

毕业设计说明书

基于 Hadoop 对某位置服务社交网络 用户访问地理位置数据

班 级：_____ 学号：_____

姓 名：_____

学 院：_____ 软件学院

专 业：_____ 软件工程（软件开发与测试方向）

指导教师：_____

2016 年 6 月

基于 Hadoop 对某位置服务社交网络用户访问地理位置数据

摘要

本文主要介绍了基于 Hadoop 的地理数据分析系统的工作原理以及各部分结构。整个系统按照功能划分为四个模块。第一个是用于存取数据的 Hadoop 集群 HDFS 系统模块；第二个是对数据进行清洗分类的 MapReduce 模块；第三个是用于查询的 Hbase 和 Hive 模块；最后一个是为了显示的前台界面模块。旨在完成对庞大数据的清洗分类查询，并以直观可见的方式反映给用户相应的结果。

本程序基于 Hadoop 集群技术设计。通过对多个主机的集群化处理完成分布式存储，统一调动和使用等目标。将数据 load 到 Hadoop 集群的数据库中，利用相对于 MapReduce 更为简单友好的 Hive 对数据进行查询和分类操作。在 Hbase 中建立临时表，以便将查询的数据写入 Hbase，通过实体类数组 entity 对 Hbase 表中 Htable 的 scan 将数据传输到实体类，以实体类与前台 Jsp 页面结合反馈给用户结果。前台结合 CSS 框架，使用 echats 插件的 map 模块，将用户的经纬坐标以显而易见的形式表示出来。

关键词：地理数据，Hive 查询，echats 插件显示

Hadoop-based access to a geographic location service social network user data

Abstract

This paper describes the working principle of Hadoop geographic data analysis system and various parts of the structure based on. According to the function of the whole system is divided into four modules. The first is the Hadoop HDFS cluster module system for accessing data; the second is the data cleansing classification MapReduce modules; the third is for the query Hbase and Hive module; the last one is the foreground for display interface module. It aims to complete the huge data cleansing classified information, and an intuitive user-visible way to reflect the results.

The program is based on Hadoop cluster technology design. By clustering process on multiple hosts complete distributed storage, unified mobilization and use of other targets. The load data into Hadoop cluster database, with respect to the use of MapReduce simpler friendly Hive data query and sorting operations. Create a temporary table in Hbase in order to query the data is written Hbase, entity by entity class array of Hbase table Htable the scan data transmission to the entity classes to entity class reception Jsp page results combined with feedback to the user. Reception combine CSS framework plugin using echats map modules will be apparent to the user's latitude and longitude coordinates of the form shown.

Keywords : geographical data , Hive queries , echats plug-in displays

目 录

1 引言.....	1
1.1. 课题的提出	1
1.1.1. 课题的背景和意义	1
1.1.2. 课题的现状 & 提出	2
1.2. 本系统的主要工作和目的	2
1.2.1. 主要工作	3
1.2.2. 最终目的	3
1.3. 本课题的技术支持	3
1.3.1. Hadoop 平台简介.....	3
1.3.2. Hadoop 集群的优点和应用.....	3
1.3.3. MapReduce	4
2 系统需求分析.....	5
2.1. 网络社交发展	5
2.2. 系统功能需求	5
2.3. 系统的性能需求	5
3 系统的详细设计.....	6
3.1. 系统流程	6
3.2. 后台模块设计	7
3.2.1. 环境和包的搭建	7
3.2.2. 创建实体类用于存储 Reduce 输出的键值对	9
3.2.3. 创建 HIVE 查询内容编写:	10
3.2.4. Map 的编写:	12
3.2.5. Reduce 的编写:	14
3.2.6. 全部数据查询方法的编写	14
3.2.7. 单一数据查询的方法	15
3.2.8. 地域查询方法	16
3.2.9. 在 Hbase 中建立 Htable.....	17
3.2.10. 扫描 Hbase 中的表	18
3.3. 前台模块设计	19

4 调试与异常处理.....	22
4.1. 异常来源	22
4.2. 异常的处理	22
5 结束语.....	23
5.1. 设计结论	23
5.2. 心得体会	23
参考文献	24
致谢	25

1. 引言

1.1. 课题的提出

1.1.1. 课题的背景和意义

时代的发展日新月异，如今以微博、社交网络应用 APP、基于位置的服务 LBS 的新型信息交流方式不断的涌现。与此同时，随着云计算、物联网等技术的兴起，数据正以前所未有的速度在不断的增长和累积，大数据时代已经来到。世界知名杂志 Nature 在 2008 年推出了 Big Data 专刊。计算社区联盟(Computing Community Consortium) 同时也在 2008 年发表了报告《Big-Data Computing: Creating revolutionary breakthroughs in commerce, science, and society》，阐述了在数据驱动的研究背景下，解决大数据问题所需的技术以及面临的挑战。麦肯锡集团(McKinsey 与)2011 年 6 月份发布了一份关于大数据的详尽报告《Big data: The next frontier for innovation, competition, and productivity》对大数据的影响、关键技术和应用领域等都进行了详尽的分析。

然而大数据的火热并不就说明人们对于大数据的了解深入，反而体现大数据存在过度炒作的风险。大数据的基本概念、关键技术以及对其的利用上均存在很多的疑问和争议。本文从大数据问题背后的本质出发，对现有的大数据研究资料进行全面的归纳和总结。首先简要介绍大数据的基本概念，阐述其同传统数据库的区别。在此基础上，对大数据处理框架进行详细解析。我们认为大数据的发展离不开云计算技术，云计算支撑着大数据存储、管理以及数据分析等。因此本文展开介绍了大数据时代不可或缺的云计算技术和工具。最后全面阐述大数据时代面临的新挑战。

就另一方面而言，位置服务(location based service ，LBS)是通过电信移动运营商的网络(如 GSM 网、CDMA 网)获取移动终端用户的位置信息(经纬度坐标)，在电子地图平台的支持下,为用户提供相应服务的一种增值业务。它是移动通信技术、空间定位技术、地理信息系统技术等多种技术融合发展到特定历史阶段的产物，是现代科学技术和经济社会发展需求的客观要求。随着移动通信技术、网络技术和测绘技术的发展，与位置相关的信息增值服务已经成为现代地理信息产业的重要组成部分，并逐渐成为其中最大的增长点。近年来，我国地理信息产业发展迅猛，2007 年，地理信息产业总规模达到 500 亿元，从业人数达到 40 多万人，其中基于位置

的服务占到很大比重，位置服务越来越得到人们的关注和重视,位置服务的概念也频频出现在各种报刊、网络和政府文献中,成为很多地理信息产业业内人士热议的话题。

1.1.2. 课题的现状并提出

进入 21 世纪以来，大数据的概念更是可谓深入人心。从谷歌的 3 篇论文开数据大数据时代到如今滴滴，Uber，网盘等结合云计算和大数据的技术产业迅猛发展。数据也产生了三个重要的变化(1) 数据量，由 TB 级升至 PB 级，并仍在持续爆炸式增长。根据 WinterCorp 的调查显示，最大的数据仓库中的数据量，每两年增加 3 倍(年均增长率为 173%)，其增长速度远超摩尔定律增长速度。照此增长速度计算，2016 年最大数据仓库中的数据量将逼近 100PB。(2) 分析需求。由常规分析转向深度分析(DeepAnalytics) .数据分析日益成为企业利润必不可少的支撑点。根据 T DWI 对大数据分析的报告企业，已经不满足于对现有数据的分析和监测，而是更期望能对未来趋势有更多的分析和预测，以增强企业竞争力。这些分析操作包括诸如移动平均线分析、数据关联关系分析、回归分析、市场篮分析等复杂统计分析，我们称之为深度分析。值得补充的是，本文中的大数据分析不仅仅指基于大数据上的深度分析，也包括常规分析。(3)硬件平台。由高端服务器转向由中低端硬件构成的大规模机群平台。由于数据量的迅速增加，并行数据库的规模不得不随之增大，从而导致其成本的急剧上升。出于成本的考虑，越来越多的企业将应用由高端服务器转向了由中低端硬件构成的大规模机群平台。同时，大数据的一个更为广泛的应用就是在地图和社交方面的应用。结合 LBS 系统。可以分析和调查一个人的行动轨迹，对对象进行数据画像。应用大数据的技术描绘出一个人的生活状态。

诸如类似的有高德地图，百度地图的应用，将大数据结合 LBS 系统，为用户推荐附近的餐馆，加油站，娱乐设施等。像这样多个数据库进行叠加和结合，便是大数据的概念。与此同时，很多社交软件，例如微信，人人网，百度贴吧等 APP 会针对用户进行每日签到。这些签到信息结合位置信息，就是我们将要分析的数据。以此为基本，可以以热力图的形式直观显示给用户。用于调查某地的人流情况，或分析某用户的习惯等。

1.2. 本系统的主要工作和目的

在我们了解了相关背景知识后，就要了解我们做的分析的目的。

1.2.1. 主要工作

本系统主要用于对数据进行分析 and 整理，并以直观的形式向用户反映出来。数据为某社交网站允许其用户签到，我们收集了 10000 位用户的大约 270 万次签到数据。主要工作划分如下：

- (1) 分析一位用户的每日签到数据
- (2) 分析同一地域的每日用户签到数据
- (3) 分析是否有用户去过同一地点
- (4) 分析同一地区的用户长期签到数据
- (5) 分析一地区的热力，是否适合商业发展

1.2.2. 最终目的

通过这些分析我们可以大致了解用户的地理位置信息。行动和生活轨迹。

1.3. 本课题的技术支持

1.3.1. Hadoop 平台简介

目前应用最广泛的开源分布式存储和计算平台之一是 Hadoop。它是根据 Google 的 GFS 分布式文件系统和 Map/Reduce 分布式计算技术而开发的开源平台，其设计目标是在普通的硬件平台上构建大容量、高性能、高可靠的分布式存储和分布式计算架构。Hadoop 目前已在 Yahoo、Facebook、亚马逊、百度、中移动等公司取得了广泛应用。其中 Yahoo、FaceBook 等公司已构建了数千至数万台普通服务器组成的大型 Hadoop 应用集群，FaceBook 上存储的图像数据量目前已超过 1 PB。

1.3.2. Hadoop 集群的优点和应用

Hadoop HDFS 分布式文件存储系统具有如下特点：

- (1) 非常适合海量数据的存储和处理；
- (2) 可扩展性高，只需简单添加服务器数量，即可实现存储容量和计算能力的线性增长；
- (3) 数据冗余度高，缺省每份数据在 3 台服务器上保留备份；
- (4) 适合“流式”访问，即一次写入，多次读取，数据写入后极少修改；

除了数据存储能力外，Hadoop MapReduce 分布式计算框架还可充分利用各服务器

CPU 的计算资源。

1.3.3. MapReduce

MapReduce 是 2004 年由 Google 提出的面向大数据集处理的编程模型, 起初主要用作互联网数据的处理, 例如文档抓取、倒排索引的建立等. 但由于其简单而强大的数据处理接口和对大规模并行执行、容错及负载均衡等实现细节的隐藏, 该技术一经推出便迅速在机器学习、数据挖掘、数据分析等领域得到广泛应用. MapReduce 将数据处理任务抽象为一系列的 Map(映射)-Reduce(化简) 操作对. Map 主要完成数据的过滤操作, Reduce 主要完成数据的聚集操作. 输入输出数据均以 $\langle \text{key}, \text{value} \rangle$ 格式存储. 用户在使用该编程模型时, 只需按照自己熟悉的语言实现 Map 函数和 Reduce 函数即可, MapReduce 框架会自动对任务进行划分以做到并行执行. MapReduce 是面向由数千台中低端计算机组成的大规模机群而设计的, 其扩展能力得益于其 shared-nothing 结构、各个节点间的松耦合性和较强的软件级容错能力: 节点可以被任意地从机群中移除, 而几乎不影响现有任务的执行。

2. 系统需求分析

2.1. 网络社交发展

结合现今互联网发展趋势，网络应用已经成为我们生活中不可缺少的一部分，同时，社交软件给我们一个很好的分析平台。互联网交往形式主要有网络聊天、BBS、电子邮件、网络游戏、网络论坛、社会化网络等。主要网络交往动机包括情感需求、信息需求、打发时间、获得社会支持、娱乐休闲等。借助网络社交的快速发展，我们可以从中挖出我们需要的数据要求。访问量，热力图等一系列数据。微信，陌陌等社交软件也给我们提供了更多更为宽广的社交平台。同时他们都有一个明显的特点。为用户提供地理位置功能。例如“附近的人”等关系，这样将网络中的虚拟社交提升到身边的现实生活中。这样复杂庞大的地理位置数据更是商业公司新的资源。

2.2. 系统功能需求

为直观反映用户的地理数据情况，系统应当做到以下几个方面：

- (1) 能直观的反映一个地区的热力图
- (2) 能直观的反映一个用户的活动迹象
- (3) 能实现所需要的查询功能

2.3. 系统的性能需求

系统的反应时间应当在毫秒级。

系统应当能对错误的输入做出相应的反应。

系统应当能够承受大流量大数据的访问和调用。

3. 系统的详细设计

3.1. 系统流程

系统流程的开始应当是，用户访问相应网址，在出现的前台界面点击相应的操作。后台对操作进行执行，然后将结果反馈给前台的页面。流程图如下：

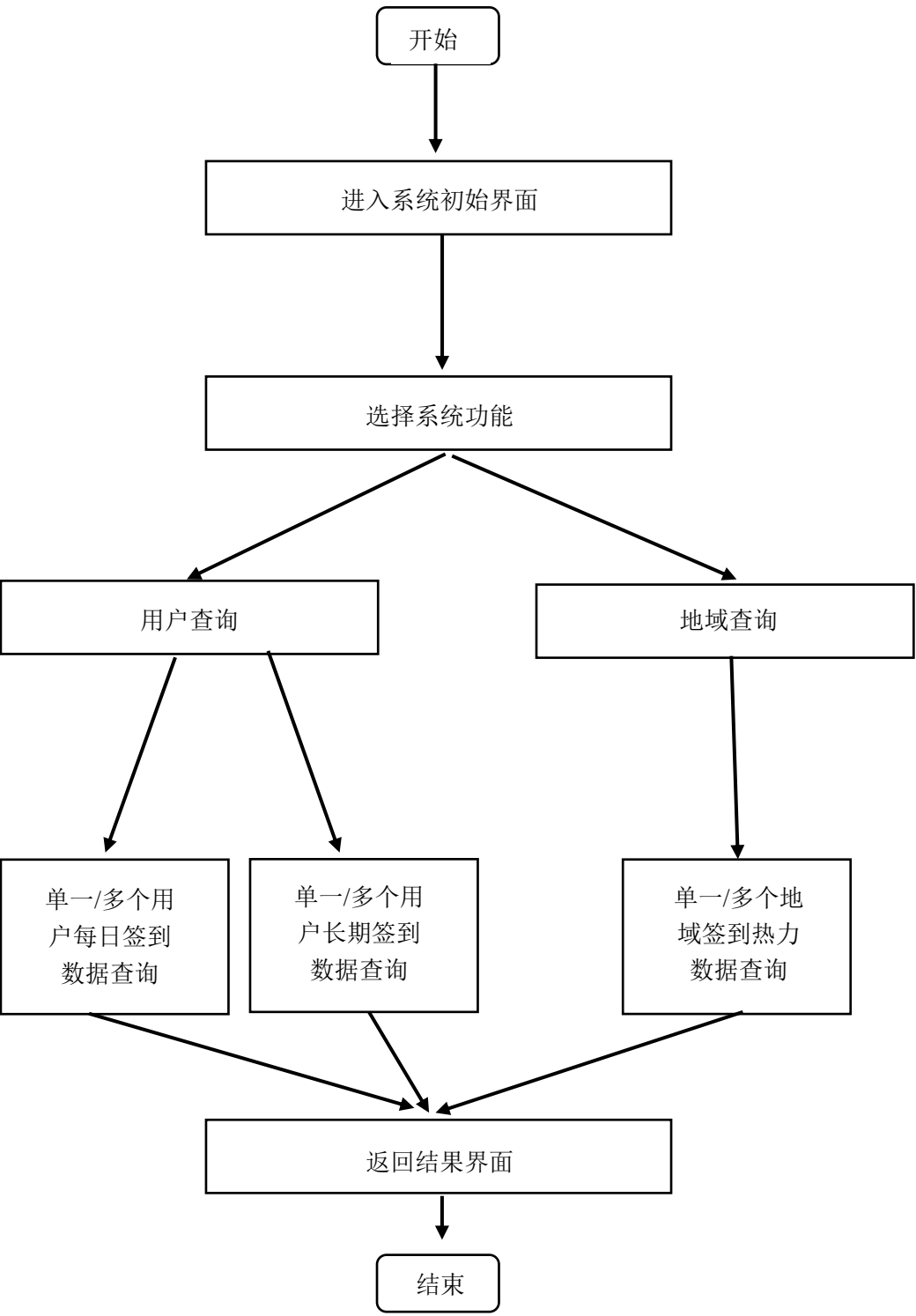


图 3.1 流程图

3.2. 后台模块设计

根据需求分析，后台应当完成 Hadoop 集群的搭建调试，以及 Data 数据的分布式存储。同时要完成在查询时的 MapReduce，以及 Hive 等操作。

3.2.1. 环境和包的搭建

搭建 Hadoop:

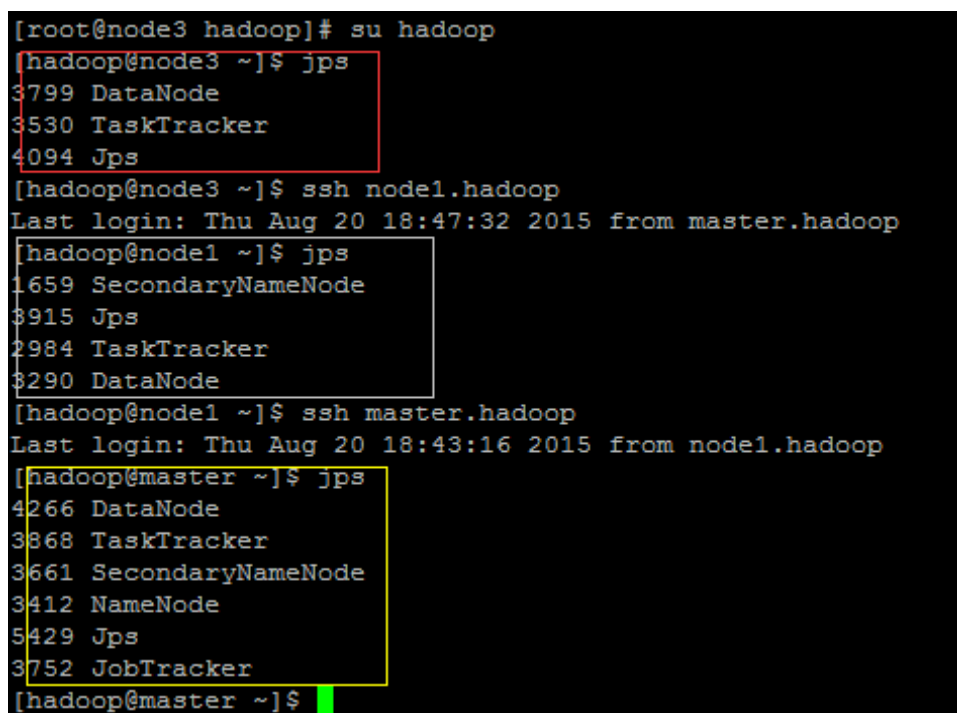
使用虚拟机安装 centos6.5 版本，安装成功后设置固定 ip，本环境设定如下:

主机名: master.hadoop 192.168.210.128

节点 1: node1 192.168.210.144

节点 2: node3 192.168.210.145

所有机器通过虚拟机桥接模式连接;



```
[root@node3 hadoop]# su hadoop
[hadoop@node3 ~]$ jps
3799 DataNode
3530 TaskTracker
4094 Jps
[hadoop@node3 ~]$ ssh node1.hadoop
Last login: Thu Aug 20 18:47:32 2015 from master.hadoop
[hadoop@node1 ~]$ jps
1659 SecondaryNameNode
3915 Jps
2984 TaskTracker
3290 DataNode
[hadoop@node1 ~]$ ssh master.hadoop
Last login: Thu Aug 20 18:43:16 2015 from node1.hadoop
[hadoop@master ~]$ jps
4266 DataNode
3868 TaskTracker
3661 SecondaryNameNode
3412 NameNode
5429 Jps
3752 JobTracker
[hadoop@master ~]$
```

图 3.2.1 Hadoop 安装

创建工程和包:

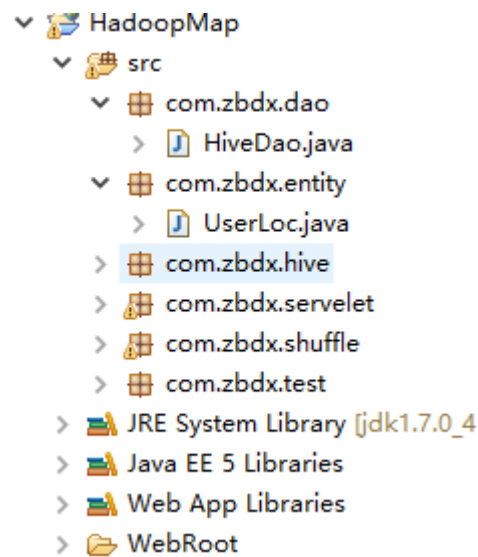


图 3. 2. 2 工程和包的搭建

Dao Hive 数据库模块作用是与 HDFS 数据存储系统进行数据连接认证。

部分代码如下：

```
public class HiveDao {  
    Connection conn=null;  
    Statement st =null;  
    ResultSet rs=null;  
  
    public Connection getConnection(){  
        try {  
            Class.forName("org.apache.hadoop.hive.jdbc.HiveDriver");  
与数据库进行连接：  
            conn=DriverManager.getConnection("jdbc:hive://192.168.210.128:10000/default",  
"hadoop", "123456");  
        } catch (ClassNotFoundException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        } catch (SQLException e) {  
            // TODO Auto-generated catch block
```

```

        e.printStackTrace();
    }
    return conn;
}
关闭数据库连接
public void closeConnection(Connection conn,Statement st,ResultSet rs){
    try {
        if(rs!=null){
            rs.close();
        }
        if(st!=null){
            st.close();
        }
        if(conn!=null){
            conn.close();
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

3. 2. 2. 创建实体类用于存储 Reduce 输出的键值对

这一步的作用是要利用实体类，对 MapReduce 之后的数据进行继承，以便将数据有效的在前后台之间传递。将 Hive 查询之后执行 MapReduce 输出的值，以数组的形式输入到 entity 中，再在 jsp 中使用 el 转译表达式，输出。

部分代码如下：

```
package com.zbdx.entity;
```

```

public class UserLoc {
    private String uname;
    private double ujing;
    private double uwei;
    public String getUname() {
        return uname;
    }
    public void setUname(String uname) {
        this.uname = uname;
    }
    public double getUjing() {
        return ujing;
    }
    public void setUjing(double ujing) {
        this.ujing = ujing;
    }
    public double getUwei() {                                     //自动生成的 doGet 方法
        return uwei;
    }
    public void setUwei(double uwei) {
        this.uwei = uwei;
    }
}

```

3.2.3. 创建 HIVE 查询内容编写：

首先将 Hive 连接到 Hadoop 集群的数据库。使用 JDBC 连接方法。我们设置的虚拟集群 master 的 ip 为：192.168.210.128.

```

public class MapHive extends HiveDao {
    public Configuration conf() {
        Configuration cfg = HBaseConfiguration.create();
    }
}

```

```

        cfg.set("hbase.zookeeper.quorum", "192.168.210.128");
        return cfg;
    }

    public void drop() {
        Connection conn = super.getConnection();
        Statement st = null;
        ResultSet rs = null;
        try {
            // 连接的异常处理
            st = conn.createStatement();
            String sql = "drop table user_hqh";
            rs = st.executeQuery(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            super.closeConnection(conn, st, rs);
        }
    }
}

```

其次，我们要将数据 load 到 Hive 的 record 中。这时就要在 Hive 中建立一个相应的表格。我们采用 HQL 中的 Create 方法。列名分别为：“用户 ID”“用户经度”“用户纬度”。数据类型依次为：“int”“double”“double”。

```

    public void createTable() {
        // 创建查询表
        Connection conn = super.getConnection();
        Statement st = null;
        ResultSet rs = null;
        try {
            st = conn.createStatement();
            String sql = "create table " + "user_hqh" + "(" + "userid string,"
                + "ujing double," + "uwei double)"
                + "row format delimited " + "fields terminated by '\t' "
                + "stored as textfile";
            //自动按照 txt 中的数据分行
            rs = st.executeQuery(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            super.closeConnection(conn, st, rs);
        }
    }
}

```

创建的表如下：

表 3.2.1 Uloc 数据表

用户 ID	用户经度	用户纬度
-------	------	------

1	30.579647	114.323888
1	30.259136	120.158161
...

在我们已经做好了查询表之后，下一步就是要将我们已经拥有的数据 load 到 Hive 数据库中，这是我们采用 load 方法。

```

public void load(String filePath) {    载入查询表
    Connection conn = super.getConnection();
    Statement st = null;
    ResultSet rs = null;
    String sql = "load data inpath " + filePath + " into table user_hqh";
    System.out.println("Running:" + sql + ":");    //采用 load 写法载入数据
    try {
        st = conn.createStatement();
        rs = st.executeQuery(sql);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } finally {
        super.closeConnection(conn, st, rs);
    }
}
}

```

3.2.4. Map 的编写：

接下来我们要对我们取到的数据进行数据的清洗和分析。这就要用到我们提到的 MapReduce 技术。Map 就是将文档中取到的数据进行拆分。例如我们的数据中有空格 “/t” 我们可以将文本以 “/t” 为分隔符号进行分割。语句如下：

```
String[] line=value.toString().split("\t");
```

语句中，将所有截取到的数据已字符的形式转存到 String 数组 line 中。

就正常情况而言，我们应当用 Reduce 对清洗过的数据进行筛选和输出，这也就是 shefful 的作用。但是，从数据的截图可以看出。我们的数据是已经做过隐私处理的数据，每一行都是已经格式化之后的。可以直接用于载入 Hive 存储表格。所以我们在 mapreduce 中加入一个 count 功能，对每一位用户的地理位置强度。进行分析。因为一位用户很可能在一个位置出现多次。例如，用户可能在家里，公司签到多次以上。这样就要引入这样的一个强度概念。我们用 mapreducde 判断是否有相同的地理位置。如果有，则计数。

由于我们用 MapReduce 系统对数据进行数据清洗的话需要更多的时间和代码。极不方便。所以，我们在 MapReduce 阶段仅仅对数据起到一个截取作用。将所有数据转存到 String 数组 Line 中。然后对 Line 进行输出。

输出数组 Line 的代码如下：

```
context.write(key, new Text(""));
```

```
package com.zbdx.shuffle;
```

```
public class UserMap extends Mapper<LongWritable, Text, Text, Text>{
```

```
    protected void map(LongWritable key, Text value,Context context){
```

```
        String[] line=value.toString().split("\t");
```

```
        String[] ujing=value.toString().split("\t");
```

```
        String[] uwei=value.toString().split("\t");
```

```
        StringBuffer str=new StringBuffer();
```

```
        for(int i=0;i<line.length;i++){
```

```
            if(line[i].length()<1){
```

```
                line[i]=null;}
```

```
            str.append(line[i]).append("\t");
```

```
        }
```

```
        try {
```

```
            context.write(new Text(str.toString()),new Text(" "));
```

```
        } catch (IOException e) {
```

```
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
```

```
        } catch (InterruptedException e) {
```

```
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
```

```
        }
```

```
    };
```

```
}
```

3.2.5. Reduce 的编写：

```
package com.zbzx.shuffle;

public class UserReduce extends Reducer<Text, Text, Text, Text>{

    protected void reduce(Text key, java.lang.Iterable<Text> value,Context context)

        throws java.io.IOException ,InterruptedException {

        context.write(key, new Text(""));

    };

}
```

3.2.6. 全部数据查询方法的编写

在创建好表之后我们就要对数据进行查询操作。这里我们有两种写的方法，可以使用 Select 查询方法对所有数据进行查询，也可以直接将我们插入的数据进行转译显示在前台上。

在这里我们列举 Select 查询方法的写法：

```
String sql="select * from user_loc ";

System.out.println("Running:"+sql+":");

try {

    st=conn.createStatement();

    rs=st.executeQuery(sql);

    HTable table=new HTable(cfg, "count_cl2");

    UserLoc userloc=null;

    while(rs.next()){

        Put put=new Put(rs.getString(1).getBytes());

        put.add("uid".getBytes(), null, rs.getString(2).getBytes());

        put.add("ujing".getBytes(), null, rs.getString(3).getBytes());

    }

}
```

```

        put.add("uwei".getBytes(), null, rs.getString(4).getBytes());
        table.put(put);
    }
    //System.out.println(11);
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} finally{
    super.closeConnection(conn, st, rs);
}

```

3.2.7. 单一数据查询的方法

在我们列出所有数据之后就要进行我们这个页面系统应当有的操作。对单一用户的数据进行查询显示。在 Hive 中，我们进行条件查询。核心语句为：

```
String sql="select * from user_loc where userid='txt_input'";
```

部分代码如下：

```

try {

    st=conn.createStatement();
    rs=st.executeQuery(sql);
    HTable table=new HTable(cfg, "count_cl1");
    UserLoc userloc=null;
    while(rs.next()){
        Put put=new Put(rs.getString(1).getBytes());
        put.add("uid".getBytes(), null, rs.getString(2).getBytes());
        put.add("ujing".getBytes(), null, rs.getString(3).getBytes());
        put.add("uwei".getBytes(), null, rs.getString(4).getBytes());
    }
}

```

```
        table.put(put);
    }
    //System.out.println(11);
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} finally{
    super.closeConnection(conn, st, rs);
}
```

3.2.8. 地域查询方法

由于我们的数据覆盖好多个省份，当我们想要查询一个省的用户访问数据时，我们可以给我们需要查询的经纬度限定范围。

下面我们以山西省为例：

核心代码语句：

```
String sql="select * from user_loc where ujing between 110.249 and 114.553 and
uwei between 34.582 and 40.720 ";
```

```
System.out.println("Running:"+sql+":");
```

部分代码：

```
Configuration cfg=conf();
```

```
Connection conn=super.getConnection();
```

```
Statement st=null;
```

```
ResultSet rs=null;
```

```
String sql="select * from user_loc where ujing between 110.249 and 114.553
and uwei between 34.582 and 40.720 ";
```

```

System.out.println("Running:"+sql+");

try {
    st=conn.createStatement();
    rs=st.executeQuery(sql);
    HTable table=new HTable(cfg, "count_cl2");
    UserLoc userloc=null;
    while(rs.next()){
        Put put=new Put(rs.getString(1).getBytes());
        put.add("uid".getBytes(), null, rs.getString(2).getBytes());
        put.add("ujing".getBytes(), null, rs.getString(3).getBytes());
        put.add("uwei".getBytes(), null, rs.getString(4).getBytes());
        table.put(put);
    }
}

```

至此，后台的 3 个任务模块已经写好。之后就是要在 Hbase 中建立新的表格，将我们查询到的数据写入到 Hbase 中。用 entity 的数组进行对表的扫描。将数据继承到数组中，通过 Servlet 把数据传入到前台的 Jsp 页面。

3.2.9. 在 Hbase 中建立 Htable

```

public void createcount_c() throws Exception{
    Configuration cfg=conf();
    HBaseAdmin admin=new HBaseAdmin(cfg); //连接 Hbase
    if(admin.tableExists("count_cl2")){
        admin.disableTable("count_cl2".getBytes());
        admin.deleteTable("count_cl2".getBytes());//若 Hbase 中已经存
    }
    HTableDescriptor tableDesc=new
    HTableDescriptor("count_cl2".getBytes());
    tableDesc.addFamily(new HColumnDescriptor("uid".getBytes()));
}

```

```

        tableDesc.addFamily(new
HColumnDescriptor("ujing".getBytes()));
        tableDesc.addFamily(new
HColumnDescriptor("uwei".getBytes()));
        admin.createTable(tableDesc);
    }else{

```

...

3. 2. 10. 扫描 Hbase 中的表

将数据继承到 entity 中

```

public List<UserLoc> scanCount() throws IOException{

    List<UserLoc> Usercount =new    ArrayList<UserLoc>();
    UserLoc a = new UserLoc();
    int i;
    Configuration cfg=conf();
    HTable table=new HTable(cfg, "count_cl2");
    Scan s=new Scan();
    ResultScanner rs=table.getScanner(s);
    Result re=rs.next();
    while(re!=null){
        a = new UserLoc();
        List<KeyValue> l=re.list();
        for( i=1;i<=4;){
            for(KeyValue kv:l){

                String fam =new String(kv.getFamily());
                if(fam.equals("uid")){
                    a.setUid(new String(kv.getValue(),"utf-8"));
                }
                if(fam.equals("ujing")){
                    a.setUjing(new String(kv.getValue(),"utf-8"));
                }
                if(fam.equals("uwei")){
                    a.setUwei(new String(kv.getValue(),"utf-8"));
                }
                i++;
            }
            re=rs.next();
        }
    }
}

```

```

        Usercount.add(a);
        System.out.println(Usercount);
    }
    return Usercount;
}

```

3.3. 前台模块设计

前台模块要做到简洁明了，大方美观，系统人性化程度高，人机交互友好。于是我在前台运用了 php 同学们使用的 CSS 模板，并对其进行优化处理，以便充分适应自己的项目。



图 3.3.1 前台首页页面

通过 div 框架对页面的长度做出调整，是页面可以向下滚动。同时对右上角的 HOME 采用<nav>标签，做出一个导航的效果。以方便用户进行直接的跳转。核心代码入下：

```

<li class="active"><a href="index.html">首页</a></li>
<li><a href="#a1" class="smoothscroll back-to-top">项目功能</a></li>
<li><a href="left-sidebar.html">Left Sidebar</a></li>
<li><a href="right-sidebar.html">Right Sidebar</a></li>

```




图 3.3.2 nav MENU 导航菜单

之后做出选择功能的界面如下图所示：



图 3.3.3 项目功能展示

部分核心代码如下：

```
<a href="show_all.jsp" class="fh5co-block-links text-center">
    <div class="icon-circle">
        <i class="ti-settings"></i>
    </div>
    <h2>全部数据演示</h2>
    <p>All data presentation</p>
    <p><span class="learn-
more">SHOEING</span></p></a>
</div>    </div>
```

全部数据演示界面，使用新的 Jsp 页面进行编写利用<div>标签搭载 echats 插件。



图 3.3.4 前台用户位置显示界面

之后写出前台的单数据查询界面，为了人机交互友好，全部写在一个界面。



图 3.3.5 前台单一数据查询界面

4. 调试与异常处理

4.1. 异常来源

页面异常的来源可能是用户的错误输入。例如 输入查询用户名时输入了从未有过的用户名，导致机器无法识别。或者是在查询过程中出现错误。网络故障或者数据库连接未能通过。所以调试是整个系统安装前的最后准备工作，包括硬件测试和程序测试，硬件测试需要进行计算机的维护，网络状况的检测。软件测试需要对各种数据进行测试，只有经过严格测试并编译成功才能将生成的键值对可以对应到 jsp 页面中，结果通过则系统的调试完成。

4.2. 异常的处理

在数据库连接中，异常采用 try catch 语句进行避免跳出。

部分代码如下：

```
public void closeConnection(Connection conn,Statement st,ResultSet rs){
    try {
        if(rs!=null){
            rs.close();
        }
        if(st!=null){
            st.close();
        }
        if(conn!=null){
            conn.close();
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

5. 结束语

5.1. 设计结论

本系统通过对各个模块的调试完成了最后的页面设计和分析。在设计中，我总是会遇到很多不可思议的困难。例如 Echats 插件和 Reduce 输出的键值对数组的结合。困扰了我很久。最后在同学的帮助下终于能够得以解决。

同时，这个系统的设计也让我明白了，其实静下心来挖掘很多信息都是有許多面的，例如我们分析的用户社交位置信息，就可以分析出一个人的习惯，生活轨迹等很多方面的因素。但在界面友好上还需要进一步的改进。

5.2. 心得体会

经过长时间的准备，整个毕业设计即将完成，而大学生活也就快结束了，这次的毕业设计使我更进一步掌握了大数据知识及其相关技术，也更深入地了解了 Hadoop 技术的前后台设计与编写。

在设计过程中我遇到不少问题。代码的编写错误，不过 Java 的自动补全编写避免了一些不必要的错误，例如拼写错误等。但同时对我们的代码逻辑性也有了更高的要求。因为我们只有在熟悉代码逻辑的情况下，才能真正意义上运用到补全功能，这也不失为我们查错的一种方法。虽然遇到问题，然而，正是在对这些问题一个一个地解决过程中我越对编程越发产生了浓厚的兴趣。从一开始牙牙学语一样的 C 语言，到现在可以熟练的敲出 Java 里的一行行代码。成长在我们的眼里是看的到的。

最后就是对设计的程序进行调试，调试要有一定的技巧，不能将一堆程序放到一起，可以将一小个模块进行调试，调试成功后，将其加入大的功能模块中，在对整体调试直至无误后，再重复进行相关步骤，依次整合各个小的模块。

总之，这次毕业设计中和同组同学互相学习，使我的能力又得到了进一步提升，虽然中间也有彷徨的时候，但是最终能顺利完成，使我对未来充满信心。

参考文献

- [1] 周雪晴, 罗亚玲. 信息化建设中大数据现状[J]. 中华图书情报杂志, 2015(11):48-51.
- [2] 王珊, 王会举, 覃雄派, 等. 架构大数据:挑战、现状与展望[J]. 计算机学报, 2011, 34(10):1741-1752.
- [3] 田丽, 安静. 网络社交现状及对现实人际交往的影响研究[J]. 图书情报工作, 2013, 57(15):13-19.
- [4] 中国互联网络信息中心. 第 32 次中国互联网络发展状况统计报告[J]. 互联网天地, 2013(10):22-24.
- [5] White T. Hadoop: The Definitive Guide[J]. O'reilly Media Inc Gravenstein Highway North, 2010, 215(11):1 - 4.
- [6] O'Driscoll A, Daugelaite J, Sleator R D. 'Big data', Hadoop and cloud computing in genomics[J]. Journal of Biomedical Informatics, 2013, 46(5):774-81.
- [7] Park K Y, Cho I S. LBS having GPS receiver and operating method thereof: US, US 20040267451 A1[P]. 2004.
- [8] 王峰, 雷葆华. Hadoop 分布式文件系统的模型分析[J]. 电信科学, 2010, 26(12):95-99.
- [9] 张田. 一种轻量级架构的利用流数据概念生成热力图的方法:, CN103927368A[P]. 2014.
- [10] 崔维福, 杨基彬. 网页热力图着色处理方法和装置: CN, CN 103559249 A[P]. 2014.
- [11] 孟小峰, 慈祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1):146-169.
- [12] 覃雄派, 王会举, 杜小勇, 等. 大数据分析--RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报, 2012, 23(1):32-45.

致谢

XXX