

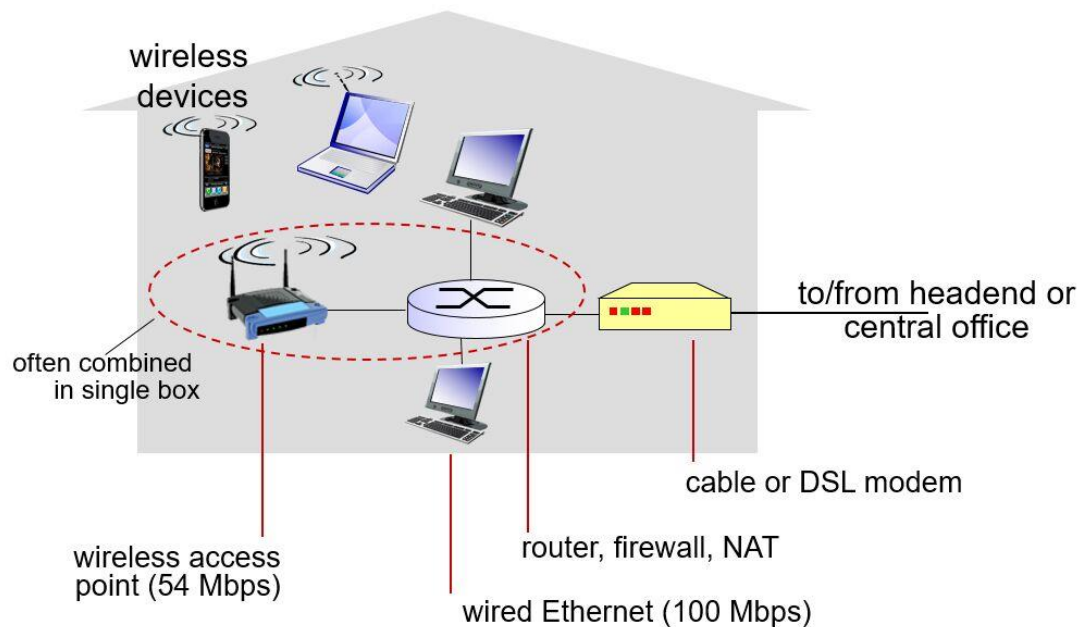
基于互联网网页的小型分布式文件系统 结题报告

王琚 夏昊琚 郑值 滕思洁
中国科学技术大学计算机学院

本项目获大学生创新创业训练计划资助

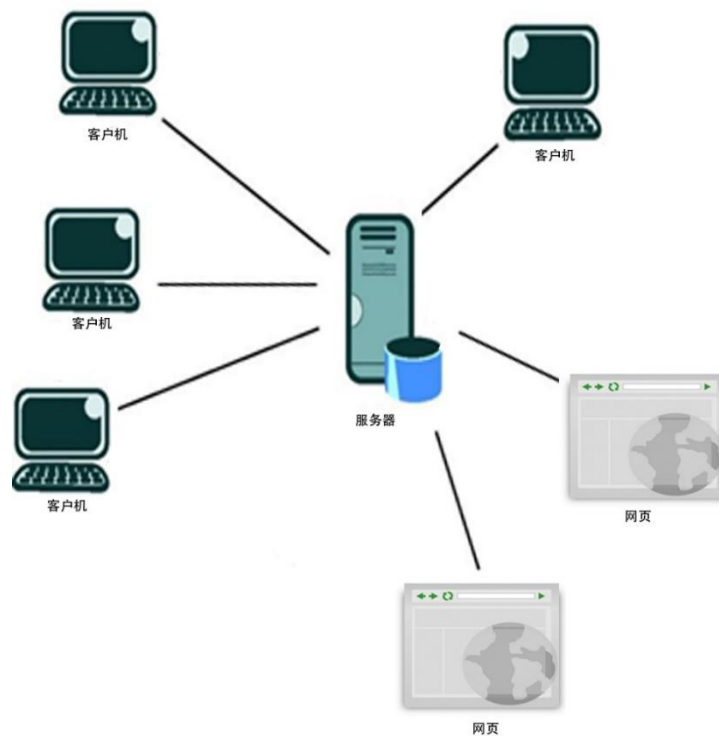
项目背景

- 个人资料多而分散
- 使用的平台不统一
- 云盘的不可靠
- 闲置存储设备多
- 有线/无线带宽的提升



项目概述

- ❑ 基于互联网的分布式文件系统
- ❑ 文件碎片分布储存在各个客户机中
- ❑ 中心服务器存储整个系统的元数据并协调各个客户机处理请求
- ❑ 通过网页访问整个文件系统



项目概述

https://github.com/IngramWang/DFS_OSH2017_USTC

54 commits

2 branches

0 releases

4 contributors

Java 94.3%

JavaScript 3.7%

HTML 2.0%

Summer-Summer Merge branch 'master' of github.com:IngramWang/DFS_OSH2017_USTC

Latest commit 91a0cee 9 minutes ago

HaojunXia_Server

完整版提交

包括全部的代码和部署范例

10 minutes ago

JunWang_Client

upload setup file

5 days ago

JunWang_Server

modified dateabase in order to support download files

6 days ago

TengSijie_Server

Merge branch 'master' of github.com:IngramWang/DFS_OSH2017_USTC

a month ago

ZhiZheng

debug file detector and uploader

a month ago

client

debug file detector and uploader

a month ago

report

modified database and rearranged files

2 months ago

server

modified dateabase in order to support download files

6 days ago

README.md

modified readme files

a month ago

优势创新

- Erasure Code
- 充分利用闲置设备
- 大容量与良好的阔放性
- 兼容多种访问平台

项目对比

GFS、AFS等等

- 面向大型文件系统
- 不支持离线访问
- 在操作系统层次提供服务

我们的设计

- 面向小型文件系统
- 在用户层次提供服务
- 使用Erasure Code编码节约空间

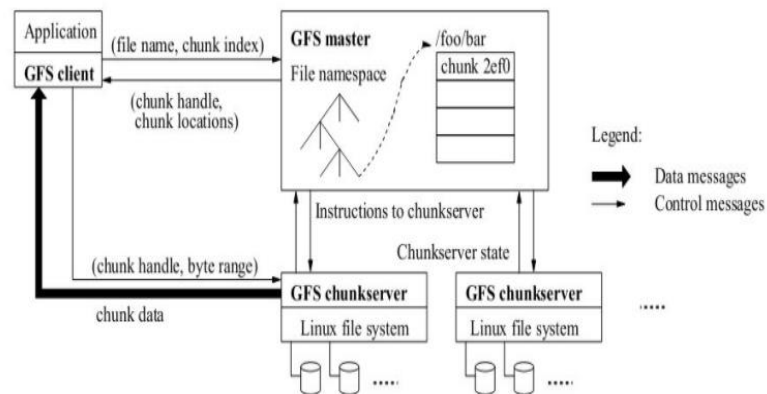
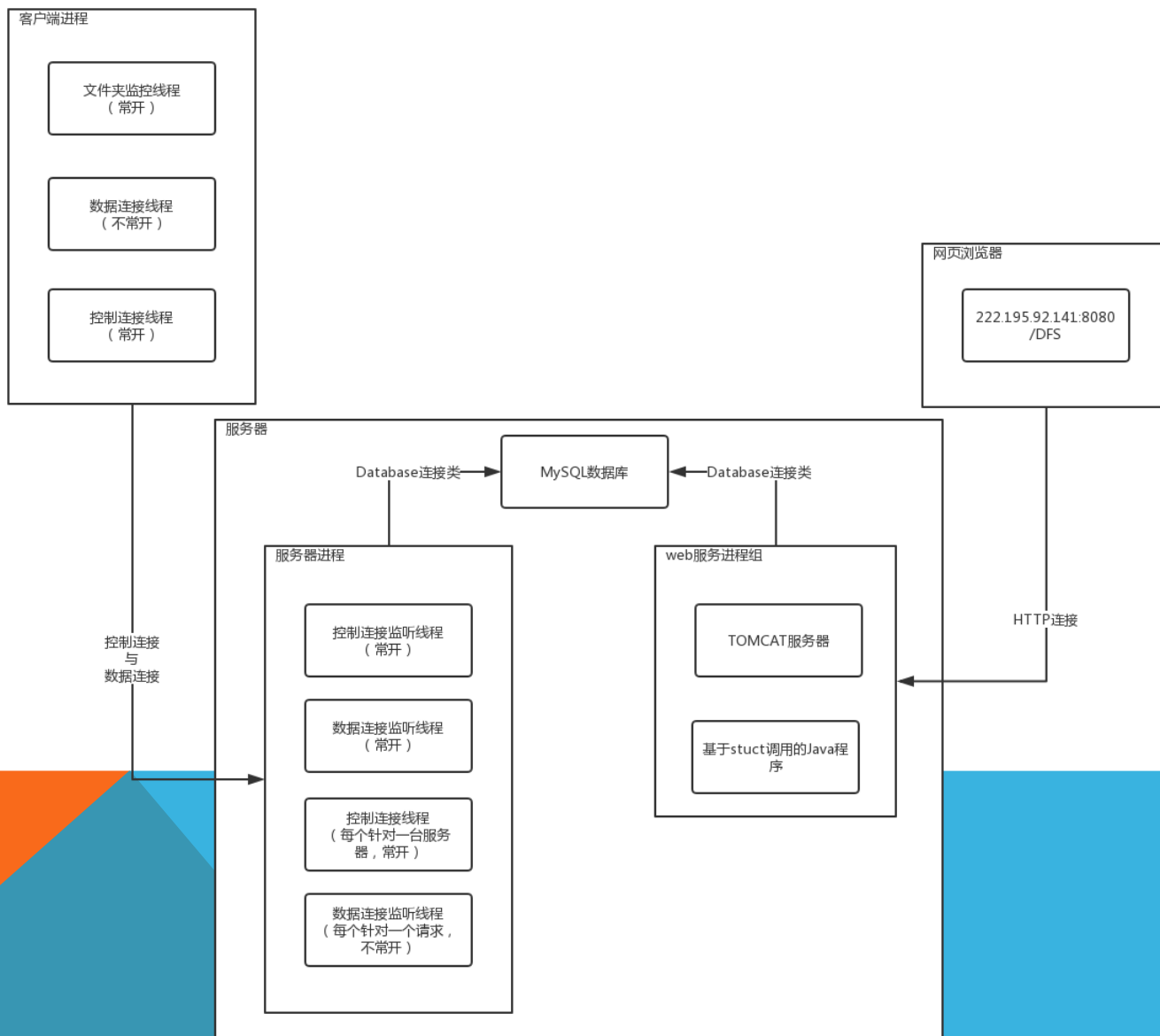


Figure 1: GFS Architecture

体系结构



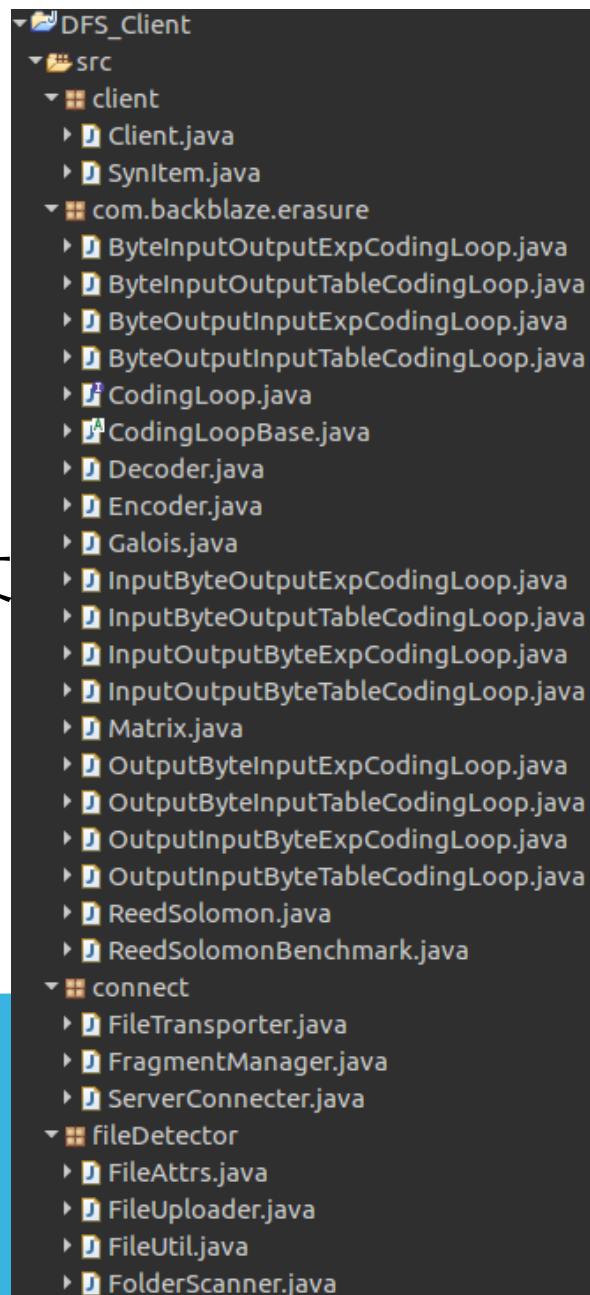
客户端设计

客户端功能：定时与服务器联络、定时扫描共享文件、根据服务器指示保存、传输本地文件块

客户端结构

客户端由4个包组成，依次是

- ❑ client: 启动与异常检测包
- ❑ com.backblaze.erasure: 文件分块包
- ❑ connect: 服务器连接包
- ❑ fileDetector: 文件夹监控包



SYMITEM

SynItem同步类用于在一个线程发生异常时向主线程传递错误信息

```
Decoder.java  FileDownloader.java  SynItem.java x
1 package client;
2
3 public class SynItem {
4     private int status;
5
6     public SynItem(int s) {
7         status = s;
8     }
9
10    public synchronized int getStatus(){
11        return status;
12    }
13
14    public synchronized void setStatus(int s){
15        status=s;
16        notifyAll();
17    }
18
19    public synchronized int waitChange(int oldValue) {
20        while (status==oldValue){
21            try{
22                wait();
23            } catch (InterruptedException e) {
24                System.out.println("interrupted");
25                return status;
26            }
27        }
28        return status;
29    }
30 }
```

COM.BACKBLAZE.ERASURE

- ❑ 基于backblaze公司提供的开源实现
- ❑ 利用Erasure Code进行文件分块
- ❑ 数据块：冗余块=1：1
- ❑ 块大小约为512kB

$$\begin{bmatrix} 1 & 0 & 0 \dots & 0 \\ 0 & 1 & 0 \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 \dots & 1 \\ 1 & 1 & 1 \dots & 1 \\ 1 & 2 & 3 \dots & n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 2^{m-1} & 3^{m-1} \dots & n^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

CONNECT

- ❑ FileTransporter类：文件传送类
- ❑ FragmentManager类：数据链接类，维护客户端上的文件碎片
- ❑ ServerConnector类：控制连接类
- ❑ 后两个类都继承了Thread类，可以在新线程中执行

```
received with 0 unread request!  
received with 0 unread request!  
received with 0 unread request!  
Encode Success  
received with 0 unread request!  
received with 4 unread request!  
Connect to server successfully(data)!  
Connect to server successfully(data)!  
Connect to server successfully(data)!  
Connect to server successfully(data)!  
received with 0 unread request!  
received with 0 unread request!  
received with 0 unread request!
```

FILEDETECTOR

- 根据配置文件监控一系列文件夹
- 自动上传文件夹中的内容到配置文件指定的逻辑位置
- 继承了Thread类，在新线程中执行

```
@Override
public void run() {
    FileUploader fUploader=new FileUploader();
    if (!fUploader.checkFolders(address)){
        System.out.println("ERR: can not register folder");
        synItem.setStatus(2);
        return;
    }
    while (detecting) {
        try {
            scanFiles();
            Thread.sleep(interval);
        } catch (InterruptedException ex) {
            ex.printStackTrace(System.err);
        }
    }
}
```

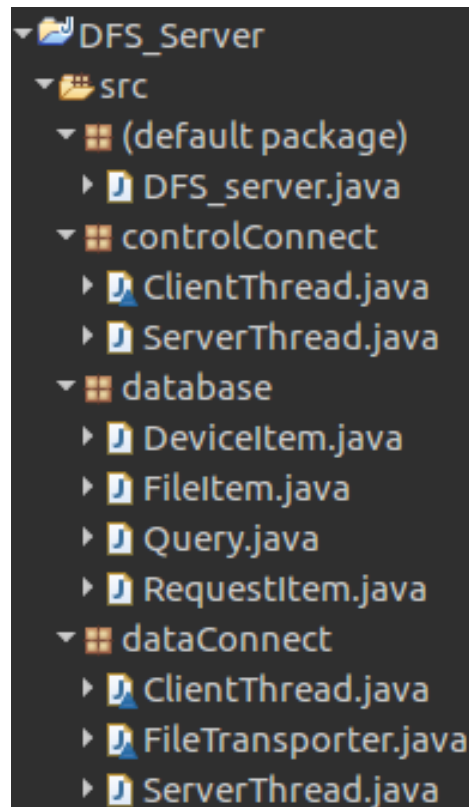
服务器设计

服务器功能：联系/管理各客户端、分发/收集数据块、维护
文件系统状态信息

服务器结构

服务器由4个包构成，依次是

- ❑ default: 服务器启动包
- ❑ controlConnect: 服务器客户端控制链接服务包
- ❑ dataConnect: 服务器客户端的数据链接服务包
- ❑ database: 数据库访问包



```
DFS_Server
├── src
│   ├── (default package)
│   │   ├── DFS_server.java
│   ├── controlConnect
│   │   ├── ClientThread.java
│   │   └── ServerThread.java
│   ├── database
│   │   ├── DeviceItem.java
│   │   ├── FileItem.java
│   │   ├── Query.java
│   │   └── RequestItem.java
│   └── dataConnect
│       ├── ClientThread.java
│       ├── FileTransporter.java
│       └── ServerThread.java
```

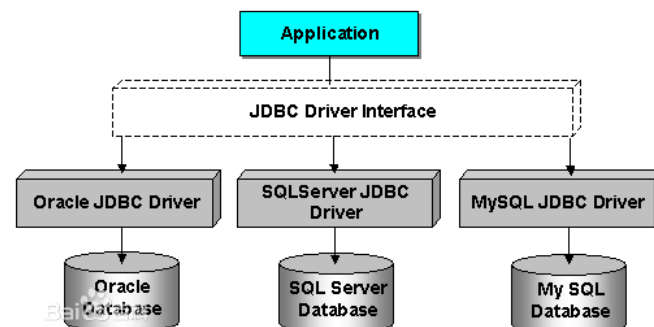
CONTROLCONNECT & DATACONNECT

- ❑ ServerThread类在一个新线程中执行，监听服务器的控制/数据链接端口
- ❑ 对于每个客户机的链接/请求，创建一个新线程并运行ClientThread类服务之

```
public void run() {  
    while (true) {  
        try {  
            Socket socket = server.accept();  
            // socket.setSoTimeout(10000);  
            ClientThread thread = new ClientThread(socket);  
            thread.start();  
            /*  
             * {catch(SocketTimeoutException s){  
             * System.out.println("Socket timed out!"); }  
             */  
            System.out.println("accepted a control link!");  
        } catch (IOException e) {  
            // TODO Auto-generated catch block  
            // e.printStackTrace();  
        }  
    }  
}
```


DATABASE

- ❑ 本服务器利用MySQL数据库存储所有的元数据
- ❑ DATABASE包定义了数据库的访问方法及相关的数据结构
- ❑ 使用Java数据库连接（JDBC）作为访问数据库的应用程序接口



元数据维护

系统中的元数据

为了维护分布式文件系统的状态，服务器的数据库中保存了以下5种类型的元数据，这些元数据都通过DATABASE包进行访问

- ❑ FILE
- ❑ DEVICE
- ❑ FRAGMENT
- ❑ REQUEST
- ❑ USER

```
mysql> use DFS
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_DFS |
+-----+
| DEVICE        |
| FILE          |
| FRAGMENT      |
| REQUEST       |
| USER          |
+-----+
5 rows in set (0.00 sec)
```

逻辑位置与物理位置

- 文件的逻辑位置通过FILE表记录
- 文件ID是文件对应的碎片的ID的前缀
- 文件碎片的物理位置通过FRAGMENT表记录

```
CREATE TABLE `FRAGMENT` (  
  `ID` int NOT NULL,  
  `PATH` char(20) NOT NULL DEFAULT '',  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `FILE` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `NAME` char(20) NOT NULL DEFAULT '',  
  `PATH` char(60) NOT NULL DEFAULT '',  
  `ATTRIBUTE` char(10) NOT NULL DEFAULT '',  
  `TIME` char(10) NOT NULL DEFAULT '',  
  `NOA` int NOT NULL DEFAULT 1,  
  `ISFOLDER` boolean NOT NULL DEFAULT false,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

设备信息

- ❑ DEVICE表记录了系统中的设备信息
- ❑ 系统为每个设备分配一个全局唯一且永久的ID，从而标识设备上碎片的物理位置
- ❑ ISONLINE记录了设备的在线情况

```
CREATE TABLE `DEVICE` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `IP` char(20) NOT NULL DEFAULT '',  
  `PORT` int NOT NULL DEFAULT 0,  
  `ISONLINE` boolean NOT NULL,  
  `RS` int NULL DEFAULT 0 ,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

请求信息

- ❑ REQUEST表记录了系统中等待客户端处理的请求
- ❑ 本系统定义了3种不同的请求类型，分别是服务器取分块、服务器将分块发送给客户端和删除客户端上的分块

```
CREATE TABLE `REQUEST` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `TYPE` int NOT NULL DEFAULT 0,  
  `FRAGMENTID` int NOT NULL DEFAULT 0,  
  `DEVICEID` int NOT NULL DEFAULT 0,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

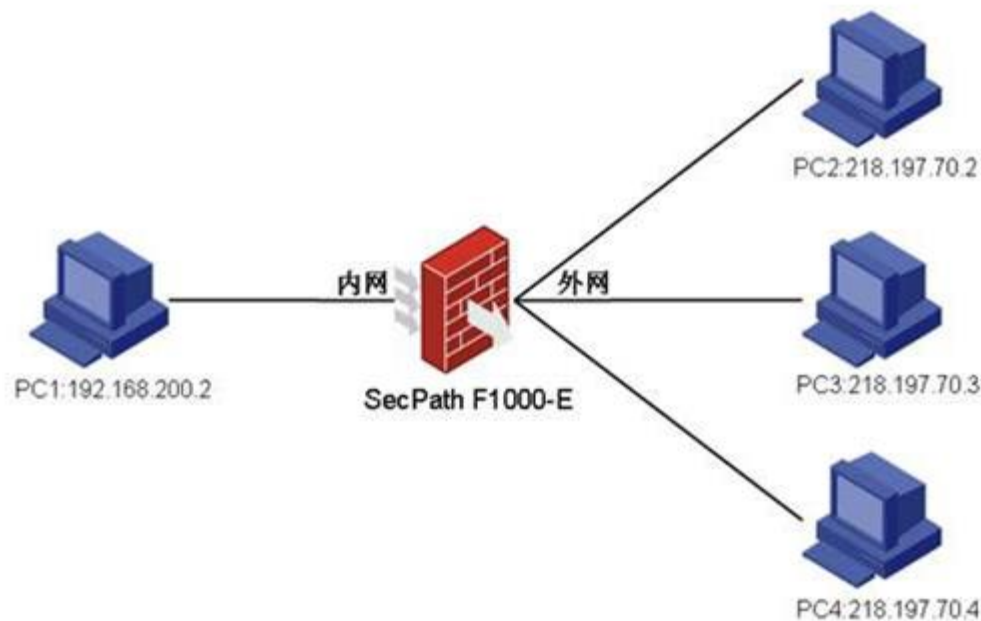
服务器-客户端通信

背景：NAT

□ 网络地址转换

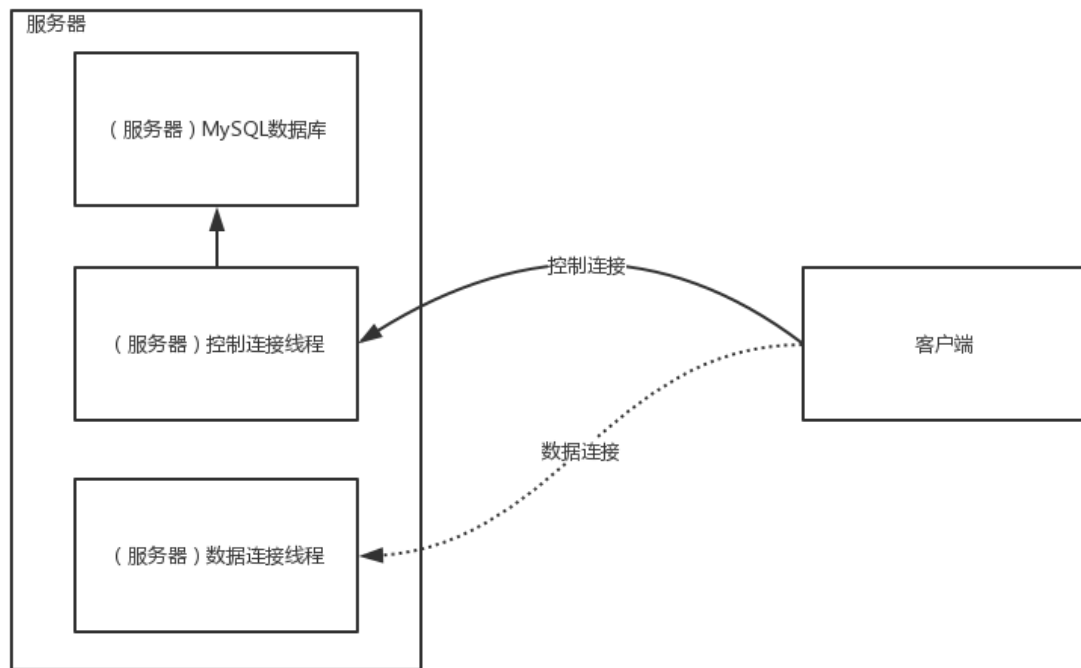
(NAT) 是一种在IP数据包通过路由器或防火墙是重写来源IP/目的IP的技术

□ NAT机制有可能使服务器无法主动向客户端发起连接



控制连接与数据连接分离的带外传输

- ❑ 服务器与客户端保持长期的控制连接
- ❑ 当服务器需要客户端响应请求时，服务器先将请求通过控制连接发往客户端，再由客户端主动建立与服务器的数据连接



控制连接报文

控制连接传送客户端状态报文与处理请求报文

客户端状态报文由客户端定期发送到服务器用于保持连接，服务器收到该报文后将回复等待其处理的请求数量

处理请求报文在客户端发现自己有未处理的请求时发送到服务器，服务器该报文后将回复具体的请求

数据连接报文

系统中定义了以下6类数据连接报文：

1. 客户机碎片上传报文
2. 客户机碎片下载报文
3. 客户机碎片删除报文
4. 文件上传报文
5. 文件碎片上传报文
6. 文件夹检查报文

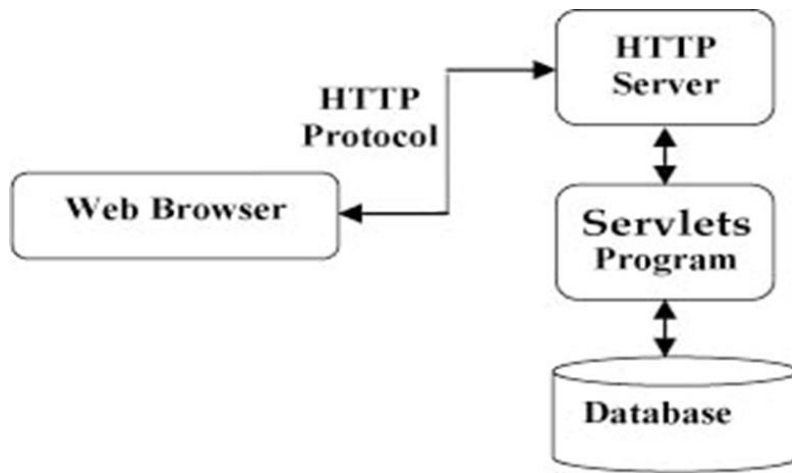
WEB服务

WEB服务

- Tomcat WEB应用服务器
- Struts2 动态网站网站应用调度框架
- BOOTSTRAP网页主题
- JQuery+AJAX异步C/S通讯+JSON+MySql
- 主要代码架构

WEB服务

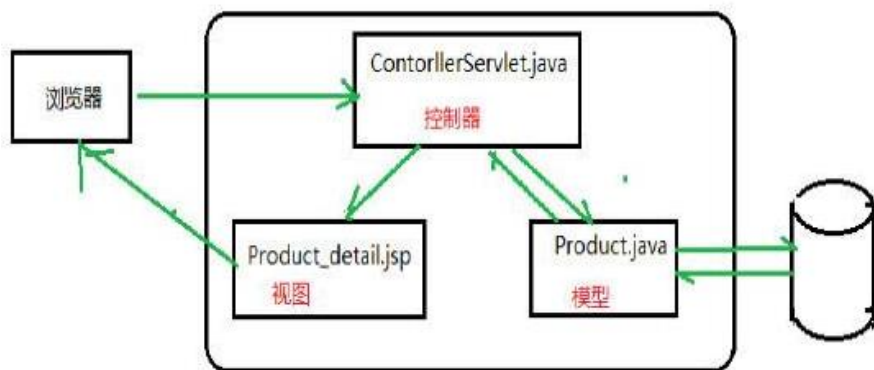
□ Tomcat WEB应用服务器



- 本身支持一般的http服务
- 提供JSP的JAVA运行环境和JavaBean的运行环境
- 响应浏览器请求，调用服务器端java函数
- 在以上基础上，搭建动态网站

WEB服务

□ Struts2 动态网站网站应用调度框架



- Tomcat基础上
- 采用MVC设计模式
- 控制器从视图读取数据，控制用户输入，向模型发送数据
- 管理复杂的应用程序，可以在一个时间内专门关注一个方面
- 简化了分组开发，不同开发人员可同时开发视图、控制器逻辑和业务逻辑

WEB服务

□ BOOTSTRAP网页主题



- 基于 HTML、CSS、JAVASCRIPT
- 适用于快速开发+可视化开发+提供可重用组件
- 响应式 CSS 能够自适应于不同尺寸电脑和手机
- 所有的主流浏览器都支持 Bootstrap

WEB服务

□ JQuery+AJAX异步C/S通讯+JSON+MySql

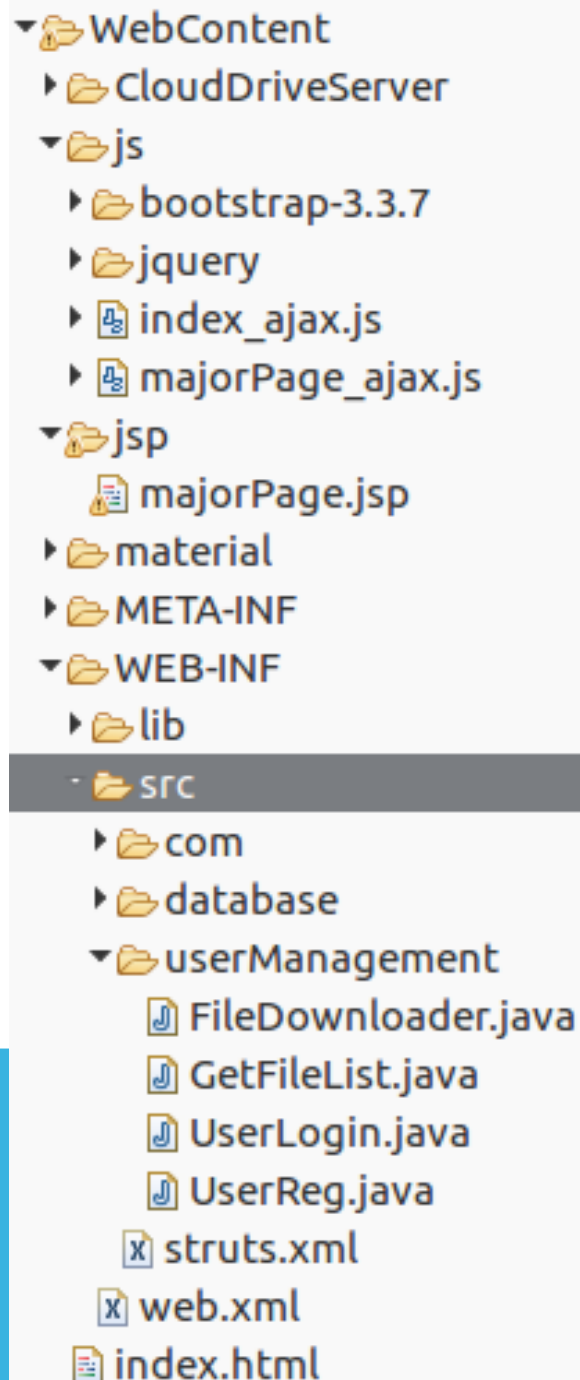
- JQuery-----"写的少，做的多"，极大地简化了 JavaScript 编程
- AJAX-----不重载全部页面，实现对部分网页的更新
- JSON-----轻量级的数据交换格式
- MySql-----关系数据库管理系统



WEB服务

□ 主要代码架构

- Html
- Js
- Jsp
- JavaBean
- 配置文件
- Material



系统演示

欢迎大家用浏览器（手机或者电脑；暂不支持IE系列）登陆
我们的网站

[HTTP://222.195.92.141:8080/DFS/](http://222.195.92.141:8080/DFS/)
进行体验

让文件触手可及

前景展望

- 文件访问和管理权限的实现（多用户）
- 在实现的基础上分析改善性能，优化文件系统设计
- 文件加密和安全 + Web服务的安全性保障
- 引入长期在线客户端 如官方云盘 高可用性
- 将文件拼接任务交给浏览器本地的js
- 服务器端微内核实现
- 界面的美观性

市场潜力

- ❑ 完全免费 + 无需app（百度云盘下载文件夹需要客户端等等限制）
- ❑ 小巧实用 便捷文件分享（提供账号密码，只要设置好文件权限（可以设置role---角色，对应一组权限）即完成了分享）
- ❑ 多人协作（网页中的文件总是最新的 只有一个最新版本）
- ❑ 完全跨平台（客户端+浏览器） 移动访问
- ❑ 文件碎片存储位置可定制化 利用好闲置的硬盘资源（比如租赁的服务器和家里的电脑）
- ❑ 文件下载速度快（针对企业和学校网络 服务器在本地局域网）
- ❑ 高可用（分布式+Erasure Code高效分割）
- ❑ 高安全（碎片在本地 仅靠部分云端碎片无法恢复文件）

THANK YOU!
谢谢！