

# 分布式文件系统 fastDFS

## 分布式文件系统 SpringBoot+FastDFS+Vue.js

### 教学目标

- 1) 了解分布式filesystem的概念及应用场景
- 2) 理解fastDFS的工作原理
- 3) 掌握fastDFS存取文件方法
- 4) 基于SpringBoot+fastDFS+vue.js实现图片服务


## 1 什么是分布式文件系统

### 1.1 技术应用场景


传智播客拥有大量优质的视频教程，并且免费提供给用户去下载，文件太多如何高效存储？用户访问量如何保证下载速度？今天讲解的分布式文件系统将解决这些问题。

yun.itheima.com


免费视频下载JavaPythonweb前端UI设计PHPC/C++云计算大数据H5+全栈工程师新媒体电商运营软件测试视频章节数: 4713




Java快速入门教程 (Java基础...)  
已经有1882人学习




JavaWeb教程\_JavaWeb入门教...  
已经有54773人学习




JavaWeb网上商城实战视频教...  
已经有1949人学习




Android项目 智慧北京项目实战...  
已经有232人学习




毕向东Java基础教程 (适合初...  
已经有90680人学习




Spring教程\_Spring视频教程从...  
已经有3728人学习




Struts2教程\_Struts2视频教程|...  
已经有2874人学习



Kotlin教程\_Kotlin视频教程|黑马...  
已经有488人学习



Hibernate教程\_Hibernate视频...  
已经有4107人学习



CRM项目实战视频 (SSH2) |...  
已经有2096人学习

# 1.2 什么是分布式文件系统

## 1.2.1 什么是文件系统

引用“百度百科”中的描述：

文件系统

编辑

本词条由“科普中国”百科科学词条编写与应用工作项目 审核。

文件系统是操作系统用于明确存储设备（常见的是磁盘，也有基于NAND Flash的固态硬盘）或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：文件系统的接口，对对象操纵和管理的软件集合，对象及属性。从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

总结：文件系统是负责管理和存储文件的系统软件，它是操作系统和硬件驱动之间的桥梁，操作系统通过文件系统提供的接口去存取文件，用户通过操作系统访问磁盘上的文件。如下图：



常见的文件系统：FAT16/FAT32、NTFS、HFS、UFS、APFS、XFS、Ext4等。

思考：如果没有文件系统我们该怎么管理自己的文件？



## 1.2.2什么是分布式文件系统

引用“百度百科”中的描述：

★ 收藏 | 240 | 24

### 分布式文件系统 锁定

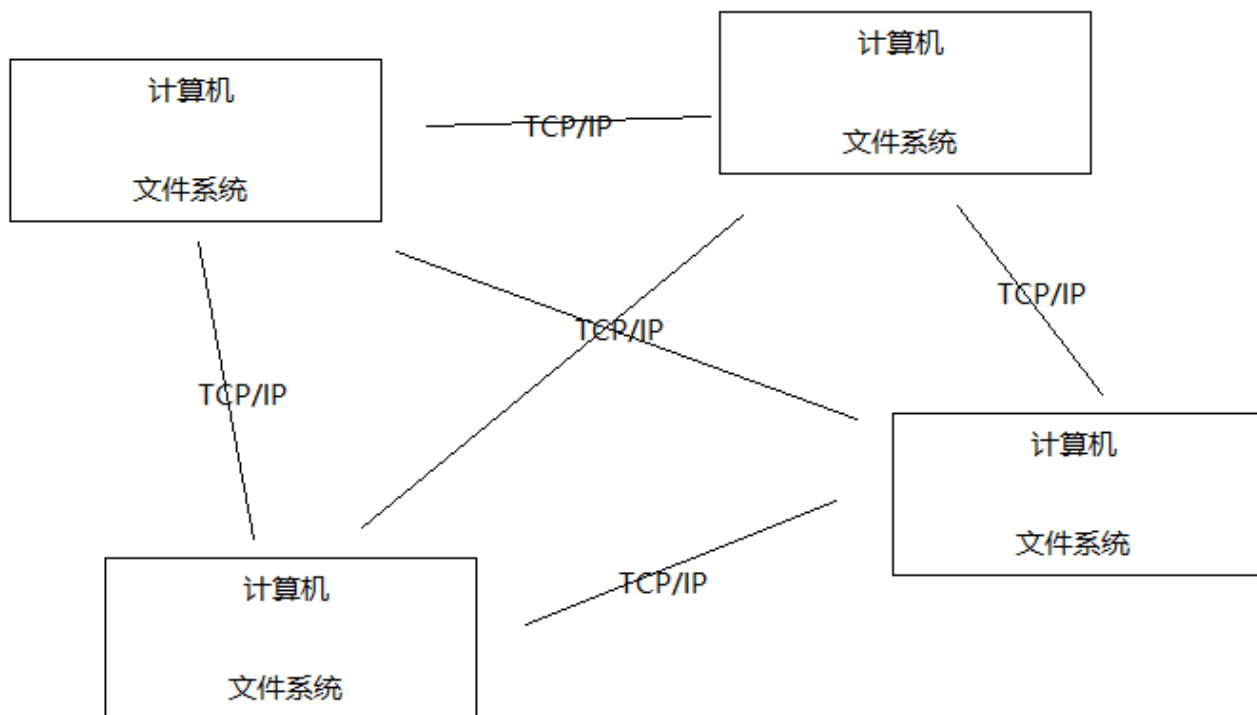
本词条由“科普中国”百科科学词条编写与应用工作项目 审核。

分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连。分布式文件系统的设计基于客户机/服务器模式。一个典型的网络可能包括多个供多用户访问的服务器。另外，对等特性允许一些系统扮演客户机和服务器的双重角色。例如，用户可以“发表”一个允许其他客户机访问的目录，一旦被访问，这个目录对客户机来说就像使用本地[驱动器](#)一样，下面是三个基本的分布式文件系统。

为什么会有分布文件系统呢？

分布式文件系统是面对互联网的需求而产生，互联网时代对海量数据如何存储？靠简单的增加硬盘的个数已经满足不了我们的要求，因为硬盘传输速度有限但是数据在急剧增长，另外我们还要要做好数据备份、数据安全等。

采用分布式文件系统可以将多个地点的文件系统通过网络连接起来，组成一个文件系统网络，结点之间通过网络进行通信，一台文件系统的存储和传输能力有限，我们让文件在多台计算机上存储，通过多台计算共同传输。如下图：



好处：

- 1、一台计算机的文件系统处理能力扩充到多台计算机同时处理。
- 2、一台计算机挂了还有另外副本计算机提供数据。
- 3、每台计算机可以放在不同的地域，这样用户就可以就近访问，提高访问速度。

## 1.3 主流的分布式文件系统

### 1、NFS

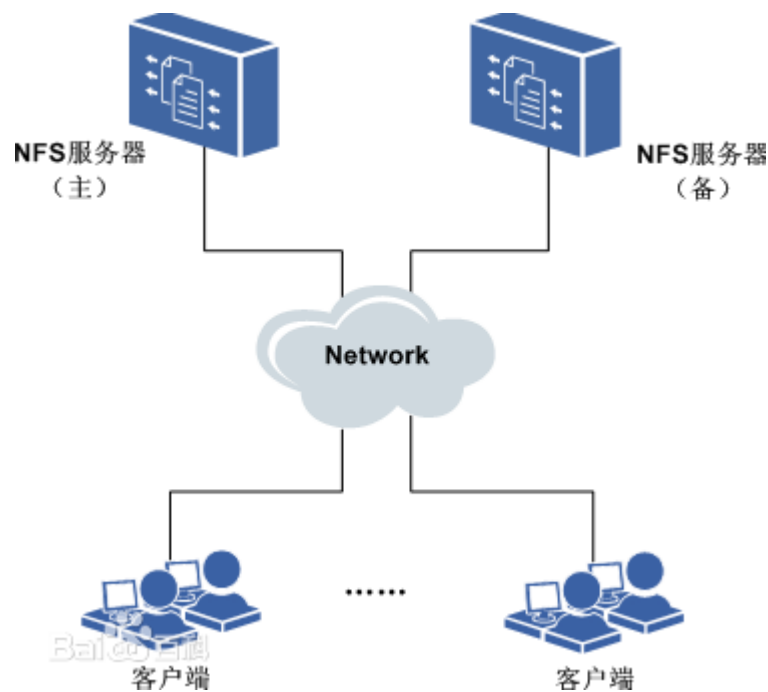
★ 收藏 | 1208 | 141

## NFS（网络文件系统）

🔒 锁定

📖 本词条由“科普中国”百科科学词条编写与应用工作项目 审核。

NFS（Network File System）即网络文件系统，是FreeBSD支持的文件系统中的一种，它允许网络中的计算机之间通过TCP/IP网络共享资源。在NFS的应用中，本地NFS的客户端应用可以透明地读写位于远端NFS服务器上的文件，就像访问本地文件一样。



- 1) 在客户端上映射NFS服务器的驱动器。
- 2) 客户端通过网络访问NFS服务器的硬盘完全透明。

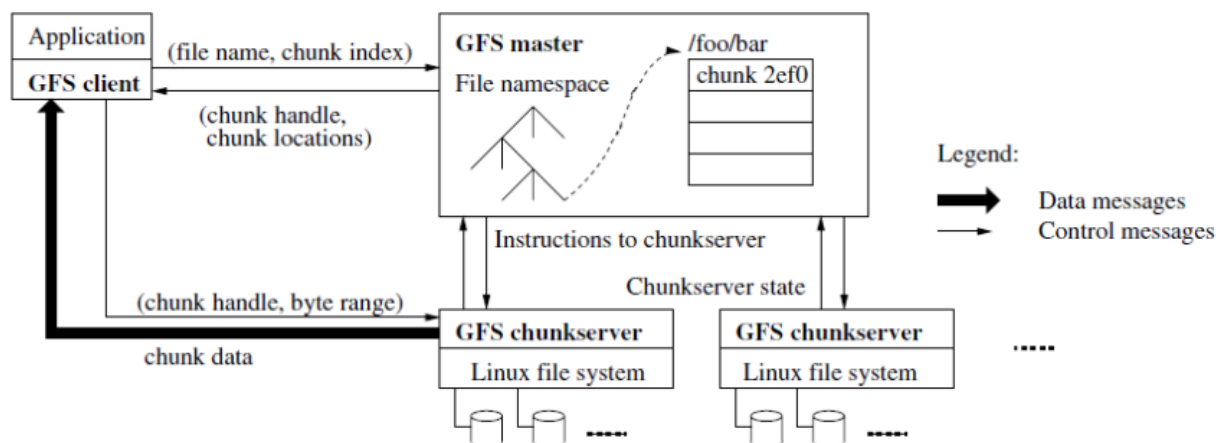
## 2、GFS

[+](#) [★ 收藏](#) [👍 3](#) [🔗 0](#)

# googleFs [编辑](#)

[🔗](#) 本词条缺少**信息栏**，补充相关内容使词条更完整，还能快速升级，赶紧来**编辑**吧！

GFS是一个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。它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。



- 1) GFS采用主从结构，一个GFS集群由一个master和大量的chunkserver组成。

- 2) master存储了数据文件的元数据，一个文件被分成了若干块存储在多个chunkserver中。
- 3) 用户从master中获取数据元信息，从chunkserver存储数据。

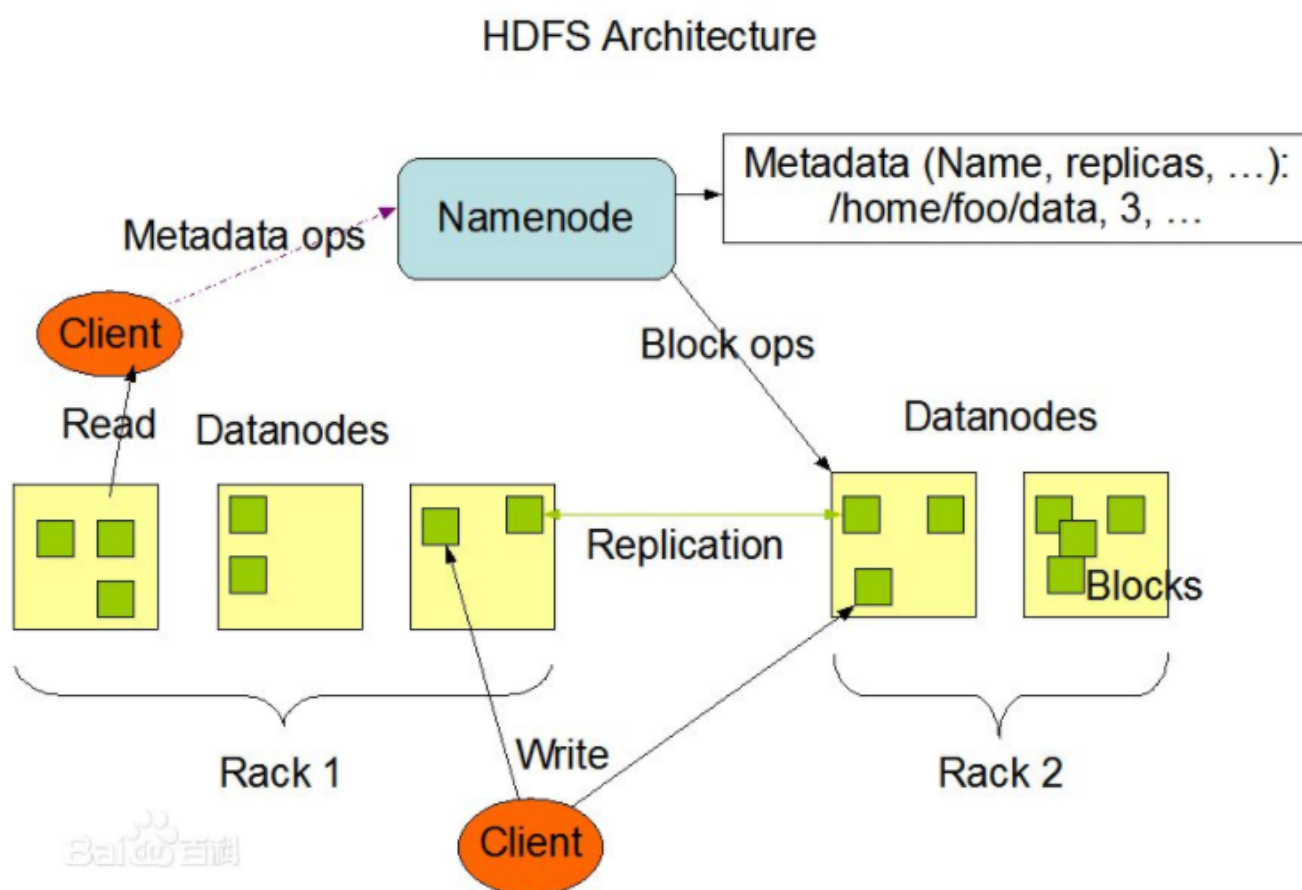
### 3、HDSF

🔖 | ★ 收藏 | 👍 197 | 🔗 52

## hdfs [编辑](#)

Hadoop分布式文件系统(HDFS)被设计成适合运行在通用硬件(commodity hardware)上的分布式文件系统。它和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。HDFS放宽了一部分POSIX约束，来实现流式读取文件系统数据的目的。HDFS在刚开始是作为Apache Nutch搜索引擎项目的基础架构而开发的。HDFS是Apache Hadoop Core项目的一部分。

HDFS有着高容错性(fault-tolerant)的特点，并且设计用来部署在低廉的(low-cost)硬件上。而且它提供高吞吐量(high throughput)来访问应用程序的数据，适合那些有着超大数据集(large data set)的应用程序。HDFS放宽了(relax)POSIX的要求(requirements)这样可以实现流的形式访问(streaming access)文件系统中的数据。



- 1) HDFS采用主从结构，一个HDFS集群由一个名称结点和若干数据结点组成。

名称结点存储数据的元信息，一个完整的数据文件分成若干块存储在数据结点。

2) 客户端从名称结点获取数据的元信息及数据分块的信息，得到信息客户端即可从数据块来存取数据。

## 1.4 分布式文件服务提供商

---

1) 阿里的OSS

### 什么是 OSS

更新时间：2018-01-29 18:15:24



阿里云对象存储服务（Object Storage Service，简称 OSS），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。它具有与平台无关的RESTful API接口，能够提供99.99999999%的服务持久性。您可以在任何应用、任何时间、任何地点存储和访问任意类型的数据。OSS适合各种网站、开发企业及开发者使用。

您可以使用阿里云提供的API/SDK接口或者OSS迁移工具轻松地将海量数据移入或移出阿里云OSS。数据存储到阿里云OSS以后，您可以选择标准类型（Standard）的阿里云OSS服务作为移动应用、大型网站、图片分享或热点音视频的主要存储方式，也可以选择成本更低、存储期限更长的低频访问类型（Infrequent Access）和归档类型（Archive）的阿里云OSS服务作为不经常访问数据的备份和归档。

2) 七牛云存储

3) 百度云存储

## 2 什么是fastDFS

---

### 2.1 fastDFS介绍

---

FastDFS是用C语言编写的一款开源的分布式文件系统，它是由淘宝资深架构师余庆编写并开源。FastDFS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。

为什么要使用fastDFS呢？

上边介绍的NFS、GFS都是通用的分布式文件系统，通用的分布式文件系统的优点是开发体验好，但是系统复杂性高、性能一般，而专用的分布式文件系统虽然开发体验性差，但是系统复杂性低并且性能高。fastDFS非常适合存储图片等那些小文件，fastDFS不对文件进行分块，所以它就没有分块合并的开销，fastDFS网络通信采用socket，通信速度很快。

### 2.2 fastDFS工作原理

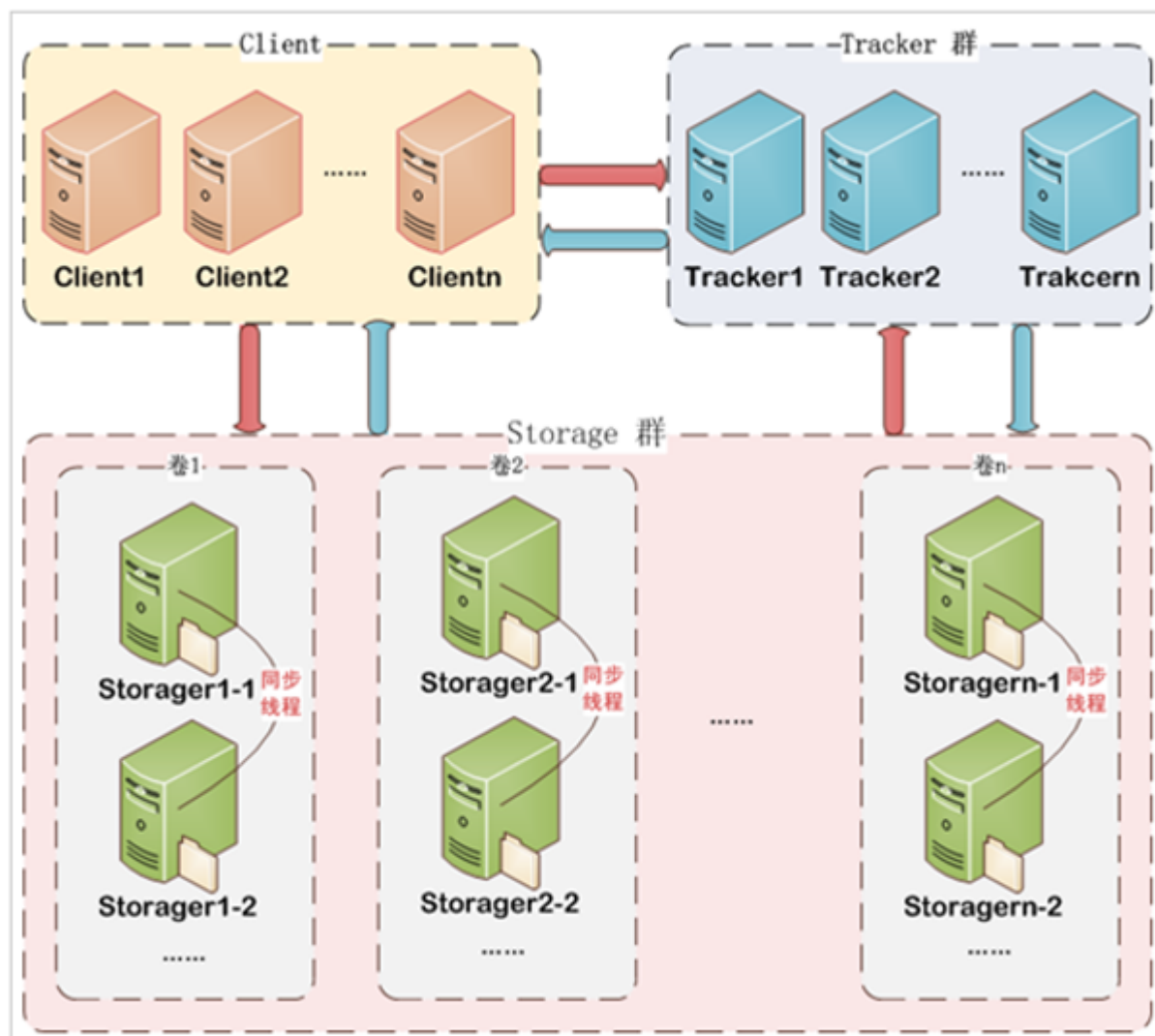
---



## 2.2.1 fastDSF架构

FastDFS架构包括 Tracker server和Storage server。客户端请求Tracker server进行文件上传、下载，通过Tracker server调度最终由Storage server完成文件上传和下载。

如下图：



### 1) Tracker

Tracker Server作用是负载均衡和调度，通过Tracker server在文件上传时可以根据一些策略找到Storage server提供文件上传服务。可以将tracker称为追踪服务器或调度服务器。

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

### 2) Storage

Storage Server作用是文件存储，客户端上传的文件最终存储在Storage服务器上，Storage server没有实现自己的文件系统而是使用操作系统的文件系统来管理文件。可以将storage称为存储服务器。



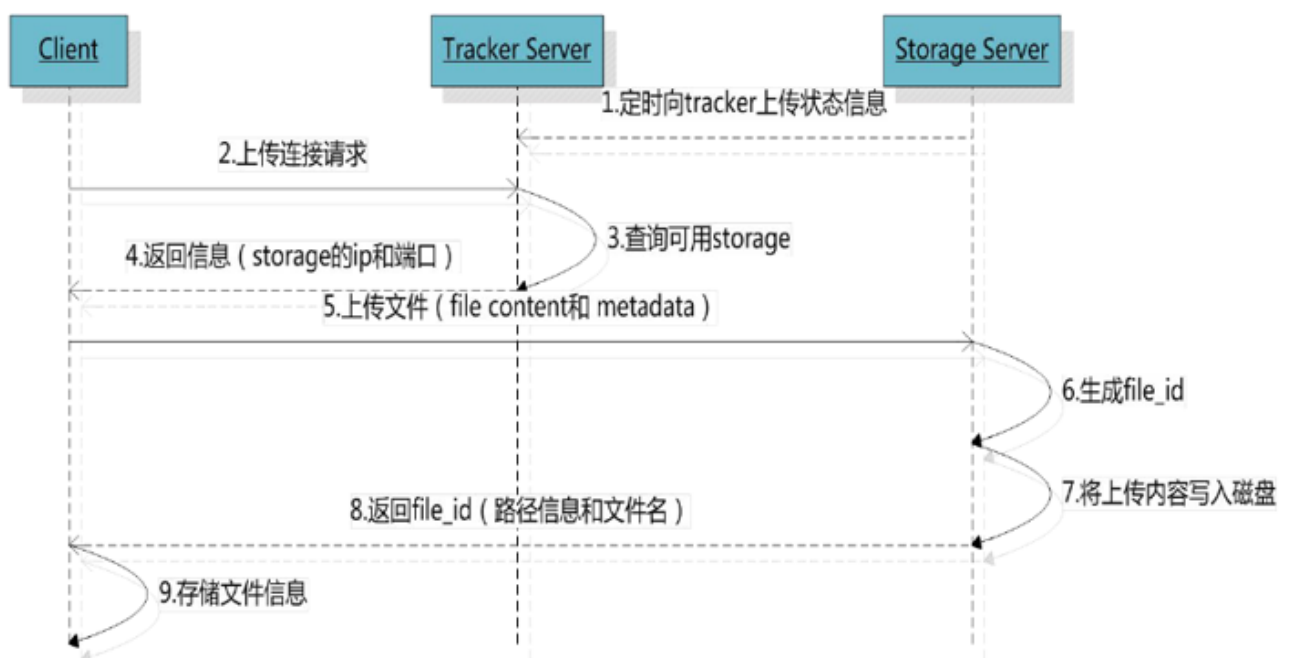
Storage集群采用了分组存储方式。storage集群由一个或多个组构成，集群存储总容量为集群中所有组的存储容量之和。一个组由一台或多台存储服务器组成，组内的Storage server之间是平等关系，不同组的Storage server之间不会相互通信，同组内的Storage server之间会相互连接进行文件同步，从而保证同组内每个storage上的文件完全一致的。一个组的存储容量为该组内存储服务器容量最小的那个，由此可见组内存储服务器的软硬件配置最好是一致的。

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

### 3) Storage状态收集

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

## 2.2.2文件上传流程



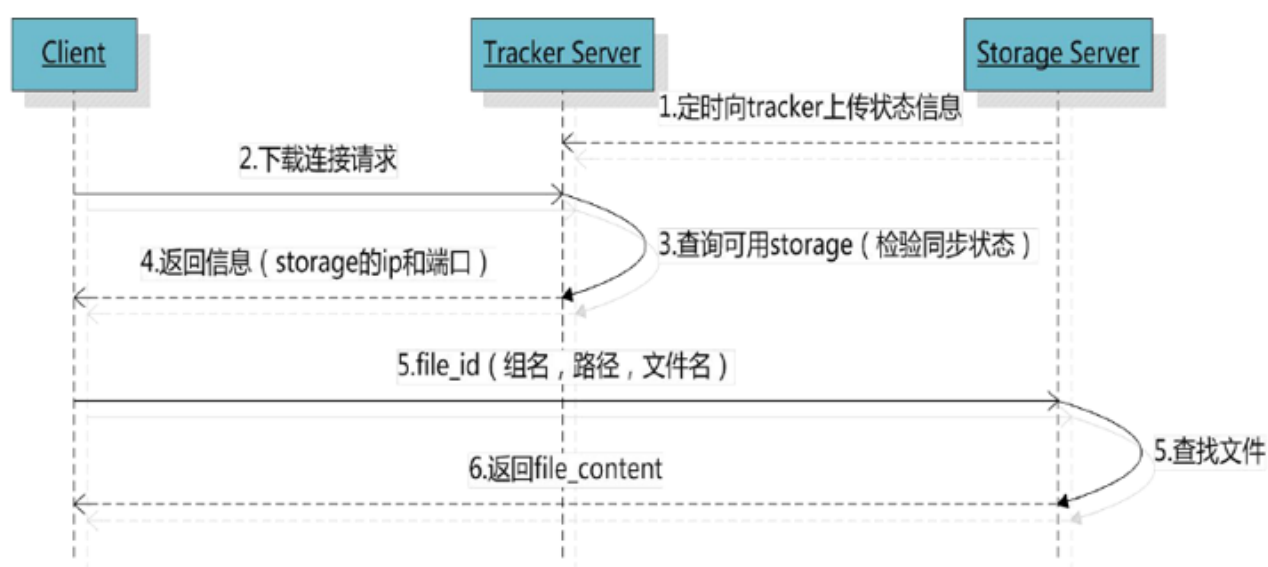
客户端上传文件后存储服务器将文件ID返回给客户端，此文件ID用于以后访问该文件的索引信息。文件索引信息包括：组名，虚拟磁盘路径，数据两级目录，文件名。

**group1 /M00 /02/44/ wKgDrE34E8wAAAAAAAAAGkEiYJK42378.sh**

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

- 虚拟磁盘路径: [storage](#)配置的虚拟路径, 与磁盘选项[store\\_path](#)\*对应。如果配置了store\_path0则是M00, 如果配置了store\_path1则是M01, 以此类推。
- 数据两级目录: storage服务器在每个虚拟磁盘路径下创建的两级目录, 用于存储数据文件。
- 文件名: 与文件上传时不同。是由存储服务器根据特定信息生成, 文件名包含: 源存储服务器IP地址、文件创建时间戳、文件大小、随机数和文件拓展名等信息。

## 2.2.3 文件下载流程



tracker根据请求的文件路径即文件ID 来快速定义文件。

比如请求下边的文件:

**group1 /M00 /02/44/ wKgDrE34E8wAAAAAAAAAGkEIYJK42378.sh**

- 1.通过组名tracker能够很快的定位到客户端需要访问的存储服务器组是group1, 并选择合适的存储服务器提供客户端访问。
- 2.存储服务器根据“文件存储虚拟磁盘路径”和“数据文件两级目录”可以很快定位到文件所在目录, 并根据文件名找到客户端需要访问的文件。

## 3 fastDFS入门

### 3.1fastDFS安装与配置

### 3.1.1fastDFS安装

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

本教程下载：FastDFS\_v5.05.tar.gz

FastDFS是C语言开发，建议在linux上运行，本教程使用CentOS7作为安装环境。

安装细节请参考“fastDFS安装教程.doc”。

### 3.1.2Tracker配置

fastDFS的配置文件目录：/etc/fdfs

主要的配置文件：/etc/fdfs/tracker.conf（tracker配置文件）；storage.conf（storage配置文件）

```
[root@localhost fdfs]# pwd
/etc/fdfs
[root@localhost fdfs]# ll
总用量 108
-rw-r--r--. 1 root root 23981 2月 6 03:39 anti-steal.jpg
-rw-r--r--. 1 root root 1455 2月 6 03:53 client.conf
-rw-r--r--. 1 root root 1461 2月 6 03:37 client.conf.sample
-rw-r--r--. 1 root root 858 2月 6 03:39 http.conf
-rw-r--r--. 1 root root 31172 2月 6 03:39 mime.types
-rw-r--r--. 1 root root 3700 2月 6 17:10 mod_fastdfs.conf
-rw-r--r--. 1 root root 7827 2月 6 07:23 storage.conf
-rw-r--r--. 1 root root 7829 2月 6 03:37 storage.conf.sample
-rw-r--r--. 1 root root 105 2月 6 03:39 storage_ids.conf
-rw-r--r--. 1 root root 7095 2月 6 07:24 tracker.conf
-rw-r--r--. 1 root root 7102 2月 6 03:37 tracker.conf.sample
```

tracker.conf配置内容如下：

端口：port=22122

tracker 基础目录：base\_path=/home/fastdfs，tracker在运行时会向此目录存储storage的管理数据。

### 3.1.3storage配置

storage.conf配置 内容如下：

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：base\_path=/home/fastdfs

磁盘存储目录：

store\_path0=/home/fastdfs/fdfs\_storage 此目录下存储上传的文件，在/home/fastdfs/fdfs\_storage/data下

store\_path1=...

...

有多个磁盘就定义多个store\_path

上报tracker的地址: tracker\_server=192.168.101.64:22122

如果有多个tracker则配置多个tracker, 比如:

tracker\_server=192.168.101.64:22122

tracker\_server=192.168.101.65:22122

....

## 3.1.4启动停止

fastDFS启动/停止脚本目录:

```
[root@localhost bin]# ll fdfs_*
-rwxr-xr-x. 1 root root 321856 2月 6 03:37 fdfs_appender_test
-rwxr-xr-x. 1 root root 321632 2月 6 03:37 fdfs_appender_test1
-rwxr-xr-x. 1 root root 308512 2月 6 03:37 fdfs_append_file
-rwxr-xr-x. 1 root root 307944 2月 6 03:37 fdfs_crc32
-rwxr-xr-x. 1 root root 308536 2月 6 03:37 fdfs_delete_file
-rwxr-xr-x. 1 root root 309304 2月 6 03:37 fdfs_download_file
-rwxr-xr-x. 1 root root 308880 2月 6 03:37 fdfs_file_info
-rwxr-xr-x. 1 root root 322712 2月 6 03:37 fdfs_monitor
-rwxr-xr-x. 1 root root 1125624 2月 6 03:37 fdfs_storaged
-rwxr-xr-x. 1 root root 331800 2月 6 03:37 fdfs_test
-rwxr-xr-x. 1 root root 326912 2月 6 03:37 fdfs_test1
-rwxr-xr-x. 1 root root 465416 2月 6 03:37 fdfs_trackerd
-rwxr-xr-x. 1 root root 309496 2月 6 03:37 fdfs_upload_appender
-rwxr-xr-x. 1 root root 310520 2月 6 03:37 fdfs_upload_file
```

fdfs\_trackerd: tracker脚本, 通过此脚本对 tracker进行启动和停止

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

fdfs\_storaged: storage脚本, 通过此脚本对 storage进行启动和停止

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

## 3.2 文件上传下载测试

### 3.2.1搭建环境

这里我们使用javaApi测试文件的上传。

java版本的fastdfs-client地址在: <https://github.com/happyfish100/fastdfs-client-java>, 参考此工程编写测试用例。

### 1) 创建maven工程

### 2) 添加依赖

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.9.RELEASE</version>
</parent>
<groupId>cn.itcast.javaee</groupId>
<artifactId>fastdfs</artifactId>
<version>1.0-SNAPSHOT</version>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <!-- https://mvnrepository.com/artifact/net.oschina.zcx7878/fastdfs-client-java -->
  <dependency>
    <groupId>net.oschina.zcx7878</groupId>
    <artifactId>fastdfs-client-java</artifactId>
    <version>1.27.0.0</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-io</artifactId>
    <version>1.3.2</version>
  </dependency>
</dependencies>
```

### 3) 配置 文件

在classpath:config下创建**fastdfs-client.properties**文件

```
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 = 192.168.101.64:22122
```

## 3.2.2 文件上传

```

//上传文件
@Test
public void testUpload() {

    try {
        ClientGlobal.initByProperties("config/fastdfs-client.properties");
        System.out.println("network_timeout=" + ClientGlobal.g_network_timeout + "ms");
        System.out.println("charset=" + ClientGlobal.g_charset);

        TrackerClient tc = new TrackerClient();

        TrackerServer ts = tc.getConnection();
        if (ts == null) {
            System.out.println("getConnection return null");
            return;
        }

        StorageServer ss = tc.getStoreStorage(ts);
        if (ss == null) {
            System.out.println("getStoreStorage return null");
        }

        StorageClient1 sc1 = new StorageClient1(ts, ss);

        NameValuePair[] meta_list = null; //new NameValuePair[0];
        String item = "C:\\Users\\admin\\Desktop\\1.png";
        String fileid;
        fileid = sc1.upload_file1(item, "png", meta_list);

        System.out.println("Upload local file " + item + " ok, fileid=" + fileid);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

```

### 3.2.3 文件查询

```

//查询文件
@Test
public void testQueryFile() throws IOException, MyException {
    ClientGlobal.initByProperties("config/fastdfs-client.properties");

    TrackerClient tracker = new TrackerClient();
    TrackerServer trackerServer = tracker.getConnection();
    StorageServer storageServer = null;

    StorageClient storageClient = new StorageClient(trackerServer,
        storageServer);

    FileInfo fileInfo = storageClient.query_file_info("group1",

```

```
"M00/00/01/wKh1QFrKBS0AW5AWAALcAg10vf4862.png");
    System.out.println(fileInfo);
}
```

### 3.2.4 文件下载

```
//下载文件
@Test
public void testDownloadFile() throws IOException, MyException {
    ClientGlobal.initByProperties("config/fastdfs-client.properties");

    TrackerClient tracker = new TrackerClient();
    TrackerServer trackerServer = tracker.getConnection();
    StorageServer storageServer = null;

    StorageClient1 storageClient1 = new StorageClient1(trackerServer,
        storageServer);
    byte[] result =
storageClient1.download_file1("group1/M00/00/01/wKh1QFrKBS0AW5AWAALcAg10vf4862.png");
    File file = new File("d:/1.png");
    FileOutputStream fileOutputStream = new FileOutputStream(file);
    fileOutputStream.write(result);
    fileOutputStream.close();
}
```

## 4 文件服务案例

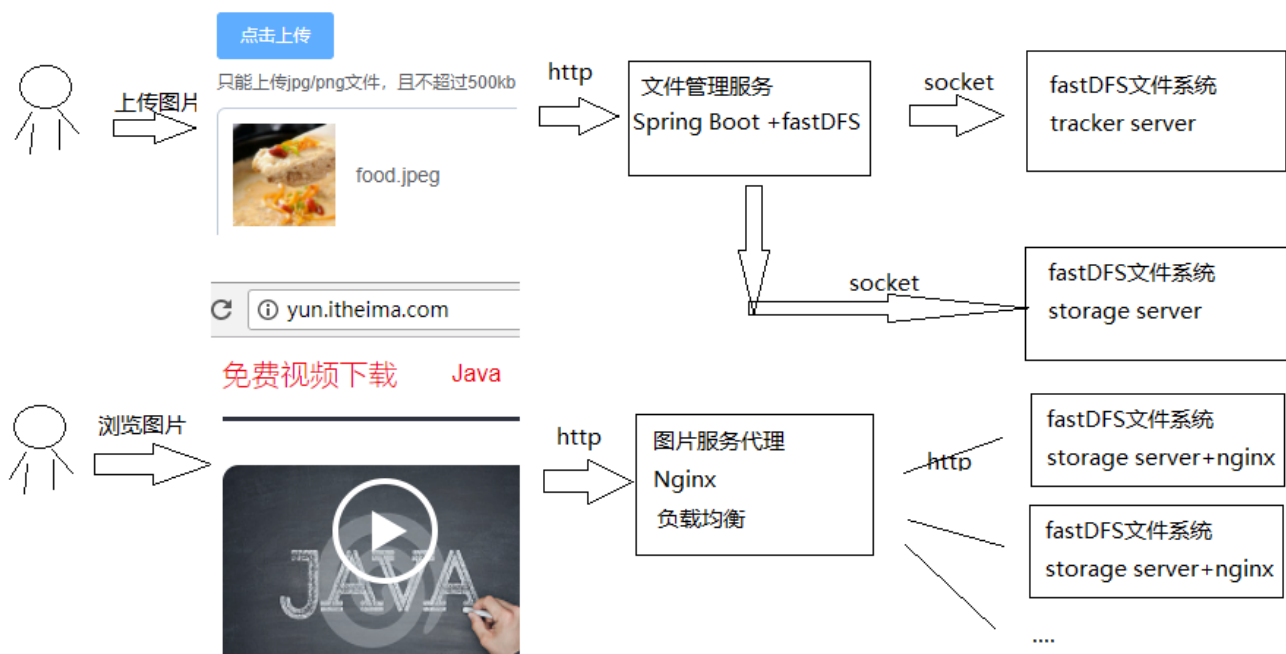
### 4.1 目标

通过文件服务案例达到以下目标：

- 1、理解fastDFS在实际项目中的使用方法。
- 2、能够使用fastDSF实现图片服务器。

### 4.2需求分析





说明如下：

- 1、管理员在管理系统中上传图片到fastDFS文件系统。
  - 1) 管理员进入管理系统，点击上传图片，选择本地图片
  - 2) 选择本地图片后，进行上传，前端浏览器会通过http调用文件管理服务的文件上传接口进行上传
  - 3) 文件管理服务通过socket请求fastDFS文件系统的上传图片，最终图片保存在fastDFS文件系统storage server中。
- 2、网友浏览免费视频的课程图片
  - 1) 网友输入[www.itcast.cn](http://www.itcast.cn)进入视频下载页面
  - 2) 前端浏览器通过图片地址请求图片服务器代理（nginx）
  - 3) 图片服务器代理根据负载情况将图片浏览请求转发到一台storage server
  - 4) storage server找到图片后通过nginx将图片响应给网友

## 4.3 功能开发

### 4.3.1 搭建fastDFS文件服务器

- 1) 安装fastDFS tracker和storage

略

- 2) 在storage server上安装nginx

在storage server上安装nginx的目的是对外通过http访问storage server上的文件。

使用nginx的模块FastDFS-nginx-module，它的作用是通过http方式访问storage中的文件，当storage本机没有要找的文件时向源storage主机代理请求文件。

详细参见 fastDFS安装教程

### 3) 在安装图片服务代理

图片服务代理的作用是负载均衡，根据storage server的负载情况将图片浏览请求均匀的转发到storage server上。

## 4.3.2 搭建文件管理服务

文件管理服务提供通过http方式上传文件，删除文件、查询文件的功能，管理员通过文件管理服务对文件服务器上的文件进行管理。

文件管理服务采用Spring Boot开发，文件管理服务通过与fastDFS交互最终将用户上传的文件存储到fastDFS上。

在fastDFS入门程序上继续开发：

### 1) 创建模型

```
package cn.itcast.java.fastdfs.model;

/**
 * Created by mrt on 2018/4/8.
 */
public class FileSystem {
    private String fileId;

    private String filePath;

    private long fileSize;

    private String fileName;

    private String fileType;

    public String getFileId() {
        return fileId;
    }

    public String getFilePath() {
        return filePath;
    }

    public long getFileSize() {
        return fileSize;
    }

    public String getFileName() {
        return fileName;
    }

    public String getFileType() {
        return fileType;
    }

    public void setFileId(String fileId) {
        this.fileId = fileId;
    }
}
```

```

    }

    public void setFilePath(String filePath) {
        this.filePath = filePath;
    }

    public void setFileSize(long fileSize) {
        this.fileSize = fileSize;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public void setFileType(String fileType) {
        this.fileType = fileType;
    }
}

```

## 2) 创建controller

```

package cn.itcast.java.fastdfs.web.controller;

import cn.itcast.java.fastdfs.model.FileSystem;
import org.csource.common.NameValuePair;
import org.csource.fastdfs.*;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

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

/**
 * Created by mrt on 2018/4/8.
 */
@RestController
@RequestMapping("/fileserver")
public class FileServerController {

    @Value("${itcast-fastdfs.upload_location}")
    private String upload_location;

    @PostMapping("/upload")
    @ResponseBody
    public FileSystem upload(@RequestParam("file")MultipartFile file){
        FileSystem fileSystem = new FileSystem();
        try {
            ClientGlobal.initByProperties("config/fastdfs-client.properties");

```

扩展名

```
TrackerClient tracker = new TrackerClient();
TrackerServer trackerServer = tracker.getConnection();
StorageServer storageServer = null;
StorageClient1 client = new StorageClient1(trackerServer, storageServer);
//MultipartFile转成File
String originalFilename = file.getOriginalFilename();//原始文件名

String extension = originalFilename.substring(originalFilename.lastIndexOf("."));//

String newFileName = UUID.randomUUID().toString() + extension;
File f = new File(upload_location + newFileName);
file.transferTo(f);

NameValuePair nvp[] = null;

String local_filename = f.getAbsolutePath();
//上传到文件系统
String fileId = client.upload_file1(local_filename, null,
    nvp);

//文件在文件系统中的路径
fileSystem.setFilePath(fileId);
fileSystem.setFileId(fileId);
long size = file.getSize();//文件大小
//文件大小
fileSystem.setFileSize(size);
String contentType = file.getContentType();
//文件类型
fileSystem.setFileType(contentType);
//文件名称
if (fileSystem.getFileName() == null || fileSystem.getFileName().equals("")) {
    //如果没有传入文件名称则存储文件的原始名称
    fileSystem.setFileName(file.getOriginalFilename());
}
//删除web服务器上的文件
f.deleteOnExit();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

return fileSystem;
}
}
```

### 3)创建application.yml

```
server:
  port: 22100
itcast-fastdfs:
  #文件上传临时目录
  upload_location: F:\\develop\\upload\\
```

#### 4) 创建spring boot启动类

```
package cn.itcast.java.fastdfs;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Created by mrt on 2018/4/8.
 */
@SpringBootApplication
public class FileServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(FileServerApplication.class);
    }
}
```

### 4.3.3 管理系统前端

管理员通过管理系统前端上传文件、查询文件、删除文件等操作。

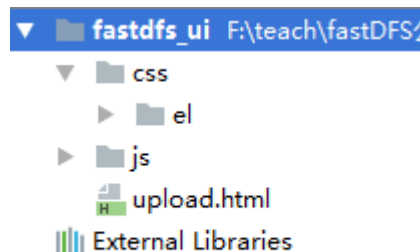
管理系统前端与文件管理服务器通过http交互，这当前流行的前后端分离的架构。

管理系统前端采用当前浏览器的vue.js前端框架实现。

#### 1、创建前端工程

使用webstorm创建前端工程

工程需要导入vue.js及element-ui



#### 2、创建upload.html

该文件实现文件上传及预览

### 1) 导入js及css

```
<link rel="stylesheet" href="css/el/index.css" />
<script src="js/vue/vue.min.js"></script>
<script src="css/el/index.js"></script>
```

### 2) 创建vue实例

```
var vm= new Vue({
  el: "#app",

  data:{

  },
  methods: {

  },

})
```

### 3) 添加element-ui的文件上传组件

```
<el-upload
  action="/fileserver/upload"
  list-type="picture-card"
  :on-preview="handlePictureCardPreview"
  :on-remove="handleRemove">
  <i class="el-icon-plus"></i>
</el-upload>
<el-dialog :visible.sync="dialogVisible">
  
</el-dialog>
```

定义数据对象及方法：

```
var vm= new Vue({
  el: "#app",

  data:{
    dialogImageUrl: '',
    dialogVisible: false
  },
  methods: {

    handleRemove(file, fileList) {
      console.log(file, fileList);
    },
    handlePictureCardPreview(file) {

      console.log(file)
    }
  }
})
```

```

        this.dialogImageUrl = 'http://192.168.101.64/'+file.response.filePath;
        this.dialogVisible = true;
    },
    })
})

```

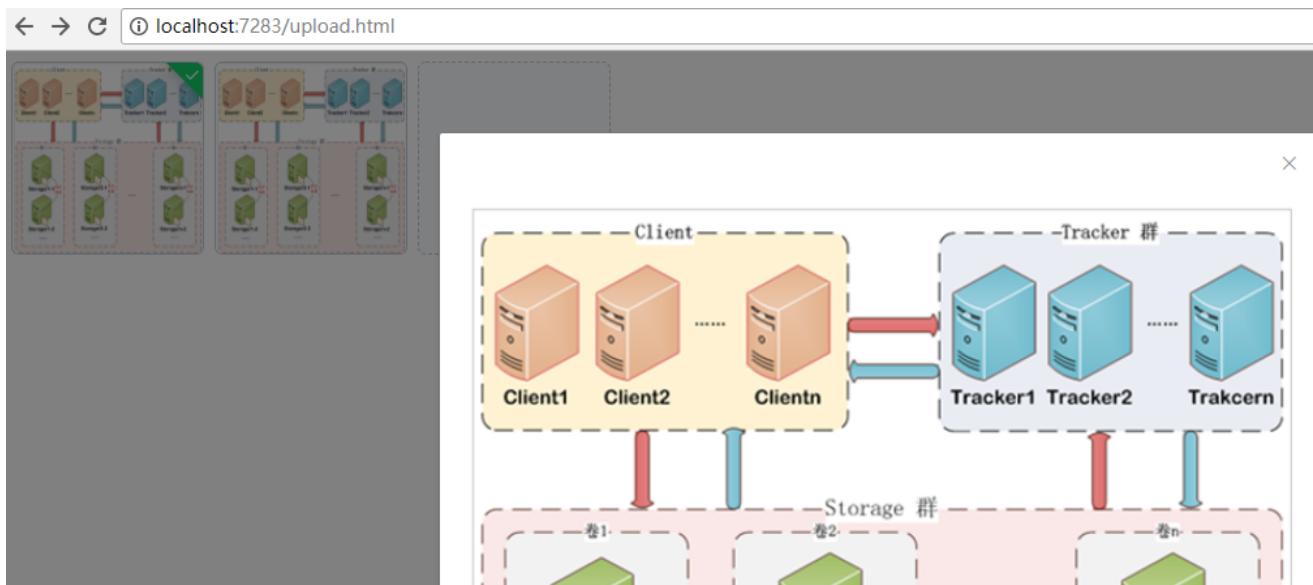
#### 4) 配置 nginx虚拟目录

```

server {
    listen      7283;
    server_name localhost;
    location / {
        alias   F:/teach/fastDFS公开课/code/fastDFSTest/fastdfs_ui/;
        index  index.html;
    }
    location ^~ /fileserver/ {
        proxy_pass http://127.0.0.1:22100/fileserver/;
    }
}

```

#### 5) 测试



### 4.3.4 课程图片浏览

网友（学习者）输入[www.itcast.cn](http://www.itcast.cn)网址进入免费视频下载页面，浏览器根据页面上的图片地址请求图片服务代理，由代理将请求转发到fastDFS storage server上，最终在浏览器看到图片。

上章节实现图片上传后已经实现图片的预览，本功能实现略。



## 5 总结

---

通过本次课程的学习您要达到以下目标：

### 1) 了解分布式文件系统的概念及应用场景

分布式文件系统是通过网络将单机上的文件系统组成一个网络文件系统。

分布式文件系统主要应用在大型互联网项目中，实现图片存储、音视频存储等服务。

分布式文件系统的优点：可以快速扩容存储，提高文件访问速度。

### 2) 理解fastDFS的工作原理

fastDFS由tracker和storage组成，它们都可以部署集群。

tracker负责调度，storage负责存储。

### 3) 掌握fastDFS存取文件方法

客户端与fastDFS采用socket协议通信，可以采用官方提供的java版本的fastDFS-client快速开发。

### 4) 能够动手搭建一个fastDFS文件服务器