产品上云和性能测试

零声教育 Darren QQ326873713

[零声教育 C/C++Linux服务器开发/高级架构师](#)

1 只允许唯一文件上传

fastdfs本身不会去检测文件内容是否重复（这里只是简单拓展）。

storage.conf 配置

if check file duplicate, when set to true, use FastDHT to store file indexes# 1 or yes: need check

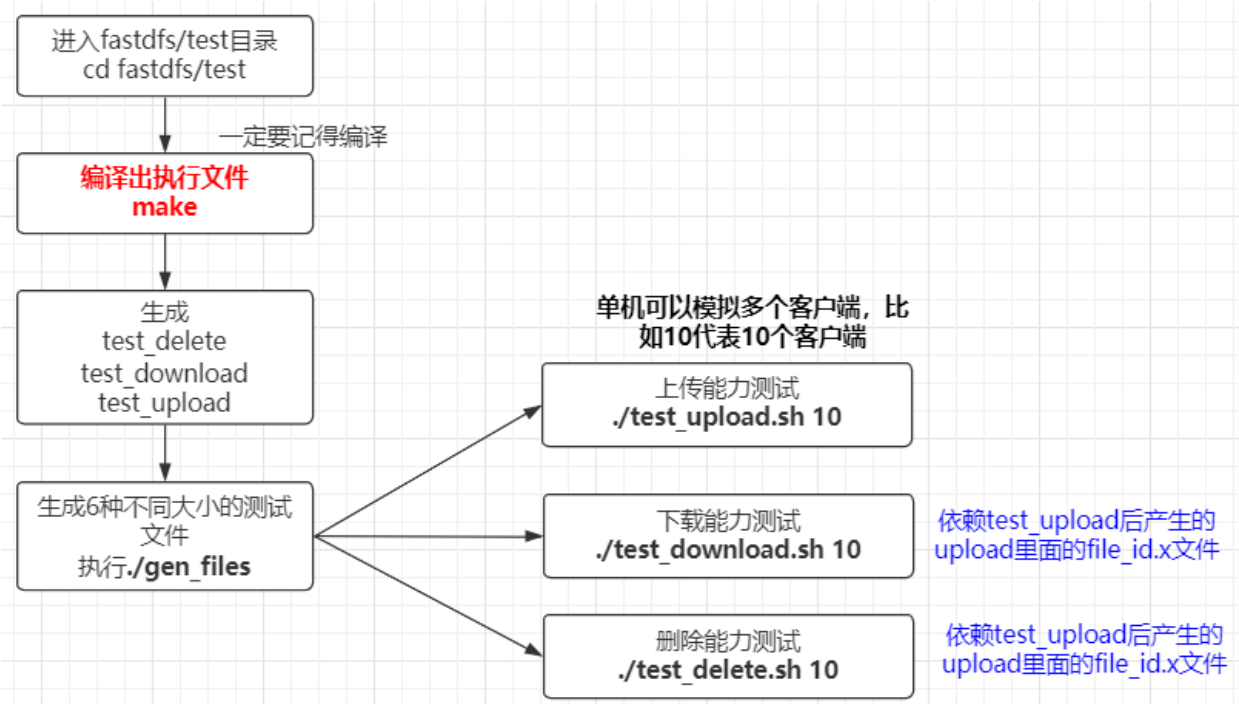
0 or no: do not check default value is 0

check_file_duplicate = 0
这里需要依赖FastDHT（也是fastdfs作者写的组件）去处理，本质而言就是对上传的文件内容进行hash计算或者md5计算出来唯一值，并保存，然后每次上传文件统计出来的唯一值都去存储的数据里面查找，如果有同样的则认为文件重复。

但这种方式导致**每个上传文件都要在服务器做实时唯一值计算**，比较消耗cpu的资源，建议还是类似我们图床的方式去做（秒传）：

- 文件MD5唯一值由客户端计算并发送给文件服务器;
- 文件服务器得到客户端提交的MD5值后，到redis+mysql的组合里面查询是否存在同样的文件。

2 文件性能测试



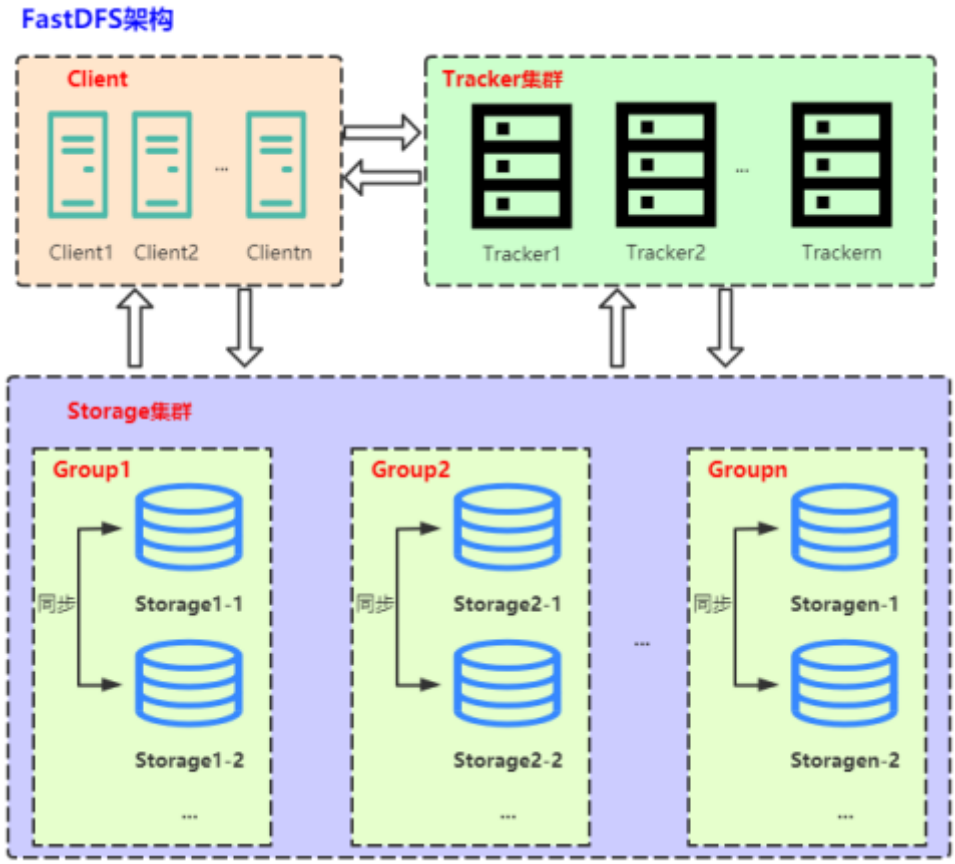
小规模测试的时候，建议一台客户端机器只模拟一个客户端 ./test_upload.sh 1 。

可以提前看看fastdfs/test 目录下的测试代码，**需要使用make进行编译。**

比如

```
cd ~/tuchuang/fastdfs/test
make
```

```
root@iZbp1h2l856zgoegc8rvnhZ:~/tuchuang/fastdfs/test# make
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -c -o common_func.o common_func.c  -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -c -o dfs_func.o dfs_func.c  -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -o gen_files gen_files.c  common_func.o dfs_func.o -L/usr/l
cal/lib -lfdfsclient -lfastcommon -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -o test_upload test_upload.c  common_func.o dfs_func.o -L/u
r/local/lib -lfdfsclient -lfastcommon -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -o test_download test_download.c  common_func.o dfs_func.o
L/usr/local/lib -lfdfsclient -lfastcommon -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -o test_delete test_delete.c  common_func.o dfs_func.o -L/u
r/local/lib -lfdfsclient -lfastcommon -I/usr/local/include
cc -g -Wall -O -D_FILE_OFFSET_BITS=64 -DDEBUG -o combine_result combine_result.c  common_func.o dfs_func.
 -L/usr/local/lib -lfdfsclient -lfastcommon -I/usr/local/include
root@iZbp1h2l856zgoegc8rvnhZ:~/tuchuang/fastdfs/test# ls
combine_result    common_func.o  dfs_func_pc.c  test_delete     test_download.c   test_upload.c
combine_result.c  dfs_func.c     gen_files      test_delete.c   test_download.sh  test_upload.sh
common_func.c     dfs_func.h     gen_files.c    test_delete.sh  test_types.h
common_func.h     dfs_func.o     Makefile       test_download   test_upload
root@iZbp1h2l856zgoegc8rvnhZ:~/tuchuang/fastdfs/test#
```



并使用**TPS**指标：

TPS `Transactions Per Second` 也就是事务数/秒。一个事务是指一个客户机向服务器发送请求然后服务器做出反应的过程。客户机在发送请求时开始计时，收到服务器响应后结束计时，以此来计算使用的时间和完成的事务个数。

可以使用：// kb/s显示磁盘信息，每2s刷新一次

iostat -d -k 2

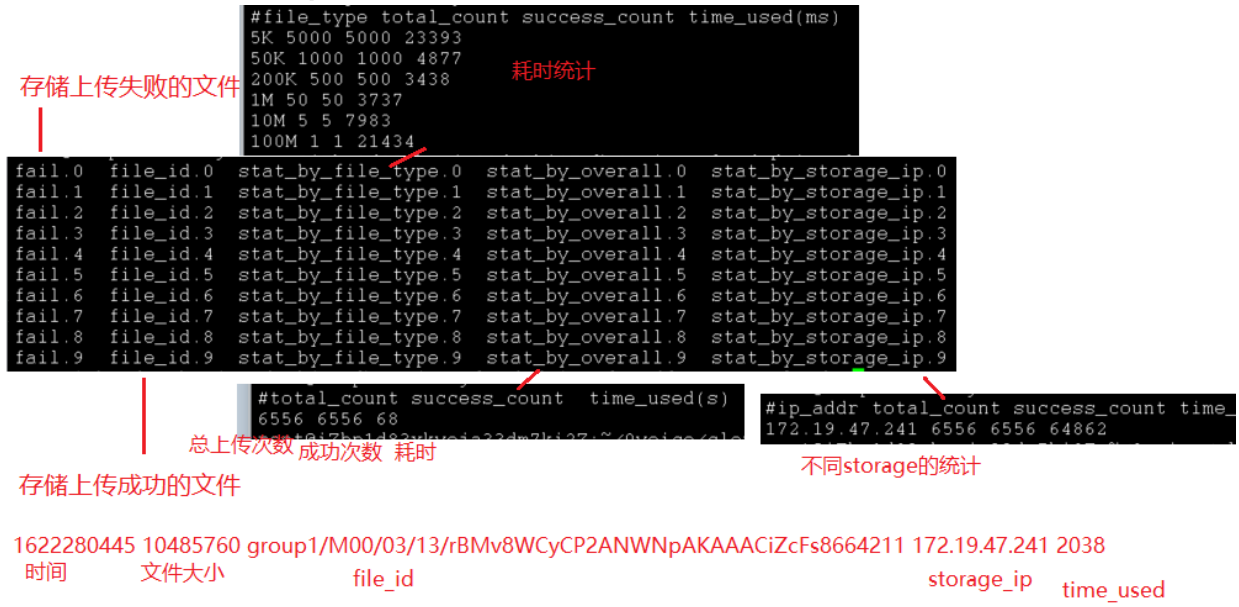
命令实时显示测试的读写情况。

2.1 上传测试

对应的测试程序test_upload.c + test_upload.sh

可以模拟十个并发客户端。.0、.1、....、.9**是客户端序号而已。**

测试结果在fastdfs/test/upload目录。



（1）生成测试文件

```
~/tuchuang/fastdfs/test# ./gen_files
生成结束打印：done.
```

使用6种不同大小的文件进行测试，分别为

文件规格	测试次数	
5K	100**0000**	
50K	2000000	
200K	1000000	
1M	200000	
10M	20000	
100M	1000	

或者

文件规格	测试次数	
5K	**50000**	

文件规格	测试次数	
50K	10000	
200K	5000	
1M	500	
10M	50	
100M	10	

（2）单客户端测试本地上传到本机服务器（受带宽限制）

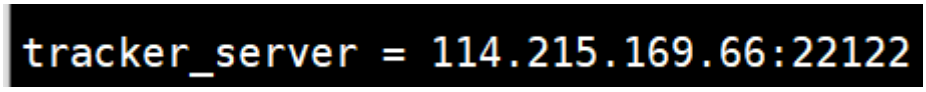
test_upload 格式，Usage: %s <process_index> [config_filename]， 默认配置文件路径**（实际就是/etc/fdfs/client.conf）**

但这里配置的是外网IP地址

```
root@iZbp1d83xkvoja33dm7ki2Z:~/Ovoice/cloud-drive/fastdfs/test# ./test_upload 0
```

测试结束后打印：

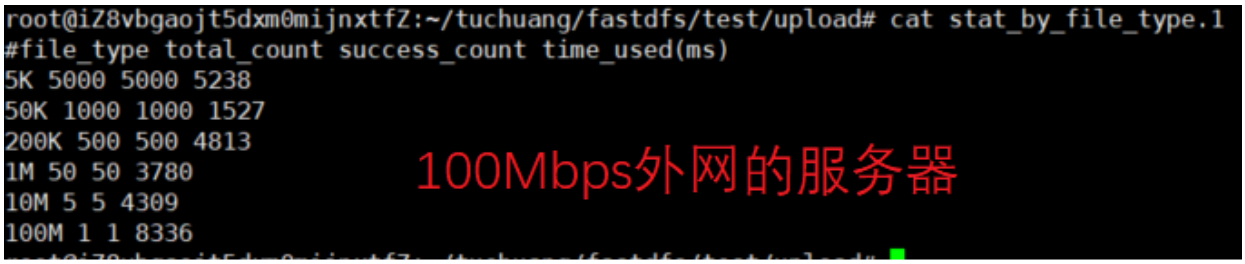
proccess 1, time used: **577s**



storage client的配置

文件规格	上传次数	成功次数	总耗时ms	平均耗时ms(每次上传)	TPS
5K	5000	5000	79779	15.9	63
50K	1000	1000	59941	59.9	17
200K	500	500	131765	263.5	3.795
1M	50	50	73525	1,470.5	0.680
10M	5	5	76394	15,278.8	0.065
100M	1	1	152530	152,530	
10MB = 10*8Mb / 15.278秒 = **5.3Mbps** 。					

另一组测试（100Mbps的服务器）



10M*5*8/4.309=92.82Mbps

100M*8/8.336=95.96Mbps

fastdfs最大的瓶颈 带宽 外网带宽。

（3）120.27.131.197往服务器114.215.169.66 发送

这里的测试结果和（1）一致，**主要受限于网络带宽。**

```
proccess 1, time used: 564s

5K 5000 5000 8107250K 1000 1000 57925
200K 500 500 125415
1M 50 50 71029
10M 5 5 75547
100M 1 1 150387
```

5M的带宽

（4）上传到本地把ip地址修改为内网的再次测试

比如我云服务器地址是：

```
root@iZbp1h2l856zgoegc8rvnhZ:~/tuchuang/fastdfs/test# ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:01:dd:02:49
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:16:3e:17:a6:b8
          inet addr:172.19.24.119  Bcast:172.19.63.255  Mask:255.255.192.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9463891 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8518678 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2280675802 (2.2 GB)  TX bytes:4638675256 (4.6 GB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:117475114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:117475114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:145498017552 (145.4 GB)  TX bytes:145498017552 (145.4 GB)
```

把storage.conf的**tracker_server改成172.19.24.119。** 并重启storage server。

```
/etc/init.d/fdfs_storaged restart
```

./test_upload 1
proccess 1, time used: 5s

#file_type total_count success_count time_used(ms)
5K 5000 5000 243
50K 1000 1000 39
200K 500 500 503
1M 50 50 639
10M 5 5 460
100M 1 1 921

文件规格	上传次数	成功次数	总耗时ms	平均耗时ms	TPS(当前测试)	TPS(5M带宽)
5K	5000	5000	243	0.0486	**20576**	63
50K	1000	1000	39	0.039	25,641	17
200K	500	500	503	1.006	994	3.795
1M	50	50	639	12.78	78	0.680
10M	5	5	460	92	10.8	0.065
100M	1	1	921	921	1.08	

PS：和硬盘写入能力有关系。

文件规格	上传次数	成功次数	总耗时 ms	平
5K	5000	5000	243	0.
50K	1000	1000	39	0.
200K	500	500	503	1.
1M	50	50	639	1
10M	5	5	460	9
100M	1	1	921	9

2核4G内存

另一组数据（云服务器，更高的8核16G内存）：

```
root@iZ8vbgaojt5dxm0mijnxtfZ:~/tuchuang/fastdfs/test# cat upload/stat_by_file_type.
#file_type total_count success_count time_used(ms)
5K 5000 5000 313
50K 1000 1000 202
200K 500 500 512
1M 50 50 243
10M 5 5 218
100M 1 1 424
root@iZ8vbgaojt5dxm0mijnxtfZ:~/tuchuang/fastdfs/test#
```

测试上传数据不经过外网

100M*8/0.424 = 1,886.79Mbps

这个和磁盘性能有关系

使用磁盘测试命令测试磁盘性能：sudo dd if=/dev/zero of=/opt/testfile bs=1M count=500 conv=fdatasync

以及使用fio命令对比测试出来的磁盘性能 bw=191MiB/s (200MB/s), 191MiB/s-191MiB/s (200MB/s-200MB/s)

具体磁盘性能测试参考：Linux测试磁盘读写性能 - Magiclala - 博客园 (cnblogs.com)

（5）如果使用集群的方式进行测试?

先考虑磁盘写入（读取）能力：

- 机械硬盘的写入速度和读写速度一般约为120MB/S
- SATA协议的固态硬盘速度约为**500MB/S**
- NVMe协议（PCIe 3.0×2）的固态硬盘速度约为1800MB/S
- NVMe协议（PCIe 3.0×4）的固态硬盘速度约为3500MB/S。

外网带宽：

- 通常是千Mbps网是极限

内网带宽（主要指云服务器厂商，以阿里云为例）

- 云服务器ECS内网间：
  - 非I/O优化的实例为**千兆共享带宽**
  - **I/O优化**的实例为**万兆或25G共享带宽。**
- 由于是共享网络，因此无法保证带宽速度是不变的。另外，阿里云服务器ECS实例规格不同，内网带宽也不同，如下图：

规格族	实例规格	vCPU	内存	处理器主频/睿频	内网带宽	内网收发包
计算型 c7 荐	ecs.c7.8xlarge	32 vCPU	64 GiB	-/3.5 GHz	最高 25 Gbps	600 万 PPS
计算型 c7 荐	ecs.c7.16xlarge	64 vCPU	128 GiB	-/3.5 GHz	32 Gbps	1200 万 PPS
计算平衡增强型 c6e	ecs.c6e.xlarge	4 vCPU	8 GiB	2.5 GHz/3.2 GHz	最高 10 Gbps	100 万 PPS
计算平衡增强型 c6e	ecs.c6e.8xlarge	32 vCPU	64 GiB	2.5 GHz/3.2 GHz	10 Gbps	600 万 PPS
计算平衡增强型 c6e	ecs.c6e.13xlarge	52 vCPU	96 GiB	2.5 GHz/3.2 GHz	16 Gbps	900 万 PPS
计算平衡增强型	ecs.c6e.26xlarge	104 vCPU	192 GiB	2.5 GHz/3.2 GHz	32 Gbps	2400 万 PPS

性能会怎么样？ 本质而言要考虑的点：

- 网络带宽
- 磁盘读写速度
- 文件大小

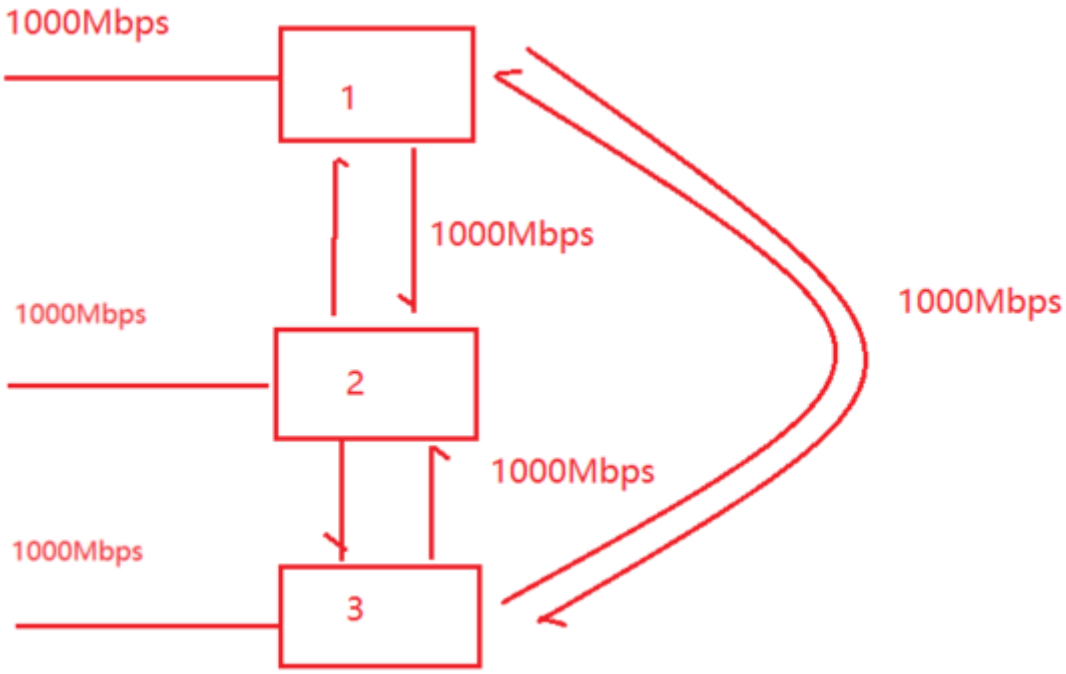
- 同组storage的个数，因为storage直接要相互同步影响源文件写入的性能。比如 1台服务器写入能力

比如三个storage：

磁盘写入能力为：500MB/s，即是4000Mbps

服务器带宽：1000Mbps

局域网带宽：10000Mbps



此时要同时三个storage上传带宽都达到1000M极限，则需要磁盘的写入能力至少是3000Mbps起，而且局域网的带宽至少要达到3000Mpbs，因为局域网之间是需要通过路由去交换数据，并不是两个服务器直连的。

再考虑到方案如果是先上传到nginx，然后再转到fastdfs，则写入能力还要再加上1000Mbps， 所以磁盘写能力至少要达到4000Mbps。

2.2 下载测试

test_download.c + test_upload.sh，他们依赖test_upload后产生的upload里面的file_id.x文件，里面记录了要下载文件的file id.

./test_download 1

测试结果在fastdfs/test/download目录。

5120（文件大小）
group1/M00/00/44/ctepQmCyRgaIS7LqAAAUAG74ecsAAAADQP7agAAABUA7824285（file_id）

由本地服务器下载

#file_type total_count success_count time_used(ms)
5K 1162088 1162088 574
50K 220357 220357 117
200K 119790 119790 116
1M 12396 12396 8630

2.3 删除测试

test_delete.c + test_delete.sh，他们依赖test_upload后产生的upload里面的file_id.x文件，里面记录了要删除的文件的file id.

./test_delete 1

5120（文件大小）
group1/M00/00/44/ctepQmCyRgaIS7LqAAAUAG74ecsAAAADQP7agAAABUA7824285（file_id）

从本地服务器删除
#file_type total_count success_count time_used(ms)5K 5000 5000 73
50K 1000 1000 9
200K 500 500 6
1M 50 50 1
10M 5 5 5
100M 1 1 15

2.4 测试性能总结

2.4.1 上传文件

提升上传性能的方法：

- 增加group（水平扩展）
- 增加带宽（带宽能力）
- 使用读写性能高的磁盘 （）

单纯增加每个group的storage只能应对上传峰值，不能从根本上提升上传能力。

2.4.2 下载文件

- 增加storage （少写多读的场景）
- **增加group**
- 增加带宽
- 使用读性能高的磁盘

具体见课上分析。

fastdfs打满千M带宽是很容易的。

3 mysql优化

课程里涉及到较多的条件语句，而且都是匹配字符串所以**需要考虑索引**的建立，具体见课堂分析。

最左匹配原则：mysql的查询器，会从最左开始向右匹配，直到遇到范围查询就停止匹配(**>、<、between、like**)，右边的索引不生效。（因为底层是B+树，从左到右建立搜索树的）：

1. 在查询时，如果不满足最左列的查询条件，索引将无法被充分利用。例如，对于一个(a, b, c)的联合索引，如果查询条件没有涉及到列a，则索引的优势无法被充分发挥。
2. 查询条件可以部分使用联合索引。例如，在(a, b, c)的联合索引中，如果查询条件仅涉及到列a，则只有列a部分的索引会被利用。如果查询条件涉及到列a和b，那么列a和b部分的索引都会被利用。如果查询a/b/c，a、b、c索引都能用。 但是，如果查询条件只涉及到列b，那么索引将不会被利用。只涉及到b/c也不能用这个索引。
3. 使用最左前缀原则时，可以使用EXPLAIN命令来分析查询计划，以确保索引被正确使用。

优化步骤：

1. 先找出所有对该表的sql语句；
2. 分析对该表的sql语句哪一条操作是最频繁的；
3. 合理创建索引。

user_info表

被调用涉及到where：

select password from user_info where **user_name**='%s'

select * from user_info where user_name='%s'

主要匹配user_name

索引：UNIQUE KEY `uq_user_name`（`user_name`）创建唯一索引：1查找更快速；2.不允许重复；

user_file_list 表

被调用涉及到where：

- select count(*) from user_file_list where **user**='%s'
- select user_file_list.*, file_info.url, file_info.size, file_info.type from file_info, user_file_list where **user** = '%s' and file_info.md5 = user_file_list.**md5** limit %d, %d"
- update user_file_list set shared_status = 1 where **user** = '%s' and **md5** = '%s' and **file_name** = '%s'
- update user_file_list set shared_status = 0 where **user** = '%s' and **md5** = '%s' and **file_name** = '%s'
- select * from user_file_list where **user** = '%s' and **md5** = '%s' and **file_name** = '%s'

联合索引： KEY `idx_user_md5_file_name` (`user`,`md5`, `file_name`)

share_picture_list 表-考虑

- select * from share_picture_list where **user**='%s' 最高
- select share_picture_list.user, share_picture_list.filemd5, share_picture_list.file_name,share_picture_list.urlmd5, share_picture_list.pv, share_picture_list.create_time, file_info.size from file_info, share_picture_list where share_picture_list.**user** = '%s' and file_info.md5 = share_picture_list.**filemd5** limit %d, %d"
- select * from share_picture_list where **urlmd5** = '%s' and **user** = '%s'
- select filemd5 from share_picture_list where **urlmd5** = '%s' 经常

索引： KEY `idx_user_filemd5` (`user`, `filemd5`),

KEY `idx_urlmd5_user` (`urlmd5`, `user`)

share_file_list表-考虑

- select share_file_list.*, file_info.url, file_info.size, file_info.type from file_info, share_file_list where file_info.md5 = share_file_list.**md5** limit %d, %d
- delete from share_file_list where **user** = '%s' and **md5** = '%s
- update share_file_list set pv = %d where **md5** = '%s' and **file_name** = '%s'
- delete from share_file_list where **user** = '%s' and **md5** = '%s' and **file_name** = '%s',
- select * from share_file_list where **md5** = '%s' and **file_name** = '%s'

索引： KEY `idx_filename_md5_user` (`file_name`, `md5`, `user`),

KEY `idx_md5_user` (`md5`, `user`)

4 上云测试注意事项

上云和本地没有太多区别，主要就是**注意开放对外的端口。**

root@iZbp1d83xkvoja33dm7ki2Z:~/0voice/cloud-drive/fastdfs/test# ifconfig eth0:
flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    内外 **inet 172.19.47.241** netmask 255.255.192.0 broadcast 172.19.63.255
      inet6 fe80::216:3eff:fe17:9305 prefixlen 64 scopeid 0x20
      ether 00:16:3e:17:93:05 txqueuelen 1000 (Ethernet)
      RX packets 9365809 bytes 3378002224 (3.3 GB)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 7581625 bytes 5072239197 (5.0 GB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

120.27.131.197 外网

部署在外网主要是端口开放：80、22122、23000等 ，可以改成只需要自己家里的**外网ip和公司的外网ip。**

如果云服务器所有的机器都在同一个集群内部，可以不对外开放22122、23000。

5 fastdfs优化策略

见课上分析

最大并发连接数设置

storage.conf
tracker.conf

参数名：max_connections （同时有个连接，流媒体服务器）
缺省值：1024
说明：FastDFS采用预先分配好buffer队列（对象池）的做法，分配的内存大小为： max_connections * buff_size，因此配置的连接数越大，消耗的内存越多。不建议配置得过大，以避免无谓的内存开销。

工作线程数设置

参数名： work_threads （网络IO的工作线程数量）
缺省值：4
说明：为了避免CPU上下文切换的开销，以及不必要的资源消耗，不建议将本参数设置得过大。为了发挥出多个CPU的效能，系统中的线程数总和，应等于CPU总数。

- 对于tracker server，公式为： work_threads + 1 = CPU数
- 对于storage server，公式为： work_threads + 1 + (disk_reader_threads + disk_writer_threads) * store_path_count = CPU数

CPU数是4 = work_threads(1) + 1 + disk_reader_threads(1) + disk_writer_threads(4) = 4cpu

同步文件线程
storage-> tracker的线程

storage目录数设置

参数名：subdir_count_per_path
缺省值：256
说明：FastDFS采用**二级目录的做法**，目录会在 FastDFS初始化时自动创建。存储海量小文件， 打开了trunk存储方式的情况下，建议将本参数适当改小，**比如设置为32**，此时存放文件的目录数为 32 * 32 = 1024。假如trunk文件大小采用缺省值**64MB**，磁盘空间为2TB，那么每个目录下存放的trunk文件数均值为：**2TB / (1024 * 64MB) = 32**个。

4TB / (1024 * 64MB) = 64个。

8T** / (1024 * 64MB) = 128* 个-》8T** / (6464* 64MB) = 32个**

storage磁盘读写线程设置

- disk_rw_separated：磁盘读写是否分离
- disk_reader_threads：单个磁盘读线程数
- disk_writer_threads：单个磁盘写线程数
- 如果磁盘读写混合，单个磁盘读写线程数为读线程数和写线程数之和
- **对于单盘挂载方式，磁盘读写线程分别设置为1即可**
- 如果磁盘做了RAID，那么需要酌情加大读写线程数，这样才能最大程度地发挥磁盘性能

storage同步延迟相关设置

- sync_binlog_buff_interval：将binlog buffer写入磁盘的时间间隔，取值大于0，缺省值为60s
- sync_wait_msec：如果没有需要同步的文件，对binlog进行轮询的时间间隔，取值大于0，缺省值为100ms
- sync_interval：同步完一个文件后，休眠的毫秒数，缺省值为0

为了缩短文件同步时间，可以将上述3个参数适当调小即可

6 磁盘性能监测和测试

实时监测

iostat

```
// kb/s显示磁盘信息，每2s刷新一次

iostat -d -k 2

// kb/s显示磁盘统计信息及扩展信息，每1s刷新 ，刷新10次结束

iostat -dkx 1 10
```

磁盘的统计参数

- **tps**：该设备每秒的传输次数（Indicate the number of transfers per second that were issued to the device.）。"一次传输"意思是"一次I/O请求"。
- 多个逻辑请求可能会被合并为"一次I/O请求"。"一次传输"请求的大小是未知的。
- kB_read/s：每秒从设备（drive expressed）读取的数据量；
- kB_wrtn/s：每秒向设备（drive expressed）写入的数据量；
- kB_read：读取的总数据量；
- kB_wrtn：写入的总数量数据量；这些单位都为Kilobytes。
- rrqm/s: 每秒对该设备的读请求被合并次数，文件系统会对读取同块(block)的请求进行合并
- wrqm/s: 每秒对该设备的写请求被合并次数
- r/s: 每秒完成的读次数
- w/s: 每秒完成的写次数
- rkB/s: 每秒读数据量(kB为单位)
- wkB/s: 每秒写数据量(kB为单位)
- avgrq-sz:平均每次IO操作的数据量(扇区数为单位)
- avgqu-sz: 平均等待处理的IO请求队列长度
- await: 平均每次IO请求等待时间(包括等待时间和处理时间，毫秒为单位)
- svctm: 平均每次IO请求的处理时间(毫秒为单位)
- %util: 采用周期内用于IO操作的时间比率，即IO队列非空的时间比率

dd命令测试性能

在Linux下，我们可以使用`dd`命令进行磁盘IO性能测试。具体步骤如下：

1. 打开终端
2. 选择一个测试文件的大小，比如1GB，用以下命令创建一个文件：

   ```
   dd if=/dev/zero of=testfile bs=1G count=1
   ```

   - dd：Linux系统中的一个命令行工具，用于复制文件和转换数据格式。
   - if=/dev/zero：表示从/dev/zero设备中读取数据作为输入源。/dev/zero是一个特殊的设备文件，它会返回一串无限长的0字节流。

- of=testfile：表示将输出写入到名为testfile的文件中。
- bs=1G：表示设置每次操作读取或写入1GB的数据块大小。这样可以加快操作速度。
- count=1：表示只执行一次操作。

3. 使用以下命令进行读取速度测试：

```
dd if=testfile of=/dev/null bs=1024k
```

这里将testfile中的内容读取到/dev/null中，bs参数指定了每次读取的块大小。执行完后会显示出读取速度等信息。

4. 使用以下命令进行写入速度测试：

```
dd if=/dev/urandom of=testfile bs=1024k count=1024
```

这里使用/dev/urandom作为源数据生成器，并将其写入到testfile中，每次写入的数据变少，磁盘写入速率也降低。

5. 清理测试文件：

```
rm testfile
```

7 参考阅读

- 海量小文件问题综述 https://blog.csdn.net/liuaigui/article/details/9981135
- FastDFS防盗链 https://www.cnblogs.com/xiaolinstudy/p/9341779.html
- FASTDFS文件服务器架构方案分析 https://wenku.baidu.com/view/3d0987452dc58bd63186bceb19e8b8f67c1cefb0.html
- 基于FastDFS的云存储文件系统性能优化设计与实现 http://www.doc88.com/p-5843806382095.html
- 基于FastDFS的目录文件系统的+研究与实现 https://www.doc88.com/p-5327654549305.html?s=rel&id=2
- 基于FastDFS云存储系统的研究与设计 https://www.doc88.com/p-7098966281892.html?s=rel&id=1
- 基于FastDFS架构的小文件存储系统的设计与实现 https://www.doc88.com/p-7098966281892.html?s=rel&id=1 （课题来源于腾讯实时生培训项目）
- 基于FastDFS的大并发问题的研究与应用 https://www.doc88.com/p-9813549702927.html?s=like&id=1

已知使用FastDFS的用户• 某大型网盘（公司名对方要求保密）
• UC（http://www.uc.cn/）
• 支付宝（http://www.alipay.com/）
• 京东商城（http://www.360buy.com/）
• 淘淘搜（http://www.taotaosou.com/）
• 飞信（http://feixin.10086.cn/）

- 赶集网（http://www.ganji.com/）
- 淘米网（http://www.61.com/）
- 迅雷（http://www.xunlei.com/）
- 蚂蜂窝（http://www.mafengwo.cn/）
- 丫丫网（http://www.iyaya.com/）
- 虹网（http://3g.ahong.com/）
- 5173（http://www.5173.com/）
- 华夏原创网（http://www.yuanchuang.com/）
- 华师京城教育云平台（http://www.hsjdy.com.cn/）
- 视友网（http://www.cuctv.com/）
- 。。。