尚马教育 JAVA 高级课程

fastDFS

文档编号：C14

创建日期： 2017-07-07

最后修改日期：2019-09-09

版 本 号：V3.5

电子版文件名：尚马教育-第三阶段-14.FastDFS 分布式文件存储.docx

文档修改记录：

更新日期	更新作者	更新说明	版本号
2017-08-25	张元林	初始版本	V1.0
2018-08-15	王绍成	Mybatis 版本更新	V2.0
2019-09-20	冯勇涛	课件格式以及课程深度加深	V3.0

目录

1. Fastdfs 介绍

1.1. 认识 fastdfs

1.1.1. 原作者版本

FastDFS 是由国人余庆所开发，其项目地址 https://github.com/happyfish100/fastdfs

FastDFS 是一个轻量级的开源分布式文件系统，主要解决了大容量的文件存储和高并发访问的问题，文件存取时实现了负载均衡。

支持存储服务器在线扩容,支持相同的文件只保存一份,节约磁盘。

FastDFS 只能通过 Client API 访问，不支持 POSIX 访问方式。

FastDFS 适合中大型网站使用，用来存储资源文件(如：图片、文档、视频等)

1.1.2. fastdfs_client 版本

tobato 项目团队在原作者 YuQing 与 yuqih 发布的 java 客户端基础上进行了大量重构工作，项目地址：https://github.com/tobato/fastdfs_client

支持对服务端的连接池管理； 支持上传图片时候检查图片格式，并且自动生成缩略图； 并且在 SpringBoot 当中自动导入依赖。

1.2. FastDFS 组成部分

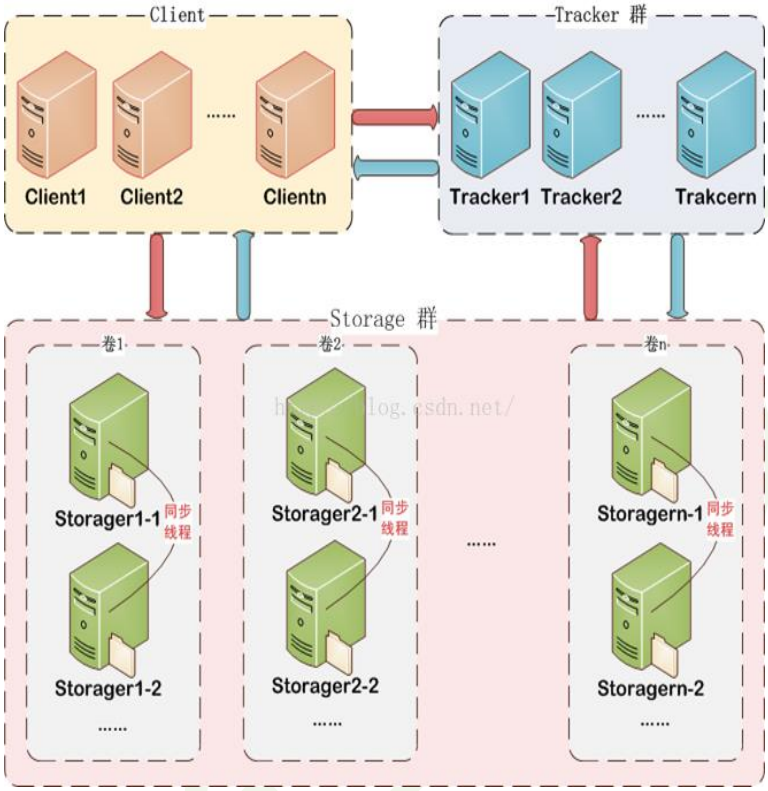
FastDFS 由跟踪服务器（tracker server）、存储服务器（storage server）和客户端（client）三个部分组成，主要解决了海量数据存储问题，特别适合以中小文件（建议范围：4KB < file_size <500MB）为载体的在线服务。

Tracker server(跟踪服务器) 跟踪服务器：用来调度来自客户端的请求。且在内存中记录所有存储组和存储服务器的信息状态。
Storage server(存储服务器)

用来存储文件(data)和文件属性(metadata)
Client 客户端 提供基本文件访问接口，比如 upload、download、append、delete 等，以客户端库的方式提供给用户使用。

1.3. FastDFS 服务器架构

- 存储服务器会定期向跟踪服务器发送状态信息（心跳包）。
- 客户端发起上传请求时,向跟踪服务器查询可用的存储服务器。
- 客户端向存储服务器上传文件内容,存储服务器保存文件并生成文件路径以及文件名
- 存储服务器将路径及文件名返回给客户端



2. FastDFS 客户端开发

2.1. jar 包准备

Maven:

```
<dependency>
  <groupId>org.csource</groupId>
  <artifactId>fastdfs-client-java</artifactId>
  <version>1.27</version>
</dependency>
```

2.2. 配置文件

fdfs_client.conf

```
connect_timeout = 2

network_timeout = 30

charset = UTF-8

http.tracker_http_port = 8111

tracker_server = 192.168.20.252:22122
```

2.3. 使用客户端 api

2.3.1. 初始化 fastdfs 服务

```
private static TrackerClient tc = null;

private static String httpPort = null;

static {

    URL resource = FastdfsUtil.class.getClassLoader().getResource("fdfs_client.conf");

    try {

        ClientGlobal.init(resource.getPath());//加载 conf 配置文件

        httpPort = ClientGlobal.getG_tracker_http_port() + "";// 得到 http 服务端口

        tc = new TrackerClient();//所有的跟踪服务器信息在 TrackerClient 对象中。

    } catch (Exception e) {

        e.printStackTrace();

    }

}
```

2.3.2. 上传文件

```
public static String upload(byte[] bytes, String fileName, int size) {
    String savePath = null;
    TrackerServer ts = null;//跟踪服务对象
    StorageServer ss = null;//存储服务对象
    try {
        ts = tc.getConnection();//得到一台具体的跟踪服务器
        StorageClient1 sc = new StorageClient1(ts, ss);//TrackerServer 分配 StorageServer
        //文件后缀
        String extName = getExtName(fileName);
        //元数据列表,(类似写的注释)上传文件习惯把上传日期，文件大小，文件真实名
        NameValuePair[] meta_list = {
                new NameValuePair(UPLOAD_TIME, System.currentTimeMillis() + ""),
                new NameValuePair(FILE_SIZE, size + ""),
                new NameValuePair(FILE_REALNAME, fileName)
        };

        savePath = sc.upload_file1(bytes, extName, meta_list);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        //释放连接
        try {
            if (ts != null) ts.close();
            if (ss != null) ss.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    }

    return savePath;

}
```

2.3.3. 下载文件

```
public static byte[] downloadFile(String savePath) {

    byte[] bs = null;

    TrackerServer ts = null;

    StorageServer ss = null;

    try {

        ts = tc.getConnection();

        StorageClient1 sc = new StorageClient1(ts, ss);

        bs = sc.download_file1(savePath);

    } catch (Exception e) {

        e.printStackTrace();

    } finally {

        try {

            if (ts != null) ts.close();

            if (ss != null) ss.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    return bs;

}
```

2.3.4. 删除文件

```
public static boolean deleteFile(String savePath) {

    boolean i=false;

    TrackerServer ts = null;

    StorageServer ss = null;

    try {

       ts = tc.getConnection();

       StorageClient1 sc = new StorageClient1(ts, ss);

       int isDel = sc.delete_file1(savePath);//表示删除文件，0 成功，2 失败

       i = isDel==0?true:false;

    } catch (Exception e) {

        e.printStackTrace();

    } finally {

        try {

            if (ts != null) ts.close();

            if (ss != null) ss.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    return i;

}
```

2.3.5. 生成 HttpUrl

```
public static String getHttpUrl(String savePath){

    String url ="http://";

    TrackerServer ts = null;
```

```
    try {
        ts = tc.getConnection();//得到一台具体的跟踪服务器
        InetSocketAddress address = ts.getInetSocketAddress();
        String hostName = address.getHostName();//得到跟踪服务器的主机名
        url = url+hostName+":"+httpPort+"/"+savePath;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (ts != null) ts.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return url;
}
```