分布式文件系统—FastDFS

---- 老孙

课程目标：

1、场景概述

2、FastDFS的上传与下载

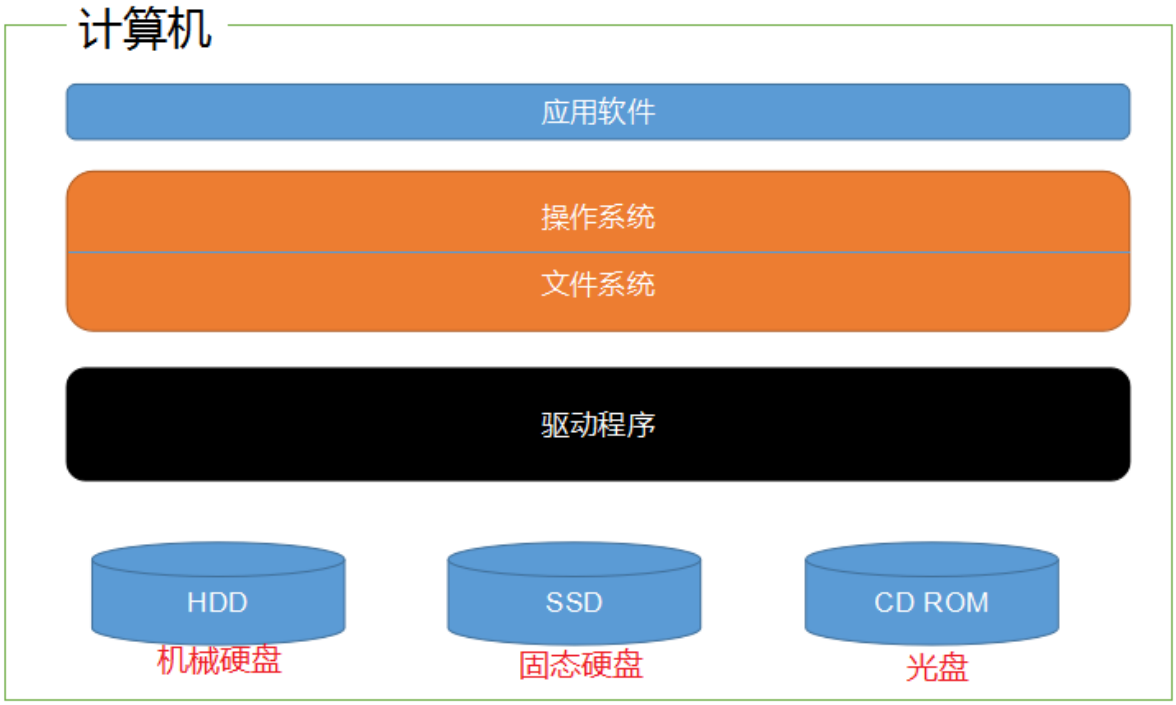
3、项目实战：搭建图片服务器

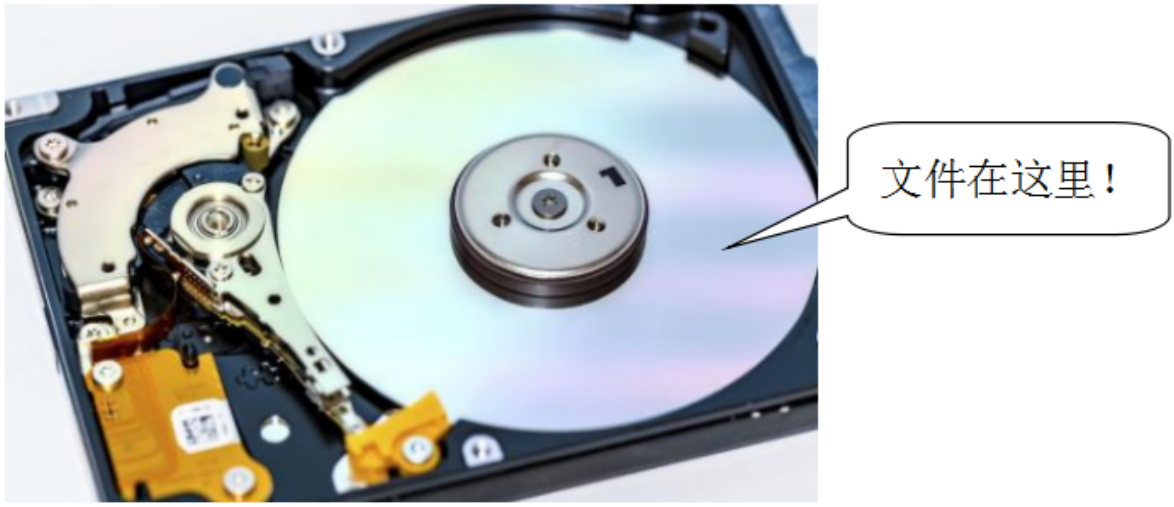
1. 场景概述

- 天猫，淘宝等购物网站，大量的图片和视频，文件太多，如何存储？
- 用户访问量大又如何保证下载速度？分布式文件系统就是解决这些问题的！
-

1.1 什么是文件系统

- 文件数据是如何存储的？？





1.2 分布式文件系统

- 一台电脑存储量有限，并且并发吞吐量也有限，如何提高性能？
- 一吨货物，我要运送到吐鲁番：
  - 1个人运，不敢想象
  - 50个人运，太难了；
  - 500个人运，每个人都很轻松；
- 这就是分布式吗？

答：这里面有集群的概念，也有分布式的概念，二者不要混淆，面试常问的经典题目

- **分布式**：不同的业务模块部署在不同的服务器上或者同一个业务模块分拆多个子业务，部署不同的服务器上。解决高并发的问题；
- **集群**：同一个业务部署在多台服务器上，提高系统的高可用
- 例如：
  - 小饭馆原来只有一个厨师，切菜洗菜备料一手抓。客人越来越多，一个厨师忙不过来，只能再请一个厨师，两个厨师都能炒菜，也就是两个厨师的作用是一样的，这样，两个厨师的关系就是“集群”；
  - 为了让厨师专心炒菜，把菜炒到极致，又请了配菜师负责切菜，备料等工作。厨师和备菜师的关系是“分布式”；
  - 一个备菜师忙不过来，要提供两份食材给两个厨师，又请了一个备菜师，两个备菜师的关系又是“集群”。

1.3 主流的分布式文件系统

1.3.1 HDFS

- (Hadoop Distributed File System)Hadoop 分布式文件系统；
- 高容错的系统，适合部署到廉价的机器上；
- 能提供高吞吐量的数据访问，非常适合大规模数据应用；
- HDFS采用主从结构，一个HDFS是由一个name节点和N个data节点组成；
- name节点储存元数据，一个文件**分割成N份**存储在不同的data节点上。

1.3.2 GFS

- Google File System
- 可扩展的分布式文件系统，用于大型的，分布式的，对大量数据进行访问的应用；
- 运行于廉价的普通硬件上，可以提供容错功能；
- 它可以给大量的用户提供总体性能较高的服务；
- GFS采用主从结构，一个GFS集群由一个master和大量的chunkserver（分块服务器）组成；
- 一个文件被**分割若干块**，分散储存到多个分块server中

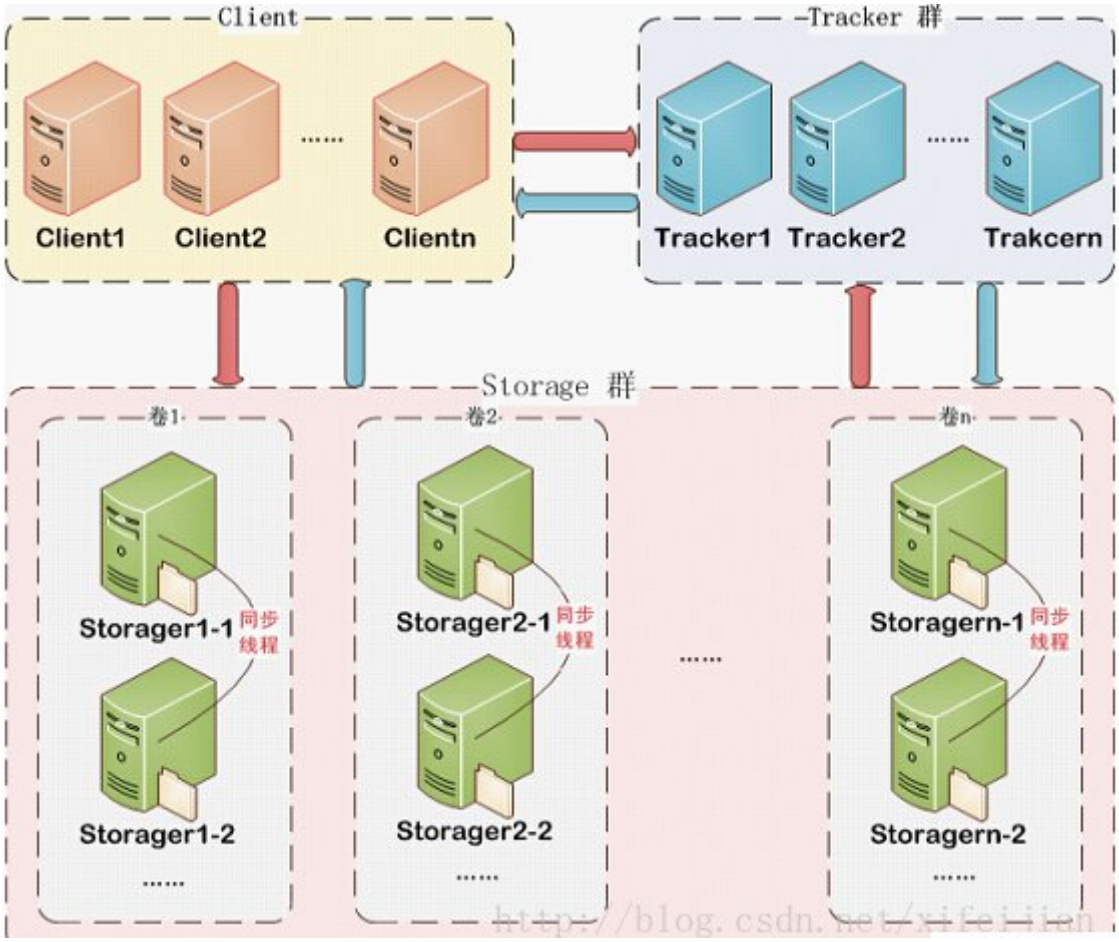
1.3.3.FastDFS

- 由淘宝资深架构师余庆编写并开源；

- 专为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，并注重高可用、高性能等指标，使用FastDFS很容易搭建一套高性能的文件服务器集群提供文件上传、下载等服务；
- HDFS，GFS等都是通用的文件系统，他们的优点是开发体验好，但是系统的复杂度较高，性能也一般；
- 相比之下，专用的分布式文件系统体验差，但是复杂度低，性能也高，尤其fastDFS特别适合图片，小视频等小文件，因为fastDFS**对文件是不分割的**，所以没有文件合并的开销；
- 网络通信用socket，速度快。

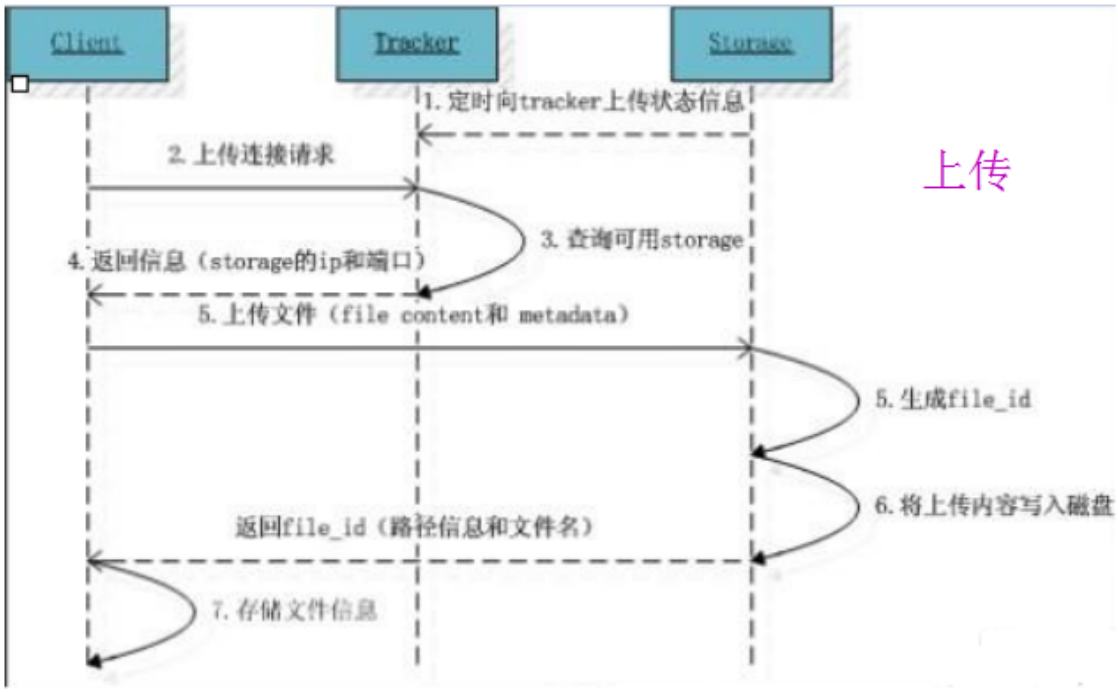
1.4 工作原理

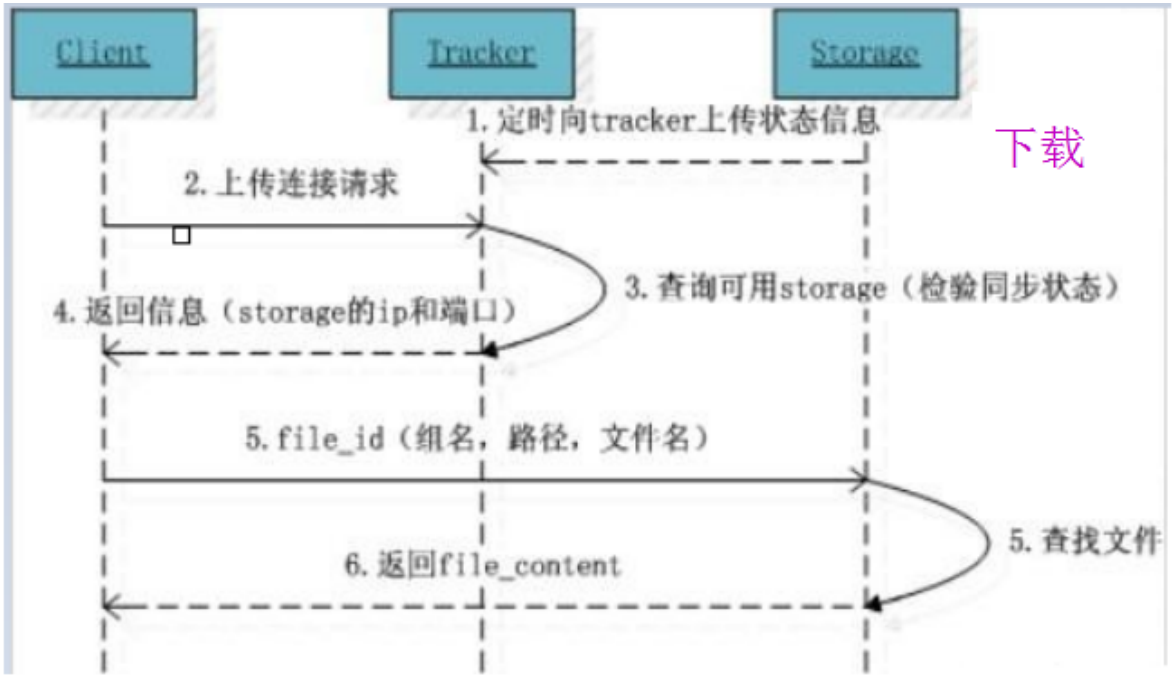
- fastDFS包含Tracker Server和Storage Server；
- 客户端请求Tracker Server进行文件的上传与下载；
- Tracker Server调度Storage Server最终完成上传与下载。



- Tracker （译：追踪者）
  - 作用是负载均衡和调度，它管理着存储服务（Storage Server），可以理解为：“大管家，追踪者，调度员”；
  - Tracker Server可以集群，实现高可用，策略为“轮询”。
- Storage （译：仓库; 贮存器）
  - 作用是文件存储，客户端上传的文件最终存储到storage服务器上；
  - storage集群采用分组的方式，同组内的每台服务器是平等关系，数据同步，目的是实现数据备份，从而高可用，而不同组的服务器之间是不通信的；
  - 同组内的每台服务器的存储量不一致的情况下，会选取容量最小的那个，所以同组内的服务器之间软硬件最好保持一致。
  - Storage Server会连接集群中的所有Tracker Server，定时向他们汇报自己的状态，例如：剩余空间，文件同步情况，文件上传下载次数等信息。

1.5 上传/下载 原理





- 客户端上传文件后，storage会将文件id返回给客户端
- **group1/M00/02/11/aJxAeF21O5wAAAAAAAAGaEIOA12345.sh**
  - **组名**：文件上传后，在storage组的名称，文件上传成功后，由storage返回，需要客户端自行保存。
  - **虚拟磁盘路径**：
    storage配置的虚拟路径，在磁盘选项storage_path对应。
    storage_path0对应M00，
    storage_path1对应M01，
  - **数据两级目录**：
    storage在虚拟磁盘下自行创建的目录。
  - **文件名**：
    与上传时不同，是用storage根据特定信息生成的，里面包含：storage服务器的ip，创建时间戳，大小，后缀名等信息

2. FastDFS的上传与下载

2.1 安装

2.1.1 安装gcc（编译时需要）

```
yum install -y gcc gcc-c++
```

2.1.2 安装libevent（运行时需求）

```
yum -y install libevent
```

2.1.3 安装 libfastcommon

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

1. 上传 libfastcommon-master.zip 到 /opt

```
安装解压zip包的命令:  yum install -y unzip
解压包:              unzip libfastcommon.zip
进入目录:            cd libfastcommon-master
```

2. 编译

```
./make.sh
```

- 如果：make.sh的权限不够，则需要授权（可执行的权利）

```
chmod 777 make.sh
```

3. 安装

```
./make.sh install
```

- libfastcommon安装好后会在/usr/lib64 目录下生成 libfastcommon.so 库文件

4. 拷贝库文件

```
cd /usr/lib64
cp libfastcommon.so /usr/lib
```

2.1.4 安装Tracker

下载 FastDFS_v5.05.tar.gz，并上传到 /opt

```
tar -zxvf FastDFS_v5.05.tar.gz
cd FastDFS
./make.sh
./make.sh install
```

安装成功将安装目录下的conf下的文件拷贝到/etc/fdfs/下

```
cp /opt/FastDFS/conf/* /etc/fdfs/
```

2.2 配置

- **Tracker配置**

```
vim /etc/fdfs/tracker.conf
```

```
#端口号
port=22122

#基础目录（Tracker运行时会向此目录存储storage的管理数据）（基础目录不存在的话，需要自行创建mkdir /home/fastdfs）
base_path=/home/fastdfs
```

- **Storage配置**

```
vim /etc/fdfs/storage.conf
```

```
#配置组名
group_name=group1
#端口
port=23000
#向tracker心跳间隔（秒）
heart_beat_interval=30
#storage基础目录
#目录不存在，需要自行创建
base_path=/home/fastdfs
#store存放文件的位置(store_path)
#可以理解一个磁盘一个path，多个磁盘，多个store_path
#fdfs_storage目录不存在，需要自行创建
#mkdir /home/fastdfs/fdfs_storage
store_path0=/home/fastdfs/fdfs_storage
#如果有多个挂载磁盘则定义多个store_path，如下
#store_path1=.....   (M01)
#store_path2=.....   (M02)

#配置tracker服务器:IP
```

```
tracker_server=10.1.220.247:22122
#如果有多个则配置多个tracker
#tracker_server=10.1.220.x:22122
```

2.3 启动服务

- 启动tracker

```
/usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf restart
```

- 启动storage

```
/usr/bin/fdfs_storaged /etc/fdfs/storage.conf restart
```

- 查看所有运行的端口：

```
netstat -ntlp
```

2.4 搭建 Java工程

使用IDEA创建maven工程

2.4.1 pom.xml

```
<!--fastdfs的java客户端-->
<dependency>
    <groupId>net.oschina.zcx7878</groupId>
    <artifactId>fastdfs-client-java</artifactId>
    <version>1.27.0.0</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-io</artifactId>
    <version>1.3.2</version>
</dependency>
```

2.4.2 创建配置文件

在resources下创建config目录，在config目录下创建 fastdfs-client.properties，内容如下：

```
##fastdfs-client.properties
fastdfs.connect_timeout_in_seconds = 5
fastdfs.network_timeout_in_seconds = 30
fastdfs.charset = UTF-8
fastdfs.http_anti_steal_token = false
fastdfs.http_secret_key = FastDFS1234567890
fastdfs.http_tracker_http_port = 80
fastdfs.tracker_servers = 10.1.220.247:22122
```

2.4.3 文件上传

```
package test;

import org.csource.common.NameValuePair;
import org.csource.fastdfs.*;


/**
 * @BelongsProject: lagou-fastdfs
 * @Author: GuoAn.Sun
 * @CreateTime: 2020-07-28 17:33
 * @Description: 文件上传
 */
public class TestUpload {
    public static void main(String[] args) {
        try {
            // 加载配置文件
            ClientGlobal.initByProperties("config/fastdfs-client.properties");

            // 创建tracker客户端
            TrackerClient trackerClient = new TrackerClient();
            // 通过tracker客户端获取tracker的连接服务并返回
            TrackerServer trackerServer = trackerClient.getConnection();
            // 声明storage服务
            StorageServer storageServer = null;
            // 定义storage客户端
            StorageClient1 client = new StorageClient1(trackerServer,
storageServer);

            // 定义文件元信息
            NameValuePair[] list = new NameValuePair[1];
            list[0] = new NameValuePair("fileName","1.jpg");

            String fileID = client.upload_file1("F:\\img\\1.jpg", "jpg", list);
            System.out.println("fileID = " + fileID);
            // group1/M00/00/00/CgHc918f8l6AFYpOAAWICfQnHuk889.jpg
            /*
            group1：一台服务器，就是一个组
            M00：  store_path0 ----> /home/fastdfs/fdfs_storage/data
            00/00：两级数据目录
             */
            trackerServer.close();
        }catch(Exception e){
            e.printStackTrace();
```

```
        }

    }
}
```

2.4.4 文件查询

```
package test;

import org.csource.fastdfs.*;

/**
 * @BelongsProject: lagou-fastdfs
 * @Author: GuoAn.Sun
 * @CreateTime: 2020-07-28 17:46
 * @Description: 文件查询
 */
public class TestQuery {
    public static void main(String[] args) throws Exception {

        // 加载配置文件
        ClientGlobal.initByProperties("config/fastdfs-client.properties");

        // 创建tracker客户端
        TrackerClient trackerClient = new TrackerClient();
        // 通过tracker客户端获取tracker的连接服务并返回
        TrackerServer trackerServer = trackerClient.getConnection();
        // 声明storage服务
        StorageServer storageServer = null;
        // 定义storage客户端
        StorageClient1 client = new StorageClient1(trackerServer,
storageServer);

        FileInfo fileInfo =
client.query_file_info1("group1/M00/00/00/CgHc918f8l6AFYp0AAWICfQnHuk889.jpg");
        if(fileInfo!=null)
            System.out.println("fileInfo = " + fileInfo);
        else
            System.out.println("查无此文件！");
        trackerServer.close();

    }
}
```

2.4.5 文件下载

```
package test;

import org.csource.fastdfs.*;

import java.io.File;
import java.io.FileOutputStream;
```

```
/**
 * @BelongsProject: lagou-fastdfs
 * @Author: GuoAn.Sun
 * @CreateTime: 2020-07-28 17:49
 * @Description: 文件下载
 */
public class TestDownload {
    public static void main(String[] args) throws Exception{
        // 加载配置文件
        ClientGlobal.initByProperties("config/fastdfs-client.properties");
        // 创建tracker客户端
        TrackerClient trackerClient = new TrackerClient();
        // 通过tracker客户端获取tracker的连接服务并返回
        TrackerServer trackerServer = trackerClient.getConnection();
        // 声明storage服务
        StorageServer storageServer = null;
        // 定义storage客户端
        StorageClient1 client = new StorageClient1(trackerServer,
storageServer);

        byte[] bytes =
client.download_file1("group1/M00/00/00/CgHc918f8l6AFYp0AAWICfQnHuk889.jpg");

        // 通过io将字节数组，转换成一个文件
        FileOutputStream fileOutputStream = new FileOutputStream(new
File("F:/xxxxxx.jpg"));
        fileOutputStream.write(bytes);
        fileOutputStream.close();
        trackerServer.close();
        System.out.println("下载完毕！");
    }
}
```

3. 项目实战

- 掌握fastDFS在真实项目中的使用方法；
- 掌握fastDFS实现图片服务器；

3.1 搭建图片服务器

3.1.1 Nginx模块安装（Storage）

1. 上传 fastdfs-nginx-module_v1.16.tar.gz 到 /opt
2. 解压nginx模块

```
tar -zxvf fastdfs-nginx-module_v1.16.tar.gz
```

3. 修改 config 文件，将文件中的 /usr/local/ 路径改为 /usr/

```
cd /opt/fastdfs-nginx-module/src
vim config
```

4. 将 fastdfs-nginx-module/src下的 mod_fastdfs.conf 拷贝至 /etc/fdfs 下

```
cp mod_fastdfs.conf /etc/fdfs/
```

5. 修改 /etc/fdfs/mod_fastdfs.conf

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

```
base_path=/home/fastdfs
tracker_server=10.1.220.247:22122
#(n个tracker配置n行)
#tracker_server=10.1.220.x:22122
#url中包含group名称
url_have_group_name=true
#指定文件存储路径（上面配置的store路径）
store_path0=/home/fastdfs/fdfs_storage
```

6. 将 libfdfsclient.so 拷贝至 /usr/lib 下

```
cp /usr/lib64/libfdfsclient.so /usr/lib/
```

7. 创建nginx/client目录

```
mkdir -p /var/temp/nginx/client
```

3.1.2 Nginx安装（Tracker）

1. 将 nginx-1.14.0.tar.gz上传到/opt（安装过nginx，此步省略）
2. 解压：tar -zxvf nginx-1.14.0.tar.gz（安装过nginx，此步省略）
3. 安装依赖库（安装过nginx，此步省略）

```
yum install pcre
yum install pcre-devel
yum install zlib
yum install zlib-devel
yum install openssl
yum install openssl-devel
```

4. 进入nginx解压的目录下 cd /opt/nginx-1.14.0
5. 安装

```
./configure \
--prefix=/usr/local/nginx \
--pid-path=/var/run/nginx/nginx.pid \
--lock-path=/var/lock/nginx.lock \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--with-http_gzip_static_module \
--http-client-body-temp-path=/var/temp/nginx/client \
--http-proxy-temp-path=/var/temp/nginx/proxy \
--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \
--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \
--http-scgi-temp-path=/var/temp/nginx/scgi \
--add-module=/opt/fastdfs-nginx-module/src
```

注意：上边将临时文件目录指定为 /var/temp/nginx，需要在 /var 下创建 temp 及 nginx 目录：mkdir /var/temp/nginx

6. 编译：make
7. 安装：make install
8. 拷贝配置文件

```
cd /opt/FastDFS/conf
cp http.conf mime.types /etc/fdfs/
是否覆盖：yes
```

9. 修改nginx配置文件

```
cd /usr/local/nginx/conf/
vim nginx.conf
```

```
server {
    listen       80;
    server_name  10.1.220.247;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location /group1/M00 {
        root   /home/fastdfs/fdfs_storage/data;
        ngx_fastdfs_module;
    }
```

10. 关闭nginx，并启动nginx

```
pkill -9 nginx
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

11. 访问nginx并查看图片

http://10.1.220.247

http://10.1.220.247/group1/M00/00/00/CgHc918f8l6AFYp0AAWICfQnHuk889.jpg

3.2 创建前端页面

```
<%--上传文件，文件与文字相比较起来，属于内容较大，必须使用post方式提交--%>
<%--上传文件，和普通文件有区别，action接收参数也会区别对待，所以声明带文件提交的表单为“多部件表单”--%>
<form action="upload" method="post" enctype="multipart/form-data">

    <input type="file" name="fname">
    <br>
    <button>提交</button>

</form>
```

3.3 搭建web服务

3.3.1 pom.xml

```
<packaging>war</packaging>

<dependencies>
    <!-- 因为有jsp页面，所以引用servlet依赖-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <scope>provided</scope>
        <version>2.5</version>
    </dependency>
    <!-- 页面提交过来的请求，使用springmvc来处理-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.2.7.RELEASE</version>
    </dependency>
    <!-- java连接fastDFS的客户端工具-->
    <dependency>
        <groupId>net.oschina.zcx7878</groupId>
        <artifactId>fastdfs-client-java</artifactId>
        <version>1.27.0.0</version>
    </dependency>
    <!-- 图片上传到FastDFS需要用的到IO工具-->
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-io</artifactId>
        <version>1.3.2</version>
    </dependency>
    <!-- 图片保存到web服务器需要用到的IO工具-->
    <dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
```

```
        <version>1.3.1</version>
    </dependency>
    <!--用来转换java对象和json字符串，注意，2.7以上版本必须搭配spring5.0以上-->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.9.8</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <configuration>
                <port>8001</port>
                <path>/</path>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

3.3.2 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">
    <servlet>
        <servlet-name>springMVC</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/spring-mvc.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springMVC</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

3.3.3 spring-mvc.xml

```
<!--扫描注解的包-->
<context:component-scan base-package="controller"/>
<!--扫描控制器中的注解：@Response-->
<mvc:annotation-driven/>
<!--上传文件的解析器（规定上传文件的大小限制）-->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!-- 上传文件最大限制：2GB-->
    <property name="maxUploadSize" value="2048000000"/>
</bean>
```

3.3.4 文件实体类

```
public class FileSystem implements Serializable {
    private String fileId;
    private String filePath;
    private String fileName;
}
```

3.3.5 控制层

```
package controller;

import entity.FileSystem;
import org.csource.common.NameValuePair;
import org.csource.fastdfs.*;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.multipart.MultipartHttpServletRequest;

import java.io.File;
import java.util.UUID;

/**
 * @BelongsProject: lagou-imageServer
 * @Author: GuoAn.Sun
 * @CreateTime: 2020-07-29 14:25
 * @Description: 处理上传文件的控制器
 */
@Controller
public class FileAction {

    /**
     * @param request 多部件表单的请求对象
     * @return 上传文件对象的json对象
```

```
     * @throws Exception
     *
     * 上传文件的流程：
     * 1、先把文件保存到web服务器上
     * 2、再从web服务器上将文件 上传 到 FastDFS上
     */

    @RequestMapping("upload")
    //MultipartHttpServletRequest：是httpservletRequest的强化版本，不仅可以装文本信息，还可以装图片文件信息
    public @ResponseBody FileSystem upload(MultipartHttpServletRequest request) throws Exception {
        FileSystem fileSystem = new FileSystem();

        /* 1、把文件保存到web服务器*/

        // 从页面请求中，获取上传的文件对象
        MultipartFile file = request.getFile("fname");
        // 从文件对象中获取 文件的原始名称
        String oldFileName = file.getOriginalFilename();
        // 通过字符串截取的方式，从文件原始名中获取文件的后缀   1.jpg
        String hou = oldFileName.substring(oldFileName.lastIndexOf(".") + 1);
        // 为了避免文件因为同名而覆盖，生成全新的文件名
        String newFileName = UUID.randomUUID().toString() + "." + hou;
        // 创建web服务器保存文件的目录(预先创建好D:/upload目录，否则系统找不到路径，会抛异常)
        File toSaveFile = new File("D:/upload/" + newFileName);
        // 将路径转换成文件
        file.transferTo(toSaveFile);
        // 获取服务器的绝对路径
        String newFilePath = toSaveFile.getAbsolutePath();

        /* 2、把文件从web服务器上传到FastDFS*/

        ClientGlobal.initByProperties("config/fastdfs-client.properties");
        TrackerClient trackerClient = new TrackerClient();
        TrackerServer trackerServer = trackerClient.getConnection();
        StorageServer storageServer = null;
        StorageClient1 client = new StorageClient1(trackerServer, storageServer);

        NameValuePair[] list = new NameValuePair[1];
        list[0] = new NameValuePair("fileName",oldFileName);
        String fileId = client.upload_file1(newFilePath, hou, list);
        trackerServer.close();

        // 封装fileSystem数据对象
        fileSystem.setFileId(fileId);
        fileSystem.setFileName(oldFileName);
        fileSystem.setFilePath(fileId);  //已经上传到FastDFS上，通过fileId来访问图片，所以fileId即为文件路径

        return  fileSystem;
    }
}
```

3.3.6 添加fastDFS的配置文件

在resources下创建config目录，在config目录下创建 fastdfs-client.properties

参考：2.4.2

3.3.7 启动fastDFS服务，测试开始

```
[root@localhost /]# /usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf restart
[root@localhost /]# /usr/bin/fdfs_storaged /etc/fdfs/storage.conf restart
[root@localhost /]# /usr/local/nginx/sbin/nginx -c
/usr/local/nginx/conf/nginx.conf
[root@localhost /]# netstat -ntlp
[root@localhost /]# systemctl stop firewalld.service
[root@localhost /]# cd /home/fastdfs/fdfs_storage/data/
[root@localhost /]# ls
```

3.6 典型错误

- 重启linux服务器，可能会到nginx启动失败：

```
[root@localhost logs]# /usr/local/nginx/sbin/nginx -c
/usr/local/nginx/conf/nginx.conf
[root@localhost /]# nginx: [emerg] open() "/var/run/nginx/nginx.pid" failed (2:
No such file or directory)
```

- 导致本次错误的原因，是没有修改pid文件的路径，编辑nginx的配置文件：

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

```
pid       /usr/local/nginx/logs/nginx.pid;
```

- 再次启动nginx，搞定！