

Java&J2EE 应用与开发

学号 1 : 201592038

姓名 1 : 阚建明

班级 1 : 软网 1503

学号 2 : 201592442

姓名 2 : 邹洋奕

班级 2 : 软网 1503

大作业得分点

编码规范及易读性 (15 分) : _____

功能实现 (35 分) : _____

异常处理 (25 分) : _____

文档质量 (25 分) : _____

大作业总成绩 (占 60%) : _____

平时作业总成绩 (占 40%) : _____

期末总成绩 (100 分) : _____

目录

1	功能要求分析	3
1.1	FILESERVER	3
1.2	STORAGE NODE	3
1.3	FILECLIENT	3
1.4	监控程序	4
2	具体实现思路	4
3	各模块及主要功能展示.....	5
3.1	FILESERVER 模块	5
3.1.1	<i>CommunicateWithStorageStrategy.....</i>	<i>5</i>
3.1.2	<i>FCStrategy</i>	<i>6</i>
3.1.3	<i>FileStorageServer+FileStorageDataDealRunnable</i>	<i>6</i>
3.2	FILECLIENT 模块.....	7
3.2.1	<i>CommunicateWithFileServerOperator.....</i>	<i>7</i>
3.2.2	<i>FileTransOperator</i>	<i>9</i>
3.2.3	<i>FileTransToStorageSupport.....</i>	<i>9</i>
3.3	FILESTORAGE	10
3.3.1	<i>CommunicateWithServerStrategy.....</i>	<i>10</i>
3.3.2	<i>TransWithClientStrategy.....</i>	<i>11</i>
3.4	MONITOR.....	11
4	不足改进.....	12
4.1	问题一及解决方案.....	12
4.2	问题二及解决方案.....	12
5	引用类库文献	13

1 功能要求分析

本应用基于 Socket 编程模拟实现文件的分布式存储，需要实现四个模块的功能。

1.1 FileServer

FileServer 服务器，主要用于负载均衡，管理文件信息和存储节点信息。在文件上传时，客户端需向 FileServer 请求存储节点的信息，此时服务器在分配存储节点时要考虑负载均衡。在 FileServer 上通过 Map 集合保存存储节点的信息以及文件的信息。服务器宕机之前将 Map 集合中存储的信息序列化到文件中，下次启动的时候再从文件中序列化回来。

1.2 StorageNode

StorageNode 服务器主要用于存储用户上传的文件，本地不能保存过多的信息。需要一个配置文件，配置 StorageNode 的 IP、端口、容量、文件存储路径等信息，在程序启动的时候要加载配置文件的信息。文件在保存的时候文件名为 FileServer 给文件分配的 UUID 的值，不能使用原文件的名字。在收到文件的时候要向 FileServer 请求同组服务器的信息，将刚收到的文件备份到同组的其他节点服务器。

1.3 FileClient

FileClient 应用程序主要实现文件的上传，下载，删除等功能。在上传时需要向 FileServer 请求可用的节点服务器组的列表，FileClient 再根据收到的节点服务器的信息，将文件上传到节点服务器。如果主节点服务器不可用，或者在传输过程中与节点服务器的连接中断，则应该有容错的功能，将文件上传到同组的备份节点服务器上。在下载的时候 FileClient 需凭借 UUID 向 FileServer 询问文件具体在哪个节点服务器上保存了，再根据收到的信息，找到对应的节点服务器下载文件。在删除文件时，FileClient 将删除文件的请求发给 FileServer，委托 FileServer 删除掉存储在节点服务器上的文件。

1.4 监控程序

监控程序，需要对节点服务器的状态进行监控，并以表格的方式显示出来。

2 具体实现思路

首先四个功能模块分属于四个项目。统一协商好一套协议，在通信过程中，根据不同的 code 执行不同的行为。

在 Socket 通信过程中，统一采用 `DataInputStream`、`DataOutputStream`。通信过程中如果传输基本数据类型，则直接调用上述两个流操作对象对应的函数即可；如果传输对象，则引入 GSON 库，将对象序列化为 JSON 格式的字符串，传到目标主机后在用 GSON 库将 JSON 串反序列化为原对象；如果传输文件，则建立缓冲数组，分配传输字节数组即可。

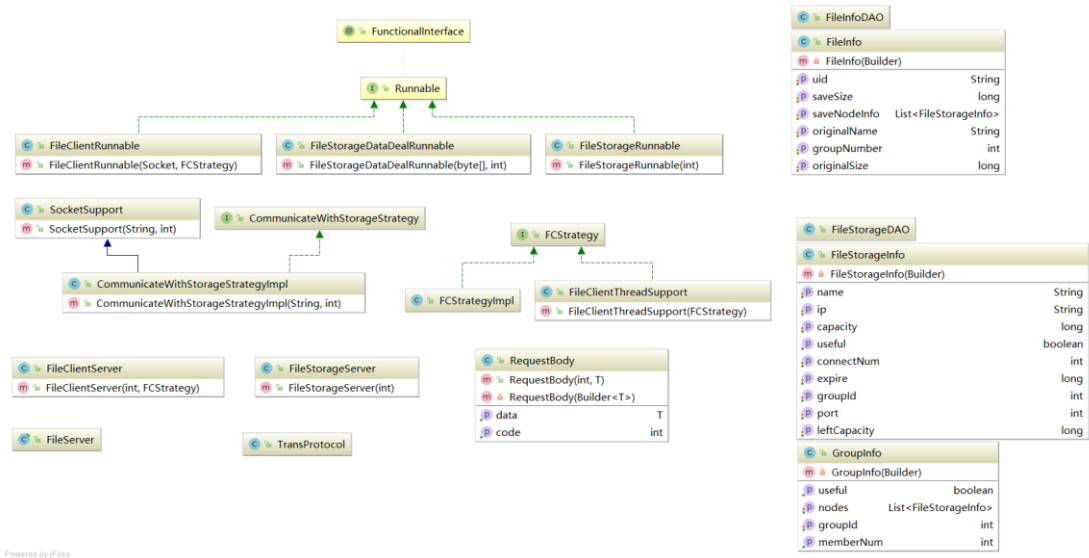
为了实现 FileServer 服务器对节点服务器的管理，节点服务器自启动开始便向 FileServer 发送 UDP packet 作为心跳包。实时向 FileServer 汇报自己的情况。对于 UDP 通信不可靠问题，在数据末尾添上 CRC 校验码，FileServer 在根据收到的信息计算 CRC 校验码，并进行比对，判断本次传输是否有效。

为了保护用户的文件安全，不能将原文件存于节点服务器，在文件上传的时候将文件分块压缩、加密，先向输出流中输出本数据块在压缩、加密以后的大小，紧接着再将处理后的数据块输出。在节点服务器上接收的时候先读取一个整形数，表示下一个传输的数据块的长度，再调用 `readFully` 方法读取下一个块，输出到文件中的时候也是先输出块的长度，紧接着输出数据块。文件下载的时候采用同样的思路读取、解压、解密，最后还原出原文件。

在监控程序方面则采用 JavaWeb 方式实现，在 JavaWeb 服务器收到 Get 请求后随即向 FileServer 发起 Socket 通信，获取到各存储节点的信息，再将数据通过 JSP 页面显示出来，同时设定每隔一秒自动刷新，以达到实时监控的效果。

3 各模块及主要功能展示

3.1 FileServer 模块



3.1.1 CommunicateWithStorageStrategy

由服务器发起的与节点服务器间的通信只有一个，那就是删除文件的时候，服务器在收到客户端的委托请求后，向对应的节点服务器发送删除文件的请求。

```
/**
 * 与节点服务器的通信策略
 * Created by Sunny on 2017/7/14 0014.
 */
public interface CommunicateWithStorageStrategy {
    /**
     * 让节点服务器删除文件
     * @param uuid 要删除文件的uuid
     * @param nodeName 要让哪个节点服务器删除文件
     * @return 返回删除的结果
     * @throws IOException
     */
    int remove(String uuid, String nodeName) throws IOException;
}
```

3.1.2 FCStrategy

- ✧ 这个策略用于应对客户端和监控程序的请求，支持各种信息的增删改查以及代理客户端删除，文件等。在 service 函数具体实现的时候根据不同的 Code 区分不同请求的目的，并作出相应的响应动作。

```
/**
 * 处理请求的策略（可能是客户端的请求，也可能是监控程序的请求）
 * Created by Sunny on 2017/7/6 0006.
 */
public interface FCStrategy {
    /**
     * 在service函数中处理来自客户端或监控程序的请求
     * 根据不同的code区分不同的请求，采取不同的响应
     * @param socket
     * @throws IOException
     */
    public void service(Socket socket) throws IOException;
}
```

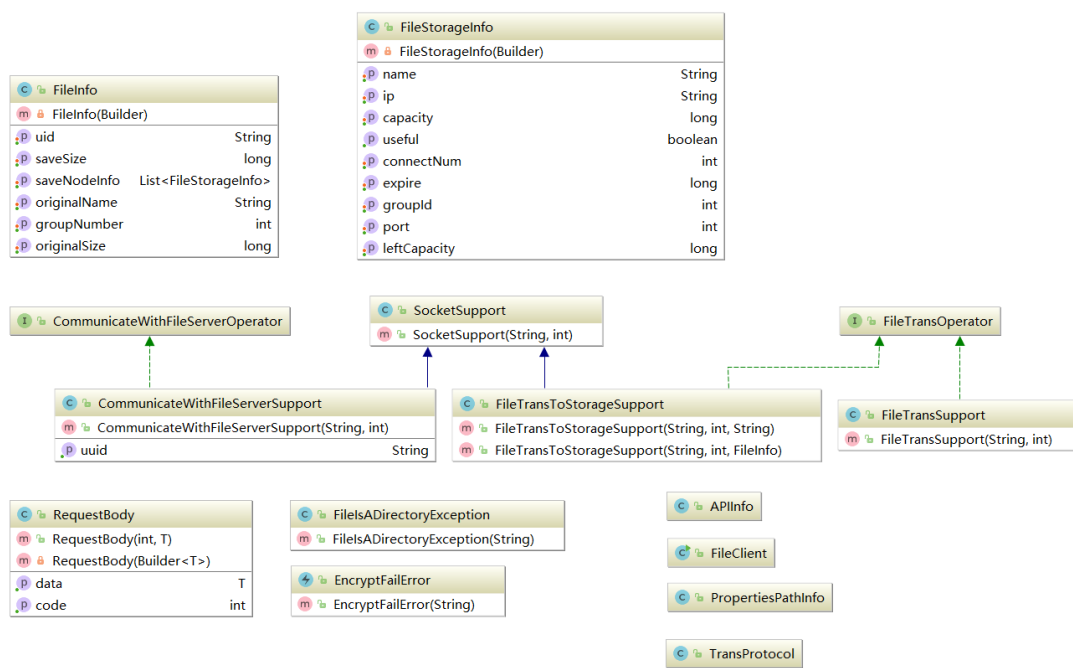
- ✧ 以下是不同的协议类中定义的不同 code 代表不同的请求

```
public static final int CODE_UPLOAD_FILE = 1; //上传文件请求
public static final int CODE_DOWNLOAD_FILE = 2; //下载文件请求
public static final int CODE_REMOVE_FILE = 3; //移除文件请求
public static final int CODE_GET_UPLOAD_STORAGE = 4; //获取可上传节点组请求
public static final int CODE_GET_DOWNLOAD_STORAGE = 5; //获取下载文件组信息
public static final int CODE_UPDATE_STORAGE_INFO = 6; //存储节点向服务器更新自己的信息
public static final int CODE_GET_FILE_INFO_BY_UUID = 7; //根据UUID获取文件的信息
public static final int CODE_GET_GROUP_INFO = 8; //存储节点向服务器询问其他节点的信息，以向其他节点备份文件
public static final int CODE_BACKUP_FILE = 9; //存储节点之前互相备份文件
public static final int CODE_UPDATE_FILE_INFO_BY_UUID = 10; //根据UUID更新文件信息
public static final int CODE_MONITOR_GET_INFO = 11; //监控程序向服务器请求所有节点组的数据
public static final int CODE_MONITOR_GET_FILE_INFO = 12; //监控程序向服务器请求所有文件的信息
```

3.1.3 FileStorageServer+FileStorageDataDealRunnable

这两个类主要负责接收来自节点服务器发送的心跳包。其中 FileStorageServer 负责接收 UDP 报文，并将其放入阻塞队列当中，而 FileStorageDataDealRunnable 则跑在另一个线程，负责从阻塞队列中取报文并处理。（心跳包有三种，一种只用于更新节点服务器信息；一携带删除文件请求；一种携带添加文件请求。）

3.2 FileClient 模块



3.2.1 CommunicateWithFileServerOperator

该策略接口主要定义了客户端与服务器之间的通信规则，要求实现该接口的类必须实现其定义的五個函数，分别为：

- ✧ **List<FileStorageInfo> getUploadAvailableStorage(String path)**
用于在上传文件时向服务器询问应该上传到哪一个节点服务器组。
- ✧ **List<FileStorageInfo> getDownloadAvailableStorage(String uuid);**
用于在下载文件的时候将 UUID 传给服务器，询问应该到哪一个节点组去下载该 UUID 对应的文件。
- ✧ **FileInfo getFileInfoByUUID(String uuid);**
通过 UUID 向服务器查询该 UUID 对应的文件信息。
- ✧ **int removeFile(String uuid);**
委托服务器删除所有存储在节点服务器上该 UUID 对应的文件。
- ✧ **void updateFileInfoByUUID(boolean isUploadSuccess, String uuid, FileInfo fi);**

用于通知服务器本次文件传输的情况，如果文件传输成功，则让服务器更新存在服务器上该文件的信息（文件在节点服务上保存的真是大小）。如果文件上传失败，则让服务器删除该文件对应的表项。

```
/**
 * 该接口主要用于FileClient向FileServer请求可用的存储节点的信息，以及委托服务器删除文件
 * Created by Sunny on 2017/7/6 0006.
 */
public interface CommunicateWithFileServerOperator {
    /**
     * 向服务器请求可用的存储节点以上传文件，
     * FileServer会返回UUID作为即将上传到存储节点的文件的UUID;
     */
    List<FileStorageInfo> getUploadAvailableStorage(String path) throws IOException;

    /**
     * 通过uuid向FileServer索取可用存储节点以下载文件
     * uuid可能无效，也有可能uuid有效但是存储节点已经宕机，同样无法下载到目标文件，需要抛出异常
     * @param uuid
     */
    List<FileStorageInfo> getDownloadAvailableStorage(String uuid) throws IOException;

    /**
     * 通过uuid从FileServer上获取文件的信息
     * @param uuid 要获取文件的uuid
     * @return 如果获取成功，返回uuid对应的文件信息
     * @throws IOException
     */
    FileInfo getFileInfoByUUID(String uuid) throws IOException;
}
```

```
/**
 * 通知服务器删除文件
 * @param uuid 要删除文件的uuid
 * @return 返回删除的结果
 * @throws IOException
 */
int removeFile(String uuid) throws IOException;

/**
 * 在文件传输到节点服务器以后，客户端要向FileServer汇报上传是否成功
 * 如果上传成功，则让服务器更新文件信息，添加saveSize属性
 * 如果上传失败，则让服务器删除该文件信息的表项
 * @param isUploadSuccess 是否上传成功
 * @param uuid 文件的uuid
 * @param fi 文件的信息
 * @throws IOException
 */
void updateFileInfoByUUID(boolean isUploadSuccess, String uuid, FileInfo fi) throws IOException;
}
```


3.2.2 FileTransOperator

该策略主要定义了提供给用户的三个接口，分别为上传、下载和删除文件。具体实现的时候会分别依赖于与服务器和节点服务器的通信策略。

```
/**
 * 该接口用于执行客户端最基本的三个功能
 * Created by Sunny on 2017/7/6 0006.
 */
public interface FileTransOperator {

    /**
     * 实现询问FileServer后将文件上传到可用的存储节点
     * 1. 首先向FileServer索要一组可用存储节点的信息
     * 2. 尝试向主节点传输文件
     * 3. 如果主节点不可用或中途宕机，要捕获异常并转而尝试将文件传输到备份节点
     * @param path 即将要上传的文件的路径
     * @return 若上传成功则返回uuid
     */
    FileInfo upload(String path) throws IOException, NoSuchAlgorithmException, EncryptFailError;

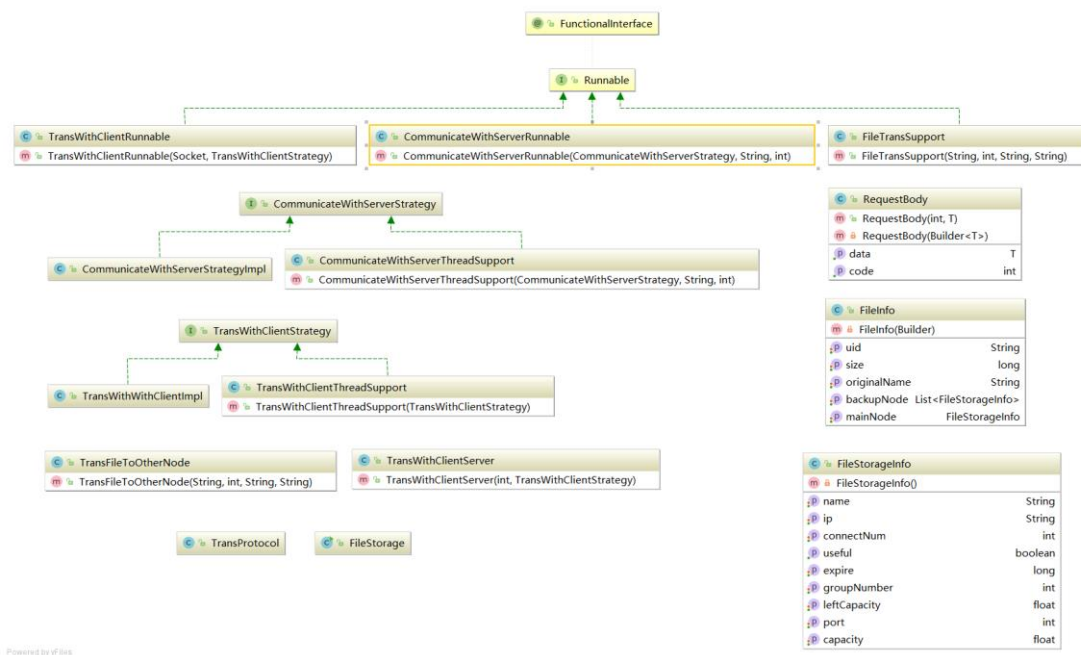
    /**
     * 实现询问FileServer后到可用节点上下载uuid对应的文件
     * 1. 首先询问FileServer，如果uuid无效，或者存储该文件的节点组都宕机，则下载失败
     * 2. 尝试从主节点下载文件，如果主节点中途宕机，这捕获异常，并转而从备份节点下载文件
     * @param uid
     */
    void download(String uid) throws IOException;

    /**
     * 实现请求FileServer帮其移除uuid对应的文件
     * @param uid
     */
    void remove(String uid) throws IOException;
}
```

3.2.3 FileTransToStorageSupport

该类提供了客户端和节点服务器之间的通信。其实现了 FileTransOperator，并实现了其中的 download 和 upload 方法，remove 方法为空方法（因为删除文件操作不是由客户端直接向节点服务器请求的，而是向服务器发请求，委托其删除文件）。

3.3 FileStorage



3.3.1 CommunicateWithServerStrategy

该策略定义了节点服务器与服务器间通信的策略，就是向服务器注册或更新自己。具体实现时是每隔两秒向服务器发一个心跳包，这个心跳包中包含了本节点服务器的 IP、端口、容量等信息。同时在心跳中也可携带其他信息，如通知服务器，本节点新添文件，或什么文件已删除。

```
/**
 * 该接口定义了存储节点和服务端交互的行为
 * Created by Sunny on 2017/7/7 0007.
 */
public interface CommunicateWithServerStrategy {

    /**
     * 在存储节点启动的时候首先要到服务器上注册
     * 或在存储节点运行过程中要定期向服务器发送心跳包更新自己的信息
     */
    public void registerOrUpdate(String ip, int port) throws IOException;
}
```

3.3.2 TransWithClientStrategy

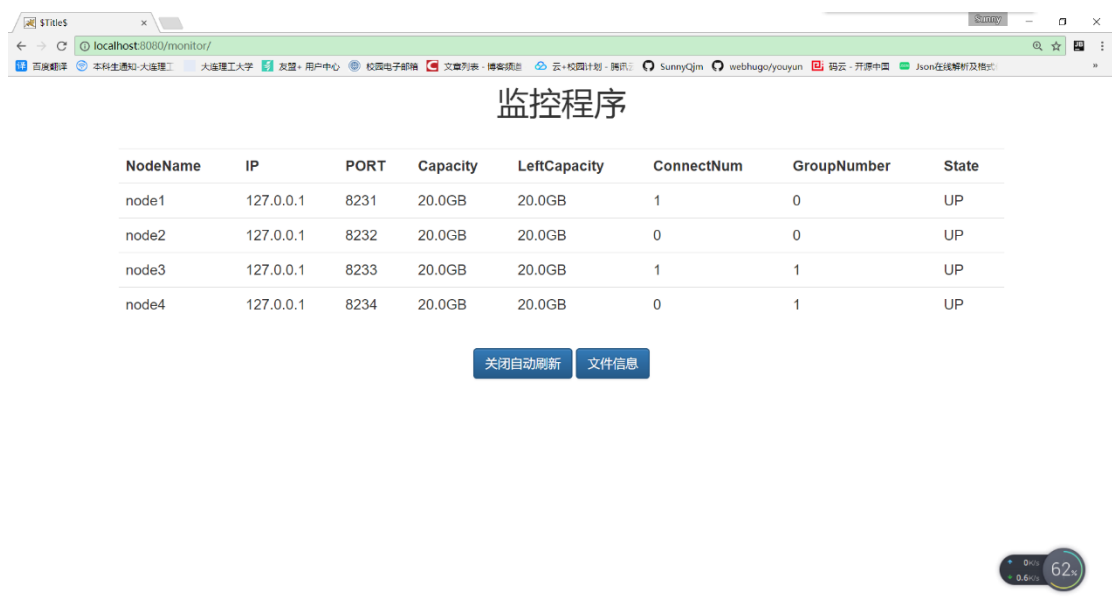
该策略接口定义了如何应对来自客户端和服务器的请求。都是把 Socket 请求传给 service 函数，函数里面再根据不同的 Code 区分不同的请求，做出不同响应。

```
/**
 * 该策略用于应对客户端上传下载文件或服务器发送的删除文件的请求
 * Created by Sunny on 2017/7/6 0006.
 */
public interface TransWithClientStrategy {
    public void service(Socket socket) throws IOException, NoSuchAlgorithmException;
}
```

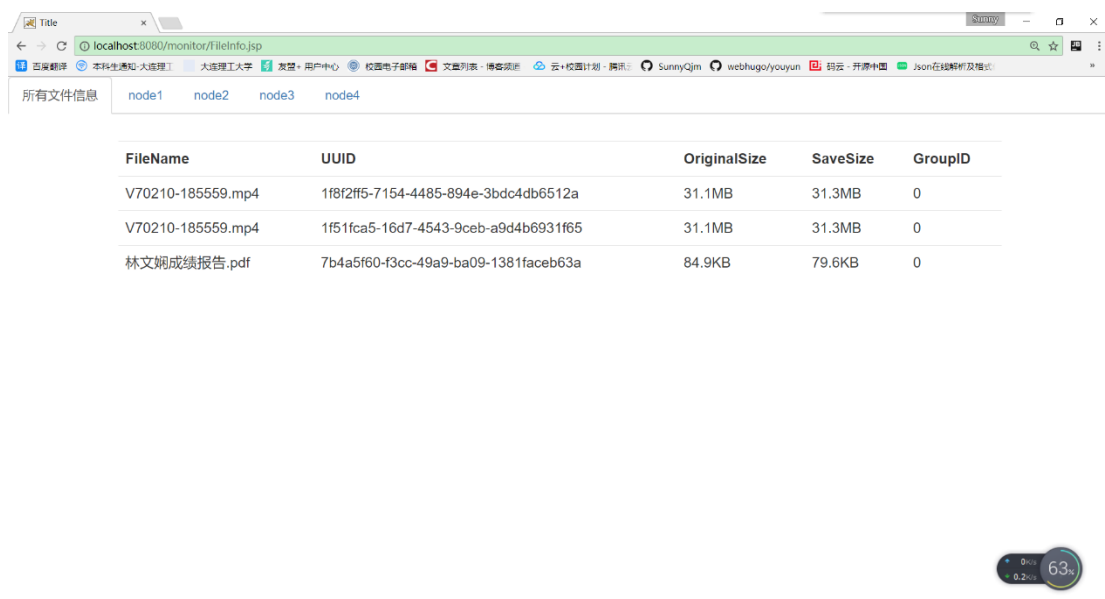
3.4 Monitor

监控程序用 JavaWeb 实现。原理相对简单，就是在用户用浏览器请求监控程序页面的时候，在 JavaWeb 服务器端用一个拦截器拦截该请求，并在拦截器中通过向 FileServer 拉取数据。并将数据通过 request 对象传给 JSP。JSP 再根据收到的数据，动态的向表格中添加表项。

整体的界面效果引入了 bootstrap 框架，使得界面稍微美观一点。



NodeName	IP	PORT	Capacity	LeftCapacity	ConnectNum	GroupNumber	State
node1	127.0.0.1	8231	20.0GB	20.0GB	1	0	UP
node2	127.0.0.1	8232	20.0GB	20.0GB	0	0	UP
node3	127.0.0.1	8233	20.0GB	20.0GB	1	1	UP
node4	127.0.0.1	8234	20.0GB	20.0GB	0	1	UP



FileName	UUID	OriginalSize	SaveSize	GroupID
V70210-185559.mp4	1f8f2ff5-7154-4485-894e-3bdc4db6512a	31.1MB	31.3MB	0
V70210-185559.mp4	1f51fca5-16d7-4543-9ceb-a9d4b6931f65	31.1MB	31.3MB	0
林文娟成绩报告.pdf	7b4a5f60-f3cc-49a9-ba09-1381faceb63a	84.9KB	79.6KB	0

4 不足改进

4.1 问题一及解决方案

问题一：由于在实现过程中每组的节点服务器有两个，在主存储节点宕机的情况下，文件只能传到备份节点服务器上，且此时无法将收到的文件备份到主存储节点。则在下次下载该文件的时候如果这个备份节点宕机，则会下载不到该文件。

解决方案：一种解决方案是每组的服务器多开几个，降低文件下载不到的概率。另一种解决方法是在 FileServer 记录每个文件保存的情况，每当一个节点服务器接入的时候，就向 FileServer 询问同组的其它节点的哪些文件自己没有备份，就向这些节点请求文件备份到本节点服务器。

4.2 问题二及解决方案

问题二：在删除文件时，可能有些存有该文件的节点服务器已经宕机，没有办法删除干净，如果直接删除 FileServer 中该文件信息的表项，则在节点服务器上就会有冗余文件，成为垃圾，无法访问到。

解决方案：在用户请求删除文件时，FileServer 在该表项上标记一下该文件已删除，然后返回给用户删除成功，接着 FileServer 遍历每个存有该文件的节点，试图删除，定期的便利一遍被标记为删除的文件试图删除它，等到所有存有该文件的节点都删除了该文件，FileServer 再将该文件的表项，从 Map 集合中删除。

5 引用类库文献

【1】Gson:2.8.1

【2】姜老师提供的 Tool.java 工具类