

分布式数据库系统课设报告

**2020/2021(2)**



课设名称 微微博系统 学生班级 大数据 1901

学生成员 罗潇 201906061515

学生成员 张思哲 201906061528

学生成员 邱继宏 201906062115

目录

[一、 背景分析 5](#_Toc9628)

[1.1 微博简介 5](#_Toc6397)

[1.2 开发目的 5](#_Toc17580)

[二、 需求分析 6](#_Toc8724)

[2.1 博主注册与登录 6](#_Toc21167)

[2.2 关注和取关用户 6](#_Toc26585)

[2.3 微博内容浏览 6](#_Toc28443)

[2.4 博主个人界面的设计 6](#_Toc4616)

[三、数据模型设计 7](#_Toc6228)

[3.1 HBase表设计 7](#_Toc11431)

[3.1.1 User 表 7](#_Toc23732)

[3.1.2 Content 表 7](#_Toc7533)

[3.1.3 Relation表 7](#_Toc24786)

[3.2 Hbase 表的设计 8](#_Toc23908)

[3.2.1 列簇设计 8](#_Toc19735)

[3.2.2 版本设计 8](#_Toc24760)

[3.2.3 数据压缩 8](#_Toc1245)

[3.3 Rowkey 设计 8](#_Toc9671)

[四、程序模块解析（罗潇写下） 9](#_Toc23471)

[4.1 Entity 类模块 9](#_Toc28146)

[4.2 Tool 模块 9](#_Toc24934)

[4.3 Service 模块 10](#_Toc30148)

[4.4 Processor 模块 11](#_Toc25644)

[4.5 Frame 模块 11](#_Toc28510)

[4.6 Stock\_Chart 模块 12](#_Toc32695)

[4.7 MapReduce 模块 12](#_Toc22868)

[五、性能优化 20](#_Toc2868)

[5.1 过滤器 20](#_Toc23900)

[5.1.1 过滤器介绍 20](#_Toc958)

[5.1.2 过滤器运行原理 20](#_Toc32120)

[5.1.3 过滤器的设计 20](#_Toc16213)

[5.1.4 过滤器的应用 21](#_Toc19341)

[六、 程序展示 22](#_Toc21673)

[6.1 用户登录界面 22](#_Toc1659)

[用户登录界面 22](#_Toc17983)

[用户登录成功界面 22](#_Toc4408)

[6.2用户注册界面 23](#_Toc13236)

[6.3 微博首页 23](#_Toc7032)

[6.4 用户个人界面 24](#_Toc10800)

[用户个人界面 24](#_Toc20716)

[个人发布界面 24](#_Toc13631)

[关注发布界面 25](#_Toc19229)

[个人微博发布界面 25](#_Toc7087)

[发布成功界面 26](#_Toc9063)

[6.5 其他用户界面 26](#_Toc20516)

[七、问题分析 27](#_Toc11694)

[7.1 问题一：微博内容表rowkey设计 27](#_Toc7037)

[7.1.1 问题描述 27](#_Toc27063)

[7.1.2 解决方法 27](#_Toc30535)

[7.2 问题二：主页推送初始化问题 27](#_Toc26701)

[7.2.1 问题描述 27](#_Toc201)

[7.2.2 解决方法 27](#_Toc2719)

[7.3 问题三：关注用户列表获取 28](#_Toc29039)

[7.3.1 问题描述 28](#_Toc30072)

[7.3.2 解决方法 28](#_Toc12195)

[7.4 问题四：未关注用户内容随机推送 29](#_Toc7996)

[7.4.1 问题描述 29](#_Toc25428)

[7.4.2 解决方法 29](#_Toc22227)

[八、课本知识及其应用 30](#_Toc5043)

[8.1 HDFS 30](#_Toc21224)

[8.1.1 HDFS 定义 30](#_Toc16377)

[8.1.2 HDFS 组成 30](#_Toc5160)

[8.1.3 HDFS 应用 31](#_Toc13099)

[8.2 HBase 31](#_Toc31327)

[8.2.1 HBase 介绍 31](#_Toc4923)

[8.2.2 HBase 数据模型 31](#_Toc11180)

[8.2.3 HBase 体系架构 32](#_Toc28154)

[8.2.4 HBase 的应用 33](#_Toc9761)

[九、总结与提高 34](#_Toc23210)

[9.1 总结 34](#_Toc22404)

[9.2 不足 34](#_Toc15683)

[十、团队分工 35](#_Toc13335)

[10.1 罗潇负责的部分 35](#_Toc24729)

[10.2 张思哲负责的部分 35](#_Toc19591)

[10.3 邱继宏负责的部分 35](#_Toc1805)

# 背景分析

## 微博简介

微博即微博客（microblog）的简称，由博客（blog）发展而来，最早也是目前最大的微博是美国的 Twitter， 中国最早开始微博业务的是门户网站新浪网。新浪网于 2009 年 8 月 推出新浪微博内测版， 从此开始了信息传播和分享的“140字的革命”。

微博，即微型博客，是随着 Web 2.0 而兴起的一类开放的互联网社交服务，它允许用户以简短文字随时随地更新自己的状态， 每条信息的长度都在140 字以内，支持图片、音频、视频等多媒体的出版， 每个用户既是微内容的创造者也是微内容的传播者和分享者。最早也是最著名的微博是美国的 twitter，根据相关公开数据，截至 2010 年 1 月 份，该产品在全球已经拥有 7500 万注册用户。2009 年 8 月 份中国最大的门户网站新浪网推出“新浪微博”内测版，成为门户网站中第一家提供微博服务的网站，微博正式进入中文上网主流人群视野。

主流微博介绍：新浪微博是一个由新浪网推出，提供微型博客服务的类Twitter 网站。用户可以通过网页、WAP 页面、手机短信/彩信发布消息或上传图片。新浪可以把微博理解为“微型博客”或者“一句话博客”。

## 开发目的

近段时间，微博这个词在网络上非常流行，大到国家领导人，小到普通市民、学生也都在网上开了微博。许多大型网站也在原来博客的基础上，增加了微博，

微博以其短小精悍，更加贴近生活而受到人们的推崇，在浏览之余不禁想问一下它们之间的区别：

1、 字数限制，微博必须在 140 字以内，这是为了手机发布阅读方便，博客没有限制，因为它主要是让人在电脑上发表和阅读的。

2、 被动阅读，看博客必须去对方的首页看，而微博在自己的首页上就能看到别人的微博。

3、 发布简便，可以通过手机网络更新，当然也可以通过电脑更新，而博客一般来说，用手机更新非常麻烦。

4、 自传播速度快，博客要是靠网站推荐带来流量，而微博通过粉丝转发来增加阅读数。

微博和博客还是有共同点的，不论是博客也好，微博也好，都是一个人所见所闻， 或编写，或转载，上至天文，下至地理，世态炎凉，悲欢离合，一篇篇一字字组成了我们的生活，同时也让我们的生活变得丰富多彩。

由此微微博系统的设计与开发有利于让我们熟悉一些工具的使用和相关功能实现的具体流程，在开发系统的过程中，也丰富我们对数据库和可视化界面的了解，以便在未来开发类似程序时能得心应手，不再犯错。

# 需求分析

## 博主注册与登录

每个博主在使用微博之前都需要注册，然后通过注册的用户名和密码来登录，进入微博首页之后就可以浏览各个用户发布的微博了。

## 关注和取关用户

进入微博首页之后，点击用户名称，就可以进入该用户界面，然后就可以关注该用户了，当然也可以取关，进入个人界面后可以看到关注的人的名称。

## 微博内容浏览

微博内容不仅在主页可以浏览，在用户界面和个人界面也可以浏览其他用户和本人的数条微博。

## 博主个人界面的设计

个人界面可以在微博主页进入，里面可以看到关注列表，个人发布的微博和关注人发布的微博，个人的微博可以进行发布和删除的操作。

# 三、数据模型设计

## HBase表设计

### User 表



### Content 表



### Relation表



## Hbase 表的设计

### 列簇设计

没有特殊需求的情况下，HBase 列簇的数量应该越少越好，因为两个及以上的列簇 HBase 性能并不是很好，一个列蔟所存储的数据达到 flush 的阈值时，表中所有列蔟将同时进行 flush 操作，这将带来不必要的 I/O 开销，列蔟越多，对性能影响越大。因此本次项目中我们除了relation表其他表只设计一个列簇：

### 版本设计

因为本暂时没有分析相关的要求。因此本次项目中只保留一个版本即可，这样可以节省大量空间。

## Rowkey 设计

本项目用户表、用户关系表的 Rowkey 设 计 采 用 用户名。微博内容表的 Rowkey 设 计 采 用 用户名拼接时间，详细设计见表。

表 6 Hbase 表 Rowkey 设计

|  |  |
| --- | --- |
| 表名 | Rowkey 设计 |
| Weibo:user | 用户名 |
| Weibo:relation | 用户名 |
| Weibo:content | 用户名\_时间 |

# 四、程序模块解析

## constants 类模块

constants类模块，声明定义了一些常量，包括HBase的配置信息，三张表的表名、列族和版本数信息。

## HBaseUtil模块

HBaseUtil模块主要负责定义一些操纵Hbase表的基本函数。详细见表。

表 8 HBaseUtil模块功能

|  |  |
| --- | --- |
| 函数名 | 功能 |
| createNameSpace | 创建命名空间 |
| isTableExist | 判断表是否存在 |
| createTable | 创建表 |
| putData | 向表插入数据 |
| deleteData | 数据删除 |
| getData | 获取数据 |
| scanTable | 扫描表 |
| filterScan | RowFilter过滤查询 |

## HBaseDao模块

HBaseDao模块负责实现对系统各种功能的实现如发布微博、删除微博，用户注册等，详细见表。

表 9 HBaseDao模块功能

|  |  |
| --- | --- |
| 函数 | 功能 |
| publishWeiBo | 发布微博 |
| deleteWeiBo | 删除微博 |
| addAttends | 关注用户 |
| deleteAttends | 取关用户 |
| userRegister | 用户注册 |
| ifRepeat | 判断用户名是否重复 |
| pswIfRight | 判断密码是否正确 |
| getWeiBo | 获取某个人所有微博详情 |
| attendList | 获取用户的关注列表 |
| attendPush | 推送(所有关注用户) |
| notAttendRandomPush | 随机推送(未关注用户) |
| userWeiBoNum | 获得当前用户发布微博数目 |
| userAttendNum | 获得当前用户关注用户数目 |

## TestWeiBo模块

TestWeiBo模块负责进行各个功能的测试。

## Frame 模块

Frame 模型负责实现用户客户端的界面设计和用户界面交互信息的处理。

表 11 Frame 模块功能

|  |  |
| --- | --- |
| 界面 | 功能 |
| LoginFrame(登录界面) | 未注册用户可以点击注册按钮进入注册，注册用户可以输入自己的用户名和密码登录进入个人信  息界面 |
| RegisterFrame(注册界面) | 游客填入自己的用户名，股市，密码，身份证，手  机号便可成为注册用户 |
| InfoFrame(个人信息界面) | 里面显示当前用户的用户名，余额，所拥有的股票以及资产估值，用户可以进行修改密码，充值， 修改个人信息及卖出股票操作。 |
| StockFrame(股票界面) | 显示当前市场上的股票，有开盘价，最高价，最低价，当前价格等信息，用户可以通过股票代码进  行股票的查询。 |
| StockSearchFrame(股票分  析界面) | 可以按股票代码查询改只股票的历史信息，也可  以按日期查询该日所有股票的信息。 |
| historyFrame(历史交易界  面) | 显示当前用户所有的交易记录，以及累计收益。  也可按股票代码查询有关该只股票的所有交易 |

## Stock\_Chart 模块

Stock\_Chart 模块负责实现股票数据的可视化。利用股票历史数据，预测股票未来走势，模拟出股票的日线图和 k 线图。其主要类及其功能见表。

表 12 Stock\_Chart 模块功能

|  |  |
| --- | --- |
| 类 | 功能 |
| TimeFigure | 利用股票历史数据，预测出当天股票的价格变化数据。  在股票交易的时间段内，股票价格以时间分为单位更新。用于绘制日线图 |
| KlineFigure | 利用股票历史数据，统计过去时间股票价格的变得，用  于绘制 k 线图 |
| DrawKlineChart | 利用股票当天的价格变化数据，绘制日线图 |
| DrawLineChart | 利用股票历史价格变化数据，绘制 k 线图 |

## MapReduce 模块

### 资产估值功能介绍

求出用户所拥有的股票具备的价值。计算方法为当前拥有的股票数量乘以股票当前的市场价，在进行求和，所得的结果就是股票的价值。

### 资产估值功能实现流程

将用户所拥有的股票及相关的的信息上传到 HDFS 上，map 阶段读取股票的信息，取其中的当前价格和数量，两者相乘得到的值就是当只股票的价值，作为value，用户名作为 key，形成<用户名，价值>的键值对。Reduce 阶段将所有的价值相加，就能得到当前用户的资产估值。

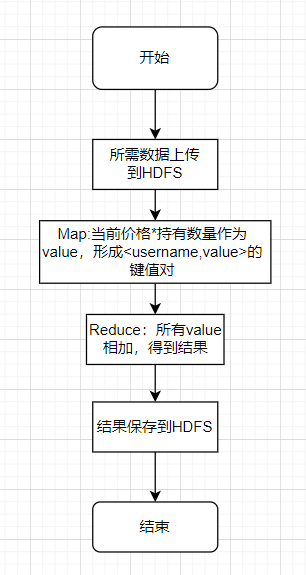


图 1 资产估值功能流程图

### 资产预估功能主要函数解析

1. Map 阶段取得当前价格和持有的数量，转为 Double 类型相乘后作为value，与用户名的 key 组成键值对，用于下游计算。

1. **public void** map(Object key, Text value, Context context) **throws** IOException, I

nterruptedException {

2. String lineValue = value.toString(); //将输入转化为字符串类型

3. String[] words = lineValue.split(" "); //进行分词

4. String user\_name=words[0]; //取得用户名

5. Double num = Double.parseDouble(words[1]);

6. //取得股票数量并转为 Double 类型

7. Double price = Double.parseDouble(words[2]);

8. //取得股票当前价格并转为 Double 类型

9. Double money =price\*num; //相乘获得当只股票的价值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10. |  | word.set(user\_name); |  |
| 11. |  | result.set(money); |  |
| 12.  游 |  | context.write(word,result); | //将用户名与价值的键值对写下来，用于下 |
|  | 13. | } |  |  |

1. Reduce 阶段将所有的 value 累加，就得到最后的结果，也就是资产的估值。上传到 HDFS。

1. **public void** reduce(Text key, Iterable<DoubleWritable> values, Context context)

**throws** IOException, InterruptedException {

2. Double sum = 0.0;

3. **for** (DoubleWritable value : values) {

4. sum += value.get();

|  |  |  |  |
| --- | --- | --- | --- |
|  | 5. |  | } //遍历 values,进行累加，值储存在 sum 中 |
| 6. |  | DecimalFormat df = **new** DecimalFormat( "0.0000"); |
| 7.  取 4 位 |  | **double** sum\_final=Double.parseDouble(df.format(sum)); //sum 小数点后 |
| 8. |  | result.set(sum\_final); |
| 9. |  | context.write(result,**new** Text("")); //将结果记录下来 |
|  | 10. | } |  |

### 累计收益功能介绍

根据用户的交易记录，算出用户在股市中的累计收益。计算方法为（股票的卖出价格-股票的买入价格）\*数量。

### 累计收益功能实现流程

将用户所有历史交易信息上传到 HDFS 上，map 阶段读取股票交易的信息，取其中的买入价格，卖出价格和数量，（股票的卖出价格-股票的买入价格）\*数量。就是当笔交易的收益，作为 value，用户名作为 key，形成<用户名，价值>的键值对。Reduce 阶段将所有的价值相加，就能得到当前用户交易的累计收益。

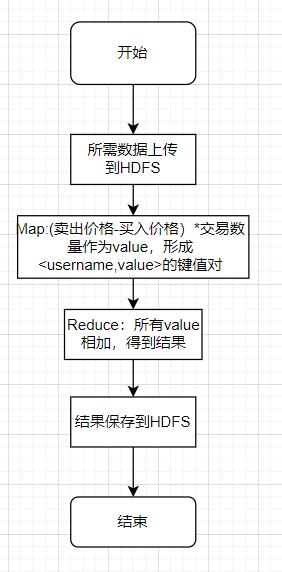


图 2 累计收益功能实现流程图

### 累计收益功能主要函数解析

1. Map 阶段取得买入价格，卖出价格和交易的数量，根据公式（股票的卖出价格-股票的买入价格）\*数量拿到的收益作为 value，与用户名的 key 组成键值对，用于下游计算。

1. **public void** map(Object key, Text value, Context context) **throws** IOException, I

nterruptedException {

2. String lineValue = value.toString(); //将输入转化为字符串类型

3. String[] words = lineValue.split(" "); //进行分词

4. String user\_name=words[0]; //取用户名

5. Double buy\_price = Double.parseDouble(words[1]); //取买入价格

6. Double sell\_price = Double.parseDouble(words[2]); //取卖出价格

7. Double num = Double.parseDouble(words[3]); //取交易数量

8. Double money =(sell\_price-buy\_price)\*num; //当笔交易的收益

9. word.set(user\_name);

10. result.set(money);

11. context.write(word,result); //将用户名与收益的键值对写下来，用于下

游

12.}

1. Reduce 阶段将所有的 value 累加，就得到最后的结果，也就是资产的

估值。上传到 HDFS。

1. **public void** reduce(Text key, Iterable<DoubleWritable> values, Context context)

**throws** IOException, InterruptedException {

2. Double sum = 0.0;

3. **for** (DoubleWritable value : values) {

4. sum += value.get();

5. } //遍历 values,进行累加，值储存在 sum 中

6. DecimalFormat df = **new** DecimalFormat( "0.0000");

7. **double** sum\_final=Double.parseDouble(df.format(sum)); //sum 小数点后取 4 位

8. result.set(sum\_final);

9. context.write(result,**new** Text("")); //将结果记录下来

10. }

### 持有股票统计功能介绍

根据用户持有的股票，统计其股票的平均买入价格和总数。总数的计算公式

为买入数量，平均价格的计算方式为（买入价格\*买入数量）

/sum 。

### 持有股票统计功能实现流程

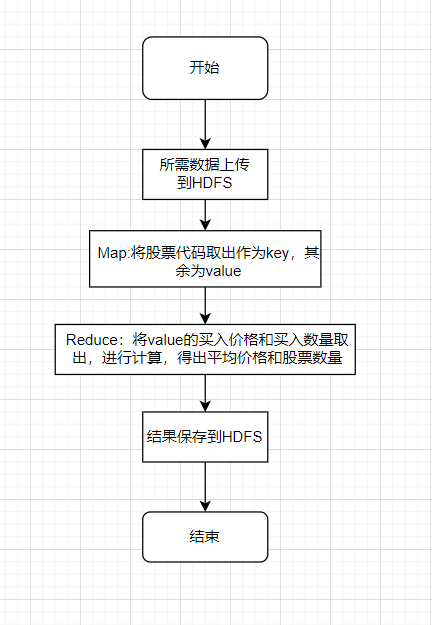
将用户持有股票信息上传到 HDFS 上，map 阶段读取股票交易的信息，取其中的股票代码当作 key，其余作为 value，形成<股票代码，信息>的键值对。Reduce 阶段将信息里的买入价格和数量提取出来，按公式进行运算，，将结果与 key 写入 HDFS，就能得到当前用户股票的统计信息。

图 3 持有股票统计功能实现流程

### 持有股票统计功能主要函数解析

1. Map 阶段取得股票代码，买入价格和买入数量，将股票代码设置为key，其余设置为 value，组成键值对，用于下游计算。

1. **public void** map(Object key, Text value, Context context) **throws** IOException, I

nterruptedException {

2. String lineValue = value.toString(); //将输入转化为字符串类型

3. String[] words = lineValue.split(" "); //进行分词

4. String code=words[0]; //取代码

5. String buy\_price=words[1]; //取买入价格

6. String num=words[2]; //取买入数量

7. stock\_code.set(code); //key 设置为股票代码

8. StringBuilder stringBuilder=**new** StringBuilder();

9. stringBuilder.append(buy\_price).append(" ").append(num); // 构 建

value 的值

10. result.set(stringBuilder.toString()); //value 设置

11. context.write(stock\_code,result); //写入结果

12.

13. }

1. Reduce 阶段将买入价格和数量取出，分别累加到总价格和总数量上， 相除得到平均价格，将结果组成字符串变成 value。

1. **public void** reduce(Text key, Iterable<Text> values, Context context) **throws** IO

Exception, InterruptedException {

2. Double sum = 0.0; //储存总的价钱

3. **int** stock\_sum=0; //储存股票数量

4. **for** (Text value : values) {

5. String lineValue = value.toString(); //变成字符串

6. String[] words = lineValue.split(" "); //分词

7. String buy\_price=words[0]; //取得买入价格

8. String num=words[1]; //取得数量

9. Double price=Double.parseDouble(buy\_price);

10. sum+=price\* Double.parseDouble(num); //取得总价格

11. stock\_sum+=Integer.parseInt(num); //累加股票数量

12. }

13. **double** avg=sum/stock\_sum; //获得平均价格

14. DecimalFormat df = **new** DecimalFormat( "0.0000");

15. **double** avg\_final=Double.parseDouble(df.format(avg)); //保留 4 位小

数

16. StringBuilder stringBuilder=**new** StringBuilder();

17. stringBuilder.append(avg\_final).append(" ").append(stock\_sum);// 构

建结果字符串

18. result.set(stringBuilder.toString());

19. //result.set(sum\_final);

20. context.write(key,result); //写入结果

21. }

# 五、性能优化

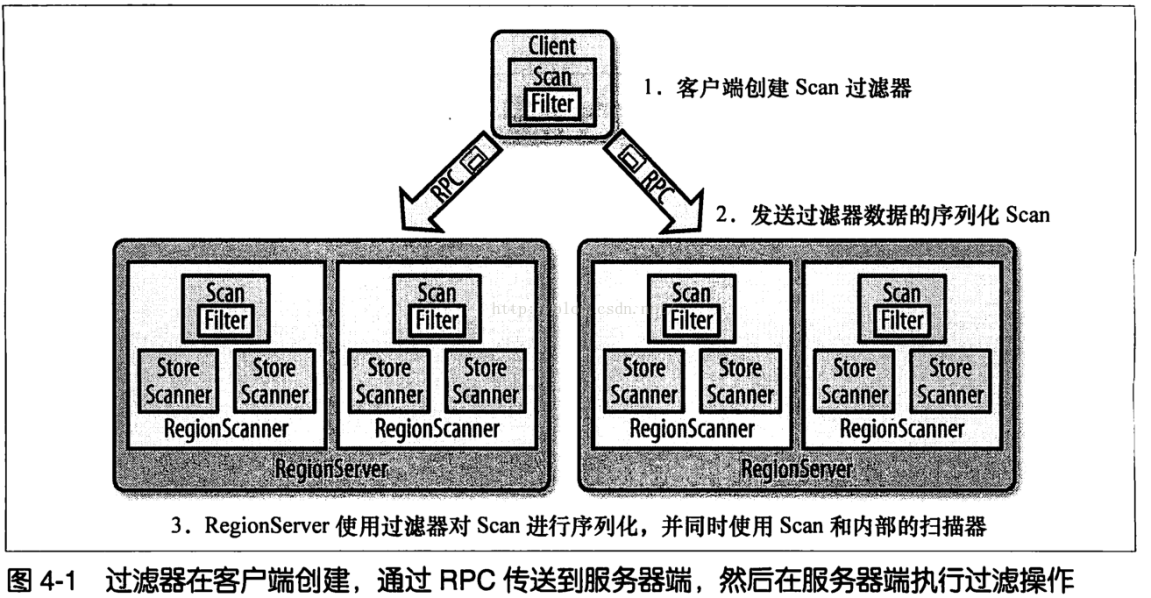
## 过滤器

### 过滤器介绍

HBase过滤器是一套为完成一些较高级的需求所提供的API借口。从过滤器的名称我们就可以看出：过滤器就是对数据库获取的数据进行过滤，将符合条件的数据返回客户端，从而减少从region服务器想客户端发送的数据，从而减少数据传输，提高效率。

### 过滤器运行原理

过滤器只能配置Get和Scan对象进行使用，前面章节也提高了Get和Scan都提供了setFilter()借口，可以通过该函数设置相应的过滤器。在发送Get或者Scan请求以后，其对象会被序列化其被传送到相应的region服务器中，这时过滤器对象也会被序列化后传入到相应的region服务器中。从而在region服务器端起到过滤数据的作用。



### 过滤器的设计

在本项目中有 Index\_Processor 和 Deal\_Processor 两个协处理器，分别负责创建二级索引和用户股票买卖操作。

在表的设计过程中，将各表进行了预分区，使用哈希策略来设计 Rowkey 避免热点问题。但是这也导致了我们无法直接使用 Rowkey 来进行查询表，因此需要建立二级索引，而维护二级索引最方便的方法就是使用协处理器。本项目在STOCK\_SYSTEM:User 表上加载了 Index\_Processor 协处理器。其原理是当STOCK\_SYSTEM:User 表进行 put 操作时，协处理器触发将其 put 进行拦截。读取put 中的数据， 将其值作为 Rowkey ，其 Rowkey 作为 value ， 将数据写入STOCK\_SYSTEM:Index 表。

在股票交易中，用户每次买卖都会涉及到多个表的修改操作。当用户买入股

票时，需要在 STOCK\_SYSTEM:UserDeal 表中增加用户交易信息记录， 在STOCK\_SYSTEM:UserStock 表中修改用户持有股票的信息，在 STOCK\_SYSTEM:User 表中修改账户余额。当用户卖出的时候同样要进行类似的多表修改操作，因此可以使用协处理器来减少人为的操作。本项目在 STOCK\_SYSTEM:UserDeal 表中加载了 Deal\_Processor 协处理器。其原理是当用户需要进行股票买卖操作时，先在STOCK\_SYSTEM:UserDeal 表中插入交易信息，此时协处理器触发，拦截 put 并读取信息，判断用户是买入操作还是卖出操作，再去修改 STOCK\_SYSTEM:UserStock 表中用户持有股票信息，计算买卖后的新余额去修改 STOCK\_SYSTEM:User 表的用户余额信息。

### 过滤器的应用

1. **程序展示**

## 6.1 用户登录界面

微微博系统运行后的初始界面是用户登录界面，用户可以在该界面登录账户或者注册新账户，登录之前需要✔用户协议。



用户登录界面



用户登录成功界面

## 6.2用户注册界面

在用户登录界面点击注册按钮后进入用户注册界面。用户在该界面填写用户名和密码后点击注册按钮便可注册成功。



用户注册界面

## 6.3 微博首页

在用户登录成功后，进入微博首页，可以看出随机用户的微博发布情况，点击刷新可以继续查看其他用户，也可以返回登录页面使用其他账号登录。



微博首页

## 6.4 用户个人界面

在微博首页点击个人中心就可以进入个人界面。在该界面用户可以查看关注列表，个人发布的微博和关注人发布的微博，还有可以通过写微博按钮发布新的微博。



用户个人界面



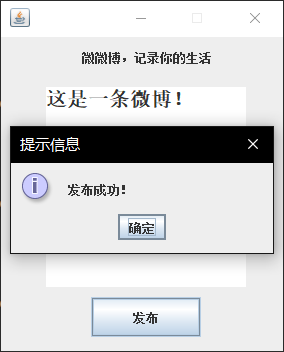
个人发布界面



关注发布界面



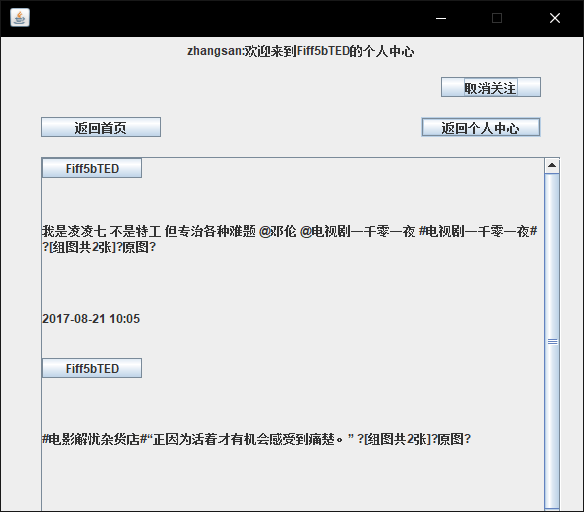
个人微博发布界面



发布成功界面

## 6.5 其他用户界面

在微博首页或个人界面点击其他用户的名称就可以进入其他用户界面。在该界面用户可以关注也可以取关，该界面有用户的所有发布的微博，且可以经此返回个人中心。



其他用户界面

# 七、问题分析

## 问题一：微博内容表rowkey设计

### 问题描述

在读取股票数据上传至 hbase 表时发生文件读取异常，导致程序终止，数据上传失败。其代码报错见图。

### 解决方法

在最初设计微博内容表rowkey时只将用户名作为rowkey，时间单独一列，但在编写代码时发现这样设计表会导致代码编写较为麻烦，在需要时间信息是需要多循环一次取出时间列的值，所以将时间列和用户名拼接，需要用户名和时间时，只需要将rowkey取出切分成用户名和时间即可

## 问题二：主页推送初始化问题

### 问题描述

将协处理器挂载在对应的表上时报错，挂载失败。

### 解决方法

在前后端对接的时候发现，新用户在登录后无法加载主页推送的内容，后来经过排查调试发现是在推送未关注用户内容函数内没有考虑到当前用户没有关注其他人的情况，在加入了判断当前用户关注列表是否为空的语句后，首页正常显示推送内容。

## 问题三：关注用户列表获取

### 问题描述

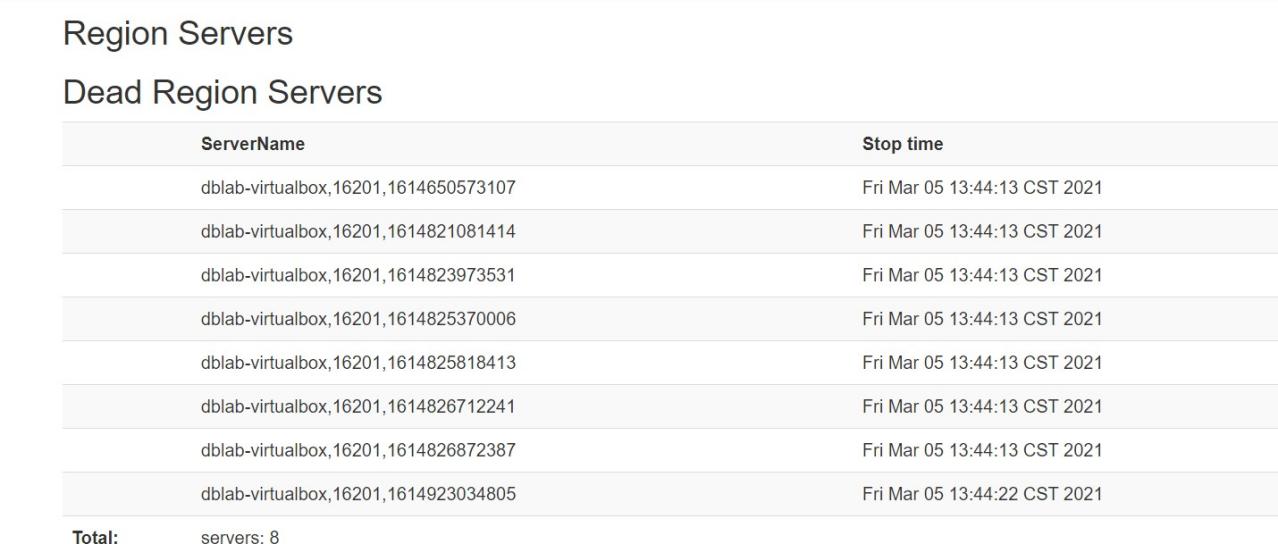
在持续批量写入 HBase 的情况下，出现了 Dead Region Servers 的情况。集群会把 dead 掉节点上的 region 自动分发到另外其他节点上，集群还能继续运行，但是节点少了。

图 36 Dead Region Servers 提示

### 解决方法

为了解决获得当前用户的关注列表这个问题，我们使用了过滤器，从用户关系表中过滤出当前用户的attend列的内容，将得到的内容加入ArrayList中，即可得到当前用户的关注列表。

## 问题四：未关注用户内容随机推送

### 问题描述

Hbase 重启后发现 shell 命令无法正常使用，在 HBase Regionserver Web UI 中发现部分表出现永久 RIT 问题。经查阅后得知永久 RIT，即由于区域由主服务器和区域服务器管理，以平衡跨服务器的负载，它们会经历短暂的过渡阶段。这适用于打开、关闭和分割区域。在执行操作之前，将区域添加到 WEB UI 上的" 过渡区域"列表中，操作完成后将其删除。只要有 region 处于 RIT 状态，balacer 就无法运行。

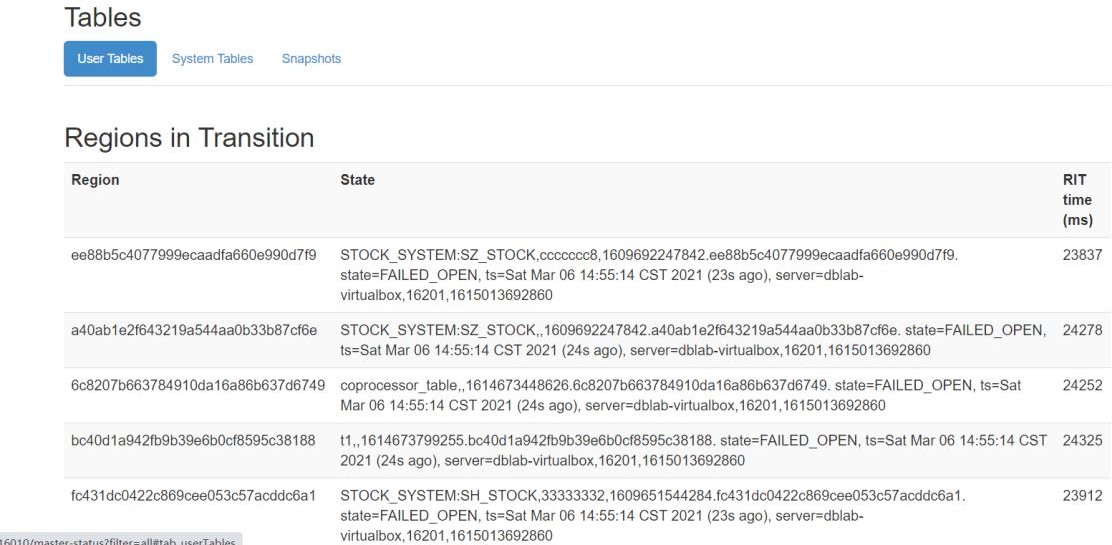


图 38 RIT 提示

### 解决方法

为了获得当前用户的未关注用户内容并且随机推送，我们使用了过滤器，并且写出一个创建不含重复数字的随机数数组生成算法，从微博内容表中过滤出不在当前用户关注列表中用户的内容，将得到的内容加入ArrayList中，再通过随机数数组生成算法生成一个不重复的随机数数组，从中取得固定的微博内容数，即可实现未关注用户内容的随机推送。

# 八、课本知识及其应用

## HDFS

### HDFS 定义

整个 Hadoop 的体系结构主要是通过 HDFS（Hadoop 分布式文件系统）来实现对分布式存储的底层支持，并通过 MR 来实现对分布式并行任务处理的程序支持。HDFS 是 Hadoop 体系中数据存储管理的基础。它是一个高度容错的系统，能检测和应对硬件故障，用于在低成本的通用硬件上运行。HDFS 简化了文件的一致性模型，通过流式数据访问，提供高吞吐量应用程序数据访问功能，适合带有大型数据集的应用程序。

### HDFS 组成

HDFS 采用主从（Master/Slave）结构模型，一个 HDFS 集群是由一个 NameNode 和若干个 DataNode 组成的。NameNode 作为主服务器，管理文件系统命名空间和客户端对文件的访问操作。DataNode 管理存储的数据。HDFS 支持文件形式的数据。

从内部来看，文件被分成若干个数据块， 这若干个数据块存放在一组DataNode 上。NameNode 执行文件系统的命名空间，如打开、关闭、重命名文件或目录等，也负责数据块到具体 DataNode 的映射。DataNode 负责处理文件系统客户端的文件读写，并在 NameNode 的统一调度下进行数据库的创建、删除和复制工作。NameNode 是所有HDFS 元数据的管理者，用户数据永远不会经过NameNode。

图展现了 HDFS 的组成，其中 NameNode 是管理者，DataNode 是文件存储者、Client 是需要获取分布式文件系统的应用程序。

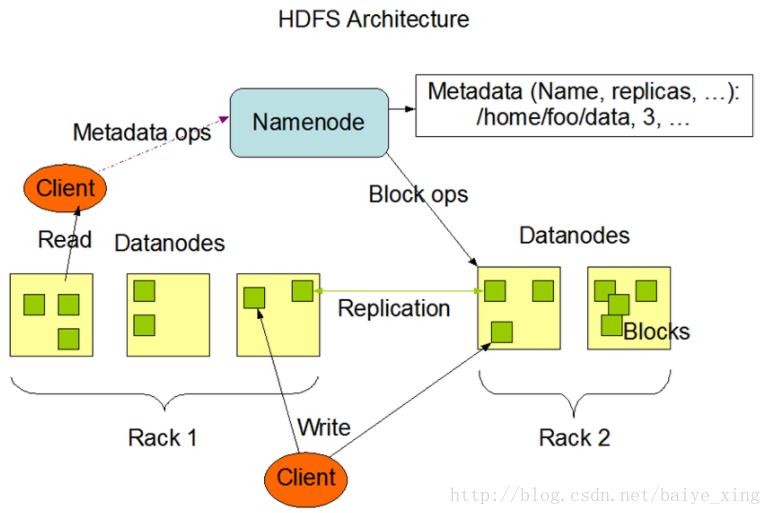


图 40 HDFS 组成

### HDFS 应用

在本项目中，在设计及使用协处理器的时候，需要先将写好的协处理器代码进行打包，上传至 HDFS，再将其挂载在对应的表中。在 MapReduce 模块中，所有MapReduce 相关的功能实现均要用到 HDFS。

## HBase

### HBase 介绍

HBase 是一个分布式的、面向列的开源数据库，该技术来源于 Fay Chang 所撰写的 Google 论文“Bigtable：一个结构化数据的分布式存储系统”。

1. HBase–Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩、实时读写的分布式数据库
2. 利用 Hadoop HDFS 作为其文件存储系统,利用 Hadoop MapReduce 来处理 HBase 中的海量数据,利用 Zookeeper 作为其分布式协同服务
3. 主要用来存储非结构化和半结构化的松散数据（列存 NoSQL 数据库）

### HBase 数据模型

RowKey:

1. 决定一行数据的唯一标识
2. RowKey 是按照字典顺序排序的；
3. Row key 最多只能存储 64k 的字节数据。

Column Family 列族:

1. HBase 表中的每个列都归属于某个列族，列族必须作为表模式(schema) 定义的一部分预先给出。
2. 列名以列族作为前缀，每个“列族”都可以有多个列成员(column，每个列族中可以存放几千~上千万个列)
3. 权限控制、存储以及调优都是在列族层面进行的
4. HBase 把同一列族里面的数据存储在同一目录下，由几个文件保存。
5. 目前为止 HBase 的列族能能够很好处理最多不超过 3 个列族。

Timestamp 时间戳：

1. 在 HBase 每个 cell 存储单元对同一份数据有多个版本，根据唯一的时间戳来区分每个版本之间的差异，不同版本的数据按照时间倒序排序，最新的数据版本排在最前面。
2. 时间戳的类型是 64 位整型。
3. 时间戳可以由 HBase(在数据写入时自动)赋值，此时时间戳是精确到毫秒的当前系统时间。
4. 时间戳也可以由客户显式赋值，如果应用程序要避免数据版本冲突， 就必须自己生成具有唯一性的时间戳。

Cell 单元格：

1. 由行和列的坐标交叉决定
2. 单元格是有版本的（由时间戳来作为版本）
3. 单元格的内容是未解析的字节数组（Byte[]），cell 中的数据是没有类型的，全部是字节码形式存贮。
4. 由{row key，column(=<family> +<qualifier>)，version}唯一确定的单元。

图 41 Hbase 表的结构

### HBase 体系架构

Client：

（1）包含访问 HBase 的接口并维护 cache 来加快对 HBase 的访问

Zookeeper：

1. 保证任何时候，集群中只有一个 master
2. 存贮所有 Region 的寻址入口。
3. 实时监控 Region server 的上线和下线信息。并实时通知 Master
4. 存储 HBase 的 schema 和 table 元数据

Master：

1. 为 Region server 分配 region
2. 负责 Region server 的负载均衡
3. 发现失效的 Region server 并重新分配其上的 region
4. 管理用户对 table 的增删改操作

RegionServer：

1. Region server 维护 region，处理对这些 region 的 IO 请求
2. Region server 负责切分在运行过程中变得过大的 region

HLog(WAL log)：

1. HLog 文件就是一个普通的 Hadoop Sequence File，Sequence File 的Key 是 HLogKey 对象，HLogKey 中记录了写入数据的归属信息，除了 table 和region 名字外，同时还包括 sequence number 和 timestamp，timestamp 是” 写入时间”，sequence number 的起始值为 0，或者是最近一次存入文件系统中sequence number。
2. HLog SequeceFile 的 Value 是 HBase 的 KeyValue 对象，即对应 HFile 中的 KeyValue

Region：

1. HBase 自动把表水平划分成多个区域(region)，每个 region 会保存一个表 里面某段连续的数据；每个表一开始只有一个 region，随着数据不断插 入表，region 不断增大，当增大到一个阀值的时候，region 就会等分会 两个新的region（裂变）
2. 当 table 中的行不断增多，就会有越来越多的 region。这样一张完整的表 被保存在多个Regionserver 上。

Memstore 与 storefile：

1. 一个 region 由多个 store 组成，一个 store 对应一个 CF（列族）
2. store 包括位于内存中的 memstore 和位于磁盘的 storefile 写操作先写入 memstore，当 memstore 中的数据达到某个阈值，hregionserver 会启动flashcache 进程写入 storefile，每次写入形成单独的一个 storefile.
3. 当 storefile 文件的数量增长到一定阈值后，系统会进行合并（minor、major compaction），在合并过程中会进行版本合并和删除工作形成更大的storefile。
4. 当一个 region 所有 storefile 的大小和超过一定阈值后，会把当前的region 分割为两个，并由 hmaster 分配到相应的 regionserver 服务器，实现负载均衡。
5. 客户端检索数据，先在 memstore 找，找不到再找 storefile
6. HRegion 是 HBase 中分布式存储和负载均衡的最小单元。最小单元就表示不同的 HRegion 可以分布在不同的 HRegion server 上。
7. HRegion 由一个或者多个 Store 组成，每个 store 保存一个 columns family。
8. 每个 Strore 又由一个 memStore 和 0 至多个 StoreFile 组成。

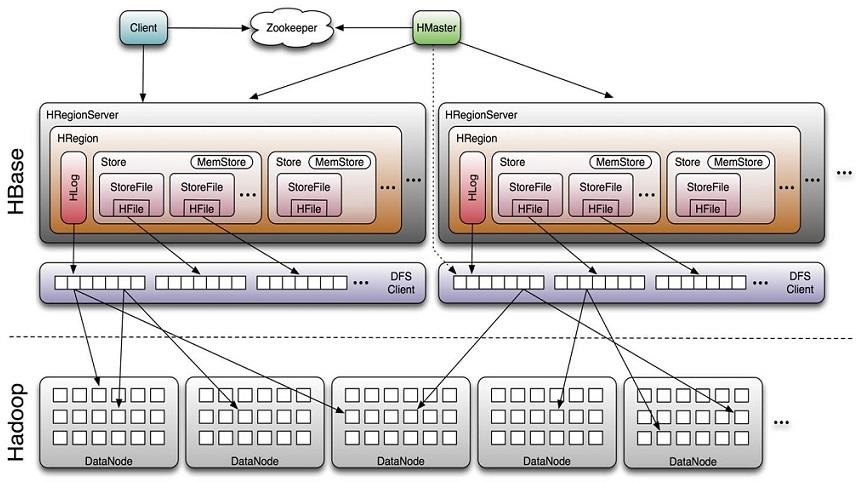


图 42 Hbase 体系架构

### HBase 的应用

本项目数据库的开发主要使用 HBase 数据库，因此用到了大量的 HBase 相关的知识。主要知识点应用如下

1. 在 HBase 表的建立时用到 HBase shell。
2. 代码实现表的增删查改时用到 HBase API
3. 在表的设计时遵循 HBase 表的设计原则
4. Hbase 表预 分区进行水平分片

# 九、总结与提高

## 总结

1. HBase–Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server或虚拟机上搭建起大规模结构化存储集群。
2. 在本次实验当中，我们进一步认识到了HBase和分布式框架的相关知识，特别是通过过滤器的使用，提高数据处理的效率，从而制作了微微博系统，在Java GUI交互界面上实现：登录注册，关注、取关用户，微博内容浏览，用户界面设计等功能。
3. 在HBase搭建当中学会了不是通内置的而是通过zookeeper实现对客户端的连接；在界面设计当中我们对于eclipse有更友好的图形界面展示，数据集的爬取也是方便HBase表的设计，有利于使代码编写变得更加简单而不冗杂。
4. 在团队当中我们互帮互助，积极进取，在编程实践中分工明确，大大提高了解决问题的效率。

## 不足

1. 功能不够完整，目前只是完成了基础的功能，有待增加更多有特色的功能
2. 用户交互界面有待美化，提升用户的使用时的舒适感
3. HBase方面有待优化，加快存取速度，提高系统流畅度

# 十、团队分工

## 10.1 罗潇负责的部分

1. 搭建虚拟机环境
2. 搭建Hadoop集群
3. 配置HBase
4. 编写具体功能实现代码

## 10.2 张思哲负责的部分

1. 参与 HBase 表的设计
2. 使用java GUI编写可视化页面
3. 对接后端数据

## 10.3 邱继宏负责的部分

1. 参与 HBase 表的设计
2. 前后端对接
3. 选题PPT制作
4. 实验报告，PPT编写