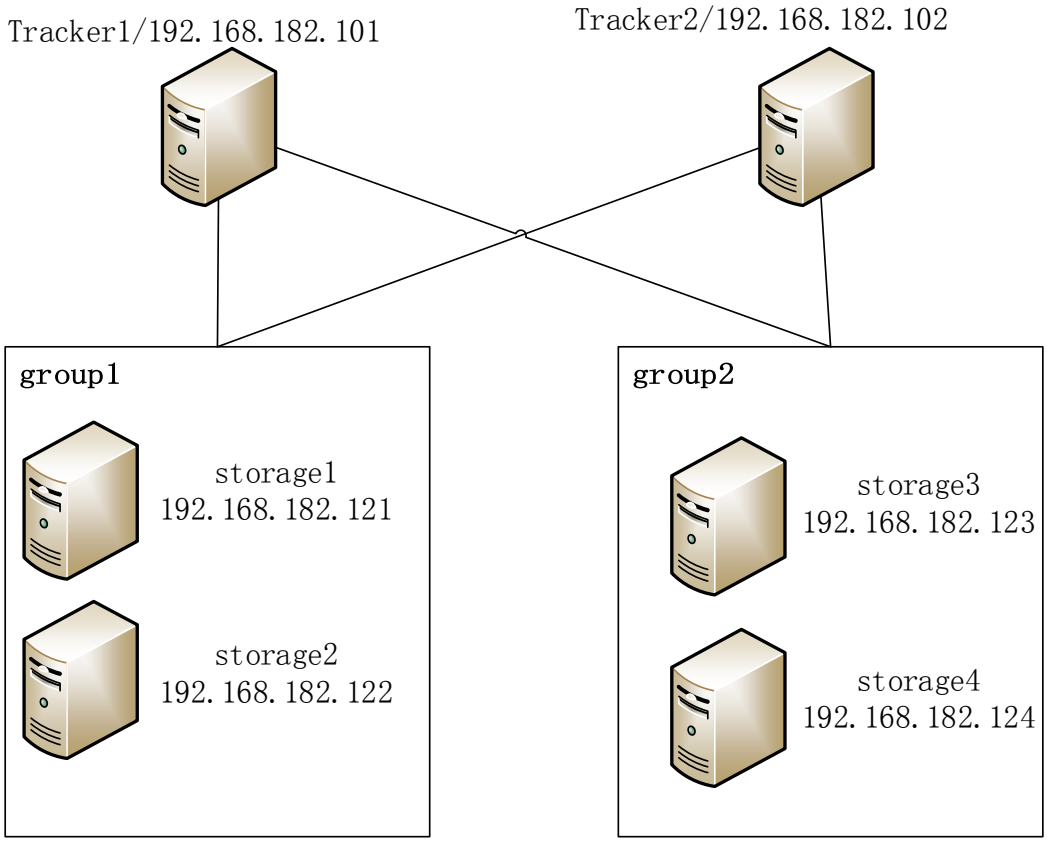
http.server_port=8888

http 的访问端口。

11 FastDFS 的安装

安装一个 FastDFS 的文件服务器集群

部署结构图：

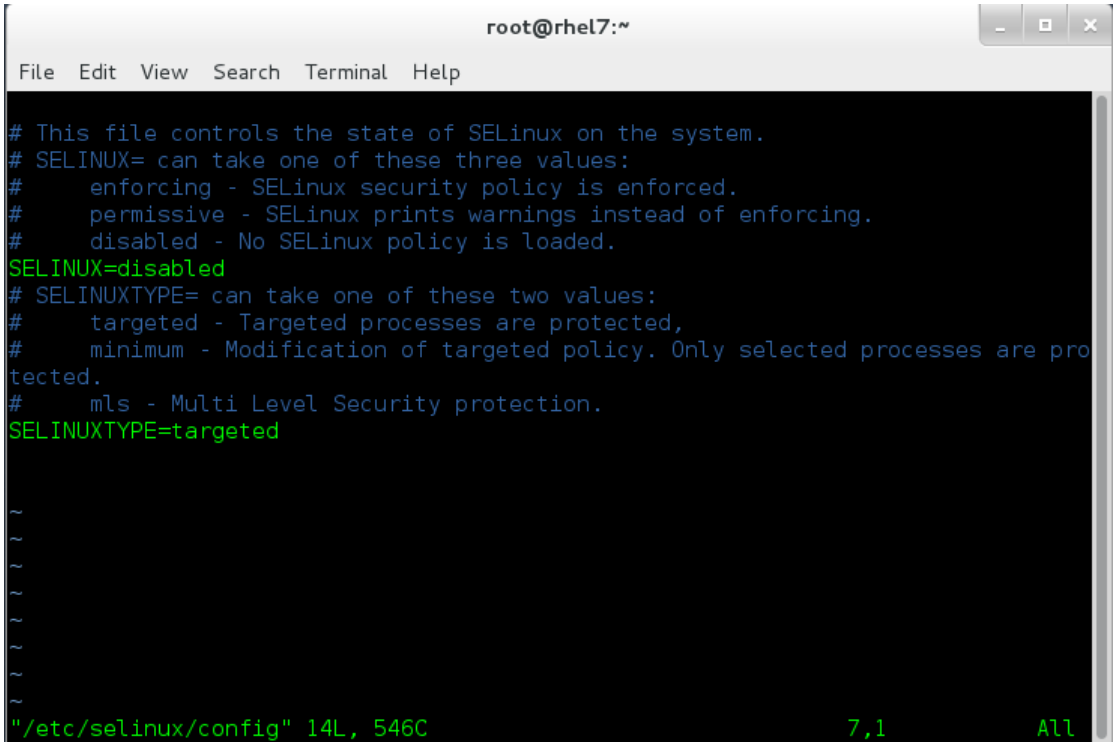


服务器操作系统为 RHEL7.0，详细信息如下：

主机名/IP 地址	卷（组）名	角色	备注	base_path
tracker1/192.168.182.101	-	跟踪器	管 理 group1 、group2	/tracker
tracker2/192.168.182.102	-	跟踪器	管 理 group1 、group2	/tracker
tracker3/192.168.182.103	-	跟踪器	用来演示 tracker 节点的添加、删除	
storage1/192.168.182.121	group1	存储节点		/storage
storage2/192.168.182.122	group1	存储节点		/storage
storage3/192.168.182.123	group2	存储节点		/storage
storage4/192.168.182.124	group2	存储节点		/storage
storage5/192.168.182.125	group1	存储节点	用来演示 storage 节点的添加、删除	/storage

安装步骤

1. 在所有服务器上关闭防火墙服务，并禁止开机自启动
   RHEL7、centOS7、OEL7 版本：
   systemctl stop firewalld.service --关闭防火墙服务
   systemctl disable firewalld.service --关闭开机自动启
   linux5、6 版本：
   service iptables stop --关闭防火墙服务
   chkconfig iptables off --关闭开机自动启
2. 在所有服务器上关闭 seLinux 服务
   vim /etc/selinux/config
   将 SELINUX 的改为 disabled



3. 在所有服务器上，安装 libfastcommon 软件包
   解压 libfastcommon-1.0.7.zip 包并将文件夹 cp 到/usr/local/src 目录下，执行命令
   cd libfastcommon-1.0.7
   ./make.sh
   ./make.sh install
4. 安装 FastDFS
   解压 FastDFS_v5.05.tar.gz 包并将文件夹 cp 到/usr/local/src 目录下，执行命令
   cd FastDFS
   ./make.sh
   ./make.sh install
5. 配置 Tracker（tracker1 和 tracker2 都做如下配置）
   FastDFS 配置文件的默认目录/etc/fdfs，操作如下：
   1） 进入配置文件目录，使用 sample 文件复制一份 tracker.conf 文件进行修改：
   cd /etc/fdfs
   cp tracker.conf.sample tracker.conf

2） 编辑 tracker.conf 配置文件，命令 vim tracker.conf，修改如下内容：

base_path=/tracker ##数据和日志的存放路径

store_lookup=2 ##存储组的选择方式，负载均衡方式，这是默认方式

store_server=0 ##存储节点的选择方式，轮询方式，这是默认方式

store_path=2 ##存储路径的选择方式，负载均衡方式

thread_stack_size = 128KB ##线程栈大小，默认 64K

storage_sync_file_max_delay=600 ##存储节点间文件同步的最大延迟时间，默认 1 天，此处测试改为 10 分钟

rotate_error_log = true ##启用错误日志

其他参数默认即可。

3） 启动 tracker 服务器：

fdfs_trackerd /etc/fdfs/tracker.conf start

如果报错，查看日志：

$base_path/logs/trackerd.log，此处配置的 base_path=/tracker，所以日志文件路径：/tracker/logs/trackerd.log

6. 配置 Storage

1） 进入配置文件目录，使用 sample 文件复制一份 storage.conf 文件进行修改：

cd /etc/fdfs

cp storage.conf.sample storage.conf

2） 编辑 storage.conf 配置文件，命令 vim storage.conf，修改如下内容：

storage1 和 storage2 的配置文件：

group_name=group1 ##卷名

base_path=/fastdfs ##基础路径，存放 storage 运行所需文件

sync_wait_msec=1000 ##同步检查间隔时间 1000ms

store_path0=/storage ##文件存放路径

tracker_server=192.168.182.101:22122 ##storage 注册到 tracker1

tracker_server=192.168.182.102:22122 ##storage 注册到 tracker2

use_access_log = true ##开启访问日志

storage3 和 storage4 的配置文件：

group_name=group2 ##卷名

base_path=/fastdfs ##基础路径，存放 storage 运行所需文件

sync_wait_msec=1000 ##同步检查间隔时间 1000ms

##store_path0=/storage ##如果不配置 store_path，那么文件默认存放在${base_path}/data 下 store_path0，实际生产环境中，尽量将所有节点的存储位置保持一致，方便管理。

tracker_server=192.168.182.101:22122 ##storage 注册到 tracker1

tracker_server=192.168.182.102:22122 ##storage 注册到 tracker2

use_access_log = true ##开启访问日志

3） 启动 storage 服务器：

在四个存储节点执行启动命令：

fdfs_storaged /etc/fdfs/storage.conf start

如果报错，查看日志：

$base_path/logs/storaged.log 此处配置的 base_path=/fastdfs，所以日志文件路径：/fastdfs/logs/storaged.log

7. 查看 FastDFS 文件服务器的状态

1） 在客户端安装 FastDFS（此处选择 tracker1 作为客户端）。

2） 配置 client.conf 文件

cp client.conf.sample client.conf

vim client.conf

client 的配置文件：

base_path=/client　　##配置 client 的基本目录

tracker_server=192.168.182.101:22122　##配置 tracker1 服务的地址

tracker_server=192.168.182.102:22122　##配置 tracker2 服务的地址

3） 使用 fdfs_monitor 命令查看 FastDFS 文件服务器的状态：

fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list

fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.102 list

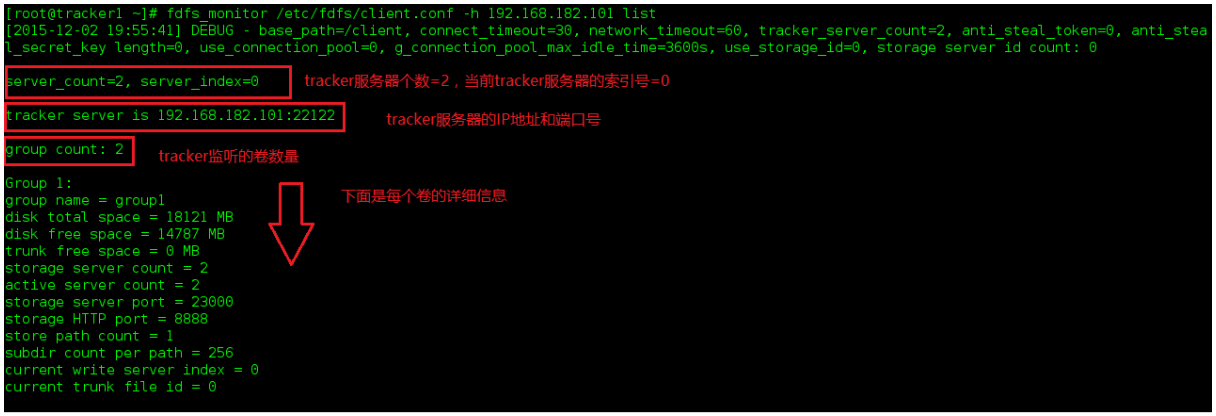
参数说明：

/etc/fdfs/client.conf：客户端配置文件

-h 192.168.182.101/-h 192.168.182.102：tracker1/tracker2 服务器的 ip 地址

list：显示指定的 tracker 的服务器信息

以 tracker1 192.168.182.101 为例，查询结果如下：



```
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.182.121
                ip_addr = 192.168.182.121  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-02 17:54:29
                up time = 2015-12-02 17:54:29
                total storage = 18121 MB
                free storage = 14787 MB
                upload priority = 10
                store_path_count = 1
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id =
                if_trunk_server = 0
                connection.alloc_count = 256
                connection.current_count = 1
                connection.max_count = 1
                total_upload_count = 0
                success_upload_count = 0
                total_append_count = 0
                success_append_count = 0
```

```
        Storage 2:
                id = 192.168.182.122
                ip_addr = 192.168.182.122  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-02 17:54:43
                up time = 2015-12-02 17:54:43
                total storage = 18121 MB
                free storage = 14787 MB
                upload priority = 10
                store_path_count = 1
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id = 192.168.182.121
                if_trunk_server = 0
                connection.alloc_count = 256
                connection.current_count = 1
```

```
Group 2:
group name = group2
disk total space = 18121 MB
disk free space = 14787 MB
trunk free space = 0 MB
storage server count = 2
active server count = 2
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.182.123
                ip_addr = 192.168.182.123  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-02 17:54:48
                up time = 2015-12-02 17:54:48
                total storage = 18121 MB
```

```
        Storage 2:
                id = 192.168.182.124
                ip_addr = 192.168.182.124  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-02 17:54:51
                up time = 2015-12-02 17:54:51
                total storage = 18121 MB
                free storage = 14787 MB
                upload priority = 10
                store_path_count = 1
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id = 192.168.182.123
                if_trunk_server = 0
                connection.alloc_count = 256
                connection.current_count = 1
                connection.max_count = 1
                total_upload_count = 0
                success_upload_count = 0
                total_append_count = 0
```

如果 tracker1 和 tracker2 的信息都能查看到 group1 和 group2 的信息，说明部署成功，如果部署过程中出现问题，需要参照日志文件中的信息进行修正。

12 文件的上传

上传文件命令：fdfs_upload_file

语法：

fdfs_upload_file <config_file> <local_filename> [storage_ip:port] [store_path_index]

<config_file>：客户端配置文件名称，必须有。

<local_filename>：要上传的文件名称，必须有。

[storage_ip:port]：上传的存储节点名称，选择项。

[store_path_index]：存储路径，选择项。

示例,上传 testupload 文件：

```
[root@tracker1 ~]# fdfs_upload_file /etc/fdfs/client.conf testupload
group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
[root@tracker1 ~]#
```

返回值文件 ID:【group 名+节点上存储目录编号+storage 节点创建的目录+加密的一个文件名】,根据文件 ID，可以查看文件信息，使用 fdfs_file_info 命令，示例：

```
[root@tracker1 ~]# fdfs_file_info /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
source storage id: 0
source ip address: 192.168.182.121
file create timestamp: 2015-12-02 21:18:35
file size: 62
file crc32: 1868958003 (0x6F660933)
[root@tracker1 ~]#
```

从返回结果中可以知道：

文件上传位置：192.168.182.121 服务器，存储节点 ID：0

创建时间：2015-12-02:21:18:35

文件大小：62bytes

crc32 校验码：1868958003，fdfs_crc32 命令可以直接查看文件的 crc32 校验码，例如：

```
[root@tracker1 ~]# fdfs_crc32 testupload
1868958003
```

文件上传前和上传后的 crc32 校验码一致。

13 文件的下载

下载文件命令：fdfs_download_file

语法：

fdfs_download_file <config_file> <file_id> [local_filename] [<download_offset> <download_bytes>]

<config_file> ：客户端配置文件，必须有

<file_id> ：要下载的文件 ID，必须有。

[local_filename] ：存储到本地的文件，选择项。

[<download_offset> <download_bytes>]：下载偏移量、下载大小，选择项。

示例：

fdfs_download_file /etc/fdfs/client.conf

group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578 /root/testdownload

```
[root@tracker1 ~]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578 /root/testdownload
[root@tracker1 ~]# ll
total 368
-rw-------.  1 root root    4273 Dec  1 13:25 anaconda-ks.cfg
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Desktop
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Documents
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Downloads
drwxr-xr-x. 10 8980 users   4096 Dec  1 19:52 FastDFS
-rwxr-xr-x.  1 root root  345400 Dec  1 19:51 FastDFS_v5.05.tar.gz
-rw-r--r--.  1 root root    4284 Dec  1 05:38 initial-setup-ks.cfg
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Music
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Pictures
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Public
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Templates
-rw-r--r--   1 root root      62 Dec  2 21:42 testdownload
-rw-r--r--   1 root root      62 Dec  2 21:18 testupload
drwxr-xr-x.  2 root root       6 Dec  1 05:39 Videos
```

测试 crc32 校验码，显示正确无误：

```
[root@tracker1 ~]# fdfs_crc32 testdownload
1868958003
[root@tracker1 ~]# fdfs_crc32 testupload
1868958003
[root@tracker1 ~]#
```

14 文件的删除

文件删除命令：fdfs_delete_file

语法：

fdfs_delete_file <config_file> <file_id>

<config_file> ：客户端配置文件，必须有。

<file_id>：文件 ID，必须有。

示例：

```
[root@tracker1 ~]# fdfs_delete_file /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
[root@tracker1 ~]# fdfs_file_info /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
source storage id: 0
source ip address: 192.168.182.121
file create timestamp: 2015-12-02 21:18:35
file size: 62
file crc32: 1868958003 (0x6F660933)
```

删除完成后，查询文件信息还能查看，但是下载时会提示文件不存在：

```
[root@tracker1 ~]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
[2015-12-02 21:57:45] ERROR - file: tracker_proto.c, line: 48, server: 192.168.182.121:23000, response status 2 != 0
download file fail, error no: 2, error info: No such file or directory
[root@tracker1 ~]# fdfs_file_info /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
source storage id: 0
source ip address: 192.168.182.121
file create timestamp: 2015-12-02 21:18:35
file size: 62
file crc32: 1868958003 (0x6F660933)
[root@tracker1 ~]#
```

到服务器存储节点的路径下查看，文件也已经删除：

```
[root@storage1 00]# pwd
/storage/data/00
[root@storage1 00]# cd 00
[root@storage1 00]# ll
total 4
-rw-r--r-- 1 root root 62 Dec  2 21:18 wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
[root@storage1 00]# ll
total 0
[root@storage1 00]# pwd
/storage/data/00/00
[root@storage1 00]# ll
total 0
[root@storage1 00]#
```

15 FastDFS Tracker 节点维护

tracker 的添加和删除比较麻烦，不光要修改 tracker 的配置文件，原有的存储节点配置文件也要修改。**所有 tracker 之间是对等的。**

15.1.1 添加 Tracker 节点

添加一个节点 tracker3，tracker3 只监控 group2，查看 tracker3 的状态并通过 tracker3 上传文件。

tracker3 主机的信息：

```
File  Edit  View  Search  Terminal  Help
[root@tracker3 Desktop]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.182.103  netmask 255.255.255.0  broadcast 192.168.182.255
```

参照 FastDFS 的安装步骤安装 FastDFS 并修改 client.conf 配置文件(将 tracker1 或 tracker2 上的 client.conf 直接 scp 到 tracker3)，启动 tracker3 服务。

```
[root@tracker3 tracker]# fdfs_trackerd /etc/fdfs/tracker.conf start
[root@tracker3 tracker]# ps -ef |grep fdfs
root       3369      1  0 23:45 ?        00:00:00 fdfs_trackerd /etc/fdfs/tracke
r.conf start
root       3379   3224  0 23:45 pts/1    00:00:00 grep --color=auto fdfs
[root@tracker3 tracker]#
```

停止 group2 中 storage3，storage4 的存储服务：

```
[root@storage3 fdfs]# fdfs_storaged /etc/fdfs/storage.conf stop
waiting for pid [3950] exit ...
pid [3950] exit.
[root@storage3 fdfs]#
```

```
[root@storage4 fdfs]# fdfs_storaged /etc/fdfs/storage.conf stop
waiting for pid [3944] exit ...
pid [3944] exit.
[root@storage4 fdfs]#
```

修改 storage3，storage4 的配置文件 storage.conf，添加 tracker3 的配置内容并保存：

```
# tracker_server can ocur more than once, and tracker_server format is
#  "host:port", host can be hostname or ip address
tracker_server=192.168.182.101:22122
tracker_server=192.168.182.102:22122
tracker_server=192.168.182.103:22122
#standard log level as syslog, case insensitive, value list:
### emerg for emergency
```

重启 storage3，storage4 节点，并在客户端配置文件中添加 tracker3 的信息：

```
# tracker_server can ocur more than once, and tracker_server format is
#  "host:port", host can be hostname or ip address
tracker_server=192.168.182.101:22122
tracker_server=192.168.182.102:22122
tracker_server=193.168.182.103:22122
#standard log level as syslog, case insensitive, value list:
```

在客户端查看 tracker3 服务器的状态信息：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list
[2015-12-03 00:52:14] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=3, 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=3, server_index=2

tracker server is 192.168.182.103:22122

group count: 2

Group 1:
group name = group1
disk total space = 0 MB
disk free space = 0 MB
trunk free space = 0 MB
storage server count = 2
active server count = 0
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0
```

发现即使在 storage1 和 storage2 中没有配置 tracker3 的信息，在 tracker3 启动后，tracker 在启动后也能管理所有的 group，测试上传文件：

```
[root@tracker1 test]# fdfs_upload_file  /etc/fdfs/client.conf file11 192.168.182.123:23000
group2/M00/00/00/wKi2e1ZgA9WAWBT6AAAAF849XuM1135897
[root@tracker1 test]# fdfs_upload_file  /etc/fdfs/client.conf file11 192.168.182.121:23000
group1/M00/00/00/wKi2eVZgA9-ARbzUAAAAF849XuM1814323
[root@tracker1 test]#
```

查看服务器存储目录下内容：

storage3 和 storage4 中的目录全部存有上传后的 file11：

```
[root@storage3 data]# cd 00/00
[root@storage3 00]# ll wKi2e1ZgA9WAWBT6AAAAF849XuM1135897
-rw-r--r-- 1 root root 23 Dec  3 00:56 wKi2e1ZgA9WAWBT6AAAAF849XuM1135897
[root@storage3 00]# pwd
/fastdfs/data/00/00
[root@storage3 00]#
```

```
[root@storage4 data]# ls
00  12  24  36  48  5A  6C  7E  90  A2  B4  C6  D8  EA  FC
01  13  25  37  49  5B  6D  7F  91  A3  B5  C7  D9  EB  FD
02  14  26  38  4A  5C  6E  80  92  A4  B6  C8  DA  EC  fdfs_storaged.pid
03  15  27  39  4B  5D  6F  81  93  A5  B7  C9  DB  ED  FE
04  16  28  3A  4C  5E  70  82  94  A6  B8  CA  DC  EE  FF
05  17  29  3B  4D  5F  71  83  95  A7  B9  CB  DD  EF  storage_stat.dat
06  18  2A  3C  4E  60  72  84  96  A8  BA  CC  DE  F0  sync
07  19  2B  3D  4F  61  73  85  97  A9  BB  CD  DF  F1
08  1A  2C  3E  50  62  74  86  98  AA  BC  CE  E0  F2
09  1B  2D  3F  51  63  75  87  99  AB  BD  CF  E1  F3
0A  1C  2E  40  52  64  76  88  9A  AC  BE  D0  E2  F4
0B  1D  2F  41  53  65  77  89  9B  AD  BF  D1  E3  F5
0C  1E  30  42  54  66  78  8A  9C  AE  C0  D2  E4  F6
0D  1F  31  43  55  67  79  8B  9D  AF  C1  D3  E5  F7
0E  20  32  44  56  68  7A  8C  9E  B0  C2  D4  E6  F8
0F  21  33  45  57  69  7B  8D  9F  B1  C3  D5  E7  F9
10  22  34  46  58  6A  7C  8E  A0  B2  C4  D6  E8  FA
11  23  35  47  59  6B  7D  8F  A1  B3  C5  D7  E9  FB
[root@storage4 data]# cd 00/00
[root@storage4 00]# ll wKi2e1ZgA9WAWBT6AAAAF849XuM1135897
-rw-r--r-- 1 root root 23 Dec  3 00:56 wKi2e1ZgA9WAWBT6AAAAF849XuM1135897
[root@storage4 00]# pwd
/fastdfs/data/00/00
[root@storage4 00]#
```

storage1 和 storage2 中的目录全部存有上传后的 file11：

```
[root@storage1 00]# ll wKi2eVZgA9-ARbzUAAAAF849XuM1814323
-rw-r--r-- 1 root root 23 Dec  3 00:57 wKi2eVZgA9-ARbzUAAAAF849XuM1814323
[root@storage1 00]# pwd
/storage/data/00/00
[root@storage1 00]#
```

```
[root@storage2 00]# ll wKi2eVZgA9-ARbzUAAAAF849XuM1814323
-rw-r--r--. 1 root root 23 Dec  3 00:57 wKi2eVZgA9-ARbzUAAAAF849XuM1814323
[root@storage2 00]# pwd
/storage/data/00/00
[root@storage2 00]#
```

删除客户端配置文件中的 tracker1 和 tracker2 的配置，只保留 tracker3 的配置：

```
# tracker_server can ocur more than once, and tracker_server format is
#  "host:port", host can be hostname or ip address
#tracker_server=192.168.182.101:22122
#tracker_server=192.168.182.102:22122
tracker_server=192.168.182.103:22122
#standard log level as syslog, case insensitive, value list:
### emerg for emergency
### alert
### crit for critical
```

尝试将文件 file11 上传到 storage1 节点，仍然成功：

```
[root@tracker1 test]# fdfs_upload_file  /etc/fdfs/client.conf file11 192.168.182.121:23000
group1/M00/00/00/wKi2eVZgBdKAV8l9AAAAF849XuM4710938
[root@tracker1 test]#
```

查看 storage1 存储目录下文件：

```
[root@storage1 00]# ll wKi2eVZgBdKAV8l9AAAAF849XuM4710938
-rw-r--r-- 1 root root 23 Dec  3 01:05 wKi2eVZgBdKAV8l9AAAAF849XuM4710938
[root@storage1 00]# pwd
/storage/data/00/00
[root@storage1 00]#
```

从添加 tracker3 节点实验得出结论：即使 tracker3 只有 group2 向其注册，tracker3 启动后，和其他的 tracker 节点是等效的，也能向其他节点监听的 group 中上传数据，即使其他节点停机。

15.1.2 删除 Tracker 节点

1. 将各个 storage 节点配置文件中的 tracker_server=IP:PORT 删除，重启 storage。这样 storage 节点就不再像 tracker 推送信息。
2. 停止 tracker 节点服务。
3. 删除客户端配置文件中的 tracker_server=IP:PORT

示例

1. 查看 tracker3（192.168.182.103）监控的信息：

   fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list | grep -iE 'group|storage|ip_addr'

```
[root@storage1 00]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list | grep -iE 'group|storage|ip_addr'
[2015-12-03 17:05:51] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=3, anti_steal_token=0, anti_stea
l_secret_key length=0, use_connection_pool=0, g_connection_pool_max_idle_time=3600s, use_storage_id=0, storage server id count: 0

group count: 2
Group 1:
group name = group1
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.121  ACTIVE
                total storage = 18121 MB
                free storage = 14786 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.122  ACTIVE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.121
```

```
Group 2:
group name = group2
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.123  ACTIVE
                total storage = 18121 MB
                free storage = 14786 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.124  ACTIVE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.123
[root@storage1 00]#
```

   可以看出 tracker3 节点监控 4 个存储节点 4 个节点都是 ACTIVE 状态。

2. 修改 4 个节点的/etc/fdfs/storage.conf 文件，删除下图中高亮部分的内容：

```
# subdir_count  * subdir_count directories will be auto created under each
# store_path (disk), value can be 1 to 256, default value is 256
subdir_count_per_path=256

# tracker_server can ocur more than once, and tracker_server format is
#  "host:port", host can be hostname or ip address
tracker_server=192.168.182.101:22122
tracker_server=192.168.182.102:22122
tracker_server=192.168.182.103:22122
#standard log level as syslog, case insensitive, value list:
### emerg for emergency
### alert
```

然后重启四个 storage 服务器，命令：fdfs_storaged /etc/fdfs/storage.conf restart，此时查看 storage 节点在 tracker3 中的状态，发现都是 OFFLINE：

```
[root@tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list | grep -iE 'group|storage|ip_addr'
[2015-12-03 17:18:52] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=3, 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

group count: 2
Group 1:
group name = group1
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.121  OFFLINE
                total storage = 18121 MB
                free storage = 14786 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.122  OFFLINE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.121
Group 2:
group name = group2
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.123  OFFLINE
                total storage = 18121 MB
                free storage = 14784 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.124  OFFLINE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.123
```

在 tracker1 和 tracker2 都是 ACTIVE 状态：

```
[root@tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.102 list | grep -iE 'group|storage|ip_addr'
[2015-12-03 17:20:03] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=3, 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

group count: 2
Group 1:
group name = group1
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.121  ACTIVE
                total storage = 18121 MB
                free storage = 14786 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.122  ACTIVE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.121
```

```
Group 2:
group name = group2
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.123  ACTIVE
                total storage = 18121 MB
                free storage = 14784 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 2:
                ip_addr = 192.168.182.124  ACTIVE
                total storage = 18121 MB
                free storage = 14787 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.123
```

3. 停止 tracker3 服务

```
[root@tracker3 Desktop]# fdfs_trackerd /etc/fdfs/tracker.conf stop
waiting for pid [3170] exit ...
pid [3170] exit.
[root@tracker3 Desktop]#
```

再次查看 tracker3 的状态：

```
[root@tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list | grep -iE 'group|storage|ip_addr|tracker'
[2015-12-03 17:24:41] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=3, anti_steal_token=0, anti_stea
l_secret_key length=0, use_connection_pool=0, g_connection_pool_max_idle_time=3600s, use_storage_id=0, storage server id count: 0

[2015-12-03 17:24:41] ERROR - file: connection_pool.c, line: 110, connect to 192.168.182.103:22122 fail, errno: 111, error info: Connection re
fused
tracker server is 192.168.182.101:22122
group count: 2
Group 1:
group name = group1
storage server count = 2
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.121  ACTIVE
                total storage = 18121 MB
                free storage = 14784 MB
```

发现 tracker3 已经拒绝连接，FastDFS 自动把 tracker1 的监听信息给返回。

删除客户端文件中的 tracker3 配置信息，即高亮内容：

```
# tracker_server can ocur more than once, and tracker_server format is
#  "host:port", host can be hostname or ip address
tracker_server=192.168.182.101:22122
tracker_server=192.168.182.102:22122
tracker_server=192.168.182.103:22122
#standard log level as syslog, case insensitive, value list:
```

再次查看 tracker3 的信息：

```
[root@tracker1 fdfs]# vim /etc/fdfs/client.conf
[root@tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.103 list | grep -iE 'group|storage|ip_addr|tracker'
[2015-12-03 17:28:16] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_secret_key length=0, use_connection_pool=0, g_connection_pool_max_idle_time=3600s, use_storage_id=0, storage server id count: 0

tracker server: 192.168.182.103 not exists!
[root@tracker1 fdfs]#
```

系统提示 tracer3 服务器找不到，表示 tracker3 删除成功。

16 FastDFS Storage 节点维护

16.1.1 添加 Storage 节点

在 group1 中多添加一个节点 storage5，查看文件同步情况：

环境准备：

准备 10 个文件 file1-file10，将文件上传到服务器，返回了 10 个文件的 ID 信息

```
[root@tracker1 test]# vim upload.sh
[root@tracker1 test]# ll
total 44
-rw-r--r-- 1 root root  34 Dec  2 22:09 file1
-rw-r--r-- 1 root root  51 Dec  2 22:11 file10
-rw-r--r-- 1 root root  48 Dec  2 22:09 file2
-rw-r--r-- 1 root root  44 Dec  2 22:09 file3
-rw-r--r-- 1 root root  44 Dec  2 22:10 file4
-rw-r--r-- 1 root root  10 Dec  2 22:10 file5
-rw-r--r-- 1 root root  55 Dec  2 22:10 file6
-rw-r--r-- 1 root root  46 Dec  2 22:10 file7
-rw-r--r-- 1 root root  49 Dec  2 22:10 file8
-rw-r--r-- 1 root root  52 Dec  2 22:11 file9
-rw-r--r-- 1 root root 451 Dec  2 22:15 upload.sh
[root@tracker1 test]# chmod 775 upload.sh
[root@tracker1 test]# ./upload.sh
group1/M00/00/00/wKi2eVZf3iCAApahAAAAIqByaB82288999
group1/M00/00/00/wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080
group1/M00/00/00/wKi2eVZf3iCABsdQAAAALEUe5vw2190367
group1/M00/00/00/wKi2elZf3iCANXmmAAAALPpsm903898002
group1/M00/00/00/wKi2eVZf3iCAHldHAAAACrhx9-U5734081
group1/M00/00/00/wKi2elZf3iCAXUQ5AAAANzBubzw6176160
group1/M00/00/00/wKi2eVZf3iCAT7R6AAAALsXob183732522
group1/M00/00/00/wKi2elZf3iCATqahAAAAMSE-2l80296494
group1/M00/00/00/wKi2eVZf3iCANopJAAAANBLU12g2152289
group1/M00/00/00/wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132
[root@tracker1 test]#
```

检查 storage1、storage2 服务器存储节点文件：

```
[root@storage1 00]# pwd
/storage/data/00/00
[root@storage1 00]# ll
total 44
-rw-r--r-- 1 root root 34 Dec  2 22:14 wKi2elZf3b-AXC3yAAAAIqByaB80309930
-rw-r--r-- 1 root root 48 Dec  2 22:16 wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080
-rw-r--r-- 1 root root 51 Dec  2 22:16 wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132
-rw-r--r-- 1 root root 44 Dec  2 22:16 wKi2elZf3iCANXmmAAAALPpsm903898002
-rw-r--r-- 1 root root 49 Dec  2 22:16 wKi2elZf3iCATqahAAAAMSE-2l80296494
-rw-r--r-- 1 root root 55 Dec  2 22:16 wKi2elZf3iCAXUQ5AAAANzBubzw6176160
-rw-r--r-- 1 root root 34 Dec  2 22:16 wKi2eVZf3iCAApahAAAAIqByaB82288999
-rw-r--r-- 1 root root 44 Dec  2 22:16 wKi2eVZf3iCABsdQAAAALEUe5vw2190367
-rw-r--r-- 1 root root 10 Dec  2 22:16 wKi2eVZf3iCAHldHAAAACrhx9-U5734081
-rw-r--r-- 1 root root 52 Dec  2 22:16 wKi2eVZf3iCANopJAAAANBLU12g2152289
-rw-r--r-- 1 root root 46 Dec  2 22:16 wKi2eVZf3iCAT7R6AAAALsXob183732522
[root@storage1 00]#
```

```
[root@storage2 00]# pwd
/storage/data/00/00
[root@storage2 00]# ll
total 44
-rw-r--r--. 1 root root 34 Dec  2 22:14 wKi2elZf3b-AXC3yAAAAIqByaB80309930
-rw-r--r--. 1 root root 48 Dec  2 22:16 wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080
-rw-r--r--. 1 root root 51 Dec  2 22:16 wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132
-rw-r--r--. 1 root root 44 Dec  2 22:16 wKi2elZf3iCANXmmAAAALPpsm903898002
-rw-r--r--. 1 root root 49 Dec  2 22:16 wKi2elZf3iCATqahAAAAMSE-2l80296494
-rw-r--r--. 1 root root 55 Dec  2 22:16 wKi2elZf3iCAXUQ5AAAANzBubzw6176160
-rw-r--r--. 1 root root 34 Dec  2 22:16 wKi2eVZf3iCAApahAAAAIqByaB82288999
-rw-r--r--. 1 root root 44 Dec  2 22:16 wKi2eVZf3iCABsdQAAAALEUe5vw2190367
-rw-r--r--. 1 root root 10 Dec  2 22:16 wKi2eVZf3iCAHldHAAAACrhx9-U5734081
-rw-r--r--. 1 root root 52 Dec  2 22:16 wKi2eVZf3iCANopJAAAANBLU12g2152289
-rw-r--r--. 1 root root 46 Dec  2 22:16 wKi2eVZf3iCAT7R6AAAALsXob183732522
[root@storage2 00]#
```

storage1 和 storage 2 上的文件全部存在，冗余完成，现在开始添加节点 storage5，并将其注册到 tracke1 和 tracker2，添加完成后，如果 storage5 上的数据能自动冗余完成，说明 storage5 添加成功。

添加 storage5：

参照 FastDFS 的安装步骤安装 FastDFS 并修改配置文件，然后启动 FastDFS 即可。

因为是测试环境，所以将 storage5 配置文件中的#store_path0=/storage 注释掉，文件会默认存放到 base_path 下的 data 目录，如下：

```
[root@storage5 data]# pwd
/fastdfs/data
[root@storage5 data]# ls
00  12  24  36  48  5A  6C  7E  90  A2  B4  C6  D8  EA  FC
01  13  25  37  49  5B  6D  7F  91  A3  B5  C7  D9  EB  FD
02  14  26  38  4A  5C  6E  80  92  A4  B6  C8  DA  EC  fdfs_storaged.pid
03  15  27  39  4B  5D  6F  81  93  A5  B7  C9  DB  ED  FE
04  16  28  3A  4C  5E  70  82  94  A6  B8  CA  DC  EE  FF
05  17  29  3B  4D  5F  71  83  95  A7  B9  CB  DD  EF  storage_stat.dat
06  18  2A  3C  4E  60  72  84  96  A8  BA  CC  DE  F0  sync
07  19  2B  3D  4F  61  73  85  97  A9  BB  CD  DF  F1
08  1A  2C  3E  50  62  74  86  98  AA  BC  CE  E0  F2
09  1B  2D  3F  51  63  75  87  99  AB  BD  CF  E1  F3
0A  1C  2E  40  52  64  76  88  9A  AC  BE  D0  E2  F4
0B  1D  2F  41  53  65  77  89  9B  AD  BF  D1  E3  F5
0C  1E  30  42  54  66  78  8A  9C  AE  C0  D2  E4  F6
0D  1F  31  43  55  67  79  8B  9D  AF  C1  D3  E5  F7
0E  20  32  44  56  68  7A  8C  9E  B0  C2  D4  E6  F8
0F  21  33  45  57  69  7B  8D  9F  B1  C3  D5  E7  F9
10  22  34  46  58  6A  7C  8E  A0  B2  C4  D6  E8  FA
11  23  35  47  59  6B  7D  8F  A1  B3  C5  D7  E9  FB
```

查看 00/00 目录下的文件信息：

```
[root@storage5 data]# cd 00/00
[root@storage5 00]# ll
total 44
-rw-r--r-- 1 root root 34 Dec  2 22:14 wKi2elZf3b-AXC3yAAAAIqByaB80309930
-rw-r--r-- 1 root root 48 Dec  2 22:16 wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080
-rw-r--r-- 1 root root 51 Dec  2 22:16 wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132
-rw-r--r-- 1 root root 44 Dec  2 22:16 wKi2elZf3iCANXmmAAAALPpsm903898002
-rw-r--r-- 1 root root 49 Dec  2 22:16 wKi2elZf3iCATqahAAAAMSE-2l80296494
-rw-r--r-- 1 root root 55 Dec  2 22:16 wKi2elZf3iCAXUQ5AAAANzBubzw6176160
-rw-r--r-- 1 root root 34 Dec  2 22:16 wKi2eVZf3iCAApahAAAAIqByaB82288999
-rw-r--r-- 1 root root 44 Dec  2 22:16 wKi2eVZf3iCABsdQAAAALEUe5vw2190367
-rw-r--r-- 1 root root 10 Dec  2 22:16 wKi2eVZf3iCAHldHAAAACrhx9-U5734081
-rw-r--r-- 1 root root 52 Dec  2 22:16 wKi2eVZf3iCANopJAAAANBLU12g2152289
-rw-r--r-- 1 root root 46 Dec  2 22:16 wKi2eVZf3iCAT7R6AAAALsXob183732522
[root@storage5 00]# pwd
/fastdfs/data/00/00
[root@storage5 00]#
```

文件同步完成，查看 tracker1（192.168.182.101）：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list
[2015-12-02 22:39:44] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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=0

tracker server is 192.168.182.101:22122

group count: 2

Group 1:
group name = group1
disk total space = 18121 MB
disk free space = 14787 MB
trunk free space = 0 MB
storage server count = 3
active server count = 3
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0
```

group1中有3个storage，当前活动的是3个，说明添加存储节点添加成功。

tracker2（192.168.182.102）的信息：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.102 list
[2015-12-02 22:44:18] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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 192.168.182.102:22122

group count: 2

Group 1:
group name = group1
disk total space = 18121 MB
disk free space = 14787 MB
trunk free space = 0 MB
storage server count = 3
active server count = 3
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0
```

group1 中 storage3 节点的信息就是刚添加的 192.168.182.125 存储节点：

```
                success_file_read_count = 1
                total_file_write_count = 12
                success_file_write_count = 12
                last_heart_beat_time = 2015-12-02 22:39:17
                last_source_update = 2015-12-02 22:15:59
                last_sync_update = 2015-12-02 22:16:01
                last_synced_timestamp = 2015-12-02 22:16:00 (0s delay)
        Storage 3:
                id = 192.168.182.125
                ip_addr = 192.168.182.125  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-02 22:29:54
                up time = 2015-12-02 22:29:54
                total storage = 18121 MB
                free storage = 14787 MB
                upload priority = 10
                store_path_count = 1
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id = 192.168.182.121
                if_trunk_server = 0
                connection.alloc_count = 256
                connection.current_count = 2
```

16.1.2 删除 Storage 节点

以删除 storage5 节点为例：

1. 停止 storage5 节点的服务

```
[root@storage5 00]# fdfs_storaged /etc/fdfs/storage.conf stop
waiting for pid [6659] exit ...
pid [6659] exit.
[root@storage5 00]#
```

2. fdfs_monitor 命令删除节点 5 的服务：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.102 delete group1 192.168.182.125
[2015-12-02 22:53:29] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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 192.168.182.102:22122

delete storage server group1::192.168.182.125 success
[root@tracker1 test]# a
```

查看 tracker 信息：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list
[2015-12-02 22:58:05] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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=0

tracker server is 192.168.182.101:22122

group count: 2

Group 1:
group name = group1
disk total space = 18121 MB
disk free space = 14787 MB
trunk free space = 0 MB
storage server count = 3
active server count = 2
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
```

```
        last_sync_update = 2015-12-02 22:16:01
        last_synced_timestamp = 2015-12-02 22:16:00 (0s delay)
    Storage 3:
        id = 192.168.182.125
        ip_addr = 192.168.182.125  DELETED
        http domain =
        version = 5.05
        join time = 2015-12-02 22:29:54
        up time =
        total storage = 0 MB
        free storage = 0 MB
        upload priority = 0
        store path count = 0
```

storage5 的状态是 deleted，重启 tracker1 服务器：

```
[root@tracker1 test]# fdfs_trackerd /etc/fdfs/tracker.conf stop
waiting for pid [3463] exit ...
pid [3463] exit.
[root@tracker1 test]# fdfs_trackerd /etc/fdfs/tracker.conf start
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list
[2015-12-02 22:59:14] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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=0

tracker server is 192.168.182.101:22122

group count: 2

Group 1:
group name = group1
disk total space = 0 MB
disk free space = 0 MB
trunk free space = 0 MB
storage server count = 2
active server count = 0
storage server port = 23000
storage HTTP port = 8888
```

重启 tracker1 后看到 active server count=0，没有活动的 storage，等待一个心跳的时间，storage 将信息推送到 tracker1 后，再次查看 tracker1 信息，可以看到 active server count=2：

```
[root@tracker1 test]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list
[2015-12-02 22:59:43] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_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=0

tracker server is 192.168.182.101:22122

group count: 2

Group 1:
group name = group1
disk total space = 18121 MB
disk free space = 14787 MB
trunk free space = 0 MB
storage server count = 2
active server count = 2
storage server port = 23000
storage HTTP port = 8888
store path count = 1
subdir count per path = 256
current write server index = 0
current trunk file id = 0
```

同理重启 tracker2，即可完成 storage5 的删除。

这种情况下，如果 storage5 中的 FastDFS 安装信息还保留这，如果启动 storage 服务，storage5 会再次加入到 group1 中，所有要完全删除 storage5，还需要将 storage5 上安装的 FastDFS 卸载。

17 FastDFS 卸载

1. 到 FastDFS 的软件目录下，执行./make.sh clean:

```
[root@tracker3 FastDFS]# pwd
/root/FastDFS
[root@tracker3 FastDFS]# ./make.sh clean
rm -f ../common/fdfs_global.o tracker_proto.o tracker_mem.o tracker_service.o tracker_status.o tracker_global.o tracker_func.o fdfs_shared_fun
c.o tracker_nio.o tracker_relationship.o tracker_dump.o ../common/linux_stack_trace.o fdfs_trackerd
rm -f ../common/fdfs_global.o ../tracker/fdfs_shared_func.o ../tracker/tracker_proto.o tracker_client_thread.o storage_global.o storage_func.o
 storage_service.o storage_sync.o storage_nio.o storage_dio.o storage_ip_changed_dealer.o storage_param_getter.o storage_disk_recovery.o trunk
_mgr/trunk_mem.o trunk_mgr/trunk_shared.o trunk_mgr/trunk_sync.o trunk_mgr/trunk_client.o trunk_mgr/trunk_free_block_checker.o ../client/clien
t_global.o ../client/tracker_client.o ../client/storage_client.o ../client/client_func.o fdht_client/fdht_proto.o fdht_client/fdht_client.o fd
ht_client/fdht_func.o fdht_client/fdht_global.o storage_dump.o ../common/linux_stack_trace.o fdfs_storaged
rm -f ../common/fdfs_global.o ../common/fdfs_http_shared.o ../common/mime_file_parser.o ../tracker/tracker_proto.o ../tracker/fdfs_shared_func
.o ../storage/trunk_mgr/trunk_shared.o tracker_client.o client_func.o client_global.o storage_client.o ../common/fdfs_global.lo ../common/fdfs
_http_shared.lo ../common/mime_file_parser.lo ../tracker/tracker_proto.lo ../tracker/fdfs_shared_func.lo ../storage/trunk_mgr/trunk_shared.lo
tracker_client.lo client_func.lo client_global.lo storage_client.lo fdfs_monitor fdfs_test fdfs_test1 fdfs_crc32 fdfs_upload_file fdfs_downloa
d_file fdfs_delete_file fdfs_file_info fdfs_appender_test fdfs_appender_test1 fdfs_append_file fdfs_upload_appender libfdfsclient.a libfdfscli
ent.so
[root@tracker3 FastDFS]#
```

2. 删除 fastDFS 文件的如下目录：

1） tracker.conf/storage.conf/client.conf 配置文件中指定的 base 目录，**特别是删除 storage.conf 中指定的存储目录时要特别注意，上传的文件都在这个目录中，必训先确认数据确实不再使用了之后再删除，因为数据删除后找不回来。**

查看 tracker.conf 目录配置：

cat /etc/fdfs/tracker.conf |grep -i base_path

```
[root@tracker3 fdfs]# cat /etc/fdfs/tracker.conf |grep -i base_path
base_path=/tracker
[root@tracker3 fdfs]#
```

查看 storage.conf 目录配置：

cat /etc/fdfs/storage.conf |grep -iE 'base_path|store_path'

#storage.conf 可以更换为自己的配置文件名称。

```
[root@tracker3 fdfs]# cat /etc/fdfs/storage.conf |grep -iE 'base_path|store_path'
base_path=/fastdfs
store_path_count=1
# store_path#, based 0, if store_path0 not exists, it's value is base_path
#store_path0=/storage
#store_path1=/home/yuqing/fastdfs2
# store_path (disk), value can be 1 to 256, default value is 256
[root@tracker3 fdfs]#
```

查看 client.conf 目录配置：

cat /etc/fdfs/client.conf | grep -i base_path

```
[root@tracker1 fdfs]# vim client.conf
[root@tracker1 fdfs]# cat /etc/fdfs/client.conf | grep -i base_path
base_path=/client
```

2） 配置文件目录：rm -rf /etc/fdfs

3） 删除/bin 目录先相关的文件命令 rm -rf /etc/fdfs_*

4） 查找根目录下 fastDFS 相关的文件还有哪些，文件查找命令：find / |grep -iE 'fdfs|fastdfs'

一般会包含：

/usr/include/fastdfs

/etc/rc.d/init.d/fdfs_storaged

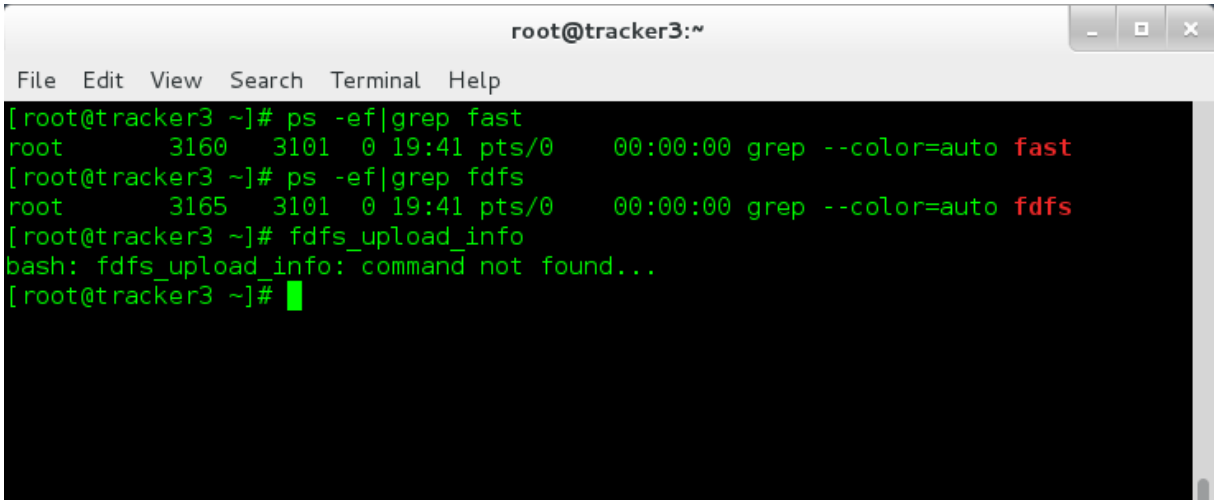
/etc/rc.d/init.d/fdfs_trackerd

/root/installFastDFS.sh

到此 FastDFS 的卸载完成。

使用如上的卸载方法是否可以将 FastDFS 卸载干净，以及是否会对以后再次在此机器上使用 FastDFS 造成干扰做了如下实验：

卸载 FastDFS 后重启服务器，查看进程信息，没有 fastDFS 进程相关信息，说明没有 FastDFS 的残留进程。



再次重新安装，不报错：

```
EVENT_USE_EPOLL -o libfdfsclient.so  -shared ../common/fdfs_global.lo ../common/
fdfs_http_shared.lo ../common/mime_file_parser.lo ../tracker/tracker_proto.lo ..
/tracker/fdfs_shared_func.lo ../storage/trunk_mgr/trunk_shared.lo tracker_client
.lo client_func.lo client_global.lo storage_client.lo -lpthread -ldl -rdynamic -
lfastcommon
[root@tracker3 FastDFS]# ./make.sh install
mkdir -p /usr/bin
mkdir -p /etc/fdfs
cp -f fdfs_trackerd /usr/bin
if [ ! -f /etc/fdfs/tracker.conf.sample ]; then cp -f ../conf/tracker.conf /etc/
fdfs/tracker.conf.sample; fi
mkdir -p /usr/bin
mkdir -p /etc/fdfs
cp -f fdfs_storaged  /usr/bin
if [ ! -f /etc/fdfs/storage.conf.sample ]; then cp -f ../conf/storage.conf /etc/
fdfs/storage.conf.sample; fi
mkdir -p /usr/bin
mkdir -p /etc/fdfs
mkdir -p /usr/lib64
cp -f fdfs_monitor fdfs_test fdfs_test1 fdfs_crc32 fdfs_upload_file fdfs_downloa
d_file fdfs_delete_file fdfs_file_info fdfs_appender_test fdfs_appender_test1 fd
fs_append_file fdfs_upload_appender /usr/bin
if [ 0 -eq 1 ]; then cp -f libfdfsclient.a /usr/lib64; fi
if [ 1 -eq 1 ]; then cp -f libfdfsclient.so /usr/lib64; fi
mkdir -p /usr/include/fastdfs
cp -f ../common/fdfs_define.h ../common/fdfs_global.h ../common/mime_file_parser
.h ../common/fdfs_http_shared.h ../tracker/tracker_types.h ../tracker/tracker_pr
oto.h ../tracker/fdfs_shared_func.h ../storage/trunk_mgr/trunk_shared.h tracker_
client.h storage_client.h storage_client1.h client_func.h client_global.h fdfs_c
lient.h /usr/include/fastdfs
if [ ! -f /etc/fdfs/client.conf.sample ]; then cp -f ../conf/client.conf /etc/fd
fs/client.conf.sample; fi
[root@tracker3 FastDFS]# cd /etc/fdfs
```

再次将其配置为 storage 服务器，添加到 group1 中，并查看 tracker 信息：

```
[root@tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf -h 192.168.182.101 list | grep -iE 'group|storage|ip_addr'
[2015-12-03 19:50:21] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=2, anti_steal_token=0, anti_stea
l_secret_key length=0, use_connection_pool=0, g_connection_pool_max_idle_time=3600s, use_storage_id=0, storage server id count: 0

group count: 2
Group 1:
group name = group1
storage server count = 3
storage server port = 23000
storage HTTP port = 8888
        Storage 1:
                ip_addr = 192.168.182.103  ACTIVE
                total storage = 18121 MB
                free storage = 14780 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id = 192.168.182.121
        Storage 2:
                ip_addr = 192.168.182.121  ACTIVE
                total storage = 18121 MB
                free storage = 14781 MB
                storage_port = 23000
                storage_http_port = 8888
                source storage id =
        Storage 3:
                ip_addr = 192.168.182.122  ACTIVE
                total storage = 18121 MB
```

192.168.182.103 节点成功添加到 group1 中，成为 Storage 1 节点，状态 ACTIVE，查看存储目录：

```
[root@tracker3 00]# pwd
/storage/data/00/00
[root@tracker3 00]# ls
wKi2elZf3b-AXC3yAAAAIqByaB80309930  wKi2elZgCCqAe853AAAAF849XuM3656860  wKi2eVZg4qaALEtfAAAAF849XuM0161655
wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080  wKi2elZg-KWEV7URAAAAAMh39_s284.txt  wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143
wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132  wKi2elZg_mWECRo7AAAAACkH2jc176.jpg  wKi2eVZgA9-ARbzUAAAAF849XuM1814323
wKi2elZf3iCANXmmAAAALPpsm903898002  wKi2eVZf3iCAApahAAAAIqByaB82288999  wKi2eVZgBdKAV8l9AAAAF849XuM4710938
wKi2elZf3iCATqahAAAAMSE-2l80296494  wKi2eVZf3iCABsdQAAAALEUe5vw2190367  wKi2eVZgBvuACQOeAAAAF849XuM3405938
wKi2elZf3iCAXUQ5AAAANzBubzw6176160  wKi2eVZf3iCAHldHAAAACrhx9-U5734081  wKi2eVZgBW-AQ7pPAAAAF849XuM5558800
wKi2elZg4oWAUbPUAAAAF849XuM7141647  wKi2eVZf3iCANopJAAAANBLU12g2152289  wKi2eVZgCDGAMFIOAAAAF849XuM0455345
wKi2elZgB3-AcfQTAAAAF849XuM4843492  wKi2eVZf3iCAT7R6AAAALsXob183732522  wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
wKi2elZgBzOAEwhAAAAAF849XuM8946663  wKi2eVZg4nmAMZnLAAAAF849XuM3985988
[root@tracker3 00]#
```

数据文件同步也已经完成，新节点成功添加。

测试附件的上传、下载、和删除命令都没问题：

```
[root@tracker1 filetest]# fdfs_upload_file /etc/fdfs/client.conf 1.jpg 192.168.182.103:23000
group1/M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg test.jpg
[root@tracker1 filetest]# ls
12.docx  12.pdf  12.txt   1.docx  1.pdf  1.txt   2.docx  2.pdf  2.txt   test.jpg
12.jpg   12.ppt  12.xlsx  1.jpg   1.ppt  1.xlsx  2.jpg   2.ppt  2.xlsx
[root@tracker1 filetest]# fdfs_delete_file /etc/fdfs/client.conf group1/M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg test1.jpg
[2015-12-03 20:03:10] ERROR - file: tracker_proto.c, line: 48, server: 192.168.182.103:23000, response status 2 != 0
download file fail, error no: 2, error info: No such file or directory
[root@tracker1 filetest]# ls
12.docx  12.pdf  12.txt   1.docx  1.pdf  1.txt   2.docx  2.pdf  2.txt   test.jpg
12.jpg   12.ppt  12.xlsx  1.jpg   1.ppt  1.xlsx  2.jpg   2.ppt  2.xlsx
[root@tracker1 filetest]#
```

通过重新安装试验，说明之前的 FastDFS 卸载干净，没有对 FastDFS 的再次使用造成干扰。

18 FastDFS 目录结构和文件的说明

18.1 tracker 服务器的目录结构和文件

完成 tracker 节点的配置并启动 tracker 服务后，在 base_path 目录下会生成 tracker 节点运行所必须的目录结构和文件，目录和文件如下：

${base_path}

|__data

|　　|__storage_groups_new.dat：存储分卷（组）信息

|　　|__storage_servers_new.dat：存储服务器列表

|__logs

　　|__trackerd.log：tracker server 日志文件

进入目录结构下，目录结构信息：

```
[root@tracker1 tracker]# ll -aR /tracker
/tracker:
total 12
drwxr-xr-x.  4 root root   28 Dec  2 16:29 .
drwxr-xr-x. 20 root root 4096 Dec  4 14:31 ..
drwxr-xr-x.  2 root root 4096 Dec  8 15:34 data
drwxr-xr-x.  2 root root 4096 Dec  8 10:57 logs

/tracker/data:
total 32
drwxr-xr-x. 2 root root 4096 Dec  8 15:34 .
drwxr-xr-x. 4 root root   28 Dec  2 16:29 ..
-rw-r--r--  1 root root    5 Dec  8 15:24 fdfs_trackerd.pid
-rw-r--r--. 1 root root   37 Dec  4 08:37 storage_changelog.dat
-rw-r--r--  1 root root  438 Dec  8 15:34 storage_groups_new.dat
-rw-r--r--  1 root root 6772 Dec  8 15:24 storage_servers_new.dat
-rw-r--r--  1 root root  204 Dec  8 15:24 storage_sync_timestamp.dat
-rw-r--r--. 1 root root   46 Dec  8 15:40 .tracker_status

/tracker/logs:
total 924
drwxr-xr-x. 2 root root   4096 Dec  8 10:57 .
drwxr-xr-x. 4 root root     28 Dec  2 16:29 ..
-rw-r--r--  1 root root 909973 Dec  8 15:34 trackerd.log
-rw-r--r--. 1 root root  10212 Dec  3 15:50 trackerd.log.20151203_000010
-rw-r--r--  1 root root   4254 Dec  4 14:31 trackerd.log.20151203_223112
-rw-r--r--  1 root root   2015 Dec  7 08:34 trackerd.log.20151207_083500
-rw-r--r--  1 root root    280 Dec  8 10:57 trackerd.log.20151208_105747
[root@tracker1 tracker]#
```

18.1.1 data 目录

18.1.1.1 storage_groups_new.dat 文件

storage_groups_new.dat 文件用来记录存储卷（组）的信息，文件信息示例：

```
[root@tracker1 data]# cat storage_groups_new.dat
# global section
[Global]
        group_count=2

# group: group1
[Group001]
        group_name=group1
        storage_port=23000
        storage_http_port=8888
        store_path_count=2
        subdir_count_per_path=256
        current_trunk_file_id=0
        trunk_server=
        last_trunk_server=

# group: group2
[Group002]
        group_name=group2
        storage_port=23000
        storage_http_port=8888
        store_path_count=1
        subdir_count_per_path=256
        current_trunk_file_id=0
        trunk_server=
        last_trunk_server=
```

storage_groups_new.dat 参数说明：

group_count：卷的个数

group_name：卷（组）名

storage_port：存储服务的端口号

storage_http_port：存储服务的 http 服务端口号

store_path_count：存储节点存储路径的个数

subdir_count_per_path：每个存储路径子目录的个数

current_trunk_file_id=0：目前 trunk 文件的 id

trunk_server=trunk 服务器

last_trunk_server=最近使用的 trunk 服务器

18.1.1.2 storage_servers_new.dat 文件

storage_servers_new.dat 文件记录了各个存储节点的详细信息，文件最后有一个节点总数的统计信息，文件信息示例：

```
[root@tracker1 data]# cat storage_servers_new.dat
# storage 192.168.182.121:23000
[Storage001]
        group_name=group1
        ip_addr=192.168.182.121
        status=5
        version=5.05
        join_time=1449107669
        storage_port=23000
        storage_http_port=8888
        domain_name=
        sync_src_server=
        sync_until_timestamp=0
        store_path_count=2
        subdir_count_per_path=256
        upload_priority=10
        total_mb=18121
        free_mb=14765
        total_upload_count=22
        success_upload_count=22
        total_append_count=3
        success_append_count=3
        total_set_meta_count=0
        success_set_meta_count=0
        total_delete_count=1
        success_delete_count=1
```

```
        total_create_link_count=0
        success_create_link_count=0
        total_delete_link_count=0
        success_delete_link_count=0
        total_upload_bytes=1973129
        success_upload_bytes=1973129
        total_append_bytes=57777
        success_append_bytes=57777
        total_download_bytes=1773815
        success_download_bytes=1773815
        total_sync_in_bytes=16241182
        success_sync_in_bytes=16241182
        total_sync_out_bytes=0
        success_sync_out_bytes=0
        total_file_open_count=52
        success_file_open_count=52
        total_file_read_count=12
        success_file_read_count=12
        total_file_write_count=111
        success_file_write_count=111
        last_source_update=1449554363
        last_sync_update=1449210593
        last_synced_timestamp=1449197171
        last_heart_beat_time=1449559467
        changelog_offset=37

# storage 192.168.182.122:23000
[Storage002]
        group_name=group1
```

.....

```
        total_delete_link_count=0
        success_delete_link_count=0
        total_upload_bytes=30479
        success_upload_bytes=30479
        total_append_bytes=685359
        success_append_bytes=685359
        total_download_bytes=715723
        success_download_bytes=715723
        total_sync_in_bytes=8216342
        success_sync_in_bytes=8216342
        total_sync_out_bytes=0
        success_sync_out_bytes=0
        total_file_open_count=14
        success_file_open_count=14
        total_file_read_count=3
        success_file_read_count=3
        total_file_write_count=44
        success_file_write_count=44
        last_source_update=1449196913
        last_sync_update=1449197545
        last_synced_timestamp=1449197538
        last_heart_beat_time=1449214245
        changelog_offset=37


# global section
[Global]
        storage_count=5
[root@tracker1 data]#
```

storage_servers_new.dat 参数说明：

group_name：存储节点所属组名

ip_addr：ip 地址

status：状态

version：版本

join_time：加入集群的时间

storage_port：存储节点的端口号

storage_http_port：存储节点 http 服务端口号

domain_name：域名

sync_src_server：向 storage server 同步已有数据文件的源服务器

sync_until_timestamp：同步已有数据文件的截至时间，服务器时间。

store_path_count：存储路径根目录数

subdir_count_per_path：每个根目录的子目录数

upload_priority：上传优先级

total_mb：总容量，单位 MB

free_mb：剩余空闲容量，单位 MB

total_upload_count：上传文件总个数

success_upload_count：成功上传的个数

total_append_count：总的附加文件个数

success_append_count：成功附加的文件个数

total_set_meta_count：总共更改 metadata（元数据）的次数

success_set_meta_count：成功更改 metadata 的次数

total_delete_count：总共删除文件的个数

success_delete_count：成功删除文件的个数

total_download_count：总共下载文件的个数

success_download_count：成功下载文件的个数

total_get_meta_count：总共获取 metdata 的次数

success_get_meta_count：成功获取 metdata 的次数

total_create_link_count：总共创建链接的个数
success_create_link_count：成功创建链接的个数
total_delete_link_count：总共删除链接的个数
success_delete_link_count：成功删除链接的个数
total_upload_bytes：总共上传的文件大小
success_upload_bytes：成功上传的文件大小
total_append_bytes：总得附加文件大小
success_append_bytes：成功附加的文件大小
total_download_bytes：总共下载的文件大小
success_download_bytes：成功下载的文件大小
total_sync_in_bytes：总共同步到该存储节点的文件大小
success_sync_in_bytes：成功同步到该存储节点的文件大小
total_sync_out_bytes：此节点作为源，总共同步到同组其他存储节点的文件大小
success_sync_out_bytes：此节点作为源，成功同步到同组其他存储节点的文件大小
total_file_open_count：文件打开总数
success_file_open_count：成功打开的文件数
total_file_read_count：文件读取总数
success_file_read_count：成功读取的文件数
total_file_write_count：写文件的总数
success_file_write_count：成功写文件的总数
last_source_update：最后一次源头更新时间（更新操作来自客户端）
last_sync_update：最后一次同步更新时间（更新操作来自其他 storage server 的同步）
last_synced_timestamp：最后同步时间戳
last_heart_beat_time：最后的心跳时间
changelog_offset：日志偏移量

18.1.1.3 storage_sync_timestamp.dat 文件

storage_sync_timestamp.dat 文件中记录了每个 storage 同步给其他 storage 的最后时间戳，数据以逗号分隔。以第一行数据为例介绍：

group1,192.168.182.121,0,1449552104,1449552104

group1：stroage 节点所在的组。

192.168.182.121：storage 节点的 IP 地址。

IP 地址后边的数字代表同组内 storage 节点同步的时间戳，第一行中 0 表示未同步，第一个 1449552104 表示从 192.168.182.121 同步到 192.168.182.122 的最后时间戳，第二个 1449552104 表示 192.168.182.121 同步到 192.168.182.123 的时间戳，因为 group1 中有 3 个 storage 节点，所以在 IP 地址后有 3 列数字。

按照纵列看，取**非零最小值**就是最后最早同步时间，最后最早同步时间用来判断一个文件在存储节点中是否存在，最后最早同步时间的详细信息可参照第 9 部分 storage 最后最早时间的介绍。

```
[root@trackerl data]# cat storage_sync_timestamp.dat
group1,192.168.182.121,0,1449552104,1449552104
group1,192.168.182.122,1449197171,0,0
group1,192.168.182.103,1449210591,1449210591,0
group2,192.168.182.123,0,1449197538
group2,192.168.182.124,1449196914,0
[root@trackerl data]#
```

18.1.1.4 storage_changelog. dat 文件

storage_changelog.dat 文件中记录的 storage 节点 IP 的变更信息。

当 storage 启动时，会将以前存在文件中的 IP(文件.data_init_flag, 不是第一次启动)和现在的 IP 比较,如果 IP 改变了,就会将 IP 变化的情况告诉 Tracker；Tracker 将更新原来的 storage 的 IP 地址, 但原来的 IP 继续保留在 server 列表中; 并将更新情况记录到 storage_changelog.dat 中。

下载文件时主要是看 group name; 如果 IP 更改了, 用指定源 IP server 下载时, 如果找不到，还是会用轮询的方式选择 storage 节点。

Tracker 每次都会将当前的 storage 节点信息保存在 storage_servers_new.dat 中，每次启动是先加载 storage_servers_new.dat 中的老的信息，后续如果 storage 的信息如果有变化就会更改 server 列表的信息(如 IP 更改等)。

storage_changelog.dat 文件信息示例如下：

```
[root@tracker1 data]# cat storage_changelog.dat
1449125608 group1 192.168.182.125 4
[root@tracker1 data]#
```

18.1.1.5 fdfs_trackerd. pid 文件

fdfs_trackerd.pid 文件记录了 tracker 服务器进程的进程号 PID，了解即可。信息内容如下：

```
[root@tracker1 data]# ps -ef |grep fdfs
root      14746      1  0 15:24 ?        00:00:02 fdfs_trackerd /etc/fdfs/tracker.conf start
root      15589   3156  0 16:37 pts/0    00:00:00 grep --color=auto fdfs
[root@tracker1 data]# cat fdfs_trackerd.pid
14746[root@tracker1 data]#
```

18.1.1.6 . tracker_status 文件

.tracker_status 文件是隐藏文件，记录 tracker 启动时间和最后一次检查时间，包含信息如下：

```
[root@tracker1 data]# cat .tracker_status;
up_time=1449559470
last_check_time=1449564300
```

18.1.2 logs 目录

18.1.2.1 trackerd. log 文件

trackerd.log 是 tracker 节点的日志文件，tracker 服务器的信息运行信息都会记录到此日志中，此日志可以用来在存储节点出错时辅助查错。信息内容样例如下：

```
[root@tracker1 logs]# tail trackerd.log
[2015-12-08 15:34:42] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.121, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:43] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.122, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:43] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.121, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:44] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.122, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:44] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.121, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:45] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.122, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:45] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.121, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:46] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.122, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:46] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.121, package size 32 is not correct, expect
length: 16
[2015-12-08 15:34:47] ERROR - file: tracker_service.c, line: 3514, cmd=84, client ip: 192.168.182.122, package size 32 is not correct, expect
length: 16
```

18.2 storage 服务器的目录结构和文件

完成 storage 节点的配置并启动 storage 服务后，在 base_path 目录下会生成 storage 节点运行所必须的目录结构和文件，目录和文件如下：

${base_path}：在 storage.conf 中配置的 base_path 路径

|__data

|　　|__.data_init_flag：当前 storage server 初始化信息，信息含义下面有详细介绍

|　　|__storage_stat.dat：当前 storage server 统计信息，信息含义下面有详细介绍

|　　|__sync：存放数据同步相关文件

|　　|　　|__binlog.index：当前的 binlog 文件索引号

|　　|　　|__binlog.###：存放更新操作的记录（日志）

|　　|　　|__${ip_addr}_${port}.mark：存放同步的完成情况

|　　|

|　　|__一级目录：256 个存放数据文件的目录，如果 store_path 不配置，文件默认存放到此处。

|　　　　|__二级目录：256 个存放数据文件的目录

|__logs

　　|__storaged.log：storage server 日志文件

　　|__storage_access.log：访问日志文件，如果开启了访问日志才会存在该日志文件，默认不存在。

进入目录结构下，目录结构信息：

```
/fastdfs/:
total 4
drwxr-xr-x.  4 root root   28 Dec  3 09:54 .
drwxr-xr-x. 19 root root 4096 Dec  3 09:20 ..
drwxr-xr-x.  3 root root   86 Dec  4 09:16 data
drwxr-xr-x.  2 root root   50 Dec  3 09:54 logs

/fastdfs/data:
total 16
drwxr-xr-x. 3 root root   86 Dec  4 09:16 .
drwxr-xr-x. 4 root root   28 Dec  3 09:54 ..
-rw-r--r--. 1 root root  232 Dec  3 09:54 .data_init_flag
-rw-r--r--. 1 root root    4 Dec  4 09:16 fdfs_storaged.pid
-rw-r--r--. 1 root root 1082 Dec  4 15:36 storage_stat.dat
drwxr-xr-x. 2 root root 4096 Dec  4 11:47 sync

/fastdfs/data/sync:
total 24
drwxr-xr-x. 2 root root 4096 Dec  4 11:47 .
drwxr-xr-x. 3 root root   86 Dec  4 09:16 ..
-rw-r--r--. 1 root root  127 Dec  4 14:30 192.168.182.103_23000.mark
-rw-r--r--. 1 root root  120 Dec  4 14:30 192.168.182.121_23000.mark
-rw-r--r--. 1 root root  126 Dec  3 14:30 192.168.182.125_23000.mark.20151202225346
-rw-r--r--. 1 root root 2434 Dec  4 14:30 binlog.000
-rw-r--r--. 1 root root    1 Dec  3 09:54 binlog.index

/fastdfs/logs:
total 28
drwxr-xr-x. 2 root root    50 Dec  3 09:54 .
drwxr-xr-x. 4 root root    28 Dec  3 09:54 ..
-rw-r--r--. 1 root root  1791 Dec  4 10:46 storage_access.log
-rw-r--r--. 1 root root 22305 Dec  4 14:39 storaged.log
[root@storage2 ~]#
```

18.2.1 data 目录

data 目录中存放了 storage 节点运行时所必须的一些文件，包括 storage 节点的初始化信息、运行状态信息、storage 节点间的同步信息等。

18.2.1.1 .data_init_flag 文件

.data_init_flag 文件中记录了 storage 节点的初始化信息：

```
[root@storage1 data]# cat .data_init_flag
storage_join_time=1449107669
sync_old_done=1
sync_src_server=
sync_until_timestamp=0
last_ip_addr=192.168.182.121
last_server_port=23000
last_http_port=8888
current_trunk_file_id=0
trunk_last_compress_time=0
[root@storage1 data]# a
```

storage_join_time：存储节点加入的时间
sync_old_done：是否成功获取到源机器 IP，1 表示成功获得。
sync_src_server：源主机的 IP 地址。
sync_until_timestamp：源主机负责的旧数据截止时间。
last_ip_addr：最后使用的 IP 地址
last_server_port：最后使用的服务器端口
last_http_port：最后使用的 http 服务端口
current_trunk_file_id：当前合并文件的 ID，默认 0
trunk_last_compress_time：合并文件最后的压缩时间，默认 0
其中 sync_old_done、sync_src_server、sync_until_timestamp 三个参数和源同步相关。

18.2.1.2 fdfs_storaged.pid 文件

fdfs_storaged.pid 文件记录了 storage 服务的进程号，这个文件不需要手动更改，了解即可：

```
[root@storage1 data]# ps -ef |grep fdfs
root       9282      1  0 Dec07 ?        00:00:29 fdfs_storaged /etc/fdfs/storage.conf start
root      22739  14288  0 11:34 pts/2    00:00:00 grep --color=auto fdfs
[root@storage1 data]# cat fdfs_storaged.pid
9282[root@storage1 data]#
```

18.2.1.3 storage_stat.dat 文件

storage_stat.dat 记录了 storage 服务的状态信息，不需要手动更改，了解信息含义即可：

```
[root@storage1 data]# cat storage_stat.dat
total_upload_count=17
success_upload_count=17
total_append_count=3
success_append_count=3
total_modify_count=0
success_modify_count=0
total_truncate_count=0
success_truncate_count=0
total_download_count=6
success_download_count=6
total_set_meta_count=0
success_set_meta_count=0
total_delete_count=1
success_delete_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=1972925
success_upload_bytes=1972925
total_append_bytes=57777
success_append_bytes=57777
total_modify_bytes=0
success_modify_bytes=0
```

```
total_download_bytes=1773815
success_download_bytes=1773815
total_sync_in_bytes=16241182
success_sync_in_bytes=16241182
total_sync_out_bytes=0
success_sync_out_bytes=0
total_file_open_count=47
success_file_open_count=47
total_file_read_count=12
success_file_read_count=12
total_file_write_count=106
success_file_write_count=106
last_source_update=1449452809
last_sync_update=1449210593
dist_path_index_high=0
dist_path_index_low=0
dist_write_file_count=17
[root@storage1 data]#
```

total_upload_count：上传文件总个数

success_upload_count：成功上传的个数

total_append_count：总上传追加个数

success_append_count：成功上传追加个数

total_modify_count：总共修改文件的个数
success_modify_count：成功修改文件的个数
total_truncate_count：总共合并文件的个数
success_truncate_count：成功合并文件的个数
total_download_count：总共下载文件的个数
success_download_count：成功下载文件的个数
total_set_meta_count：总共设置 metadata（元数据）的个数
success_set_meta_count：成功设置 metadata 的个数
total_delete_count：总共删除文件的个数
success_delete_count：成功删除文件的个数
total_get_meta_count：总共获取 metdata 的次数
success_get_meta_count：成功获取 metdata 的次数
total_create_link_count：如果文件存在，创建连接个数，用连接来访问文件
success_create_link_count：如果文件存在，成功创建连接个数， 用连接来访问文件
total_delete_link_count：总共删除链接的次数
success_delete_link_count：成功删除连接的次数
total_upload_bytes：上传文件的总大小
success_upload_bytes：成功上传文件的大小
total_append_bytes：附加文件的总大小
success_append_bytes：成功附加文件大小
total_modify_bytes：修改文件的总大小
success_modify_bytes：成功修改文件的大小
total_download_bytes：下载文件的总大小
success_download_bytes：成功下载文件的大小
total_sync_in_bytes：同步同组下机器文件的总大小
success_sync_in_bytes：成功同步同组下机器文件的总大小
total_sync_out_bytes：本机器同步到同组别的机器文件大小
success_sync_out_bytes：本机器成功同步到同组别的机器文件大小
total_file_open_count：打开文件总数量(同步文件个数+上传次数+流量次数)
success_file_op
en_count：成功打开文件数量(同步文件个数+上传次数+流量次数)
total_file_read_count：读取文件总个数
success_file_read_count：成功读取文件个数
total_file_write_count：上传文件数，可能还有同步别的机器过来的文件数量
success_file_write_count：成功上传文件数，可能还有同步别的机器过来的文件数量
last_source_update：最后一次上传文件时间
last_sync_update：同步别的机器数据过来的时间
dist_path_index_high：
dist_path_index_low：
dist_write_file_count：

18.2.1.4 sync 目录

sync 目录中的文件用来记录 storage 之间同步的信息。

18.2.1.4.1 ${ip_addr}_${port}.mark 文件

mark 文件的格式：${ip_addr}_${port}.mark，每个 storage 包含一个或多个 mark 文件，每个 mark 文件对应同组中的一个 storage 节点，mark 文件中信息如下：

```
[root@storage1 sync]# cat 192.168.182.122_23000.mark
binlog_index=0
binlog_offset=2617
need_sync_old=1
sync_old_done=1
until_timestamp=1449107684
scan_row_count=44
sync_row_count=22
[root@storage1 sync]#
```

增量同步相关参数：

binlog_index：已处理（同步）到的 binlog 索引号，记录当前使用的 Binlog 文件序号，如为 1，则表示使用 binlog.001，binlog.001 就是真实地 Binlog 文件。

binlog_offset：已处理（同步）到的 binlog 文件偏移量（字节数），表示上次同步给对方的最后一条 binlog 偏移量，若程序重启了，也只要从这个位置开始向后同步即可。

源同步相关参数：

need_sync_old：是否需要同步旧文件给对方，1 需要同步，0 不需要同步。

sync_old_done：若需要同步旧文件，那么旧文件是否同步完成，1 完成，0 未完成。

until_timestamp：源主机同步数据的截止时间戳。

其他参数：

scan_row_count：已扫描的 binlog 记录数。

sync_row_count：已同步的 binlog 记录数。

18.2.1.4.2 binlog.index 文件

binlog.index 中只有一个数据项：当前 binlog 的文件索引号。如下：

```
[root@storage1 sync]# cat binlog.index
0[root@storage1 sync]#
```

18.2.1.4.3 binlog.###文件

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：

timestamp：更新发生时间（Unix 时间戳）。

op_type：操作类型，一个字符。操作类型说明：

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

源表示客户端直接操作的那个 Storage 即为源，其他的 Storage 都为副本。

filename：文件名，如：M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx。其中的 M01 为 storepath 索引，紧接着 00/00/为路径，后面 M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx 为文件系统中实际的文件名（不采用合并存储时，若采用合并存储并不是一个实际的文件名）。

文件名组成：M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx，这个文件名中，除了 docx 为文件后缀，CgAHl1SJUR6AZqSHAAAF1vgN0rw59 这部分是一个 base64 编码缓冲区，组成如下：

Storage_id（ip 的数值型）

timestamp（创建时间）

file_size（若原始值为 32 位则前面加入一个随机值填充，最终为 64 位）

crc32（文件内容的检验码）

示例：

```
-rw-r--r-- 1 root root  129 Dec  8 13:21 192.168.182.103_23000.mark
-rw-r--r-- 1 root root  129 Dec  8 13:21 192.168.182.122_23000.mark
-rw-r--r-- 1 root root  128 Dec  3 14:30 192.168.182.125_23000.mark.20151202225332
-rw-r--r-- 1 root root 2849 Dec  8 13:59 binlog.000
-rw-r--r-- 1 root root    1 Dec  3 09:54 binlog.index
[root@storage1 sync]# cat binlog.000
1449119915 C M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
1449121910 D M00/00/00/wKi2eVZf0KuAEPaaAAAAPm9mCTM1183578
1449123263 c M00/00/00/wKi2elZf3b-AXC3yAAAAIqByaB80309930
1449123360 C M00/00/00/wKi2eVZf3iCAApahAAAAIqByaB82288999
1449123360 C M00/00/00/wKi2eVZf3iCABsdQAAAALEUe5vw2190367
1449123360 C M00/00/00/wKi2eVZf3iCAHldHAAAACrhx9-U5734081
1449123360 C M00/00/00/wKi2eVZf3iCAT7R6AAAALsXob183732522
1449123360 C M00/00/00/wKi2eVZf3iCANopJAAAANBLU12g2152289
1449123360 c M00/00/00/wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080
1449123360 c M00/00/00/wKi2elZf3iCANXmmAAAALPpsm903898002
1449123360 c M00/00/00/wKi2elZf3iCAXUQ5AAAANzBubzw6176160
1449123360 c M00/00/00/wKi2elZf3iCATqahAAAAMSE-2l80296494
1449123360 c M00/00/00/wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132
1449133023 C M00/00/00/wKi2eVZgA9-ARbzUAAAAF849XuM1814323
```

18.2.2 logs 目录

18.2.2.1 storaged.log 文件

storaged.log 是 storage 节点的日志文件，storage 服务器的信息运行信息都会记录到此日志中，此日志可以用来在存储节点出错时辅助查错。信息内容样例如下：

```
[root@storage1 logs]# tailf storaged.log
[2015-12-08 15:34:45] INFO - file: tracker_client_thread.c, line: 310, successfully connect to tracker server 192.168.182.102:22122, as a trac
ker client, my ip is 192.168.182.121
[2015-12-08 15:34:45] INFO - file: tracker_client_thread.c, line: 1235, tracker server 192.168.182.102:22122, set tracker leader: 192.168.182.
102:22122
[2015-12-08 15:34:46] INFO - file: tracker_client_thread.c, line: 310, successfully connect to tracker server 192.168.182.101:22122, as a trac
ker client, my ip is 192.168.182.121
[2015-12-08 15:34:46] INFO - file: tracker_client_thread.c, line: 310, successfully connect to tracker server 192.168.182.102:22122, as a trac
ker client, my ip is 192.168.182.121
[2015-12-08 15:34:46] INFO - file: tracker_client_thread.c, line: 1235, tracker server 192.168.182.102:22122, set tracker leader: 192.168.182.
102:22122
[2015-12-08 15:34:47] INFO - file: tracker_client_thread.c, line: 310, successfully connect to tracker server 192.168.182.101:22122, as a trac
ker client, my ip is 192.168.182.121
[2015-12-08 15:34:47] INFO - file: tracker_client_thread.c, line: 310, successfully connect to tracker server 192.168.182.102:22122, as a trac
ker client, my ip is 192.168.182.121
[2015-12-08 15:34:47] INFO - file: tracker_client_thread.c, line: 1235, tracker server 192.168.182.102:22122, set tracker leader: 192.168.182.
102:22122
[2015-12-08 15:34:48] INFO - file: storage_sync.c, line: 2698, successfully connect to storage server 192.168.182.103:23000
[2015-12-08 15:35:18] INFO - file: storage_sync.c, line: 2698, successfully connect to storage server 192.168.182.122:23000
```

18.2.2.2 storage_access.log 文件

storage_access.log 访问日志记录了文件的操作记录，信息样例如下：

```
[root@storage1 logs]# tailf storage_access.log
[2015-12-03 22:10:17.788] 192.168.182.101 upload M00/00/00/wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg 0 13 263759 70
[2015-12-06 17:45:52.763] 192.168.182.101 upload M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 0 1 540 70
[2015-12-06 17:46:50.488] 192.168.182.101 append M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 0 0 6675 10
[2015-12-06 17:47:05.787] 192.168.182.101 download M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 0 0 86 7130
[2015-12-06 17:47:10.206] 192.168.182.101 download M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 0 0 86 7130
[2015-12-07 21:21:44.228] 192.168.182.101 upload M00/00/00/wKi2eVZmaOiAdUFBAAAAIqByaB83040598 0 0 59 70
[2015-12-08 13:55:26.377] 192.168.182.101 upload M01/00/00/wKi2eVZmcM6ANY-AAAAAIqByaB80521498 0 0 59 70
[2015-12-08 13:56:56.787] 192.168.182.101 upload M01/00/00/wKi2eVZmcSiAUoUhAAAAIqByaB87391837 0 0 59 70
[2015-12-08 13:57:49.740] 192.168.182.101 upload M01/00/00/wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379 0 0 76 70
[2015-12-08 13:59:24.149] 192.168.182.101 upload M01/00/00/wKi2eVZmcbyAd2W2AAAAM3gt4Uo9171221 0 0 76 70
```

19 FastDFS 自动备份和恢复

19.1 FastDFS 自动备份

FastDFS 的同一卷（组）内的 Storage 节点之间数据完全一致，互相作为备份。同一个组内的存储节点越多，数据冗余分数越多，数据的安全性越高。如果想要添加一个冗余备份，只需要向卷内

添加一个存储节点即可，FastDFS 会采用源同步的方式在新增加的节点中完全冗余一份数据。

19.2 FastDFS 恢复

FastDFS 的磁盘恢复属于自动完成，不需要手动干预，这里只介绍原来，方便出现问题后查错。

磁盘恢复也就是一台服务器的某个磁盘坏掉，换了一个新的硬盘，然后将旧磁盘本应该有的数据拷贝到新硬盘的过程。FastDFS 原生支持该过程，可以自动检测，并完成。FastDFS 推荐的方式是一个磁盘挂载成一个 Store-Path，换了一个新的磁盘后，该 Store_path 目录下的数据丢失了，但是这部分数据在同组的其他 Storage 都有，因此只要从对应的 Storage 上拷贝到目录下的数据即可。

FastDFS-Storaged 程序在启动时会检查每个 Store_path 目录下的子目录个数，默认情况下，每个 Store_path 下面会创建两级 256 个子目录。**当非首次启动时，发现某个 Store_path 下没有任何目录，则会进入磁盘恢复过程。**

磁盘恢复过程包括三个步骤：RecoveryStart、RecoveryStore、RecoveryFinish；这三个步骤之中的 RecoveryStore 可以多次执行，因此在恢复过程中记录了中间状态，即使在磁盘恢复过程中宕机或停止，下次再次启动时还是可以从上次的状态继续恢复数据。而 RecoveryStart 与 RecoveryFinish 都是只执行一次。

磁盘恢复过程的两个状态文件：**.recovery.mark**、**.binlog.recovery**

这两个状态文件保存与 StorePath 目录下，用于记录恢复过程的状态。

.recovery.mark 文件包括三个字段：

saved_storage_status=　　##表示状态

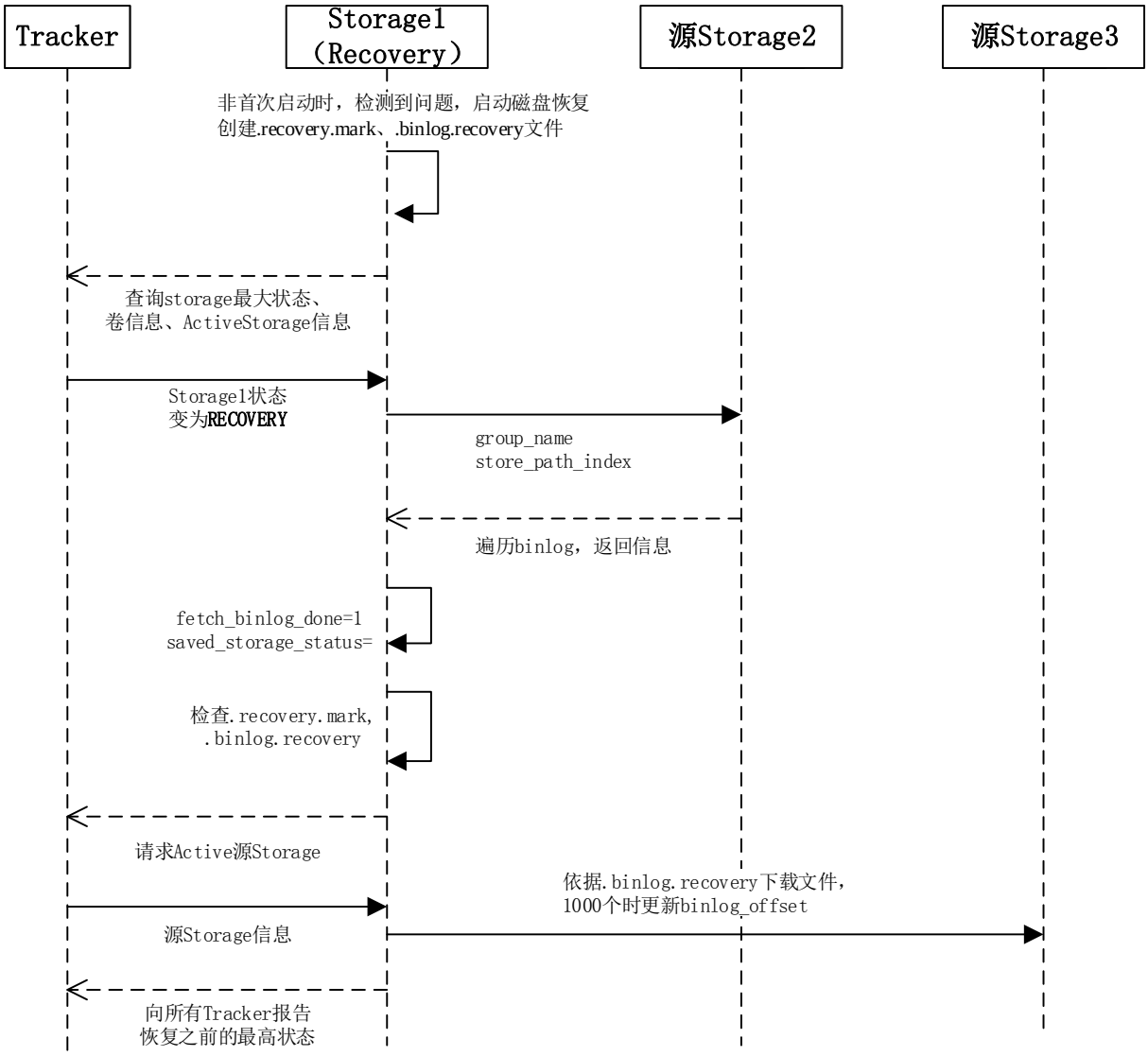
binlog_offset=　　##表示恢复的 binlog 偏移量

fetch_binlog_done=　　##表示是否已经下载 binlog

.binlog.recovery 文件，记录的是需要恢复的文件 binlog，内容类似下面这样的记录：

1406858030 C M00/00/00/CmQPRlPa8y6AbW3ZAAADWjN_G5k15.conf

恢复过程



恢复过程

1. 磁盘恢复开始 RecoveryStart

在 Storaged 启动时，会依次检查每一个 StorePath，首次启动时会为每个 StorePath 创建两级 256 个子目录（默认 256，可配置），因此当非首次启动时，检测到 StorePath 下不存在这两级的 256 个子目录，那么程序就会认为该 StorePath 数据丢失，开始进行这个 StorePath 的磁盘恢复。

RecoveryStart 执行如下操作：

1） 创建状态文件：**.recovery.mark**、**.binlog.recovery**，并初始化。

2） 获取源 Storage：连接 Tracker，查询该 Storage 的最大状态值（该值用于在磁盘恢复完成后，恢复到之前的状态），查询所在组的 Storage 列表，从中获取一个状态为 Active 的 Storage，若获取不到 Active 的 Storage 则睡眠 5 秒后再次尝试；获取到的该 Storage 我们称为源 Storage。

3） Tracker 将要恢复的 storage 设置为 **RECOVERY** 状态。

4） 向源 Storage 发送 FETCH_ONE_PATH_BINLOG 命令，参数为 group_name、store_path_index；源 Storage 收到该命令后从头遍历整个 binlog，过滤出符合下列条件的 binlog 返回来：

binlog 文件的目录为指定的 store_path；

该 binlog 的操作类型为 Create 方式；

该 binlog 对应的文件在文件系统中还存在；

5） 将从源 Storage 返回的 binlog 写入到.binlog.recovery 文件之中，若完成收到源 Storage 的整个回复，则设置.recovery.mark 文件中的 fetch_binlog_done 标志为 1；表示已经完成 binlog 文件的获取，设置 saved_storage_status 为第二步中查询到的最大状态值。

6） 检查若采用合并存储，则需要将该 binlog 进行分割。

2. 磁盘恢复下载文件

这个步骤也就是真实地从一个源 Storage 上下载文件，过程如下：

1） 检查是否存在.recovery.mark、 .binlog.recovery 文件，若任何一个不存在则返回。

2） 向 RecoveryStart 一样，向 Tracker 请求，查找该组内的一个状态为 Active 的 Storage 作为源 Storage。

3） 打开.recovery.mark 文件，读取 binlog_offset 值，打开.binlog.recovery 文件，定位到 binlog_offset 这个位置，也就是从上次最后保存的状态开始继续下载文件。

4） 依次读取.binlog.recovery 文件的后面记录，对于每一条记录都从源 Storage 中去下载对应文件

5） 每下载 1000 个文件，将状态值 binlog_offset 写入到.recovery.mark 文件之中

6） 当所有的 binlog 对应文件都下载完成后，向所有 Tracker 报告恢复之前的最高状态。

3. 磁盘恢复结束

该步骤清理开始时创建的临时文件 .recovery.mark, .binlog.recovery。

20 FastDFS 手动迁移

FastDFS 手动迁移将集群中的存放的数据单独拿出来再部署一个新的环境，实际工作中可以最为一个故障恢复的方法、也可以用来搭建测试用的集群环境。

20.1 手动迁移分类

手动迁移分三种情况：

第一种：新集群机器的机器数量和原来集群的数量一样，部署结构一样，IP 地址也一样。

在新机器上安装 FastDFS 后，直接将 base_path 和 store_path 目录下的内容复制到新目录中，启动服务即可。

第二种：新集群机器的机器数量和原来集群的数量一样，部署结构一样，但是 IP 地址变更。

如果使用了 server ID 特性，直接将在新机器上 tracker 上的 IP 和 ID 映射文件 storage_ids.conf 修改好即可。

如果没有使用 server ID 特性，那么需要将相关文件中的 IP 地址全部更改。需要更改的文件：

tracker 服务器：

data/storage_groups_new.dat：记录集群卷（组）信息，注意端口号是否一致。

data/storage_servers_new.dat：记录集群 storage 节点信息，将旧 IP 修改为新 IP 地址。

data/storage_sync_timestamp.dat：记录同步时间戳信息，将旧 IP 修改为新 IP 地址。

storage 服务器：

data/.data_init_flag：记录服务器初始化信息，将旧 IP 修改为新 IP 地址。

data/sync/${ip_addr}_${port}.mark：同组其他服务器的同步记录文件，修改文件名中的 IP 地址。

第三种：不管集群机器的数据和部署方式，也不管 IP 地址是什么样的，做一个通用迁移。此时的迁移需要注意的事项比较多，修改的文件也比较多，下面以一个迁移的案例介绍说明

20.2 通用手动迁移方法

以上面安装 FastDFS 章节的环境为基础，假如现在想利用 192.168.182.121 storage1 存储节点中的数据搭建另外的一个集群测试环境，新的集群环境服务器信息如下：

主机名/IP 地址	卷（组）名	角色	备注	base_path	store_path
back_tracker1/192.168.182.111	-	跟踪器	管理 group1	/tracker	-
back_storage1/192.168.182.131	group1	存储节点		/fastdfs	/base/storage /base/storage1
back_storage2/192.168.182.132	group1	存储节点		/fastdfs	/base/storage /base/storage1

新的测试环境中只有 1 个 tracker 和 1 个 group1，group1 中包含 2 个存储节点，按照如下步骤开始手动迁移 FastDFS 中的文件并搭建一个测试库。

20.2.1 安装配置 FastDFS

在 back_tracker1、back_storage1、back_storage2 三台机器上按照安装章节中的说明安装 FastDFS，在配置 back_storage1、back_storage2 中的 store_path_count、store_path 参数时注意目录的个数要和旧服务器一样，文件保存路径可以随便写，以此次迁移为例：

旧服务器/etc/fdfs/storage.conf 的 store_path_count、store_path 的设置为：

```
# path(disk or mount point) count, default value is 1
store_path_count=2

# store_path#, based 0, if store_path0 not exists, it's value is base_path
# the paths must be exist
store_path0=/storage
store_path1=/storage1
#store_path1=/home/yuqing/fastdfs2
```

新服务器的/etc/fdfs/storage.conf 的 store_path_count、store_path 的设置为：

```
# path(disk or mount point) count, default value is 1
store_path_count=2

# store_path#, based 0, if store_path0 not exists, it's value is base_path
# the paths must be exist
store_path0=/base/storage
store_path1=/base/storage1
#store_path1=/home/yuqing/fastdfs2
```

此处 back_storage2 和 back_storage1 配置相同

注：back_storage2 和 back_storage1 两台服务器的 store_path0 和 store_path1 也可以不一样。

20.2.2 关闭旧的 FastDFS 服务器并复制文件

关闭全部旧的 tracker 和 storage 服务器。

将旧 tracker1 的${base_path}路径下的 data 和 logs 文件夹复制到新 back_racker1 下${base_path}路径下。

将旧 storage1 的${base_path}和${store_path}路径下的内容复制到新 back_storage1 的${base_path}和${store_path}路径下。

20.2.3 调整新 tracker 上的文件

新 back_tracker1 服务器上需要调整的文件列表：

data/storage_groups_new.dat：记录集群组的信息

data/storage_servers_new.dat：记录集群 storage 节点信息

data/storage_sync_timestamp.dat：记录同步时间戳信息

data/storage_changelog.dat：记录 IP 变更历史，如果文件存在需要清空文件内容。

20.2.3.1 storage_groups_new.dat 文件

storage_groups_new.dat 中记录了组的信息，旧集群中是 2 个 group，新集群中只有 1 个 group，所以修改 storage_groups_new.dat 中的集群信息，如果端口等信息修改了，也需要将此文件的对应的项修改

tracker1 中 storage_groups_new.dat 的内容

```
# global section
[Global]
        group_count=2

# group: group1
[Group001]
        group_name=group1
        storage_port=23000
        storage_http_port=8888
        store_path_count=2
        subdir_count_per_path=256
        current_trunk_file_id=0
        trunk_server=
        last_trunk_server=

# group: group2
[Group002]
        group_name=group2
        storage_port=23000
        storage_http_port=8888
        store_path_count=1
        subdir_count_per_path=256
        current_trunk_file_id=0
        trunk_server=
        last_trunk_server=
```

back_tracker1 中 storage_groups_new.dat 的内容

```
# global section
[Global]
        group_count=1

# group: group1
[Group001]
        group_name=group1
        storage_port=23000
        storage_http_port=8888
        store_path_count=2
        subdir_count_per_path=256
        current_trunk_file_id=0
        trunk_server=
        last_trunk_server=

~
~
~
~
~
~
~
~
~
~
```

20.2.3.2 storage_servers_new.dat 文件

storage_servers_new.dat 文件中记录了所有 storage 服务器的详细信息，手动迁移时需要将没用的信息删除，同时将 ip 地址修改。

tracker1 中 storage_servers_new.dat 的内容：

```
# storage 192.168.182.121:23000
[Storage001]
        group_name=group1
        ip_addr=192.168.182.121
        status=5
        version=5.05
        join_time=1449107669
        storage_port=23000
        storage_http_port=8888
        domain_name=
        sync_src_server=
        sync_until_timestamp=0
        store_path_count=2
        subdir_count_per_path=256
        upload_priority=10
        total_mb=36242
        free_mb=29480
        total_upload_count=22
        success_upload_count=22
        total_append_count=3
        success_append_count=3
        total_set_meta_count=0
        success_set_meta_count=0
        total_delete_count=1
        success_delete_count=1
        total_download_count=6
```

```
        success_download_count=6
        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=1973129
        success_upload_bytes=1973129
        total_append_bytes=57777
        success_append_bytes=57777
        total_download_bytes=1773815
        success_download_bytes=1773815
        total_sync_in_bytes=16241182
        success_sync_in_bytes=16241182
        total_sync_out_bytes=0
        success_sync_out_bytes=0
        total_file_open_count=52
        success_file_open_count=52
        total_file_read_count=12
        success_file_read_count=12
        total_file_write_count=111
        success_file_write_count=111
        last_source_update=1449554363
        last_sync_update=1449210593
        last_synced_timestamp=1449197171
```

```
        success_file_write_count=111
        last_source_update=1449554363
        last_sync_update=1449210593
        last_synced_timestamp=1449197171
        last_heart_beat_time=1449624955
        changelog_offset=37

# storage 192.168.182.122:23000
[Storage002]
        group_name=group1
        ip_addr=192.168.182.122
        status=5
        version=5.05
        join_time=1449107683
        storage_port=23000
        storage_http_port=8888
        domain_name=
        sync_src_server=192.168.182.121
        sync_until_timestamp=1449107684
        store_path_count=2
        subdir_count_per_path=256
        upload_priority=10
        total_mb=36242
        free_mb=29484
        total_upload_count=12
        success_upload_count=12
```

tracker1 中的 storage_servers_new.dat 文件记录了 5 个 storage 节点的信息，而新服务器集群中一开始只有一个节点，所以将其余的 storage 删除，修改 ip_addr 的地址。

back_tracker1 中 storage_servers_new.dat 修改后的内容：

```
# storage 192.168.182.131:23000
[Storage001]
        group_name=group1
        ip_addr=192.168.182.131
        status=5
        version=5.05
        join_time=1449107669
        storage_port=23000
        storage_http_port=8888
        domain_name=
        sync_src_server=
        sync_until_timestamp=0
        store_path_count=2
        subdir_count_per_path=256
        upload_priority=10
        total_mb=36242
        free_mb=29480
        total_upload_count=22
        success_upload_count=22
        total_append_count=3
        success_append_count=3
        total_set_meta_count=0
        success_set_meta_count=0
        total_delete_count=1
        success_delete_count=1
        total_download_count=6
-- INSERT --
```

```
        success_download_count=6
        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=1973129
        success_upload_bytes=1973129
        total_append_bytes=57777
        success_append_bytes=57777
        total_download_bytes=1773815
        success_download_bytes=1773815
        total_sync_in_bytes=16241182
        success_sync_in_bytes=16241182
        total_sync_out_bytes=0
        success_sync_out_bytes=0
        total_file_open_count=52
        success_file_open_count=52
        total_file_read_count=12
        success_file_read_count=12
        total_file_write_count=111
        success_file_write_count=111
        last_source_update=1449554363
        last_sync_update=1449210593
        last_synced_timestamp=1449197171
-- INSERT --                                                    52,18-25
```

```
        last_synced_timestamp=1449197171
        last_heart_beat_time=1449624955
        changelog_offset=37
# global section
[Global]
        storage_count=1
                                                                57,16-23
```

20.2.3.3 storage_sync_timestamp.dat 文件

storage_sync_timestamp.dat 文件中记录的是同步时间戳，旧集群中有 5 个 storage 节点，新集群中刚开始只有 1 个，另一个 storage 节点采用添加节点的方式自动同步数据。

tracker1 中 storage_sync_timestamp.dat 的内容：

```
[root@tracker1 data]# cat storage_sync_timestamp.dat
group1,192.168.182.121,0,1449554364,1449554364
group1,192.168.182.122,1449197171,0,0
group1,192.168.182.103,1449210591,1449210591,0
group2,192.168.182.123,0,1449197538
group2,192.168.182.124,1449196914,0
[root@tracker1 data]#
```

back_tracker1 中 storage_sync_timestamp.dat 的内容：

```
group1,192.168.182.131,0
~
~
~
~
~
~
~
```

20.2.3.4 storage_changelog.dat 文件

storage_changelog.dat 如果存在并且其中有内容，迁移时需要将新 tracker 中的 storage_changelog.dat 文件内容清空。

20.2.4 调整新 storage1 上的文件

新 storage1 服务器 back_storage1 上需要调整的文件列表：

data/.data_init_flag

data/sync/${ip_addr}_${port}.mark

data/sync/index.xxx

20.2.4.1 .data_init_flag 文件

.data_init_flag 文件需要修改 IP 地址，对应参数 last_ip_addr，修改后内容如下：

```
storage_join_time=1449107669
sync_old_done=1
sync_src_server=
sync_until_timestamp=0
last_ip_addr=192.168.182.131
last_server_port=23000
last_http_port=8888
current_trunk_file_id=0
trunk_last_compress_time=0
~
~
```

20.2.4.2 ${ip_addr}_${port}.mark 文件

${ip_addr}_${port}.mark 如果整齐集群迁移，此类文件，需要将此文件名中的 IP 地址改为新地址即可。

但是采用只复制一个 storage 节点中的数据，然后再添加新节点自动同步数据的方式，所以不需要${ip_addr}_${port}.mark 文件，sync 目录下只保留 binlog.xxx 文件和 binlog.index 文件，其余文件可以删除。

```
[root@back_storage1 sync]# ll
total 8
-rw-r--r--. 1 root root 2849 Dec  9 13:57 binlog.000
-rw-r--r--. 1 root root    1 Dec  9 13:57 binlog.index
[root@back_storage1 sync]#
```

20.2.4.3 binlog.xxx 文件

binglog.xxx 文件在数据同步时会用到，当 storage1 节点加入集群后，再加入 storage2 节点时，会根据 binglog.xxx 文件中的内容将 storage1 节点的文件同步到 storage2，如果没有此文件，在 storage2 加入集群后，storage1 中的文件不会同步到 storage2 中。

修改 binglog.xxx 文件，将第二列的字符串全部替换为大写。

修改前：

```
1449190054 C M00/00/00/wKi2eVZg4qaALEtfAAAAF849XuM0161655
1449190021 c M00/00/00/wKi2elZg4oWAUbPUAAAAF849XuM7141647
1449194054 C M00/00/00/wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143
1449194100 A M00/00/00/wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143 51 23
1449195685 c M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt
1449195733 a M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt 338 110
1449196251 C M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
1449196277 A M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx 1708216 51149
1449197157 c M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg
1449197171 a M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg 263734 441516
1449201695 c M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
1449201781 d M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
1449207121 c M00/00/00/wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
1449207142 a M00/00/00/wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt 360448 357376
1449209405 c M00/00/00/wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
1449209417 C M00/00/00/wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
1449209780 c M00/00/00/wKi2Z1ZhL7SAQm1iAGegs9vls40995.pdf
1449210271 c M00/00/00/wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
1449210591 c M00/00/00/wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
1449452752 C M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
1449452810 A M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 515 6605
1449552104 C M00/00/00/wKi2eVZmaOiAdUFBAAAAIqByaB83040598
1449554126 C M01/00/00/wKi2eVZmcM6ANY-AAAAAIqByaB80521498
1449554216 C M01/00/00/wKi2eVZmcSiAUoUhAAAAIqByaB87391837
1449554269 C M01/00/00/wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379
```

vim 命令编辑文件，执行下面的替换命令：

:%s/ c / C /g　　##c 替换为 C

:%s/ d / D /g　　##d 替换为 D

:%s/ a / A /g　　##a 替换为 A

:%s/ t / T /g　　##t 替换为 T

注意字母 c 和 C、d 和 D、a 和 A、t 和 T 前后都带空格，要不然会把文件名中的字母也替换掉，这样就麻烦大了。

```
1449209417 C M00/00/00/wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
1449209780 C M00/00/00/wKi2Z1ZhL7SAQmliAGegs9vls40995.pdf
1449210271 C M00/00/00/wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
1449210591 C M00/00/00/wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
1449452752 C M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
1449452810 A M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 515 6605
1449552104 C M00/00/00/wKi2eVZmaOiAdUFBAAAAIqByaB83040598
1449554126 C M01/00/00/wKi2eVZmcM6ANY-AAAAAIqByaB80521498
1449554216 C M01/00/00/wKi2eVZmcSiAUoUhAAAAIqByaB87391837
1449554269 C M01/00/00/wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379
1449554364 C M01/00/00/wKi2eVZmcbyAd2W2AAAAM3gt4Uo9171221
:%s/ c / C /g
```

修改后：

```
1449190054 C M00/00/00/wKi2eVZg4qaALEtfAAAAF849XuM0161655
1449190021 C M00/00/00/wKi2elZg4oWAUbPUAAAAF849XuM7141647
1449194054 C M00/00/00/wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143
1449194100 A M00/00/00/wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143 51 23
1449195685 C M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt
1449195733 A M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt 338 110
1449196251 C M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
1449196277 A M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx 1708216 51149
1449197157 C M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg
1449197171 A M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg 263734 441516
1449201695 C M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
1449201781 D M00/00/00/wKi2Z1ZhEB-ADtvZAAQGNikH2jc837.jpg
1449207121 C M00/00/00/wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
1449207142 A M00/00/00/wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt 360448 357376
1449209405 C M00/00/00/wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
1449209417 C M00/00/00/wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
1449209780 C M00/00/00/wKi2Z1ZhL7SAQmliAGegs9vls40995.pdf
1449210271 C M00/00/00/wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
1449210591 C M00/00/00/wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
1449452752 C M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
1449452810 A M00/00/00/wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt 515 6605
1449552104 C M00/00/00/wKi2eVZmaOiAdUFBAAAAIqByaB83040598
1449554126 C M01/00/00/wKi2eVZmcM6ANY-AAAAAIqByaB80521498
1449554216 C M01/00/00/wKi2eVZmcSiAUoUhAAAAIqByaB87391837
1449554269 C M01/00/00/wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379
1449554364 C M01/00/00/wKi2eVZmcbyAd2W2AAAAM3gt4Uo9171221
```

保存文件后退出。

20.2.5 启动集群服务

上述文件全部调整完成后才能启动集群服务。

1. 启动 back_tracker1 服务器，并检查服务器状态。

启动 back_tracker1 上的 tracker 服务并查看服务器状态信息：

```
[root@back_tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf list
[2015-12-09 11:39:54] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=1, anti_steal_token=0, anti_stea
l_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 192.168.182.111:22122

group count: 1

Group 1:
group name = group1
disk total space = 0 MB
disk free space = 0 MB
trunk free space = 0 MB
storage server count = 1
active server count = 0
storage server port = 23000
storage HTTP port = 8888
store path count = 2
subdir count per path = 256
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.182.131
                ip_addr = 192.168.182.131  OFFLINE
```

```
        Storage 1:
                id = 192.168.182.131
                ip_addr = 192.168.182.131  OFFLINE
                http domain =
                version = 5.05
                join time = 2015-12-03 09:54:29
                up time =
                total storage = 36242 MB
                free storage = 29480 MB
                upload priority = 10
                store_path_count = 2
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id =
                if_trunk_server = 0
                connection.alloc_count = 0
                connection.current_count = 0
                connection.max_count = 0
                total_upload_count = 22
```

启动不报错，并且可以看到只有 1 个 group1，group1 中只有一个 storage1，状态是 OFFLINE。

2. 启动 back_storage1 服务器，并检查服务器状态。

启动命令：fdfs_storaged /etc/fdfs/storage.conf start

```
[root@back_storage1 fastdfs]# fdfs_storaged /etc/fdfs/storage.conf start
[root@back_storage1 fastdfs]#
```

如果启动过程不报错，启动后查看 tracker 服务器的状态：

```
[root@back_tracker1 fdfs]# fdfs_monitor /etc/fdfs/client.conf list
[2015-12-09 13:16:44] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=1, anti_steal_token=0, anti_stea
l_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 192.168.182.111:22122

group count: 1

Group 1:
group name = group1
disk total space = 27026 MB
disk free space = 20240 MB
trunk free space = 0 MB
storage server count = 1
active server count = 1
storage server port = 23000
storage HTTP port = 8888
store path count = 2
subdir count per path = 256
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.182.131
                ip_addr = 192.168.182.131  ACTIVE
```

发现此时 storage 节点的状态变为 active。

3. 启动 back_storage2 服务器，并检查服务器状态。

back_tracker1 的状态：

```
[root@back_tracker1 ~]# fdfs_monitor /etc/fdfs/client.conf list
[2015-12-09 13:42:14] DEBUG - base_path=/client, connect_timeout=30, network_timeout=60, tracker_server_count=1, anti_steal_token=0, anti_stea
l_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 192.168.182.111:22122

group count: 1

Group 1:
group name = group1
disk total space = 27026 MB
disk free space = 20062 MB
trunk free space = 0 MB
storage server count = 2
active server count = 2
storage server port = 23000
storage HTTP port = 8888
store path count = 2
subdir count per path = 256
current write server index = 0
current trunk file id = 0

        Storage 1:
                id = 192.168.182.131
                ip_addr = 192.168.182.131  ACTIVE
```

```
        Storage 2:
                id = 192.168.182.132
                ip_addr = 192.168.182.132  ACTIVE
                http domain =
                version = 5.05
                join time = 2015-12-09 13:40:14
                up time = 2015-12-09 13:40:14
                total storage = 27026 MB
                free storage = 20062 MB
                upload priority = 10
                store_path_count = 2
                subdir_count_per_path = 256
                storage_port = 23000
                storage_http_port = 8888
                current_write_path = 0
                source storage id = 192.168.182.131
                if_trunk_server = 0
                connection.alloc_count = 256
                connection.current_count = 1
                connection.max_count = 1
                total_upload_count = 0
                success_upload_count = 0
                total_append_count = 0
                success_append_count = 0
                total_modify_count = 0
```

状态信息显示 storage2 成功添加到集群中，对比 storage1 和 storage2 存储目录下的数据：

back_storage1 两个存储目录文件列表：

```
[root@back_storage1 00]# pwd
/base/storage/data/00/00
[root@back_storage1 00]# ls
wKi2elZf3b-AXC3yAAAAIqByaB80309930  wKi2eVZf3iCAApahAAAAIqByaB82288999  wKi2eVZgCDGAMFIOAAAAF849XuM0455345
wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080  wKi2eVZf3iCABsdQAAAALEUe5vw2190367  wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132  wKi2eVZf3iCAHldHAAAACrhx9-U5734081  wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
wKi2elZf3iCANXmmAAAALPpsm903898002  wKi2eVZf3iCANopJAAAANBLU12g2152289  wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
wKi2elZf3iCATqahAAAAMSE-2l80296494  wKi2eVZf3iCAT7R6AAAALsXob183732522  wKi2eVZmaOiAdUFBAAAAIqByaB83040598
wKi2elZf3iCAXUQ5AAAANzBubzw6176160  wKi2eVZg4nmAMZnLAAAAF849XuM3985988  wKi2glZnyrWAXKIMAAAZzbH0pNg297.txt
wKi2elZg4oWAUbPUAAAAF849XuM7141647  wKi2eVZg4qaALEtfAAAAF849XuM0161655  wKi2hFZnytuAZhYLAAACA5u0fik131.txt
wKi2elZgB3-AcfQTAAAAF849XuM4843492  wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143  wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
wKi2elZgBzOAEwhAAAAAF849XuM8946663  wKi2eVZgA9-ARbzUAAAAF849XuM1814323  wKi2Z1ZhL7SAQm1iAGegs9vls40995.pdf
wKi2elZgCCqAe853AAAAF849XuM3656860  wKi2eVZgBdKAV8l9AAAAF849XuM4710938  wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
wKi2elZg-KWEV7URAAAAAMh39_s284.txt  wKi2eVZgBvuACQOeAAAAF849XuM3405938  wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
wKi2elZg_mWECRo7AAAAACkH2jc176.jpg  wKi2eVZgBW-AQ7pPAAAAF849XuM5558800  wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
```

```
[root@back_storage1 00]# pwd
/base/storage1/data/00/00
[root@back_storage1 00]# ls
wKi2eVZmcbyAd2W2AAAAM3gt4Uo9171221  wKi2eVZmcSiAUoUhAAAAIqByaB87391837  wKi2g1ZnyuKALZ32AAAZzbH0pNg286.txt
wKi2eVZmcM6ANY-AAAAAIqByaB80521498  wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379
[root@back_storage1 00]#
```

back_storage2 两个存储目录文件列表：

```
[root@back_storage2 00]# pwd
/base/storage/data/00/00
[root@back_storage2 00]# ls
wKi2elZf3b-AXC3yAAAAIqByaB80309930  wKi2eVZf3iCAApahAAAAIqByaB82288999  wKi2eVZgCDGAMFIOAAAAF849XuM0455345
wKi2elZf3iCAFHOUAAAAMDGl-oQ4152080  wKi2eVZf3iCABsdQAAAALEUe5vw2190367  wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
wKi2elZf3iCANlZ5AAAAM3gt4Uo2221132  wKi2eVZf3iCAHldHAAAACrhx9-U5734081  wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
wKi2elZf3iCANXmmAAAALPpsm903898002  wKi2eVZf3iCANopJAAAANBLU12g2152289  wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
wKi2elZf3iCATqahAAAAMSE-2l80296494  wKi2eVZf3iCAT7R6AAAALsXob183732522  wKi2eVZmaOiAdUFBAAAAIqByaB83040598
wKi2elZf3iCAXUQ5AAAANzBubzw6176160  wKi2eVZg4nmAMZnLAAAAF849XuM3985988  wKi2g1ZnyrWAXKIMAAAZzbH0pNg297.txt
wKi2elZg4oWAUbPUAAAAF849XuM7141647  wKi2eVZg4qaALEtfAAAAF849XuM0161655  wKi2hFZnytuAZhYLAAACA5u0fik131.txt
wKi2elZgB3-AcfQTAAAAF849XuM4843492  wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143  wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
wKi2elZgBzOAEwhAAAAAF849XuM8946663  wKi2eVZgA9-ARbzUAAAAF849XuM1814323  wKi2Z1ZhL7SAQmliAGegs9vls40995.pdf
wKi2elZgCCqAe853AAAAF849XuM3656860  wKi2eVZgBdKAV8l9AAAAF849XuM4710938  wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
wKi2elZg-KWEV7URAAAAAMh39_s284.txt  wKi2eVZgBvuACQOeAAAAF849XuM3405938  wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
wKi2elZg_mWECRo7AAAAACkH2jc176.jpg  wKi2eVZgBW-AQ7pPAAAAF849XuM5558800  wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
[root@back_storage2 00]#
```

```
[root@back_storage2 00]# pwd
/base/storage1/data/00/00
[root@back_storage2 00]# ls
wKi2eVZmcbyAd2W2AAAAM3gt4Uo9171221  wKi2eVZmcSiAUoUhAAAAIqByaB87391837  wKi2g1ZnyuKALZ32AAAZzbH0pNg286.txt
wKi2eVZmcM6ANY-AAAAAIqByaB80521498  wKi2eVZmcV2ANil6AAAAM3gt4Uo8758379
[root@back_storage2 00]#
```

数据一样说明同步没有问题。

20.2.6 测试文件的上传、下载和删除功能

20.2.6.1 测试文件的上传功能

```
[root@back_tracker1 ~]# fdfs_upload_file /etc/fdfs/client.conf /root/2.txt
group1/M00/00/00/wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
[root@back_tracker1 ~]#
```

提示文件上传成功，到 storage 服务器存储目录下查看：

```
[root@back_storage1 storage]# cd data/00/00/
[root@back_storage1 00]# pwd
/base/storage/data/00/00
[root@back_storage1 00]# ll wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
-rw-r--r--. 1 root root 6605 Dec  9 13:19 wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
[root@back_storage1 00]#
```

文件确实存在，这说明迁移后上传功能正常。

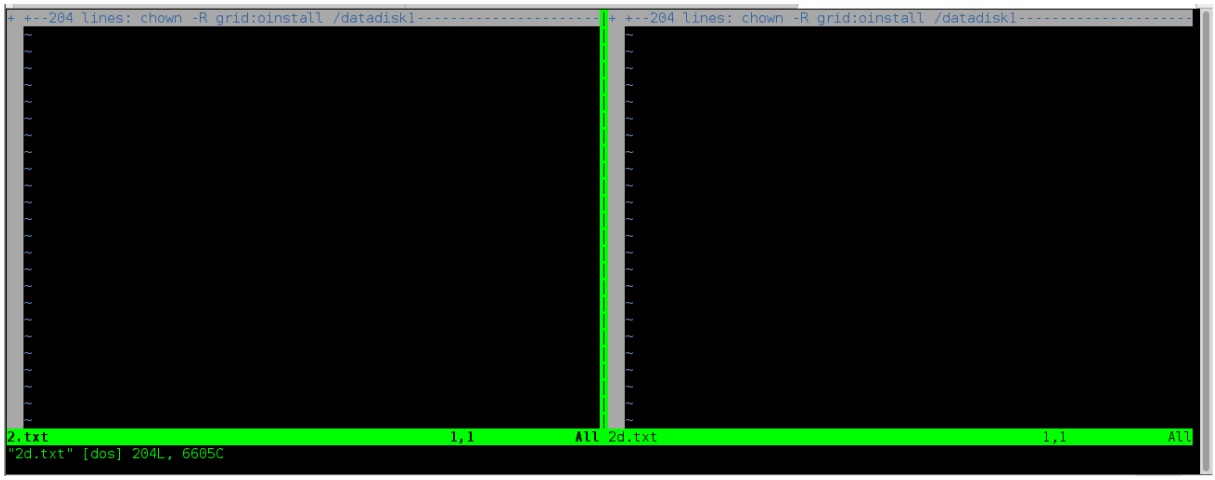
20.2.6.2 测试文件的下载功能

```
[root@back_tracker1 ~]# fdfs_download_file /etc/fdfs/client.conf group1/M01/00/00/wKi2g1ZnurWAUt3QAAAZzbH0pNg194.txt 2d.txt
[root@back_tracker1 ~]# ll
total 476
-rw-------. 1 root root     515 Dec  4 15:25 1.txt
-rw-r--r--. 1 root root    6605 Dec  9 13:24 2d.txt
-rw-------. 1 root root    6605 Dec  4 15:25 2.txt
```

文件下载成功，查看上传前文件和下载后文件的 crc32 校验码也是一样：

```
[root@back_tracker1 ~]# fdfs_crc32 2d.txt
2985600216
[root@back_tracker1 ~]# fdfs_crc32 2.txt
2985600216
[root@back_tracker1 ~]#
```

使用 vimdiff 2.txt 2d.txt 命令查看两个文件的内容是否变化，发现内容没有差异：



这说明迁移后下载功能正常。

20.2.6.3 测试文件的删除功能

删除前 sotrage 节点中的文件：

```
[root@back_storage1 00]# ll -St
total 17716
-rw-r--r--. 1 root root    6605 Dec  9 13:19 wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
-rw-r--r--. 1 root root      34 Dec  9 09:41 wKi2eVZmaOiAdUFBAAAAIqByaB83040598
-rw-r--r--. 1 root root    7120 Dec  9 09:41 wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
-rw-r--r--. 1 root root  707125 Dec  9 09:41 wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
-rw-r--r--. 1 root root 6791347 Dec  9 09:41 wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
-rw-r--r--. 1 root root 6791347 Dec  9 09:41 wKi2Z1ZhL7SAQmliAGegs9vls40995.pdf
-rw-r--r--. 1 root root  263734 Dec  9 09:41 wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
-rw-r--r--. 1 root root  263734 Dec  9 09:41 wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
-rw-r--r--. 1 root root  717824 Dec  9 09:41 wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
-rw-r--r--. 1 root root  705250 Dec  9 09:41 wKi2elZg_mWECRo7AAAAACkH2jc176.jpg
-rw-r--r--. 1 root root 1759365 Dec  9 09:41 wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
```

执行删除命令：

```
[root@back_tracker1 ~]# fdfs_delete_file /etc/fdfs/client.conf group1/M00/00/00/wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
[root@back_tracker1 ~]#
```

查看 sotrage 节点中的文件，wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt 文件已删除：

```
[root@back_storage1 00]# ll -St
total 17708
-rw-r--r--. 1 root root      34 Dec  9 09:41 wKi2eVZmaOiAdUFBAAAAIqByaB83040598
-rw-r--r--. 1 root root    7120 Dec  9 09:41 wKi2eVZk5NCEZtiNAAAAAJu0fik687.txt
-rw-r--r--. 1 root root  707125 Dec  9 09:41 wKi2Z1ZhMt-AITT7AArKNe-jPsE885.pdf
-rw-r--r--. 1 root root 6791347 Dec  9 09:41 wKi2Z1ZhMZ-AK4GHAGegs9vls40230.pdf
-rw-r--r--. 1 root root 6791347 Dec  9 09:41 wKi2Z1ZhL7SAQmliAGegs9vls40995.pdf
-rw-r--r--. 1 root root  263734 Dec  9 09:41 wKi2eVZhLkmAIULKAAQGNikH2jc512.jpg
-rw-r--r--. 1 root root  263734 Dec  9 09:41 wKi2Z1ZhLj2ABEA5AAQGNikH2jc044.jpg
-rw-r--r--. 1 root root  717824 Dec  9 09:41 wKi2Z1ZhJVGEQ_HaAAAAAFd86ZI305.ppt
-rw-r--r--. 1 root root  705250 Dec  9 09:41 wKi2elZg_mWECRo7AAAAACkH2jc176.jpg
-rw-r--r--. 1 root root 1759365 Dec  9 09:41 wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
-rw-r--r--. 1 root root     448 Dec  9 09:41 wKi2elZg-KWEV7URAAAAAMh39_s284.txt
-rw-r--r--. 1 root root      74 Dec  9 09:41 wKi2eVZg8kaEEmqEAAAAAHgt4Uo4489143
```

再次尝试下载出错，提示文件不存在：

```
[root@back_tracker1 ~]# fdfs_delete_file /etc/fdfs/client.conf group1/M00/00/00/wKi2g1Znud6AKhPbAAAZzbH0pNg715.txt
[2015-12-09 13:36:53] ERROR - file: tracker_proto.c, line: 48, server: 192.168.182.131:23000, response status 2 != 0
delete file fail, error no: 2, error info: No such file or directory
[root@back_tracker1 ~]#
```

这说明迁移后删除功能正常。

到此 FastDFS 文件的手动迁移完成。

21 FastDFS 性能调整

21.1 关于 Storage 之间的同步速度

由于到每台机器的同步都是由单线程负责的，因此即使主机上有很多个磁盘也不能加快同步速度，因此单线程并不能最大化磁盘 IO，**最大的速度也就是一个磁盘的速度**。

对于重要文件，为了防止机器宕机或者坏磁盘的文件丢失，及时的同步备份是必须得。若同步太慢，还可能导致 Tracker 不能正确估计可用空间，而导致空间溢出。

由于 Linux 系统的文件缓存机制，若一个刚写入的文件马上读，那么很可能就命中在系统缓存上，此时读数据几乎不耗费时间，若不能命中缓存，从磁盘读，将耗费很长时间。因此写入速度，最好与同步速度相当，这样可以保证及时同步。若写入速度超过同步速度，从某个时刻开始，将会使得所有的读无法命中缓存，同步性能将大幅下滑。

影响 storage 同步速度的参数：

sync_binlog_buff_interval=10

同步 binglog（更新操作日志）到硬盘的时间间隔，单位为秒，默认值 60 秒。本参数会影响新上传文件同步延迟时间。

sync_wait_msec=50

同步等待时间，单位毫秒。同步文件时，如果从 binlog 中没有读到要同步的文件，休眠 N 毫秒后重新读取。0 表示不休眠，立即再次尝试读取。出于 CPU 消耗考虑，不建议设置为 0。如果希望同步尽可能快一些，可以将本参数设置得小一些，比如设置为 10ms。

21.2 关于磁盘读写速度

FastDFS 的推荐是多个 group，每个 group 包括多个 storage，每个 storage 中单独挂载一个磁盘或多个磁盘（注意是单独的磁盘，不是一个磁盘的多个分区），这样可以将文件系统的 I/O 分散到多个磁盘，适当提升服务器的 I/O 性能。

21.3 可以提升 storage 性能的参数

use_connection_pool = false

是否使用连接池，在资源比较紧张的情况下建议使用连接池，但是具体性能的提升程度是多少，这个还需要测试。

connection_pool_max_idle_time = 3600

连接池中连接的最大空闲时间，单位秒，默认值 3600，可以适当调低连接的最大空闲时间，提升连接的利用率。

max_connections=256

服务器支持的最大连接数，最大连接数越大，能同时处理的任务量越大，但是消耗的资源也越多，可以根据系统的资源适当调整此值来获取最佳性能。

buff_size = 256KB

设置队列结点的 buffer 大小。工作队列消耗的内存大小 = buff_size * max_connections。设置得大一些，系统整体性能会有所提升。对于 32 位系统，请注意使用到的内存不要超过 3GB。

work_threads=4

工作线程数，通常设置为 CPU 数，配置的是 NIO 线程，也就是网络线程数，此值越大，服务器处理能力越强，但是不能超过 CPU 数据，超过后性能无提升。

thread_stack_size=512KB

线程栈的大小。FastDFS server 端采用了线程方式。 线程栈越大，一个线程占用的系统资源就

越多。torage server 线程栈不应不小于 512KB，可以适当调大。

22 FastDFS 常用命令介绍

FastDFS 安装后，命令在/usr/bin 目录下，以 fdfs_开头，如下所示：

```
[root@oel7 bin]# ll |grep fdfs
-rwxr-xr-x  1 root root     320715 Nov 25 16:23 fdfs_appender_test
-rwxr-xr-x  1 root root     320492 Nov 25 16:23 fdfs_appender_test1
-rwxr-xr-x  1 root root     307372 Nov 25 16:23 fdfs_append_file
-rwxr-xr-x  1 root root     306824 Nov 25 16:23 fdfs_crc32
-rwxr-xr-x  1 root root     307407 Nov 25 16:23 fdfs_delete_file
-rwxr-xr-x  1 root root     308166 Nov 25 16:23 fdfs_download_file
-rwxr-xr-x  1 root root     307748 Nov 25 16:23 fdfs_file_info
-rwxr-xr-x  1 root root     321569 Nov 25 16:23 fdfs_monitor
-rwxr-xr-x  1 root root    1119020 Nov 25 16:23 fdfs_storaged
-rwxr-xr-x  1 root root     330659 Nov 25 16:23 fdfs_test
-rwxr-xr-x  1 root root     325780 Nov 25 16:23 fdfs_test1
-rwxr-xr-x  1 root root     458636 Nov 25 16:23 fdfs_trackerd
-rwxr-xr-x  1 root root     308358 Nov 25 16:23 fdfs_upload_appender
-rwxr-xr-x  1 root root     309378 Nov 25 16:23 fdfs_upload_file
```

fdfs_appender_test：测试命令，正式生产环境中不使用。
fdfs_appender_test1：测试命令，正式生产环境中不使用。
fdfs_append_file：
fdfs_crc32：
fdfs_delete_file：删除文件。
fdfs_download_file：下载文件。
fdfs_file_info：查看文件信息
fdfs_monitor：FastDFS 管理命令。
fdfs_storaged：启动、关闭、重启存储节点（Storage）的命令。
fdfs_test：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，正式生产环境中不使用。
fdfs_test1：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，正式生产环境中不使用。
fdfs_trackerd：启动、关闭、重启跟踪器（Tracker）的命令。
fdfs_upload_appender：
fdfs_upload_file：上传文件。

在终端中直接输入命令，不带任何参数，可以查看命令的使用方法，如下：

```
[root@16:04:58@rhel7 /usr/bin]# fdfs_trackerd
Usage: fdfs_trackerd <config_file> [start | stop | restart]
```

主从文件说明：

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：

先上传主文件（如原文件），得到主文件 ID 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。

22.1 文件附加：fdfs_append_file

语法：

fdfs_append_file <config_file> <appender_file_id> <local_filename>

<config_file>：客户端配置文件

<appender_file_id>：使用 fdfs_upload_appender 命令上传文件后返回的 ID

<local_filename>：本地文件

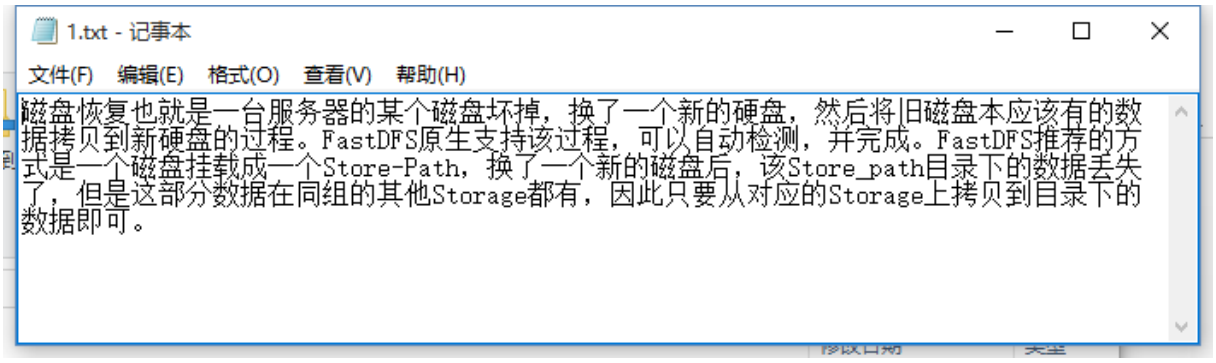
文件附加命令在开源社区中资料比较少，介绍的不是很详细，所以针对一些常用的文件格式做了如下测试，测试文件类型包括：txt、docx、xlsx、jpg、ppt，测试结果：

针对字符类型的文本文件可以合并到一块，其他特殊文件格式都不支持合并。

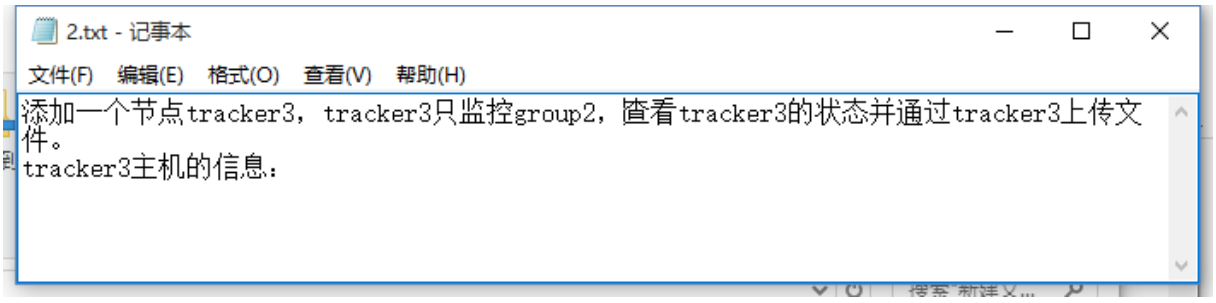
测试过程如下：

1. txt 文件测试：

1.txt 文件内容



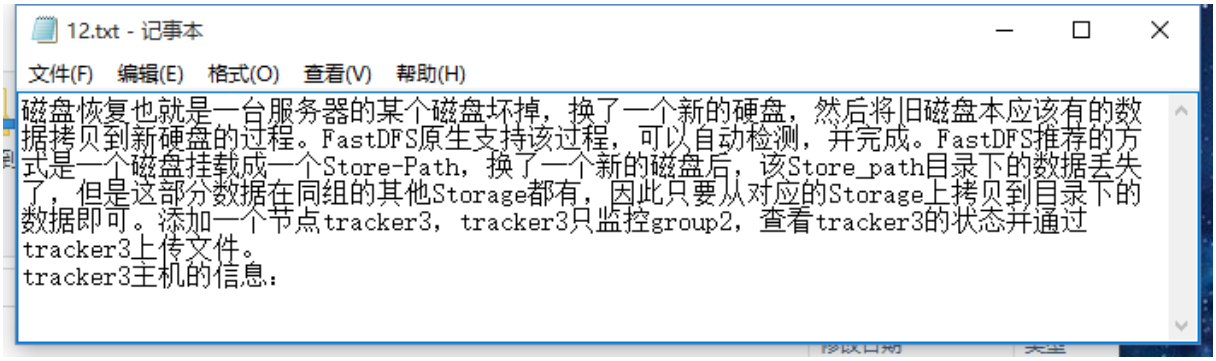
2.txt 文件内容



文件进行附加上传后，再下载到本地，查看文件内容：

```
[root@tracker1 filetest]# ll
total 11160
-rwxrw-rw- 1 root root 1708165 Dec  1 19:40 1.docx
-rwxrw-rw- 1 root root  263734 Jul 28 08:35 1.jpg
-rwxrw-rw- 1 root root 6791347 Aug 15 22:54 1.pdf
-rwxrw-rw- 1 root root  360448 Dec  3 17:59 1.ppt
-rwxrw-rw- 1 root root     338 Dec  3 18:01 1.txt
-rwxrw-rw- 1 root root   30290 Dec  3 18:02 1.xlsx
-rwxrw-rw- 1 root root   51149 Oct 27 08:28 2.docx
-rwxrw-rw- 1 root root  441516 Dec 18  2014 2.jpg
-rwxrw-rw- 1 root root  707125 Dec 20  2014 2.pdf
-rwxrw-rw- 1 root root  357376 Dec  3 18:00 2.ppt
-rwxrw-rw- 1 root root     110 Dec  3 18:01 2.txt
-rwxrw-rw- 1 root root  685359 Nov  1 19:25 2.xlsx
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.txt
group1/M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group1/M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt 2.txt
```

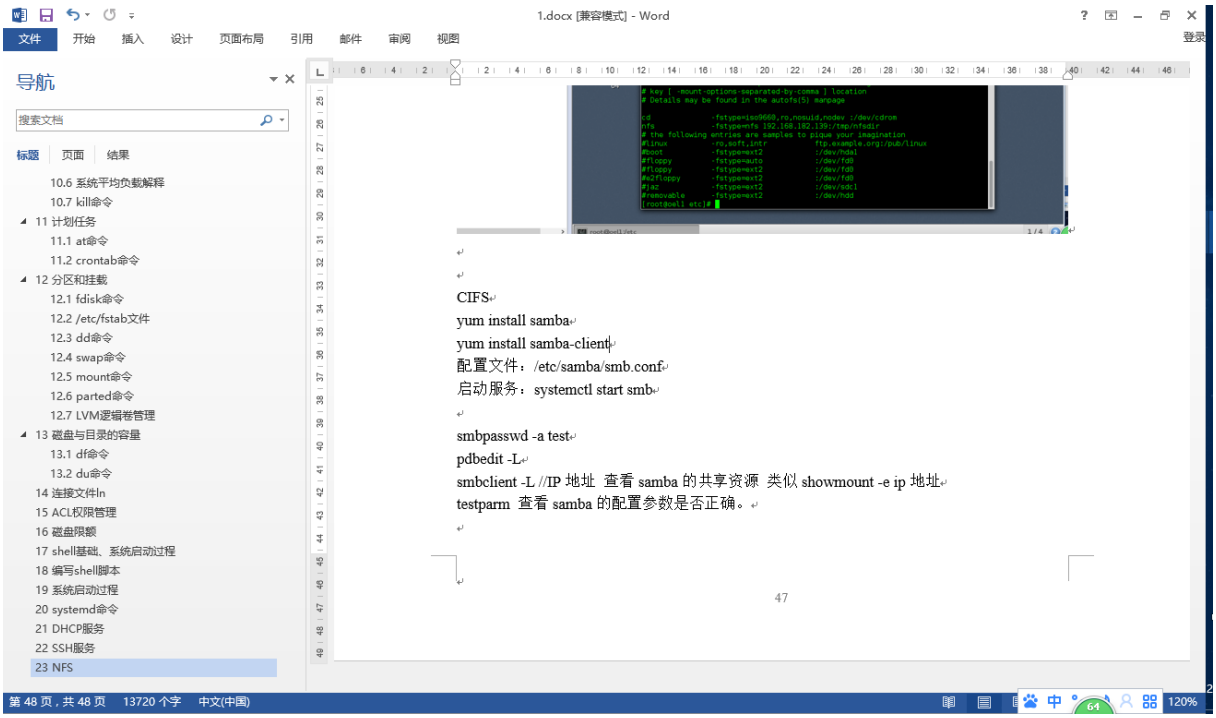
```
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2elZg-KWEV7URAAAAAMh39_s284.txt 12.txt
[root@tracker1 filetest]#
```



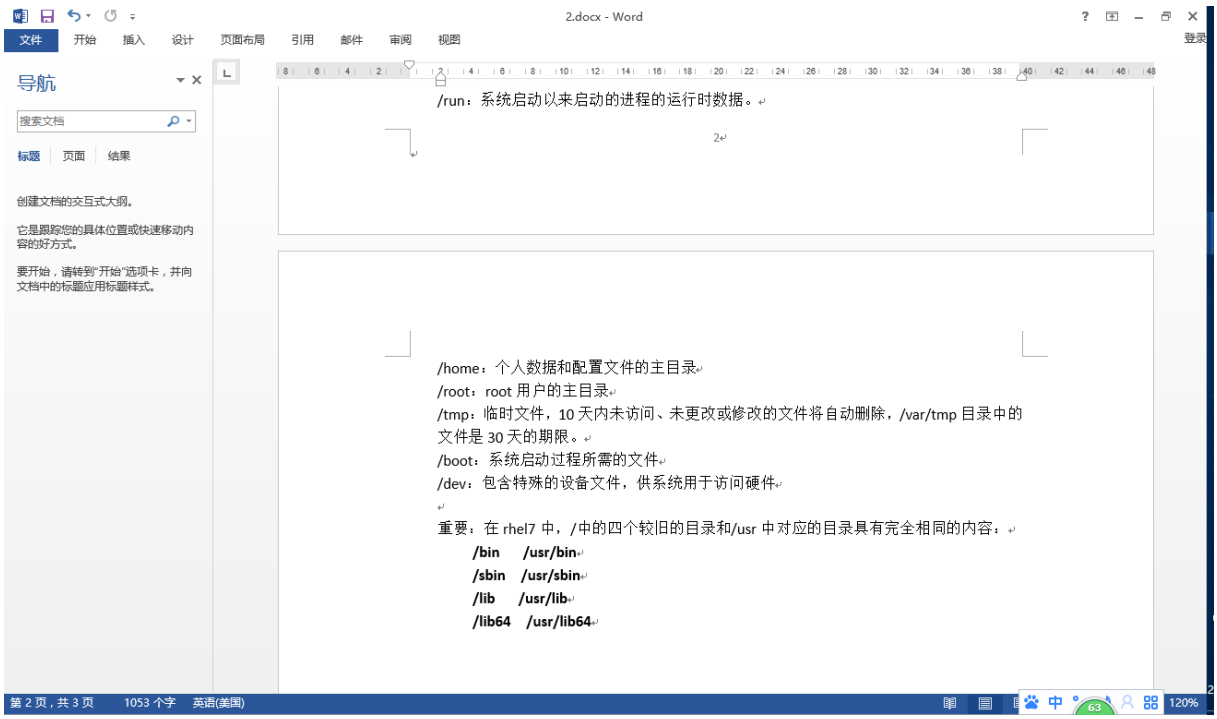
可以看到是 txt 文件是将两个文件的内容直接合并到了一个文件中。

2. docx 文件测试：

1.docx 内容：



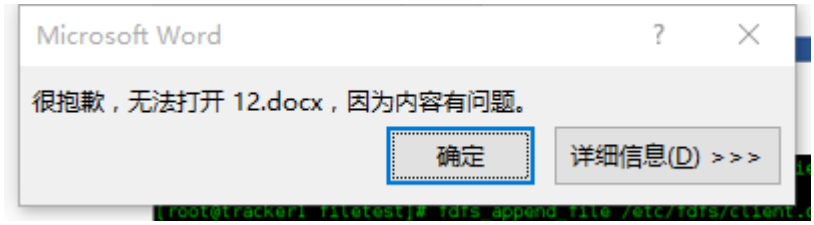
2.docx 内容：



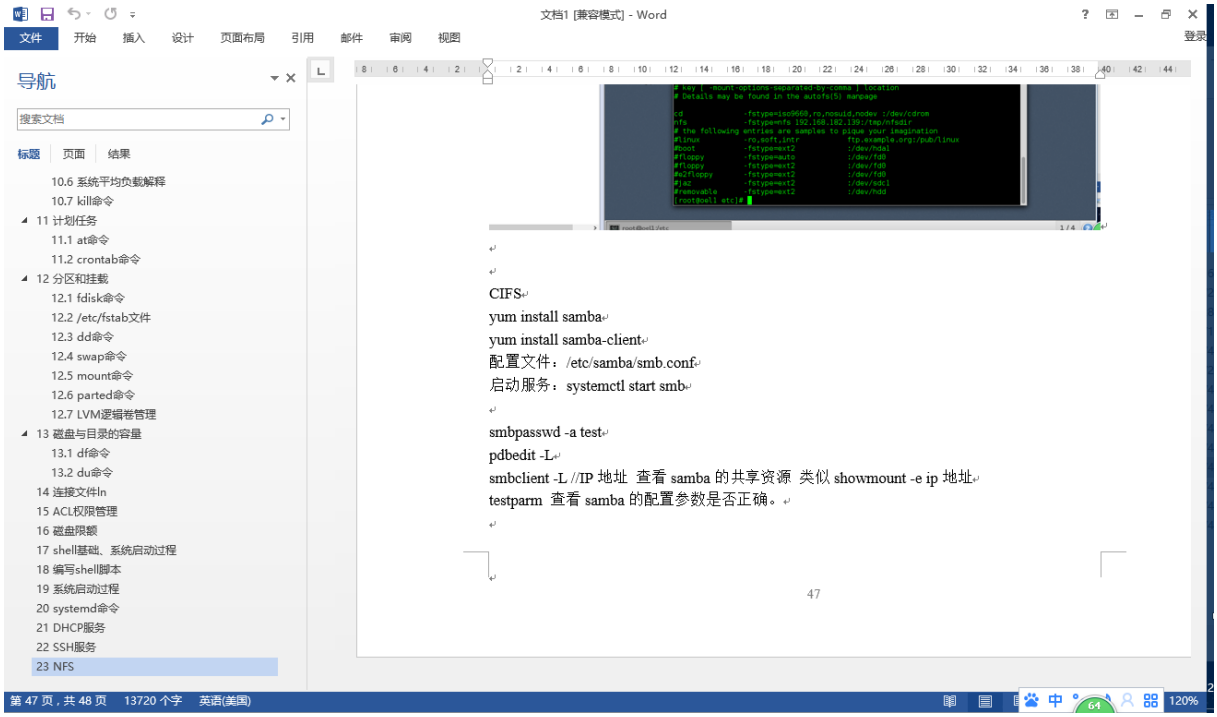
测试过程：

```
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.docx
group1/M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx 2.docx
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2eVZg-tuEesvaAAAAAJ0Wlpo33.docx 12.docx
[root@tracker1 filetest]#
```

下载查看合并文件报错，：



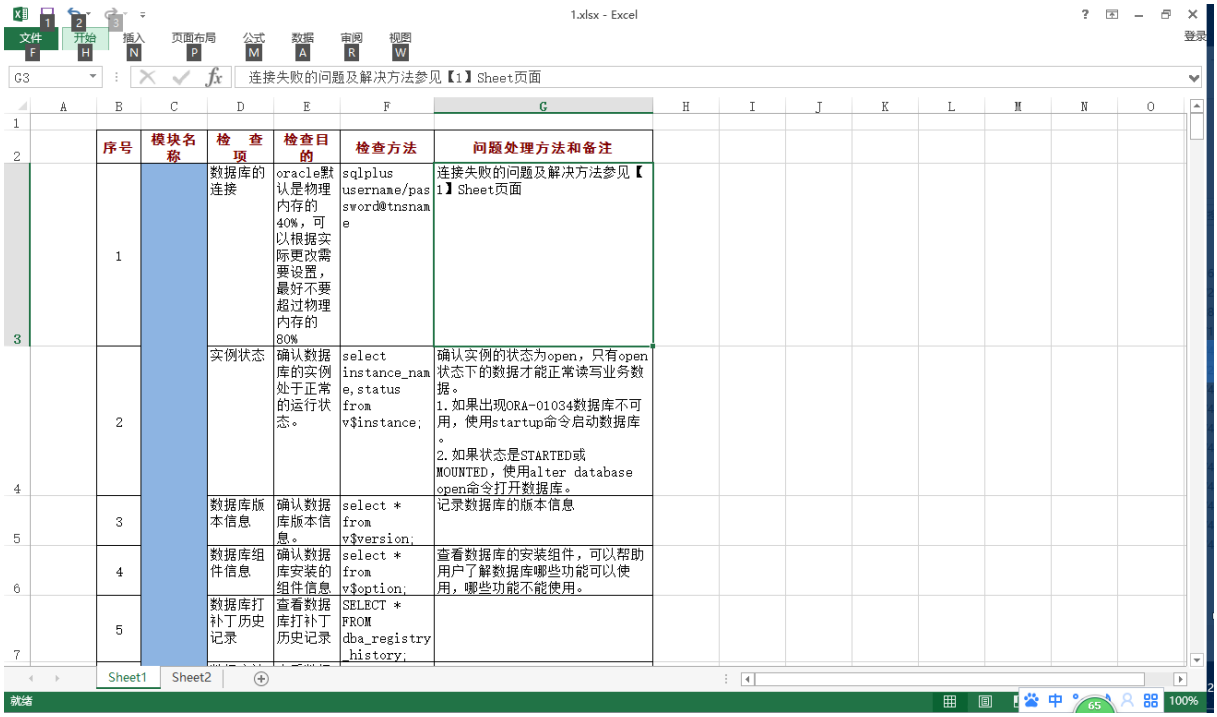
强制打开发现还是原来 1.dox 的内容：



说明 fdfs_append_file 命令对 word 类文档不支持。

3. xlsx 文件测试：

1.xlsx 内容：



序号	模块名称	检 查 项	检查目的	检查方法	问题处理方法和备注
1		数据库的连接	oracle默认是物理内存的40%，可以根据实际更改需要设置，最好不要超过物理内存的80%	sqlplus username/password@tnsname	连接失败的问题及解决方法参见【1】Sheet页面
2		实例状态	确认数据库的实例处于正常的运行状态。	select instance_name,status from v$instance;	确认实例的状态为open，只有open状态下的数据才能正常读写业务数据。 1.如果出现ORA-01034数据库不可用，使用startup命令启动数据库。 2.如果状态是STARTED或MOUNTED，使用alter database open命令打开数据库。
3		数据库版本信息	确认数据库版本信息。	select * from v$version;	记录数据库的版本信息
4		数据库组件信息	确认数据库安装的组件信息	select * from v$option;	查看数据库的安装组件，可以帮助用户了解数据库哪些功能可以使用，哪些功能不能使用。
5		数据库打补丁历史记录	查看数据库打补丁历史记录	SELECT * FROM dba_registry_history;	

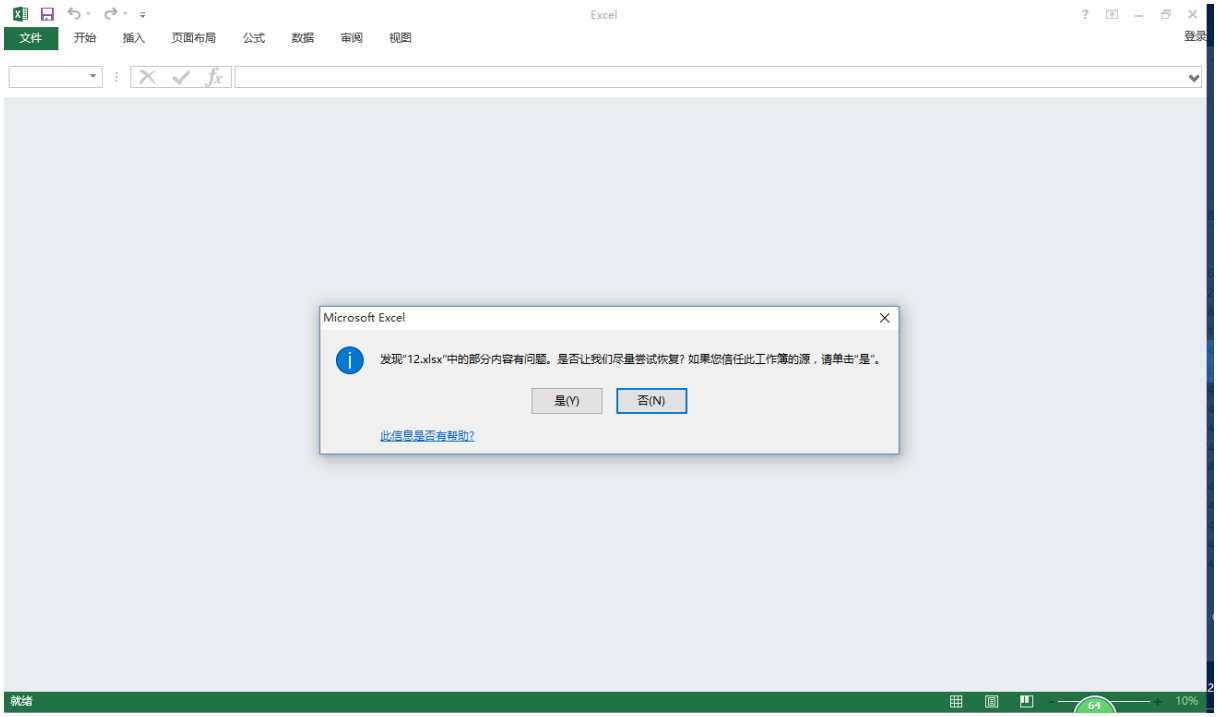
2.xlsx 内容：

序号	模块名称	检　查　项	检查目的	检查方法	问题处理方法和备注
1	数据库基本信息检查	数据库的连接	oracle默认是物理内存的40%，可以根据实际更改需要设置，最好不要超过物理内存的80%	sqlplus username/password@tnsname	连接失败的问题及解决方法参见【1】Sheet页面
2		实例状态	确认数据库的实例处于正常的运行状态。	select instance_name,status from v$instance;	确认实例的状态为open，只有open状态下的数据才能正常读写业务数 1.如果出现ORA-01034数据库不可用，使用startup命令启动数据库。 2.如果状态是STARTED或MOUNTED，使用alter database open命令打
3		数据库版本信息	确认数据库版本信息。	select * from v$version;	记录数据库的版本信息
4		数据库组件信息	确认数据库安装的组件信息	select * from v$option;	查看数据库的安装组件，可以帮助用户了解数据库哪些功能可以使用
5		数据库打补丁历史记录	查看数据库打补丁历史记录	SELECT * FROM dba_registry_history;	
6		数据库补丁信息	查看数据库补丁信息	$ORACLE_HOME/OPatch/opatch lsinventory	
7		连接用户	查看数据库当前连接的用户数。	select inst_id,username,count(*) from gv$session group by inst_id,username;	系统后台进程对应的username为null。
8		会话信息	查看数据库当前连接的会话数和历史最高会话数，以便设置合理的sessions值。	select sessions_current,sessions_highwater from v$license;	历史最大连接数主要用来作为修改sessions参数的参考值。Sessions并发用户数。
9		数据库的实例名	确保登录的正确的数据库	show parameter instance_name;	当一个数据库服务器上部署了多个数据库服务时，登录数据库后要先确数据库。
10		当前登录的用户	确认使用正确的用户登录。	show user;	可以避免一些不必要的失误，例如：登录用户不对，查询时数据库提现象。
11		数据库的归档模式	确认数据是否开启归档模式。	archive log list;	开启归档模式后，可以使用RMAN对数据库进行备份和恢复，但是需要否开始归档模式可以根据实际情况决定。 开启归档模式后需要注意的内容： 1.磁盘预留空间需要充足，空间的大小由实际的业务量、库文件的大小素决定。 2.备份的命令中添加备份完成后将归档日志文件删除，否则时间一长住后，无法提供服务。 3.最好不要手动物理删除归档日志文件，否则会导致备份失败，如果

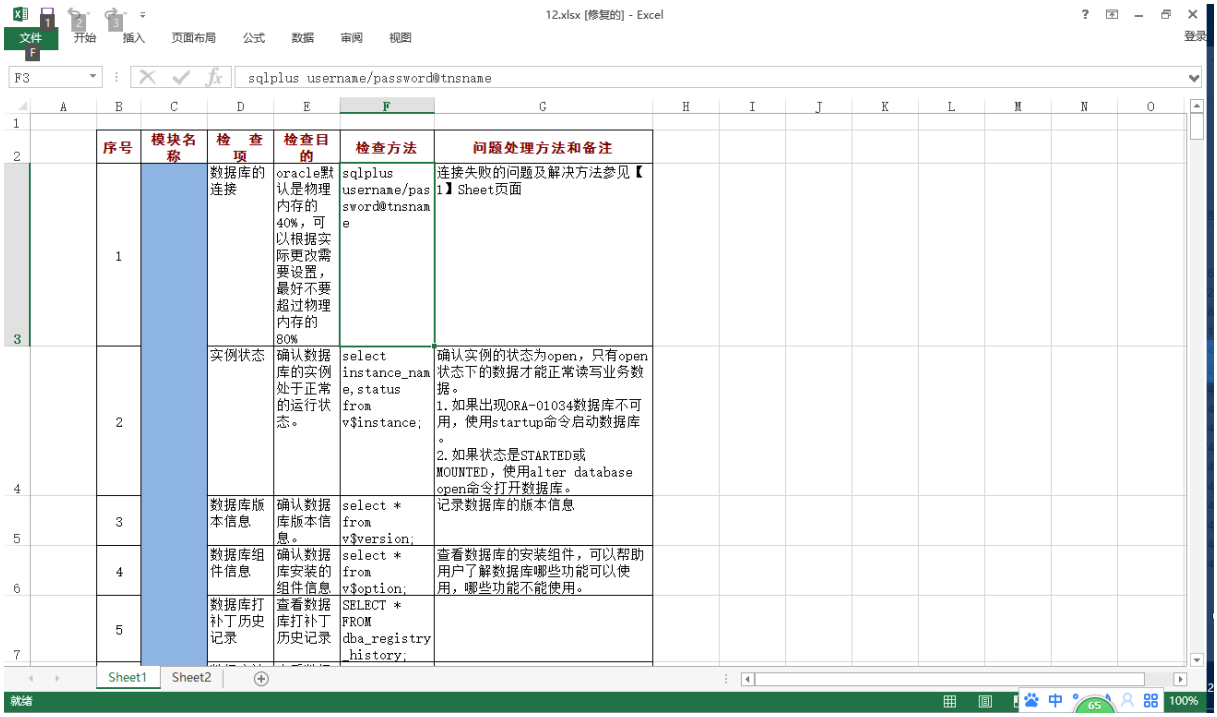
测试过程：

```
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.xlsx
group2/M00/00/00/wKi2fFZg_WOEF08YAAAAAKraBeI68.xlsx
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group2/M00/00/00/wKi2fFZg_WOEF08YAAAAAKraBeI68.xlsx 2.xlsx
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group2/M00/00/00/wKi2fFZg_WOEF08YAAAAAKraBeI68.xlsx 12.xlsx
[root@tracker1 filetest]#
```

合并后文件打开报错：



强制打开和原来 1.xlsx 相同：

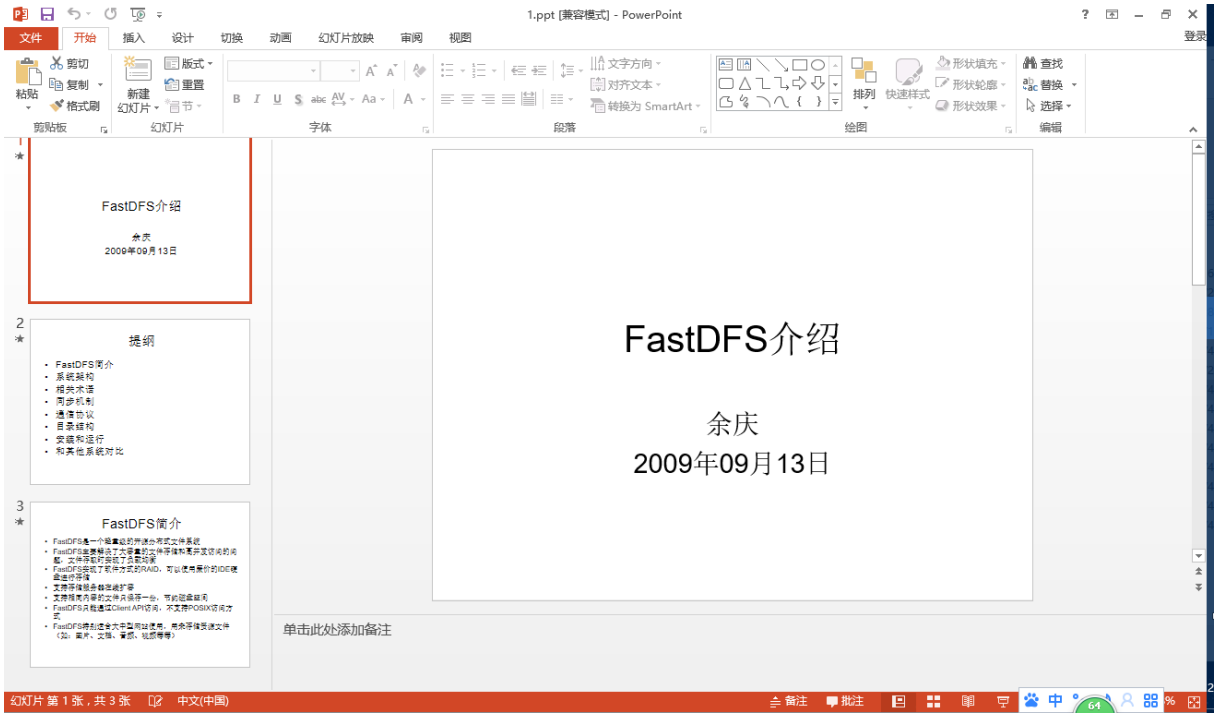


序号	模块名称	检查项	检查目的	检查方法	问题处理方法和备注
1		数据库的连接	oracle默认是物理内存的40%，可以根据实际更改需要设置，最好不要超过物理内存的80%	sqlplus username/password@tnsname	连接失败的问题及解决方法参见【1】Sheet页面
2		实例状态	确认数据库的实例处于正常的运行状态。	select instance_name,status from v$instance;	确认实例的状态为open，只有open状态下的数据才能正常读写业务数据。 1.如果出现ORA-01034数据库不可用，使用startup命令启动数据库。 2.如果状态是STARTED或MOUNTED，使用alter database open命令打开数据库。
3		数据库版本信息	确认数据库版本信息。	select * from v$version;	记录数据库的版本信息
4		数据库组件信息	确认数据库安装的组件信息	select * from v$option;	查看数据库的安装组件，可以帮助用户了解数据库哪些功能可以使用，哪些功能不能使用。
5		数据库打补丁历史记录	查看数据库打补丁历史记录	SELECT * FROM dba_registry_history;	

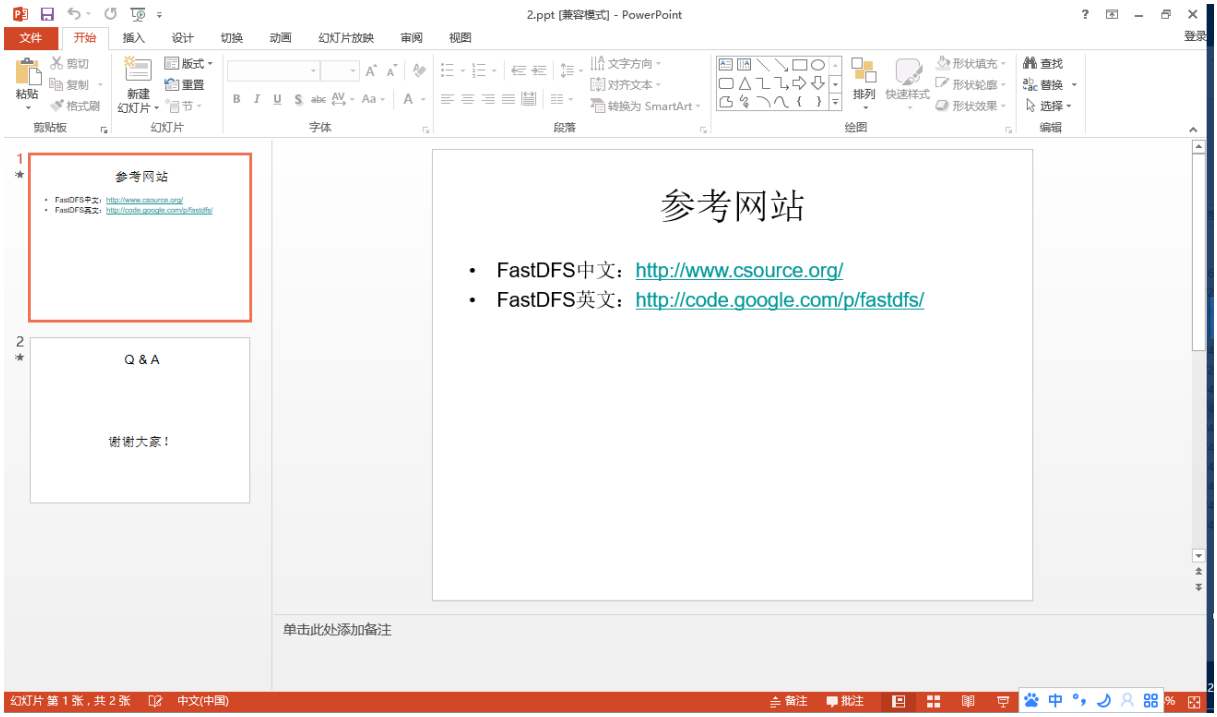
文件合并失败，所以不支持 excel 类文件的合并。

4. ppt 文件测试：

1.ppt 内容：



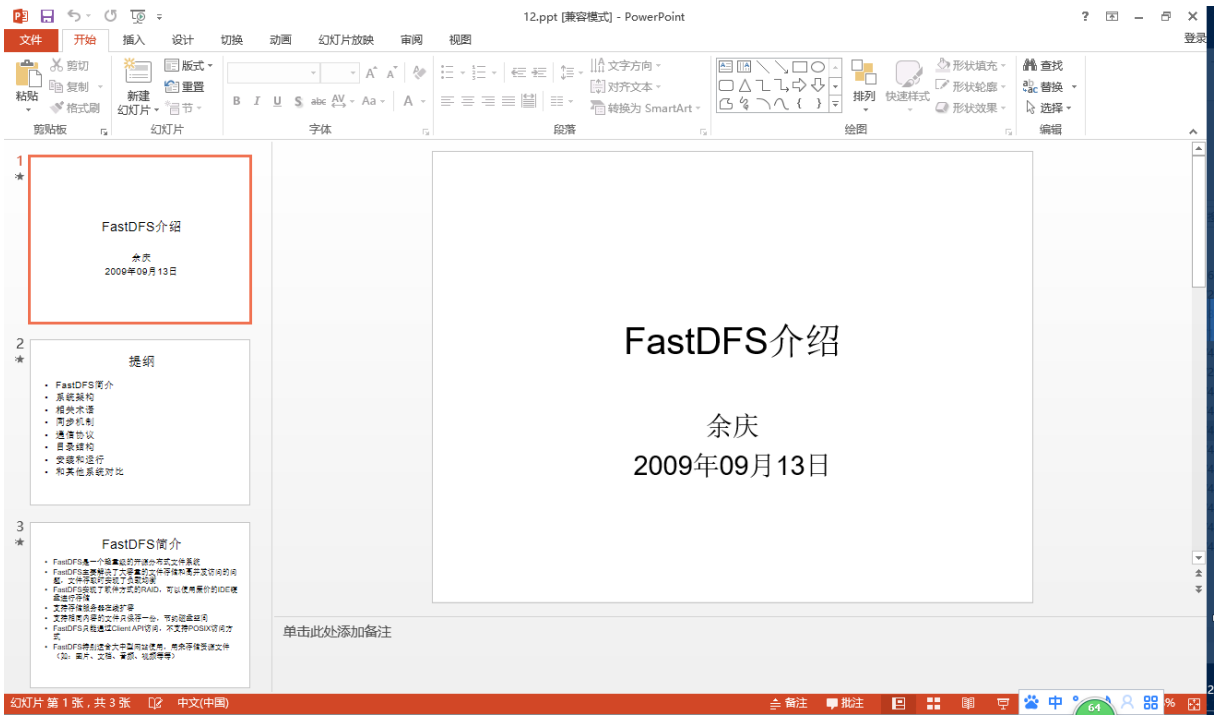
2.ppt 内容：



测试过程：

```
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.ppt
group2/M00/00/00/wKi2e1Zg_BqESiFuAAAAAFd86ZI518.ppt
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group2/M00/00/00/wKi2e1Zg_BqESiFuAAAAAFd86ZI518.ppt 2.ppt
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group2/M00/00/00/wKi2e1Zg_BqESiFuAAAAAFd86ZI518.ppt 12.ppt
[root@tracker1 filetest]#
```

查看测试结果：

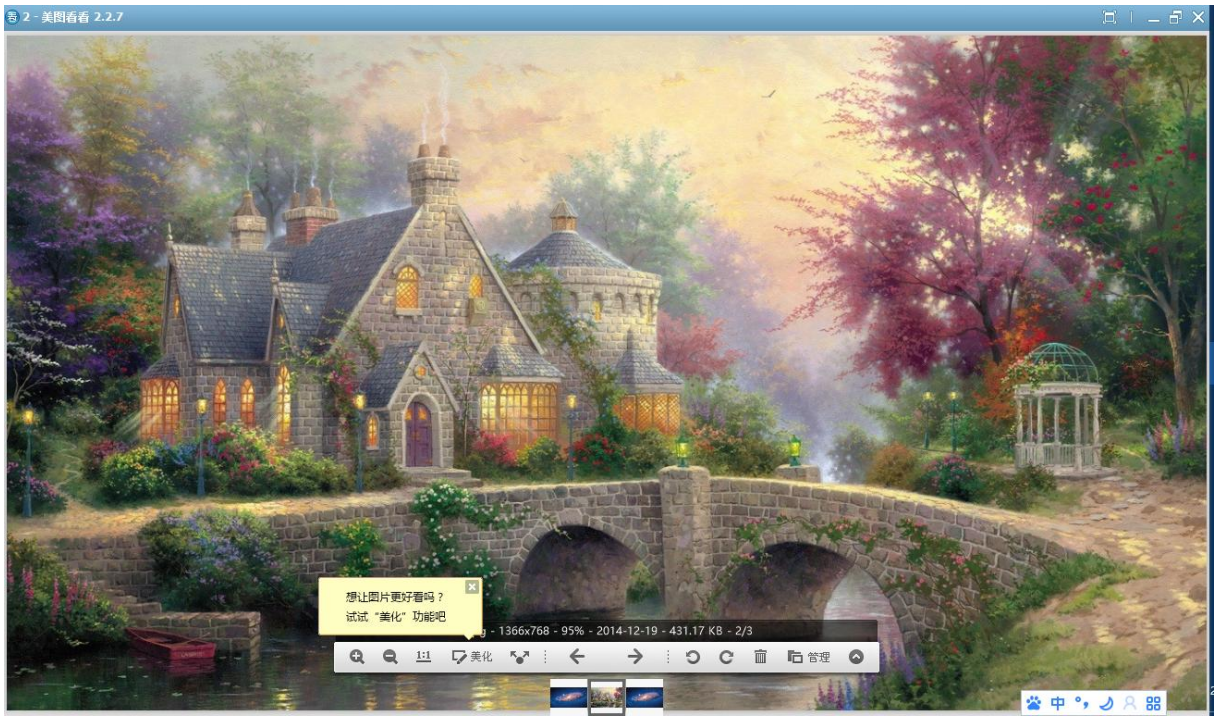


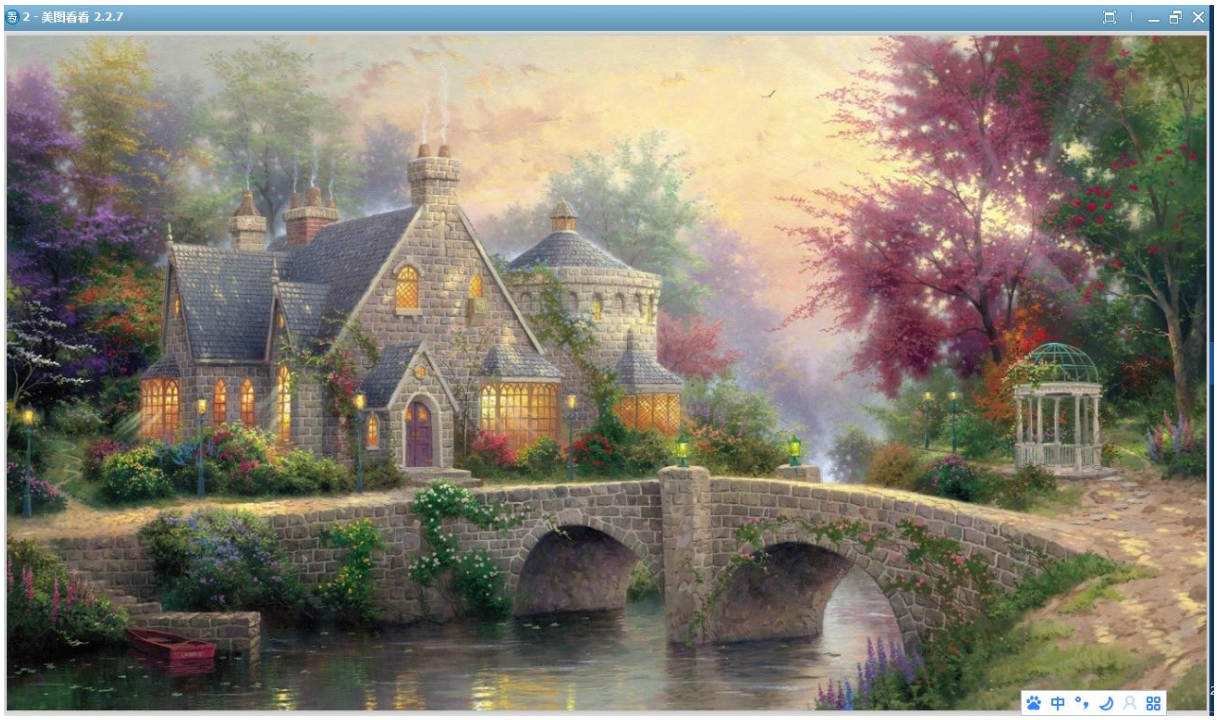
打开不报错，但是还是原理 1.ppt 内容，所以合并失败，所以不支持 ppt 文件的合并。

5. jpg 文件测试：

1.jpg 图片内容：

2.jpg 图片内容：





测试过程：

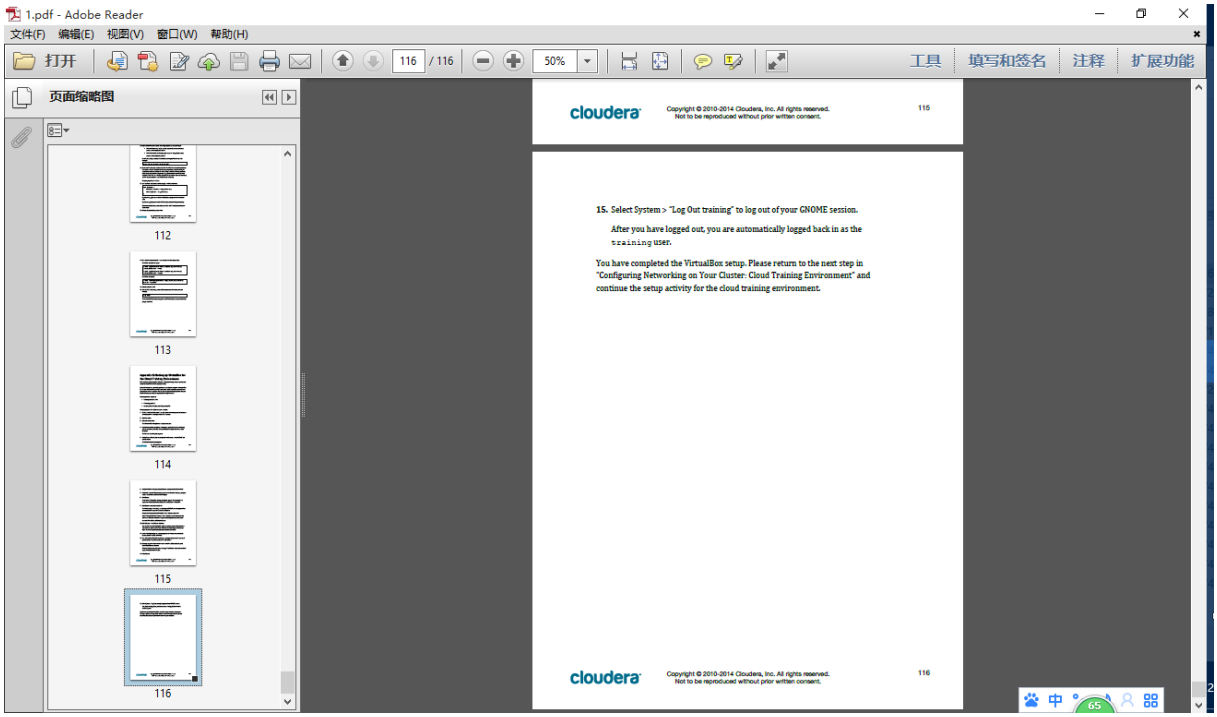
```
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.jpg
group1/M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group1/M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg 2.jpg
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group1/M00/00/00/wKi2elZg_mWECRo7AAAAACkH2jc176.jpg 12.jpg
[root@tracker1 filetest]#
```

合并后图片：

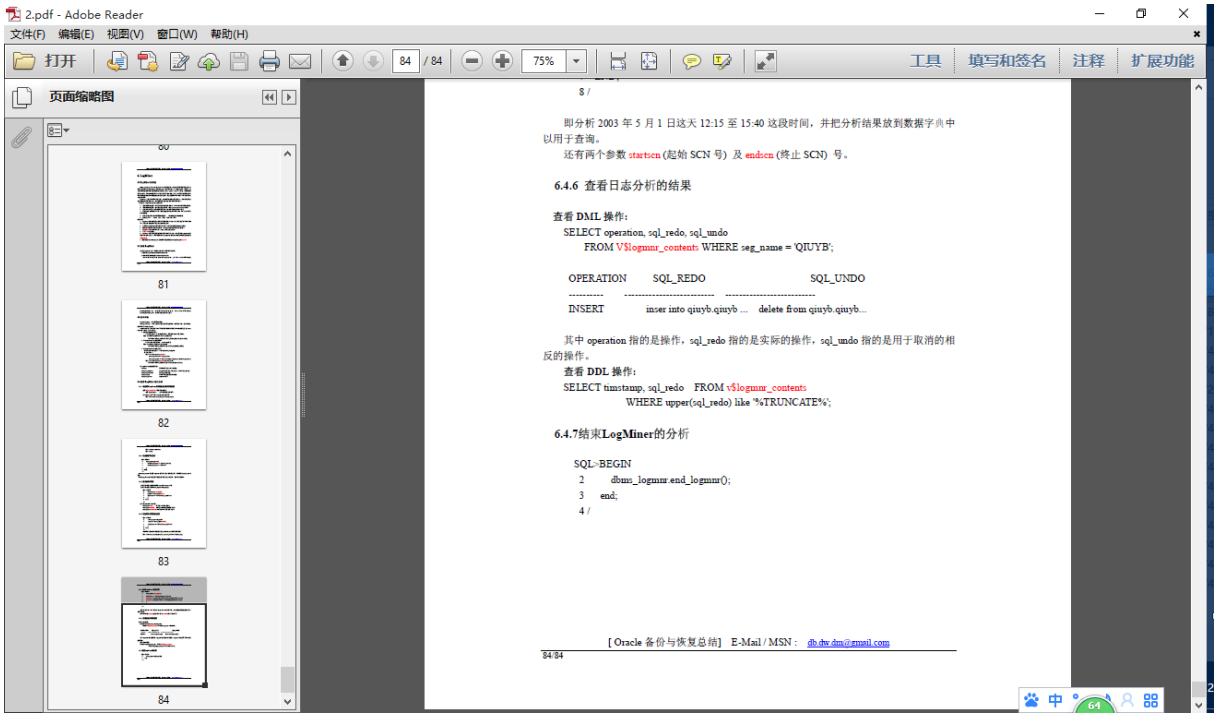
可以看出图片的合并也失败，合并后图片还是原来的 1.jpg 图片。

6. pdf 文件测试：

   1.pdf 内容：



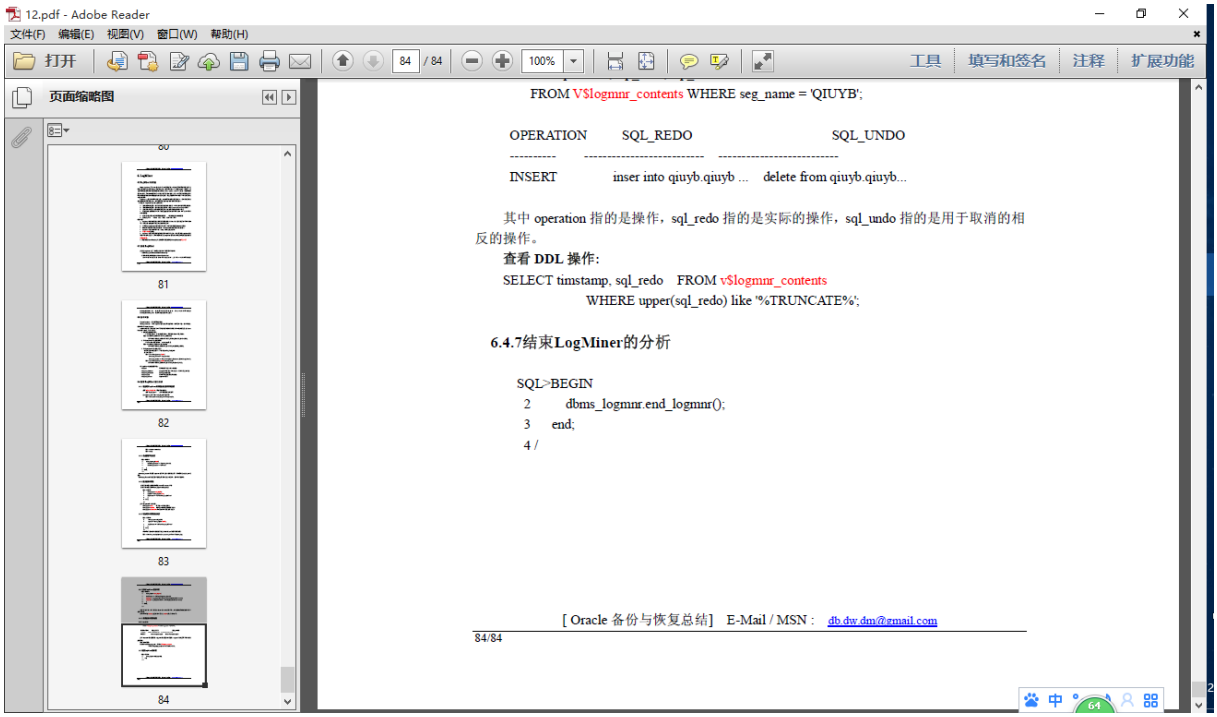
2.pdf 内容：



测试过程：

```
[root@tracker1 filetest]# fdfs_upload_appender /etc/fdfs/client.conf 1.pdf
group2/M00/00/00/wKi2e1Zg_8iETf2UAAAAANvls40545.pdf
[root@tracker1 filetest]# fdfs_append_file /etc/fdfs/client.conf group2/M00/00/00/wKi2e1Zg_8iETf2UAAAAANvls40545.pdf 2.pdf
[root@tracker1 filetest]# fdfs_download_file /etc/fdfs/client.conf group2/M00/00/00/wKi2e1Zg_8iETf2UAAAAANvls40545.pdf 12.pdf
[root@tracker1 filetest]#
```

合并后 pdf 内容：



合并后变成了 2.pdf 的内容，合并失败。

22.2 查看文件校验码：fdfs_crc32

fdfs_crc32 命令用来查看文件的 crc32 校验码

语法：

fdfs_crc32 <filename>

<filename>：文件名称

22.3 删除文件：fdfs_delete_file

语法：

fdfs_delete_file <config_file> <file_id>

<config_file>：客户端配置文件，必须有。

<file_id>：文件 ID，必须有。

22.4 下载文件：fdfs_download_file

语法：

fdfs_download_file <config_file> <file_id> [local_filename] [<download_offset> <download_bytes>]

<config_file> ：客户端配置文件，必须有

<file_id> ：要下载的文件 ID，必须有。

[local_filename] ：存储到本地的文件，选择项。

[<download_offset> <download_bytes>]：下载偏移量、下载大小，选择项。

22.5 查看文件：fdfs_file_info

语法:
fdfs_file_info <config_file> <file_id>
<config_file>：客户端配置文件，必须有。
<file_id>：文件 ID，必须有。

22.6 监控管理：fdfs_monitor

FastDFS 的监控器，可以用来监控 tracker 或 storage 的状态执行删除等操作。
语法：
fdfs_monitor <config_file> [-h <tracker_server>] [list|delete|set_trunk_server <group_name> [storage_id]]
<config_file> ：客户端配置文件名称
[-h <tracker_server>] ：tracker 服务器
[list|delete|set_trunk_server <group_name> [storage_id]]：操作选项 list：查看、delete 删除、set_trunk_server：设置合并服务器

```
[root@16:21:18@rhel7 /usr/bin]# fdfs_monitor
Usage: fdfs_monitor <config_file> [-h <tracker_server>] [list|delete|set_trunk_server <group_name> [storage_id]]
[root@16:22:23@rhel7 /usr/bin]#
```

示例：
删除无效的存储节点：
fdfs_mointor /etc/fdfs/client.conf delete group1 192.168.126.132
如果存储节点的状态是 ACTIVE，即还在线上提供服务，则无法删除。删除后，如果再次启动 Storage 存储节点，存储节点可以再次注册到 Tracker 上。

22.7 Storage 服务器启动：fdfs_storaged

Storage（存储节点）的启动、关闭和重启命令。
语法：
fdfs_storaged <config_file> [start | stop | restart]
语法查看：直接在终端中执行 fdfs_storaged

```
[root@16:12:35@rhel7 /usr/bin]# fdfs_storaged
Usage: fdfs_storaged <config_file> [start | stop | restart]
```

启动存储节点示例：fdfs_ storaged /etc/fdfs/storage.conf start

22.8 Tracker 服务器启动：fdfs_trackerd

Tracker（跟踪器）的启动、关闭和重启命令。
语法：
fdfs_trackerd <config_file> [start | stop | restart]
语法查看：直接在终端中执行 fdfs_trackerd

```
[root@16:21:13@rhel7 /usr/bin]# fdfs_trackerd
Usage: fdfs_trackerd <config_file> [start | stop | restart]
```

启动跟踪器示例：fdfs_trackerd /etc/fdfs/tracker.conf start

22.9 文件附加：fdfs_upload_appender

语法：

fdfs_upload_appender <config_file> <local_filename>
<config_file> ：客户端配置文件
<local_filename>：本地文件名称

22.10 文件上传：fdfs_upload_file

语法：
fdfs_upload_file <config_file> <local_filename> [storage_ip:port] [store_path_index]
<config_file>：客户端配置文件名称，必须有。
<local_filename>：要上传的文件名称，必须有。
[storage_ip:port]：上传的存储节点名称，选择项。
[store_path_index]：存储路径，选择项。

23 常见错误和疑问说明

23.1 Storage 服务启动时，一直卡住启动不了

启动 storage 服务，storage 将连接 tracker 服务，如果连不上，将一直重试。直到连接成功，启动才算真正完成，所以需要首先确保 tracker 服务启动。

如果启动过程中，执行命令卡住，可能是由于某些问题导致的启动失败，可以使用命令 tailf $base_path/logs/storaged.log 跟踪日志信息，$base_path 是在 storage.conf 中配置的存储跟目录。

注：在杀 tracker 和 storage 服务器进程时，请使用 killall fdfs_trackerd/killall fdfs_storaged 命令，千万不要使用 kill -9 命令强杀，否则可能会导致 binlog 数据丢失的问题。

23.2 上传文件失败，返回错误码 28

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。

可以酌情设置 reserved_storage_space 这个参数，比如可以设置为磁盘总空间的 20%左右。

23.3 日志中出现 malloc task buff failed 字样的错误

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。

23.4 FastDFS 的文件 ID 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：

- 文件创建时间（unix 时间戳，32 位整数）
- 文件大小
- 上传到的源 storage server IP 地址（32 位整数）
- 文件 crc32 校验码
- 随机数（这个字段用来避免文件重名）

24 FastDFS 原理博客推荐

《FastDFS 之 Binlog 同步》 http://blog.csdn.net/hfty290/article/details/42041155
《FastDFS 之添加机器同步》 http://blog.csdn.net/hfty290/article/details/42041953

《FastDFS 之磁盘恢复过程》 http://blog.csdn.net/hfty290/article/details/42032817
《FastDFS 之 Storage 程序框架》 http://blog.csdn.net/hfty290/article/details/42048001
《FastDFS 之客户端与 Tracker 通讯》http://blog.csdn.net/hfty290/article/details/42064429
《FastDFS 合并存储原理分析》 http://blog.csdn.net/hfty290/article/details/42026215
《FastDFS 之 Tracker-Leader 选择》 http://blog.csdn.net/hfty290/article/details/42030339
《FastDFS 之合并存储缺陷导致数据丢失或错误》 http://blog.csdn.net/hfty290/article/details/42030481