基于 Hadoop 的出租车营运行为分析

专业：计算机科学与技术(专升本)　学生：谢小烽　指导老师：黄风华

摘要

随着互联网、大数据技术飞速发展与广泛应用，大数据正成为信息技术的新热点与新的发展方向，对于人类社会的生活与生产将产生巨大的影响。大数据来源于互联网、物联网和企业系统等信息系统，经过大数据平台分析与挖掘，产生新知识数据用于业务的智能化运转，大数据时代的到来给数据管理、分析提出了新的挑战，数据处理方法的合理性与时效性成为大数据的研究方向。近年来，基于数据挖掘算法的大数据分析是研究的重要方向，但都以传统单机环境下的数据挖掘算法改进为主，受到单机模式下计算能力、存储能力、扩展性等多方面的限制，不能有效地满足海量的数据处理需求，因此本文研究传统数据挖掘算法在 MapReduce 并行编程环境下的实现方法。

在平台的实现上，采用 Hadoop 作为数据处理平台，使用 Apriori 算法对出租车的营运数据进行关联分析，分析出出租车载客热点、载客峰值与营运规则等有价值的重要信息。采用 HDFS 存储挖掘分析的出租车营运源数据、挖掘分析过程中产生的中间数据、挖掘分析的结果数据，最后将结果数据从 HDFS 存储到 MySQL 关系数据库保存，MySQL 数据库通过与网站的实时交互动态将分析结果以百度地图、简数 CDN 图表的方式展现。针对单机环境下传统数据挖掘算法在分析处理大规模数据时存在计算性能低、拓展性弱、内存消耗高等问题，提出基于 MapReduce+Apriori 的关联规划并行实现方法。在 Hadoop 云计算平台下进行实验验证，并通过精度和效率这两个指标评价算法的性能。平台采用主从架构,具有较强的可靠性、高扩展性、高容错性、高效性。

本文以大规模出租车 GPS 数据为实例，首先阐述了基于 Hadoop 的大数据分析、研究与实现的关键技术。其次，详细介绍了 Hadoop 平台的架构、平台搭建、平台数据库的设计与实现、Apriori 算法的并行实现和 JSP 网站开发过程。最后，基于 Hadoop 平台的出租车营运分析结果进行了全面总结，并提出了本课题存在的问题及不足之处，对系统未来需要发展及完善的内容做了展望。系统中存在的不足之处，有待于今后在工作和学习中进一步深入研究。

关键字：Hadoop　出租车　Apriori　大数据

目　录

1 绪论

随着移动互联网、物联网、云计算等新型技术的迅猛发展，数据规模成爆炸性增长，数据模式更高度复杂化，标志着信息社会已经进入到大数据时代[1-2]。随着智慧城市化的推进以及国民经济的快速发展，出租车作为城市交通的重要部分，其营运信息的交流与传递变得越来越频繁，出租车营运也面临重大挑战。通过大数据手段来挖掘与分析出租车营运数据是极具特色的解决方案。Hadoop 云计算平台作为大数据处理的代表性技术之一日益成熟，在各行业得到广泛应用，基于 Hadoop 的出租车营运行为分析可以采用 MapReduce 分布式并行编程框架实现快速数据分析，大大提高了数据分析的效率。此外，MapReduce 计算模型编写并行程序相对简单，在并行编程分析数据时无需考虑如分布式存储、工作调度、负载平衡、容错处理，网络传输等，基于 Hadoop 云计算平台的数据分析具有很好的帮助。

1.1 国内外研究现状

城市现代化建设迅速，城市正在不断扩张。传统的交通方式已经很难满足城市居民现在化出行的多样化需求。出租车，作为一种快速直接的出行方式，具备方便、灵活、快速等特点，受到城市居民的青睐[3] 。近年来，国内外出租车发展迅速，规模不断增大，但是出租车的营运方式确是一直没有较大的改变。传统的出租车营运方式有固定的出租车停候车点打车、招手打车、电话预约打车方式等，其中招手打车为主要搭乘方式。这几种方式的随机性非常大，出租车司机的载客路线完全按照历史的经验，司机与乘客无直接性的交流，同时也没直接性的数据可以借鉴与依据，所以通常出租车在传统方式下的空载率较高。所以，很多城市单纯的通过增加出租车的数量来改善城市居民出行打车便利的问题。但是，通过简单的增加出租车数量的方式不仅避免不了运营效率低的问题，还带来了许多方面的问题。如车辆过多的冗余问题、营运成本直线增高的问题等。不仅浪费了出租车的时间和资源，还给城市交通带来了压力[4]。

随着无线通行技术的快速发展和普及，出租车 GPS 数据采集技术日渐成熟。目前，出租车普遍上都安装 GPS 终端，这些终端在固定时间上给出租车调度中心发送一次数据，如出租车当前所在的位置、时间、是否载客等，出租车调度中心则可以根据此信息进行实时监控。如何从庞大的出租车营运信息中提取出能为出租车行业服务的黄金数据是如今出租车营运数据研究的一个热点与方向。所以本文针对出租车营运数据进行挖掘研究，获取知识并进行智能化的推荐。

与此同时，随着计算技术与网络技术的广泛应用，各个行业都存储积累了大量、复杂的数据。如何将积累的大数据转化成为各种形式的有用知识数据，是迫切的问题。在大数据面前，单纯的依靠数据统计软件或者数据库的查询、筛选机制是很难获得这些有用的知识数据，而是应该自动化、智能化地将数据转化成为有价值的知识数据，并为管理决策提供服务。因此，数据挖掘应运而生，将机器学习、专家系统、人工智能、数据库技术和信息检索等技术整合在一起，广泛应用于各个领域[5]

关联分析算法是数据挖掘中的一个功能，在研究对于提高数据发现、搜索效率，推广数据库在社会各领域的应用等方面具有十分重大的理论价值和实际意义[6]。关联分析是一种简单、实用的分析技术，发现存在于大量数据集中的关联性或相关性，从而描述了一个事物中某些属性同时出现的规律和模式。本文通过关联规则算法来实现有价值信息的获取。

基于上述背景，国内外对数据挖掘关联规则的研究如火如荼。由于城市交通已面临诸多严重问题，必须采取更有效的措施，保证城市交通系统有效运行。为了应对城市交通运行困境,美、欧、日三国取得了长足发展，这三个国家及地区也成为了世界智能交通研究应用的主要基地。我国近年来也一直在充分利用物联网、云计算、大数据、移动互联等技术，大力推进我国交通运输领域的信息化。在这样的形势之下，交通网络优化、智能化出行服务以及交通应急保障等方面将形成巨大的市场，大数据技术将成为市场新趋势[7]。

2006 年 8 月 9 日，在搜索引擎大会上，谷歌公司首席执行官埃里克·施密特首次提出了云计算的概念。届时谷歌公司在发表的学术论文中提出了云计算的三个重要概念，结构化存储（BigTable）、分布式编程模型（MapReduce）以及分布式文件存储系统（GFS）[8]。正是由于谷歌公司发表的三篇学术论文，大数据技术迅速崛起，通过现有的云计算作为底层技术，支撑着上层的大数据处理，谷歌公司在学术论文中提到，未来只要动一下鼠标就可以在秒极操作 PB 级别的数据。

在开源框架方面，Apache 基金会的 Hadoop 云计算平台因其优良的特性被许多企业和机构大量引入使用来作为大数据分析平台，它包括分布式存储系统（HDFS）和分布式编程模型（MapReduce）。目前，许多国内外大型公司，如 Facebook、雅虎、推特、华为、淘宝等都在使用 Hadoop 平台。 本文也是基于 Hadoop 平台，利用分布式存储（HDFS）出租车营运数据，运用分布式编程模型（MapReduce）将关联分析算法进行并行化实现。

1.2 论文研究内容

通过上述对国内外研究现状的研究与分析，在深入了解大数据与数据挖掘知识后，提出了基于 Hadoop 的出租车营运行为分析系统；在了解了 Apriori 关联分析算法的基础上，将 Apriori 关联分析算法与 Hadoop 平台相结合的方式运用到出租车营运行为分析中，通过 MapReduce 分布式计算模型对 Apriori 关联分析算法进行并行化实现，很好地解决了海量数据因为单机计算能力和内存不足等带来的性能问题。

本文的具体研究内容如下：

（1）研究与分析 Hadoop 分布式计算框架，对 Hadoop 云计算平台下的分布式文件系统（HDFS）与分布式计算模型（MapReduce）的核心架构、相关机制进行研究分析。

（2）对数据挖掘知识进行了深入了解，通过大量查阅对出租车营运行为的书籍与文献后，给出了基于 Hadoop 的出租车营运行为分析系统的设计思想与模型结构。基于出租车营运行为的信息挖掘系统，实现出租车载客热点的推荐、载客峰值分析和营运规则分析。

（3）对关联分析算法进行了深入了解，学习了数据挖掘中关联分析算法中的经典 Apriori 算法。结合 Apriori 算法与 MapReduce 分布式计算编程模型，设计了并行化的出租车营运行为的关联分析算法，并基于 Hadoop 云计算平台进行实现。

（4）通过南京市的出租车营运数据对基于 Hadoop 云计算平台的关联分析算法的效率、正确性进行了验证，并给出了出租车载客热点、载客峰值与营运规则的实验结果，证实了通过并行关联分析算法对大量出租车营运数据进行分析，并得到载客热点的推荐、载客峰值的统计、营运规则的推荐的可行性与设计的合理性。

1.3 研究技术方案

结合本课题的特点，本文的研究技术方案如下：

（1）通过搭建 Hadoop 云计算平台实现分布式云存储(HDFS)和分布式并行计算编程模型（MapReduce），该平台用于存储业务源数据、挖掘分析后的中间数据和最终结果数据，还用于运行 MapReduce 程序实现对源数据的挖掘。

（2）通过 Eclipse IDE 工具、Java 语言和 MapReduce 分布式并行计算编程模型进行 Apriori 算法的并行化业务逻辑实现，编程分析挖掘出租车的营运规则信息。

（3）将分析结果从 HDFS 中存储到 MySQL 数据库中，方便不同用户读取与访问。

（4）通过 JSP 技术实现动态网站的开发，主要实现将分析完成的数据通过网页报表、地图叠加的方式展现出来。

1.4 论文组织结构

本文的组织结构如下：

第 1 章，绪论。本章节首先对本文的背景意义和国内外的研究现状进行了简单的分析介绍，然后对出租车营运数据、关联分析算法和大数据技术研究现状进行了简要的概述，最后介绍了本文所做的主要内容工作和论文的组织结构。

第 2 章，系统研发的相关技术。本章首先对数据挖掘进行了介绍，从数据挖掘的概念到数据挖掘的过程。然后，重点分析介绍了 Hadoop 云计算平台、Hadoop 分布式存储系统(HDFS)、分布式计算模型(MapReduce)、关联分析算法(Apriori)。最后，简单介绍了 JSP 技术与 MySQL 技术。

第 3 章，基于 MapReduce+Apriori 数据挖掘模式设计。本章节首先介绍了关联分析算法（Apriori）并行实现的方式与原理。然后，介绍了通过分布式编程模型（MapReduce）对 Apriori 算法的并行实现。最后，介绍了 Apriori 算法的参数优化策略。

第 4 章，基于 Hadoop 的出租车营运行为分析系统设计。本章节首先介绍了系统的流程设计、功能模块设计、体系结构设计。然后，介绍了 HDFS 云存储设计、MySQL 关系数据库设计。最后，介绍了系统软硬件配置。

第 5 章，系统实现与结果分析。本章节首先介绍了系统性能评价指标，对实验数据与样本数据进行了说明。描述了系统的前后台的基本实现，验证了基于 Hadoop 云计算平台的并行化关联分析算法的正确性与高效性。最后，结合百度地图，对载客峰值、载客热点、营运规则进行了可视化现实，验证了基于 Hadoop 与关联分析算法的出租车营运规则分析的可行性。

第 6 章，总结与展望。从四个方面详细总结了本文所做的主要工作，并介绍了未来需要进一步学习研究的内容。

2 系统研发相关技术

2.1 数据挖掘概述

2.1.1 数据挖掘的产生

从 20 世纪 60 年代到 90 年代，数据处理技术先后经历了数据搜索，数据访问，数据仓库、决策支持三个阶段。近年来数据库技术日益发展与成熟，数据库应用的范围、深度和规模在不断地扩大。数据挖掘与传统的数据分析(如查询、报表、联机应用分析)的本质区别是数据挖掘是在没有明确假设的前提下去挖掘信息、发现知识数据挖掘所得到的信息应具有先未知，有效和可实用三个特征。随着信息的急剧增长，仅单纯的依靠传统的数据分析检索机制等方法已经远不能满足现实需要，它迫切要求自动、智能地将待处理的数据转化为有用的信息和知识。数据挖掘就是为迎合这种要求而产生并迅速发展起来的、可用于开发信息资源的一种新的数据处理技术。

2.1.2 数据挖掘的概念

数据挖掘（Data Mining）就是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。发现的是用户感兴趣的知识；发现的知识要可接受、可理解、可运用；并不要求发现四海皆准的知识，仅支持特定的发现问题。数据挖掘是一种新的商业信息处理技术，其主要特点是对商业数据库中的大量业务数据进行抽取、转换、分析和其他模型化处理，从中提取辅助商业决策的关键性数据。数据挖掘的方法主要有分类、回归分析、聚类、关联规则、特征、变化和偏差分析、Web 页挖掘等，它们分别从不同的角度对数据进行挖掘。数据挖掘常用技术：人工神经网络、决策树、遗传算法、近邻算法、规则推导。

2.2 Hadoop 云计算平台

Hadoop 是一款开源的云计算平台，是当前大数据处理的一项重要技术，能够对大量数据进行分布式并行处理，能够处理数据达 PB 以上级别。Hadoop 是一种离线计算，平台擅长存储大量的半结构化的数据集、分布式计算，快速地跨多台机器处理大型数据集合，数据可以随机存放，所以一个磁盘的失败并不会带来数据丢失。Hadoop 作为一款有着强大的可靠性、高效性、可伸缩性、高容错性

的支持数据密集型分布式应用，在处理海量数据方面有着极强的优势，并可构建在不同平台与廉价机器之上，正是这些设计上与生俱来的优点，才使得 Hadoop 一出现就受到众多大公司的青睐，同时也引起了研究界的普遍关注。

2.3 Hadoop 分布式存储（HDFS）

分布式存储系统（HDFS）是 Hadoop 平台的重要组成部分，它是一个高容错性的系统，对硬件的需求比较低，适合构建在大量的廉价 PC 机器之上。HDFS 能提供高吞吐量的数据访问，为海量数据处理提供了高可靠的存储性，对普通的用户分析处理海量的数据提供了便利，图 2-1 给出了 HDFS 的体系结构。

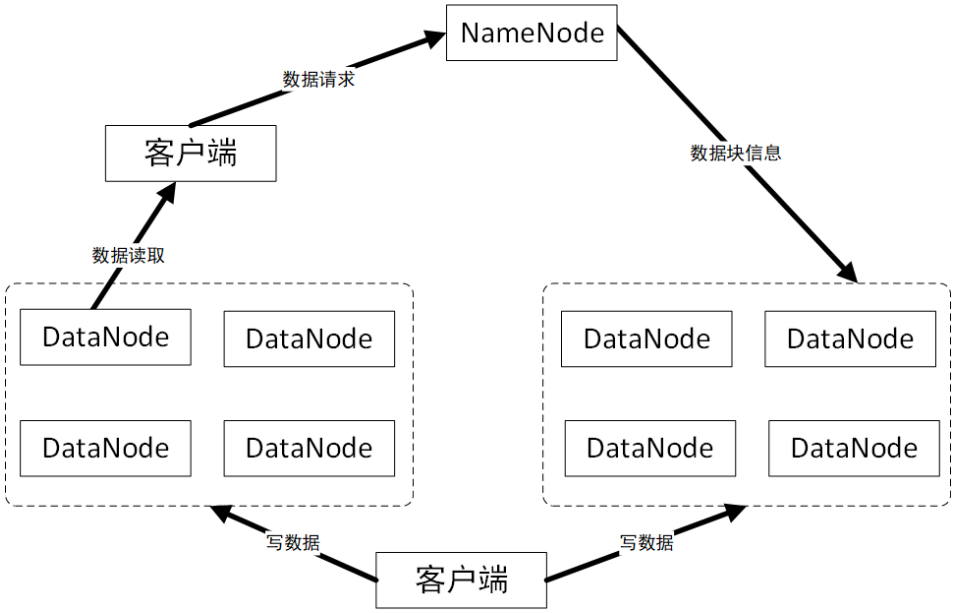


图 2-1 HDFS 体系结构

HDFS 采用主从（Master/Slave）结构对文件进行管理。但从用户的角度看，就像在操作一个本地文件系统一样。HDFS 支持 Linux 系统下的通过 Shell 命令行对数据进行查看、编辑、删除等操作，其操作命令方式类 Linux 系统下的 Shell 命令，站在用于角度来看你，非常容易学习掌握；HDFS 还可以 Java、Python 等主流的编程语言通过开放的文件系统 API 对文件系统进行操作。一个 HDFS 集群由一个名称节点（NameNode）和一定数目的数据节点（DataNode）组成。名称节点负责管理文件系统的组织结构，成为元数据，数据节点只负责实际数据的存储工作。客户端通过名称节点与数据节点进行文件系统的交互访问。客户端通过从名称节点获取数据实际存储的地址，通过这个获得的地址与真实存储数据的数据节点直接进行 I/O 操作，从而得到数据信息。

由图 2-1 可以看出，HDFS 分布式存储具有高拓展性、高吞吐率、低网络拥塞率、高可靠性等特性。由于 HDFS 是通过 Java 语言开发，具备很好的平台移植性，可以部署在不同的系统下，如主流的的 Windows、Linux 等操作系统下，这对 HDFS 的推广起到了很大的作用。

2.4 MapReduce 分布式并行编程模型

MapReduce 是 Google 公司发明的分布式并行编程模型，也是一个处理和生成超大规模数据集的算法模型的相关实现。MapReduce 架构的程序可以实现在大量普通配置的设备上实现分布式并行计算，通过网络通信与传输，将海量数据进行分片后分发到节点服务器进行分析与处理[9]。该模型很容易理解与实现，但需要编写符合自定义需求的业务程序确实困难的。MapReduce 支持多语言编程，可以通过 Java、Python 等主流的编程语言进行分布式编程，丰富了开发者的开发选择。图 2-2 给出了 MapReduce 架构图。

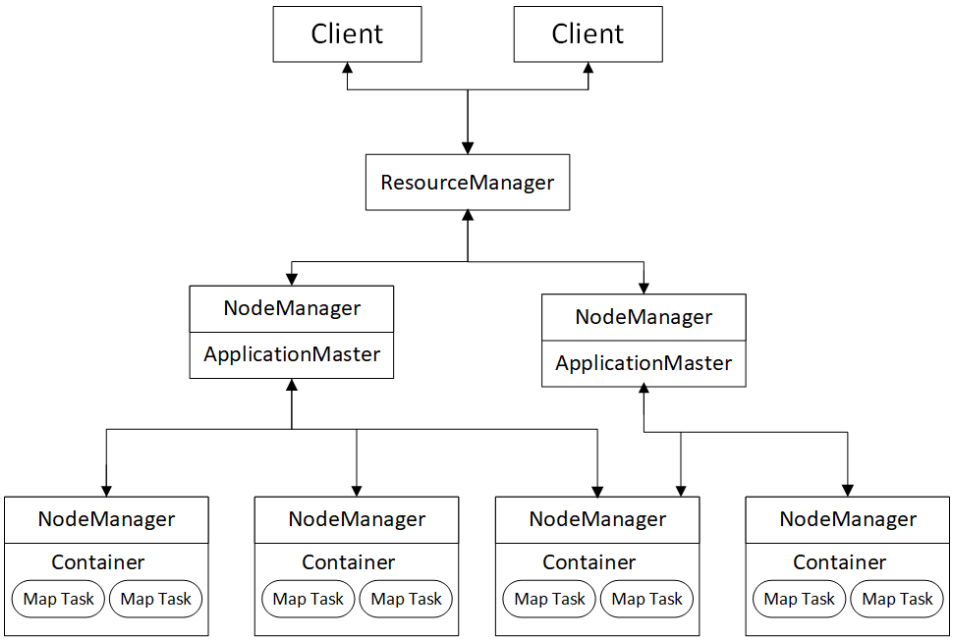


图 2-2 MapReduce 架构图

从 MapReduce 自身的命名特点可以看出，MapReduce 由两个阶段组成：Map 和 Reduce。开发者只需编写 map()函数和 reduce()函数，将自己的业务处理逻辑在 map()函数和 reduce()函数实现即可实现分布式编程。Map 和 Reduce 两个阶段在处理用户业务逻辑时输入与输出通过键值的对数据进行处理。该模型运行在 Hadoop 云计算平台上，所以数据的分析计算能力与 Hadoop 云计算平台的服务

节点数量有关，Hadoop 集群中的服务节点越多分析处理的速度越快。图 2-3、2-4 给出了 MapReduce 数据处理过程。

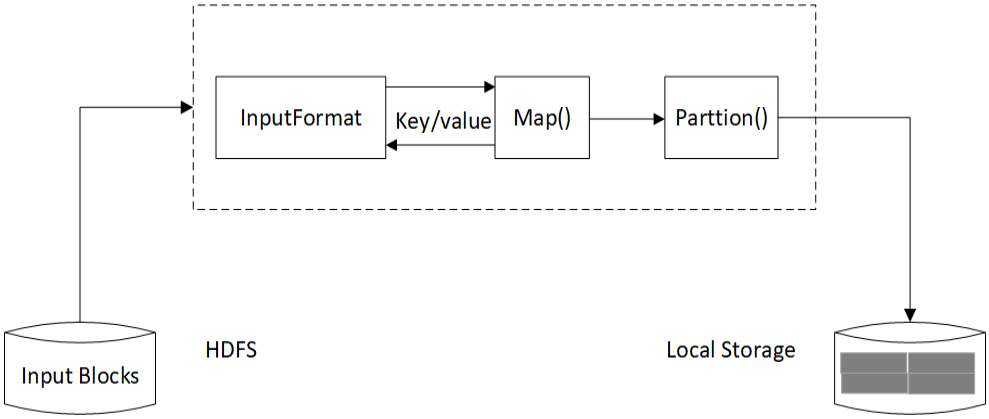


图 2-3 Map 阶段数据处理过程

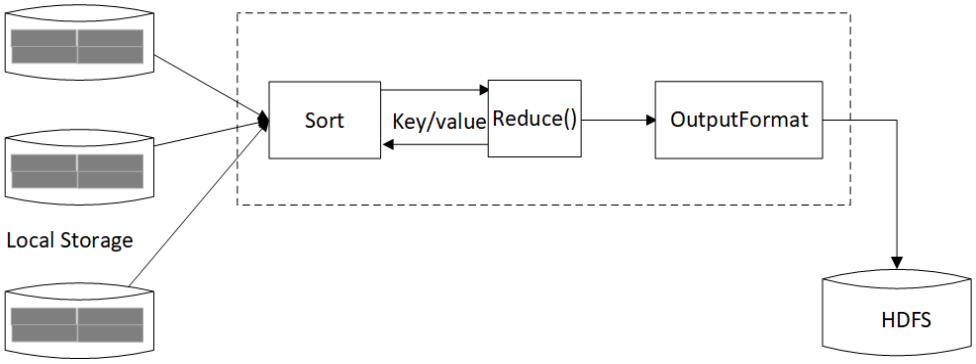


图 2-4 Reduce 阶段数据处理过程

MapReduce 模型的出现使人们对数据的看法有了很大的改变，数据不再是冷冰冰的存放在存储设备上，它是可以说话的，通过对大量数据的分析处理，可以得到潜藏在数据中的一些规律，我们称为黄金数据，通过这些数据使各行业找出优势与不足或者潜藏的规律等等，使人们获益。

2.5 Apriori 关联规则算法原理

Apriori 算法属于关联分析中的算法，是 Agrawal 在 1993 提出来的[10]。作为数据挖掘中经典的关联规则算法 Apriori 算法它的应用领域非常的广泛，如：网络安全领域、移动通信领域、商城推荐系统、高校图书管理系统等。Apriori 的核心是基于频集的递归关联算法，算法的基本思想是首先找出所有的频繁项集，频繁项集与用户定义的最小支持度比较，小于最小支持度的频繁项集将被剔除，

由筛选后的频繁项集产生包含集合项的关联规则，通过迭代方式不停的计算，直到计算出候选集为空时候停止[11-12]。

2.6 JSP 技术

JSP 技术由 Sun 公司开发，是一种跨平台基于 B/S 技术的动态网页的技术，它是一种服务端的脚本语言，在静态页面中嵌入 Java 代码片段，再由 Web 服务器中的 JSP 引擎来进行编译并执行嵌入的 Java 代码片段，生成的页面信息返回给客户端，最大的好处就是开发效率高。JSP 可以使用基于 MVC 的 Web 架构，通过 MVC 的 Web 架构，可以弱化各个部分的耦合关系，并将业务逻辑处理与页面以及数据分离开来，这样当其中一个模块的代码发生改变时，并不影响其他模块的正常运行，所以基于 MVC 的 Web 架构更适应于大型应用开发的潮流。

2.7 MySQL 技术

MySQL 是 Oracle 公司旗下的产品，是互联网中最受用户欢迎的关系型数据库管理系统，是一个真正多用户、多线程的数据库服务器，具有强大的功能、灵活性、丰富的编程 API，由于体积小、速度快、免费的特性成为广泛的被人们使用。MySQL 自称采用 C 语言开发，具备很强的可移植性，支持几乎所有的操作系统。

2.8 本章小结

本章节简单介绍了数据挖掘和 Hadoop 云平台的产生和发展，从信息时代如何处理海量数据方面介绍了数据挖掘技术和 Hadoop 云平台面对海量数据的机遇与挑战。接着详细介绍了 Hadoop 下的两大核心内容（HDFS 和 MapReduce）和关联规则算法 Apriori。阐述 HDFS 和 MapReduce 的体系结构、运行原理，阐述 Apriori 算法的基本原理。最后简单介绍了 JSP 技术与 MySQL 技术。

3 基于 MapReduce+Apriori 数据挖掘模式设计

3.1 Apriori 算法的并行化实现

传统的挖掘算法 Aprioir 无论运算效率、运算能力都不能满足人们的需求，本文第 2 章中介绍的基于 Hadoop 云平台的数据挖掘模式给出了对数据挖掘的新思路。通过 MapReduce 模型实现 Apriori 算法的并行化，从而解决传统 Apriori 挖掘效率不足的问题。另外，通过 MapReduce 模型很好地解决了海量数据的存储问题，同时基于 Hadoop 云平台还具有负载均衡与强大的容错能力。

基于 MapReduce+Apriori 并行挖掘算法实现的挖掘模式图 3-1 所示。

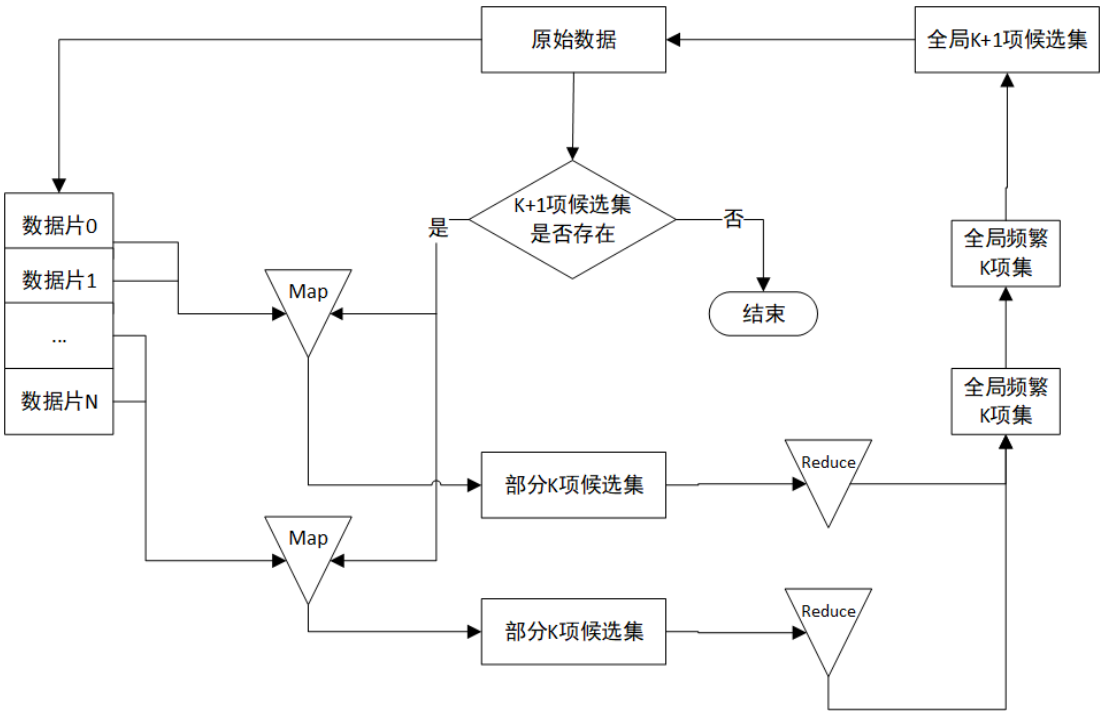


图 3-1 MapReduce+Apriori 并行数据挖掘模式设计

基本思想是：首先利用 MapReduce 分布式编程模型的键值对作为原始数据的输入，通过 map()函数功能对数据进行分块处理，并将分块的数据分发到各个节点上进行计算，然后通过 Apriori 算法对每个节点上的块数据进行如下处理：

（1）通过 Map 阶段对数据进行分割，生成候选项集 C_1；

（2）通过 Reduce 阶段对分块数据集与原始数据进行对比生成频繁项集 L_1；

（3）通过 Map 阶段由频繁项集 L_1 与原始数据进行对比，生成候选项集 C_2

（4）通过 Reduce 阶段计算出支持度，并通过计算的支持度与用户输入的最小支持度相比较，得到 2-项非空频繁项集 L_2；

（5）循环步骤 3-4，生成候选项集 C_k（k>=2），如果 C_k 小于最大迭代次数 Lk 则继续执行，否则 K++，如此迭代直到产生 C_k 项候选集，C_k 项候选集与原数据集对比，如果还存在 C_k 项候选集，则继续执行；如果不存在，则最终得到 K-1 候选集；

由于 MapReduce 模型的运行机制，上述过程每个节点在 Map 阶段对分块数据集进行计算，然后该模型的 Reduce 阶段获取每个节点 Map 阶段产生的 K 项候选集支持数并计算产生得到全局的支持数，通过得到的全局支持数计算出 K 项频繁项集。如果当前计算的数据分块全局的 K 项频繁项集出来后，每个节点会继续执行 Map 阶段处理下一份数据分块，则通过循环的方式，直到架构处理完成所有的数据分块。

3.2 MapReduce+Apriori 并行数据挖掘模式设计

通过 Hadoop 云平台对数据进行处理时，数据结果以文件形式替代传统 Apriori 使用数据库存储数据结果，数据结果存储在 HDFS 中。通过 MapReduce 模型对数据处理的方式，采用键值对的方式对文件进行读取和写入，所以在这些文件中，每一行的行号可以标记一条数据，每一行的数据通过 tab 键对属性条目进行隔开。当对数据进行关联分析作业时，每一次迭代将会启动一次 MapReduce 作业，Map 阶段对数据进行读入后按数据的属性条目分开，将候选集与该候选集出现的次数作为 Key、Value 的值并通过一个上下文将数据输出给 Reduce 阶段。Reduce 阶段通过合并候选项集的值，并通过计算得到支持度，通过上下文写入到文件中并存储在 HDFS，得到频繁项集。Map 阶段与 Reduce 阶段键值对（Key，Value）的设置如表 3-1 所示。

表 3-1 Map 与 Reduce 函数中 key/value 的类型和值的设置

输入/输出	Map 函数	Reduce 函数
输入:<key,value>	key:行号 value:一行数据	key:候选项集 value:1
输出:<key,value>	key:候选项集 value:1	key:频繁项集 value:支持度

Map 阶段主要用来计算候选项 K-项集的集合 Ck 中的每个候选集是否在一条数据记录中存在，如果存在则输出<候选项集,1>键值对，否则不输出[13]。Reduce 阶段主要来统计相同键的值，并将统计的值进行计算得这个候选项集的支持度，最后将<候选项集,支持度>键值对作为输出到 HDFS 文件中。通过 Map 和 Reduce

两阶段的并行处理，通过循环将所有候选项合并计算输出后就得到频繁 K-项集与对应的支持度，大大的提升了 Apriori 算法的分析效率。

MapReduce+Apriori 核心实现代码：

```
while(nivelAtual <= maxNivel){
    countMinSup = 0;
    //通过迭代方式生成频繁项集，并写到 HDFS
    isCompleted=apriori.agrupamento(inFile,outFile+nivelAtual);
    if(!isCompleted)
        System.exit(0);
    if(countMinSup == 0) break;
    nivelAtual++;
    //生成候选项集
    produtoCartesiano(conjAtual,conjItens.keySet(),nivelAtual);
 }
```

Apriori 主运行类函数用于频繁一项集的计算，通过 MapReduce 作业在 Map 阶段中，数据记录的任意行为 key，而行对应的值为 value，通过键值对的关联输出格式为<value,1>。在 Reduce 阶段，将相同 key 的 value 值进行相加、计算得到某个所有 value 值对用的支持频度，如果该支持频度不小于最小支持频度则以<value,支持频度>的键值对输出到 HDFS 上。

Map 阶段具体的执行过程代码：

```
map(LongWritable key, Text value, Context context){
    String line = value.toString();
    String [] tokens = line.split("[ \t]");     //数据按空格和制表符切割
    List<Integer> valores = new ArrayList<>();
    …
    for(String s : tokens)                          //循环找出频繁项
        StringBuilder strConj = new StringBuilder();
       int k = 0;
       for(int n : conj){
            for(int valor : valores){…}
            strConj.append(n);
            strConj.append(',');
       }
      if(k == Apriori.nivelAtual)
      context.write(newText(strConj.substring(0,strConj.length()-1).toString()),ne
w Text("1"));,
```

```
}
Reduce 阶段具体执行过程代码：
public void reduce(Text key,Iterable<Text> values,Context context){
  ...
  for(Text value :values){
    sum += Integer.valueOf(value.toString());//频繁值统计
     }
  ...
   double suporte = (double)sum/Apriori.lineNumber;//计算支持度
   if(suporte >= Apriori.minSup){
      ...
      context.write(key,new Text(String.valueOf(suporte)));
      ...
   }}
```

3.3 MapReduce+Apriori 参数优化策略

基于 Hadoop 云平台的 MapReduce+Apriori 并行挖掘算法模型虽然在执行速度上较传统单机系统有了很大的提升，但其挖掘精度和效率也受到 MapReduce 设置与 Apriori 算法参数取值的影响，在运行前一般要考虑模型优化问题。

（1）在 MapReduce 编程阶段，在处理海量数据期间通过自定义 Partitioner 类继承 Hadoop 的 Partitioner 类，Partitioner 类主要作用就是将 Map 阶段的结果发送给 Reduce 阶段，因此，为了体现速度，该类执行越简单越好；另外，自定义 Partitioner 类还能通过自定义业务逻辑实现对 Map 阶段数据的分配控制，实现整个集群的负载均衡。

（2）Apriori 算法参数优化。Apriori 算法进行关联分析时，需要输入 minSup、minConf、maxNivel 三个参数，参数不同对数据关联分析后的结果数据存在差异，所以数据分析师需要根据数据量、维度等信息合理分配这些参数。

（3）Apriori 算法参数设置。Apriori 算法的参数在实验是手动进行设置的，通过对数据的关联分析得出，将 minSup 下界设置为过大则无法找到满足条件的关联规则，maxNivel 参数设置过大则关联分析耗时非常长。

3.4 本章小结

本章节首先分析和介绍了 Apriori 并行化实现的基本思想和基于 Hadoop 云计算平台的并行实现的思路；其次，阐述了 MapReduce+Apriori 算法并行实现的

基本算法设计与数据存储方式；然后，分析和介绍了 MapReduce+Apriori 参数优化策略；最后，介绍了挖掘分析过程中对 Apriori 算法参数的设置。

4 基于 Hadoop 的出租车营运行为分析系统设计

4.1 系统流程设计

基于 Hadoop 的出租车营运行为分析，其面对的是全市的出租车所采集的数据。利用出租车 GPS 实时向出租车调度中心提供的时间、位置、载客状态、方向等信息,通过Hadoop平台实现并行的关联规则算法Apriori对数据的挖掘分析，并将结果数据存储到数据库,通过 Web 站点将挖掘分析的结果数据通过网页报表的形式展现，为出租车调度中心提供支持和服务。

系统设计为了简化用户的操作、提升系统的体验，用户能够直接通过网页表单提交的方式提交数据挖掘分析任务，系统整体结构如图 4-1。

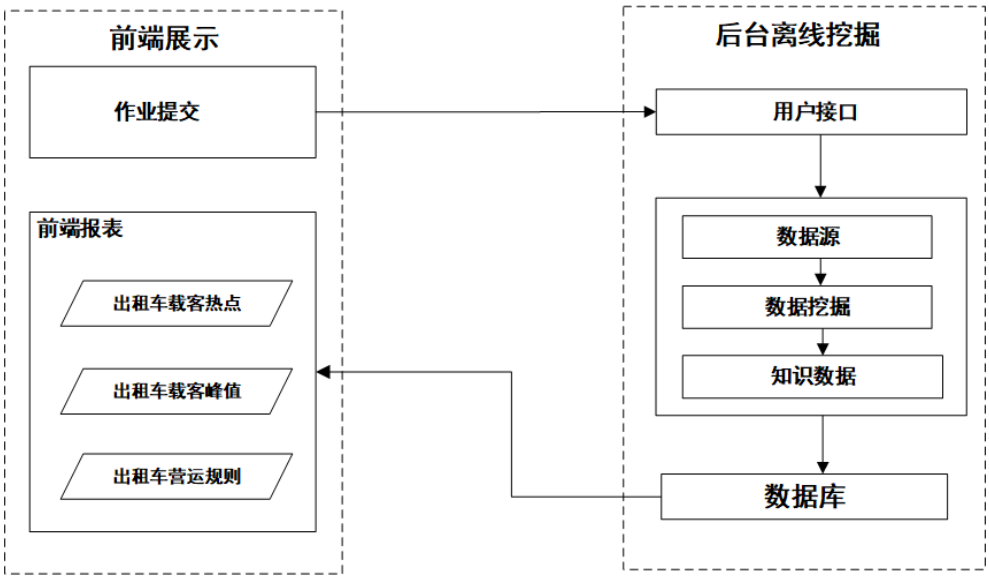


图 4-1 出租车营运行为分析整体架构图

基于 Hadoop 的出租车营运行为分析系统挖掘流程如下：

（1）全市的出租车 GPS 信息上传到 HDFS 分布式存储中；

（2）系统后台开放用户接口，实时监听接受前端的作业提交；

（3）用户通过本地浏览器使用作业提交功能，发起挖掘任务；

（4）一旦后台的用户接口检测到用户的作业提交，检测合法性，通过则发起 MapReduce 作业任务通过，否则返回用户一个错误代码；

（5）通过 MapReduce+Apriori 模型对数据进行关联分析，将结果存储到数据库中；

（6）前端系统通过数据库获取挖掘后的结果，通过百度地图或网页报表的

形式将结果展现在用户本地浏览器上。

4.2 系统功能模块设计

系统对出租车营运数据的分析可概括为三个主要模块，数据挖掘前的预处理、数据挖掘、数据挖掘后的处理。基于这三个步骤，本文提出了三层结构的出租车营运行为数据的挖掘系统。系统基础模块如下图 4-2 所示。

（1）数据层

数据层分为两种存储形式，分别是 HDFS 存储和 MySQL 存储。HDFS 主要用于存放出租车 GPS 设备产生的数据、数据挖掘产生的中间数据等，MySQL 主要存储数据挖掘的结果数据。

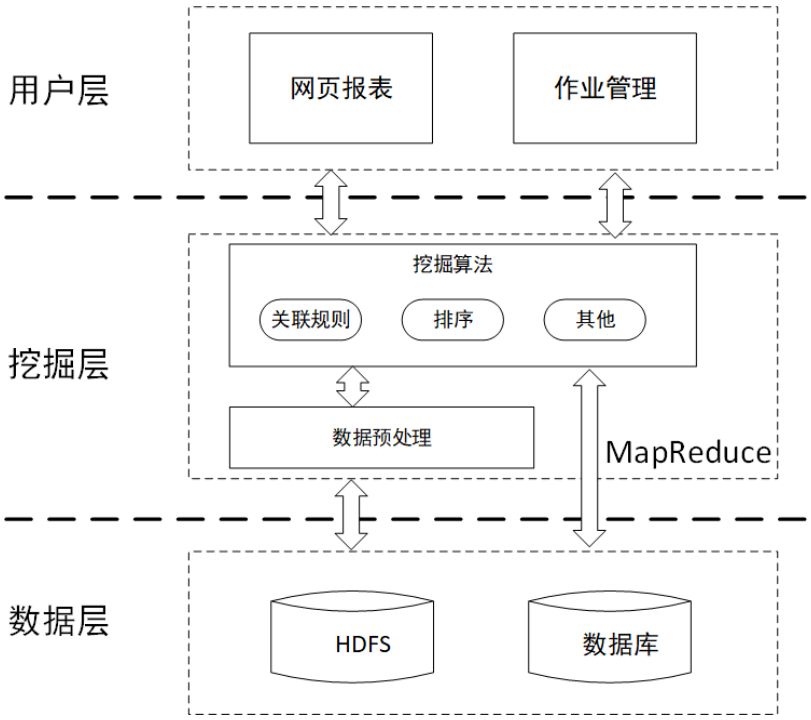


图 4-2 出租车营运分析系统模块设计图

（2）挖掘层

挖掘层主要包含数据预处理和基于 MapReduce+Apriori 的并行关联规则挖掘算法模块。首先，将出租车的源数据进行预处理，筛选掉 GPS 设备在提交数据时的噪声数据，保证挖掘分析过程中数据的质量。另外，针对不同的数据挖掘需要不同的数据，为了快速的对数据进行关联分析，必须提前对数据进行精心的筛选处理。最后，通过关联规则算法 Apriori 针对不同的数据集进行关联分析，将关联结果与实际数据相匹配得到最终的结果。

（3）用户层

用户层主要实现用户的作业提交和数据挖掘结果的报表功能展示。作业提交实现对用户文件的提交以及作业任务的提交，报表功能实现对作业结果的可视化展示，如折线图、地图、散点图等。

4.3 系统体系结构设计

系统的体系结构为三层浏览器/服务器结构（B/S 结构），该架构的优点是普通用户只需通过 Web 浏览器和网络，即可使用客户端的所有功能，不需要安装其它软件，减轻服务器负担，B/S 架构设计如图 4-3。

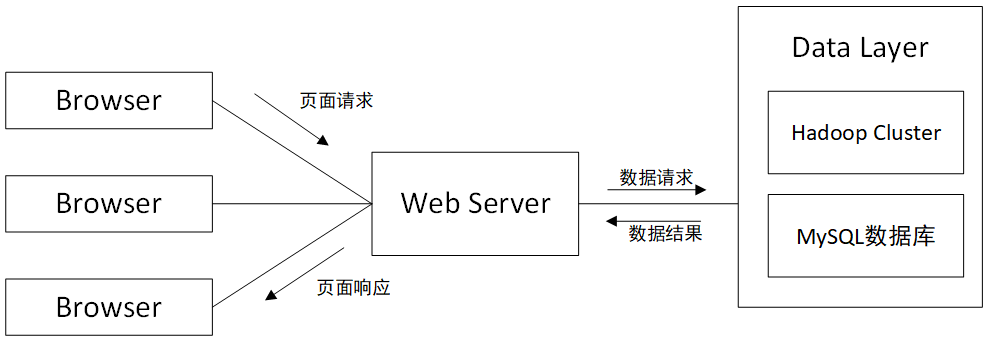


图 4-3 B/S 的架构设计

使用 B/S 架构维护和升级方式简单，客户端即浏览器不需要任何维护，只需要针对服务器进行维护即可。该架构没有平台限制，可以运行在 Linux 和 Windows 下。

4.4 HDFS 云存储设计

HDFS 是数据存储的中心，在 Hadoop 集群部署时需要考虑到 HDFS 在设计之出的不足，通过合理配置和其它技术手段进行弥补，提供实时的服务，具体如下：

（1）HDFS 高可用功能

HDFS 是由 NameNode 和 DataNode 两种角色组成，而默认情况下 NameNode 在整个集群中有且只有一个，是整个分布式文件系统的大脑，一旦出现问题将造成数据无法访问存在单点故障。HDFS 的高可用功能就解决了上述的问题，通过在配置文件中启用 HDFS 高可用功能，选择同一集群中的两个节点配置成为 NameNodes，在任何时候，一个节点 active 状态，另一个处于 standby 状态。处于 active 状态的节点响应客户端的操作，而处于 standby 状态的节点仅仅作为

一个备用节点，一旦 active 节点出现故障则可手动将备用节点状态设置成 active，保证分布式文件系统的正常服务。HDFS 高可用功能参数配置如表 4-1 所示。

表 4-1 HDFS 高可用功能参数配置

HDFS 高可用功能参数配置

<property>
<name>dfs.ha.namenodes.YGSDMI1</name> //NameNode 集群的唯一标识
<value>NN1,NN2</value> //最多只允许两台 NameNode
</property>
<property>
<name>dfs.namenode.rpc-address.YGSDMI1.NN1</name>
<value>NN1:8020</value> //NameNode 通信地址
</property> //还需要配置 NN2 的地址，简化操作不做配置
//还需要配置 NN2 的通讯地址
<property>
<name>dfs.namenode.http-address.YGSDMI1.NN1</name>
<value>NN1:50070</value> //NameNode 管理地址
</property> //还需要配置 NN2 的地址，简化操作不做配置
//还需要配置 NN2 的管理地址

（2）HDFS 元数据同步功能

HDFS 通过启动 HA 功能可以解决单节点故障问题，但无法解决两个 NameNode 数据同步的问题。由于 NameNode 将分布式文件系统元数据保存在本地，维护整个集群的文件信息，一旦 active 的 NameNode 出现故障，standby 的节点无法获取元素即使状态变成 active 也无法管理维护整个 HDFS 集群。

HDFS 通过启用 JournalNodes 功能解决了元数据同步的问题，NameNode 通过与一组 JournalNodes 的服务进程相互通信。当 active 状态的 NameNode 的元数据出现任何改变，会告知 JournalNodes 进程。standby 状态的 NameNode 将会通过 JournalNodes 进程将元数据同步到本地，与 avtive 状态的 NameNode 元数据保持一致，HDFS 元数据同步功能配置如表 4-2 所示。

表 4-2 元数据同步功能参数配置

元数据同步功能参数配置
<property>
<name>dfs.namenode.shared.edits.dir</name>
//使用 JournalNode 方式存放元数据
<value>qjournal://node1:8485;node2:8485;node3:8485/YGSDMI1</value>

续表 4-2

</property>
<property>
<name>dfs.journalnode.edits.dir</name>
//元数据存放的位置
<value>/data/hadoop-2.7.3/tmp/journal</value>
</property>
<property>
<name>dfs.qjournal.write-txns.timeout.ms</name>
//元数据写入的超时时间，单位毫秒
<value>60000</value>
</property>

（3）HDFS 自动故障转移

通过以上两点对 HDFS 的功能配置，实现了高可用与元数据同步的问题，但 NameNode 无法通过自动化迁移到备用的 NameNode 节点中，HDFS 在部署的时候需要增加两个组件：ZooKeeper 和 ZKFailoverController（ZKFC）进程，HDFS 自动故障转移配置如表 4-3 所示。

表 4-3 HDFS 自动故障转移

HDFS 自动故障转移参数配置
<property>
<name>dfs.ha.automatic-failover.enabled.YGSDMI1</name>
<value>true</value> //开启自动故障转移
</property>
<property>
<name>dfs.client.failover.proxy.provider.YGSDMI1</name> //故障转移实现类

```
    <value>
        org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider
    </value>
</property>
<property>
    <name>dfs.ha.fencing.methods</name>       //隔离机制的实现方式
```

续表 4-3

```
        <value>
        sshfence
    shell(/bin/true)
        </value>
</property>
<property>
<name>dfs.ha.fencing.ssh.private-key-files</name>
    <value>/root/.ssh/id_rsa</value>          //使用隔离机制时需要免密码登陆
</property>
```

（4）HDFS 副本策略

通过配置 HDFS 的副本策略，提高系统可靠性，实现任务在执行 MapReduce 任务的负载均衡和提高访问的效率，减少网络传输的开销。HDFS 默认的副本数是 3 份，客户端上传数据到 HDFS 将随机选择一个负载较小的 DataNode 节点存放数据，根据集群配置的副本策略，则从该数据节点所在的机架中选择一个合适的 DataNode 节点存放一份副本，最后一份副本将选择一个远程机架节点存放。如果 HDFS 配置的副本策略超过 3 份，HDFS 集群则超出部分副本数随机选择若干集群负载较小的 DataNode 节点存放，HDFS 副本策略配置如表 4-4 所示。

表 4-4 HDFS 副本放置策略配置

HDFS 副本放置策略配置

```
    <property>
      <name>dfs.replication</name>
      <value>3</value>          //数据副本 3 份，可根据集群大小动态调整
    </property>
```

4.5 MySQL 关系数据库设计

系统选择 MySQL 关系型数据库作为数据挖掘结果的属性的存储，系统主要分为 GPSPoint、TimeCount、TaxiRules 三个表，分别表示出租车载客热点表、出租车载客峰值表和出租车营运规则分析表。

GPSPoint 表有 id、lit、dim、count 四个字段，其中 id 字段为整数类型且值是自增，lit 字段字符类型用于存储经度数据、dim 字段字符类型用于存储维度数据、count 字段字符类型用于存储载客数量，表结构如表 4-5 所示。

表 4-5 GPSPoint 表结构

名	类型	长度	小数点	允许空值	主键
id	int	10		是	是
lit	varchar	20		是	
dim	varchar	20		是	
count	varchar	10		是	

TimeCount 表有 id、count 两个字段，其中 id 字段为整数类型且值是自增，count 字段存储每个时间段的载客统计值，表结构如表 4-6 所示。

表 4-6 TimeCount 表结构

名	类型	长度	小数点	允许空值	主键
id	int	10		是	是
count	varchar	10		是	

TaxiRules 表有 id、Sup、Confidence 三个字段，其中 id 字段为整数类型且值是自增，Sup 字段存储每条数据的支持度，Confidence 字段存储每个数据的置信度，表结构如表 4-7 所示。

表 4-7 TaxiRules 表结构

名	类型	长度	小数点	允许控制	主键
id	int	10		是	是
Sup	double	20		是	
Confidence	double	20		是	

4.6 系统软硬件配置

系统的软件配置如下：

（1）集群操作系统：RedHat Enterprise Linux 6.5

（2）编程语言：JAVA

（3）编程工具：Eclipse Mars 2015

（4）Hadoop 集群相关软件及版本：Hadoop 2.7.3，zookeeper3.4.5，JDK1.7

（5）数据库服务器：MySQL 5.6

（6）网站开发：JSP 技术

（7）WEB 服务器：Apache Tomcat 8

（8）虚拟机软件：VMware Workstation 11

系统的硬件配置如下：

（1）DELL 服务器：1 台

（2）CPU：Intel(R) Xeon CPU E5-2603 v2 @ 1.60GHz

（3）内存：64GB

（4）硬盘：2TB

4.7 本章小结

本章节首先介绍了系统的流程设计，阐述了系统数据挖掘分析的主要步骤；然后介绍了系统的功能模块，说明了系统各个功能模块的功能，接着详细介绍了 HDFS 云存储的设计思路与设计方法，增强 HDFS 的安全性与可靠性，并简单介绍了 MySQL 数据库，以及表的结构设计；最后介绍了系统构建中所需要的软硬件配置。

5 系统实现与结果分析

5.1 系统性能评价指标

针对单机环境下传统数据挖掘算法在分析处理大规模数据时存在计算性能低、拓展性弱、内存消耗高等问题，提出通过 MapReduce 模型并行实现 Apriori 关联规则算法的方案。以速度（时效性）为性能指标，通过 Hadoop 云平台的分布式计算对海量数据进行快速处理，通过合理的配置数据的副本数以提高 Hadoop 云平台在处理数据时的效率，减少不必要的网络开销，并且根据业务数据规模动横向扩展集群，以加快数据处理的速度。在保证数据快速的处理的同时必须以精度（正确性）为性能指标，保证数据在处理过程中的准确性。

5.2 实验数据与样本选择

5.2.1 数据介绍

本文所使用的数据源是 2010-2011 年江苏省南京市出租车的营运数据，数据包含了 7726 辆出租车，共包含 33042225 条记录。

（1）数据格式

数据主要内容通过车载 GPS 设备产生的车辆和载客相关的实时信息，数据字段包含出租车 ID 字段、数据收集时间、经度、维度、速度、方向和载客状态字段，数据集的格式与部分内容如表 5-1 所示。

表 5-1 营运数据部分数据源

VehicleId	Time	Longitude	Latitude	Speed	Direction	State
806584008859	2010-09-01 00:01:42	118.8550	31.9394	0	0	0
806770549907	2010-09-01 00:01:45	118.7490	32.0891	30	6	1
806770943693	2010-09-01 00:01:46	118.8562	32.0942	26	318	1
806914743134	2010-09-01 00:01:48	118.8368	31.9316	0	0	0
806451847007	2010-09-01 00:01:51	118.7955	32.0393	20	10	0
806451847099	2010-09-02 12:01:49	118.6556	32.0791	30	245	1
806451847056	2010-09-02 12:01:51	118.5397	32.0599	50	60	1

（2）数据各个字段的含义

表 5-1 中各字段的含义如下：

VehicleId：出租车标识符，出租车在出租车管理部门中唯一的编号。

Time：数据收集时间，时间格式为 YYYY-MM-DD hh:mm:ss。

Longitude/ Latitude：经度/维度，以地球表面某点随地球自转所形成的轨迹。

Speed：速度，以千米/小时计算。

Direction：方向，与正北方向的夹角度数。

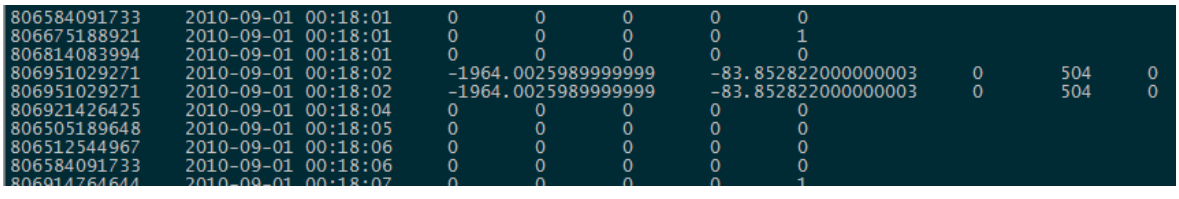
PassengerState：载客状态，1 为载客，0 为空车。

5.2.2 数据预处理

针对本文的研究，在挖掘分析时需要使用不同数据字段集，需要预先对原始数据进行相应的属性过滤，通过属性过滤将挖掘分析过程中使用不到的属性字段过滤掉，生成不同的样本数据；另外，还需要对原始数据集进行分析，并对部分属性字段进行数据调整等，具体步骤如下：

（1）由于 GPS 设备在通信过程中容易受到周围环境的影响，如天气、建筑物、隧道等因素，会产生不正确的数据或影响 GPS 通信的中断，造成数据丢失。另外，GPS 设备长期的使用会出现设备老化，影响 GPS 设备定位精度，以上情况都有可能导致 GPS 数据中存在噪声数据。

（2）经纬度异常：通过对江苏省南京市的经纬度分析，南京市位于江苏省西南部，北纬 31 度 14 分至 32 度 37 分，东经 118 度 22 分至 119 度 14 分，所以，不满足南京市所在的经纬度的数据均视为不合格的数据，如图 5-1 的数据记录就超过南京市的地理位置，数据不符合要求，不能用于挖掘分析。



806584091733	2010-09-01 00:18:01	0	0	0	0	0		
806675188921	2010-09-01 00:18:01	0	0	0	0	1		
806814083994	2010-09-01 00:18:01	0	0	0	0	0		
806951029271	2010-09-01 00:18:02	-1964.0025989999999	-83.852822000000003	0	504	0		
806951029271	2010-09-01 00:18:02	-1964.0025989999999	-83.852822000000003	0	504	0		
806921426425	2010-09-01 00:18:04	0	0	0	0	0		
806505189648	2010-09-01 00:18:05	0	0	0	0	0		
806512544967	2010-09-01 00:18:06	0	0	0	0	0		
806584091733	2010-09-01 00:18:06	0	0	0	0	0		
806814764644	2010-09-01 00:18:07	0	0	0	0	1		

图 5-1 部分不正确格式数据

5.3 系统前台与后台的实现

5.3.1 系统前台实现

本文使用 JSP 技术开发前台系统，通过 JavaScript 调用百度地图、简数 CDN 图表提供的 API 接口，通过读取数据库的数据实现对数据挖掘结果可视化展示。

（1）通过 JavaScript 调用百度地图 API 构建出一张中国地图，通过 JavaScript 创建的地图实例，设置南京市的经纬度，实现对南京市地图的展现。系统实现代码如下：

```
<script type="text/javascript">
    var map = new BMap.Map("container");                    // 创建地图实例
    var point = new BMap.Point(118.8109, 31.973);
    map.centerAndZoom(point, 15);              // 初始化地图，设置南京市坐标和地图级别
    map.enableScrollWheelZoom();                            //允许滚轮缩放
    var points =[{"lng":118.450, "lat":31.9373, "count": 4},
                 {"lng":118.4830, "lat":32.1011, "count": 301,}
                 {"lng":119.1114, "lat":31.0331, "count": 5,];
    heatmapOverlay = new BMapLib.HeatmapOverlay({"radius":20});
    map.addOverlay(heatmapOverlay);
    heatmapOverlay.setDataSet({data:points,max:100});
    function openHeatmap() { heatmapOverlay.show();}      //显示热力图
    function closeHeatmap(){ heatmapOverlay.hide();}      //关闭热力图
    closeHeatmap();
    //设置热力图颜色、范围等参数
    function setGradient(){
        var gradient = {};
        var colors = document.querySelectorAll("input[type='color']");
        colors.forEach(function(ele){
            gradient[ele.getAttribute("data-key")] = ele.value; });
        heatmapOverlay.setOptions({"gradient":gradient});
    }
    //判断浏览器是否支持 canvas
   function isSupportCanvas(){
        var elem = document.createElement('canvas');
        return !!(elem.getContext && elem.getContext('2d'));
    }
</script>
```

系统界面如图 5-2 所示。

图 5-2 南京地图界面的实现

（2）通过 JavaScript 调用简数 CDN 的图表接口，实现对数据的展现，实现核心代码如下：

```
<script type="text/javascript">
    $(function() {
        $('#container').highcharts(                    //实例图表
                {
                    xAxis : {                          //设置 x 轴
                        categories : [ '00', '01', '02', '03', '04', '05',
                                '06', '07', '08', '09', '10', '11', '12', '13',
                                '14', '15', '16', '17', '18', '19', '20', '21',
                                '22', '23', '24' ]
                    },
                    //设置 y 轴
                    yAxis : {plotLines:[{value:0,width:1,color:'#808080'}]},
                    tooltip : {valueSuffix : '人'},
                    series : [{ name : '2010-09-01',data : []},
                            { name : '2010-09-02',data : []} ]
                });
    });
</script>
```

系统图标界面的实现如图 5-3 所示。

图 5-3 图表界面的实现

5.3.2 系统后台 Hadoop 集群的配置

本文系统后台使用 Hadoop 云平台实现对数据挖掘分析，通过 8 台计算机来搭建 Hadoop 集群，8 台节点安装使用 RedHat Enterprise 6.5 操作系统。在第四章节中介绍过 HDFS 云存储的设计，通过配置 HDFS 高可用（HA）、自动故障转移、元数据同步功能，解决 NameNode 的单点故障,在 MapReduce 中，整个集群只有一个 ResoucerManager，可以有多个 NodeManager[14]。所以，8 台机器中，两台作为 NameNode 节点（其中一台用作 HA 的备用节点），三台作为 DataNode 和 JournalNode 节点（作为元数据同步节点），最后三台作为 Zookeeper 仲裁节点实现集群服务的自动故障切换功能。集群中 8 台机器的设置如表 5-2 所示。

表 5-2 集群角色配置表

机器名	IP 地址	角色
namenode1	192.168.100.1	NameNode,ResoucerManager,ZKFC
namenode2	192.168.100.2	NameNode,ZKFC
datanode1	192.168.100.3	DataNode,JournalNode,NodeManager
datanode2	192.168.100.4	DataNode,JournalNode,NodeManager
datanode3	192.168.100.5	DataNode,JournalNode,NodeManager
datanode3	192.168.100.5	DataNode,JournalNode,NodeManager

续表 5-2

机器名	IP 地址	角色
zookeeper1	192.168.100.6	QuorumPeerMain
zookeeper2	192.168.100.7	QuorumPeerMain
zookeeper3	192.168.100.8	QuorumPeerMain

以上对八台机器的角色安排，在开始配置集群前需要保证所有机器的网络正常，可以在各计算机之间通过 ping 命令进行网络检测。

（1）JDK 安装与配置

考虑到系统的稳定性与编程开发，都选用了 64 位的 JDK 1.7 版本。通过从 Oracle 官网下载获取 jdk-7u79-linux-x64.tar.gz 压缩包，将 JDK 压缩包上传到 Linux 服务器，在根目录建立一个/data 目录，用于存放所有的数据文件。通过 tar 命令对文件解压至/data 目录下，最后通过配置/etc/profile 文件末尾添加配置 JAVA_HOME 环境变量，实现对 java 命令的直接访问。以上所述配置需要在所有机器上都执行，具体配置如下表 5-3：

表 5-3 JDK 具体配置

JDK 具体配置
[ygsdmi@zk1 ~]$mkdir -p /data
[ygsdmi@zk1 ~]$tar zxvf /root/jdk-7u79-linux-x64.tar.gz –C /data
[ygsdmi@zk1 ~]$vim /etc/profile
export JAVA_HOME=/data/jdk1.7.0_79 //编辑 profile 文件添加
export PATH=$PATH:$JAVA_HOME/bin

（2）SSH 免密码配置

NameNode 需要通过 SSH 协议与其它机器进行交互，通过 SSH 协议连接其它节点需要密码授权，在集群管理中不能自动完成，为了节点间的通信变得简单，通过配置 SSH 免密码登入，实现免密码交互。

在两台 NameNode 上使用 ssh-keygen 命令生成密钥，通过 ssh-copy-id 命令将生成的密钥串拷贝到其它所有节点。通过上述配置，集群中 NameNode 通过 SSH 协议连接到其它节点都不需要输入密码就可以执行命令了。

（3）Hadoop 配置

通过第四章对 HDFS 云存储的设计，所以选择使用 Hadoop 2.0 分支版本，选择目前官方最新的稳定版本 Hadoop2.7.3。由于 Apache 官方没有提供 64 位的

Hadoop 软件包，只提供了源码包，所以需要对源码进行编译后生成。通过编译生成的软件包 hadoop-2.7.3.tar.gz。下面系统需求对 Hadoop 进行配置过程介绍。

将编译好的 Hadoop 文件通过 tar 命令解压至/data 目录，并在 profile 文件下配置 HADOOP_HOME 环境变量，方便 Hadoop 命令的访问。以上配置需要在所有的 NameNode、DataNode 节点上执行，具体配置如表 5-4 所示：

表 5-4 Hadoop 环境变量配置

Hadoop 环境变量配置
[ygsdmi@zk1 ~]$tar zxvf /root/hadoop-2.7.3.tar.gz -C /data/
[ygsdmi@zk1 ~]$vim /etc/profile //环境变量文件
export HADOOP_HOME=/data/Hadoop-2.7.3 //添加环境变量
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin

在 HADOOP_HOME 下定位到 etc/Hadoop 目录，通过配置 hadoop_env.sh 文件指定系统的安装的 JDK，指定绝对路径。以上操作在 NameNode、DataNode 上都要执行。

在 HADOOP_HOME 下定位到 etc/Hadoop 目录，将所有 DataNode 的 IP 地址填写到 slaves 文件。

按表 5-5、5-6 中的配置方式，配置 HADOOP_HOME 下 etc/Hadoop 目录的 hdfs-site.xml、core-site.xml、mapred-site.xml 文件。由于在第四章 HDFS 云存储的设计中介绍过 HDFS 的配置，所以对 hdfs-site.xml 配置可参考第四章。以上步骤在 NameNode 和 DataNode 上都需要执行。

表 5-5 core-site.xml 配置

core-site.xml 配置
<property>
<name>fs.defaultFS</name>
<value>hdfs://YGSDMI1:8020</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/data/hadoop-2.7.3/tmp</value>
</property>

```
<property>
    <name>fs.viewfs.mounttable.default.link./YGSDMI1</name>
```

续表 5-5

```
    <value>hdfs://YGSDMI1:8020</value>
</property>
<property>
    <name>ha.zookeeper.quorum</name>
    <value>zk1:2181,zk2:2181,zk3:2181</value>
</property>
```

表 5-6 mapred-site.xml 配置

mapred-site.xml 配置
<property> <name>MapReduce.framework.name</name> <value>yarn</value> </property> <property> <name>MapReduce.admin.map.child.java.opts</name> <value>-Xmx2048m</value> </property> <property> <name>MapReduce.admin.reduce.child.java.opts</name> <value>-Xmx4096m</value> </property> <property> <name>mapred.map.child.java.opts</name> <value> -Xmx1024m </value> </property> <property> <name>mapred.reduce.child.java.opts</name> <value> -Xmx2048m </value>

```
</property>
```

从 Apache 下载获取 zookeeper 并上传到对应的节点上，通过 tar 命令将软件解压至/data 目录下，在解压后的目录下定位到 bin 目录下通过指定 zkServer.sh start 命令启动服务，通过 jps 命令可以对启动的进程进行查看，以上操作在所有 zookeeper 节点上都需要执行，具体配置如下所示：

```
[ygsdmi@zk1   ~]$source /etc/profile
[ygsdmi@zk1   ~]$tar zxvf /root/zookeeper-3.4.5.tar.gz -C /data/
[ygsdmi@zk1   ~]$cd /data/zookeeper-3.4.5/conf/
[ygsdmi@zk1 conf]$mv ./zoo_sample.cfg ./zoo.cfg
[ygsdmi@zk1 conf]$echo server.1=zookeeper1:2888:3888 >>./zoo.cfg
[ygsdmi@zk1 conf]$eecho server.2=zookeeper2:2888:3888 >>./zoo.cfg
[ygsdmi@zk1 conf]$eecho server.3=zookeeper3:2888:3888 >>./zoo.cfg
[ygsdmi@zk1 conf]$/data/zookerper-3.4.5/bin/zkServer.sh start
[ygsdmi@zk1 conf]$/data/zookerper-3.4.5/bin/zkServer.sh status
[ygsdmi@zk1 conf]$jps
```

最后，通过格式化 NameNode 和 ZKFC 为启动做准备，在格式化没有出错的情况下，需要通过两步骤将集群启动：

（1）在充当 JournalNodes 的节点上启动该进程，具体如下：

```
[ygsdmi@datanode1 ~]$source /etc/profile
[ygsdmi@datanode1 ~]$cd /data/Hadoop-2.7.3/sbin/
[ygsdmi@datanode1 ~]$/hadoop-daemon.sh start journalnode
[ygsdmi@datanode2~]$cd /data/Hadoop-2.7.3/sbin/
[ygsdmi@datanode2 ~]$/hadoop-daemon.sh start journalnode
[ygsdmi@datanode3 ~]$cd /data/Hadoop-2.7.3/sbin/
[ygsdmi@datanode3 ~]$/hadoop-daemon.sh start journalnode
[ygsdmi@datanode3 ~]$jps
```

（2）在 NameNode 上启动 ZKFC、NameNode、ResoucerManager 服务，并通过配置好的 SSH 免密钥登入远程启动所有的 DataNode，NodeManager 具体如下：

```
[ygsdmi@namenode1 ~]$source /etc/profile
[ygsdmi@namenode1 ~]$/data/Hadoop-2.7.3/sbin/
[ygsdmi@namenode1 ~]$./hadoop-daemon.sh start zkfc
[ygsdmi@namenode1 ~]$./hadoop-daemon.sh start namenode
[ygsdmi@namenode1 ~]$./hadoop-daemon.sh start namenode
```

```
[ygsdmi@namenode1 ~]$./hadoop-daemons.sh start datanode
[ygsdmi@namenode1 ~]$./start-yarn.sh start resoucermanager
[ygsdmi@namenode2 ~]$/data/Hadoop-2.7.3/sbin/
[ygsdmi@namenode2 ~]$./hadoop-daemon.sh start zkfc
[ygsdmi@namenode2 ~]$./hadoop-daemon.sh start namenode1
[ygsdmi@namenode2 ~]$jsp
```

5.4 出租车载客热点分析的实现

以时间、经纬、载客状态为数据统计和分类的基本单元。通过对南京全市两天里载客热点路段的统计展现，将经纬度与载客峰值通过百度地图叠加以热力图方式得到图 5-4 效果。可以看出，载客热点路段以莫愁湖公园为中心周边路段为南京城市载客热点的主要区域。

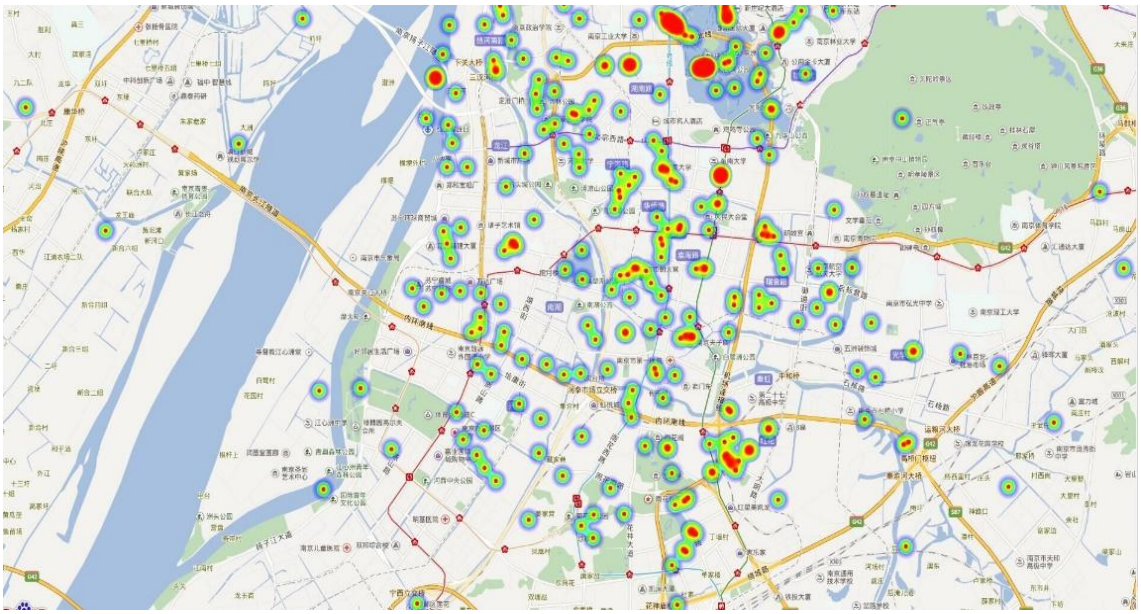


图 5-4 南京市出租车载客热点

5.5 出租车载客峰值分析的实现

以时间、经纬、载客状态为数据统计和分类的基本单元。图 5-5 是对南京全市两天里各个时间段范围内的载客统计量。可以看出，每天早晨 6 点以后载客量直线增加，由于城市居民陆续的出行，在早晨 9 点左右会出现第一个载客峰值；下午 14 点左右会出现第二个载客峰值，下午 17 点左右会出现第三个载客峰值，到了晚上由于居民陆续回到居住地休息在 21 点左右会出现第四个载客峰值。

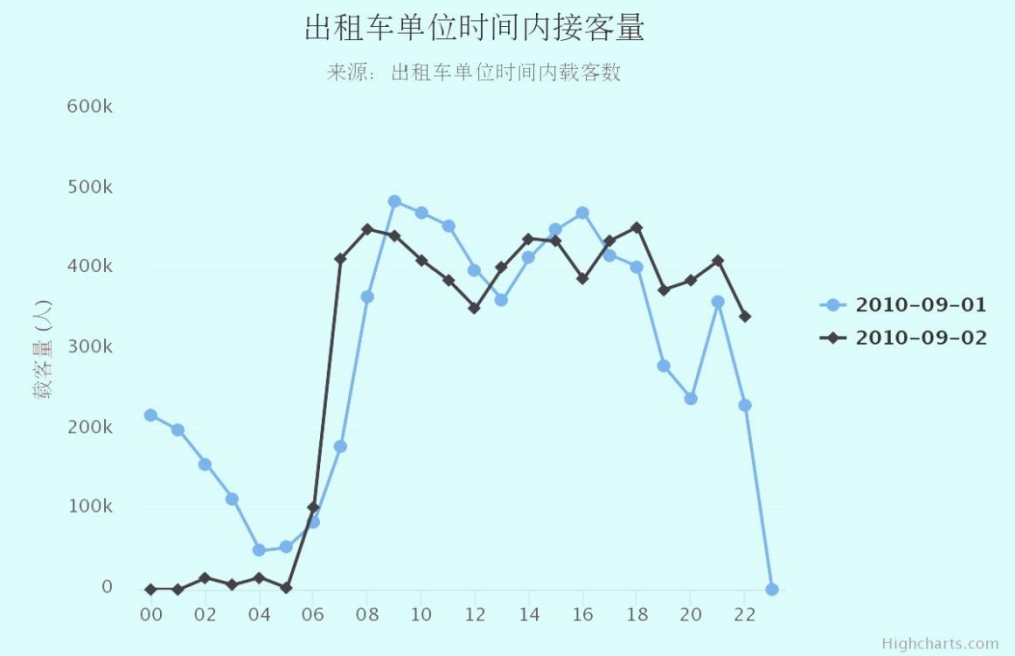


图 5-5 出租车载客峰值

5.6 出租车营运规则分析的实现

通过关联规则算法 Apriori 对营运数据的经度、维度和载客状态进行关联分析，发现载客热点与载客热点之间存在的关系。由于数据量太大，单机环境在挖掘分析中消耗大量时间，所以 MapReduce+Apriori 进行并行挖掘分析。图 5-6 是通过关联分析后的结果展示，通过分析可推演和感知调度相应的城市出租车运行方案。

图 5-6 出租车营运规则

5.7 实验结果与分析

5.7.1 系统时效性验证

为了验证 MapReduce+Apriori 并行数据挖掘的时效性，对原始数据处理，分别筛选出两万条、五万条、十万条、三十万和一百万条数据进行实验研究。通过对单机系统与 Hadoop 云平台进行速度测试，实验结果如图 5-7 所示。

图 5-7 单机系统和 Hadoop 云平台运行速度对比

通过图 5-7 的观察可以看出，在五万条数据左右，两者在处理相同数据时的时间没有太大差别，当数据继续增长后两者之间速度差异很大，单机系统处理数据耗时随着数据的增长而直线上升，Hadoop 云平台则缓慢上升，没有很大的波动。实验表明，MapReduce+Apriori 并行数据挖掘的时效性比单机系统要高，在对海量数据挖掘分析时具有很大的优势。

5.7.2 精度验证

为了验证 MapReduce+Apriori 并行数据挖中数据处理精度，对原始数据处理，将经度、维度统一截取 8 位，便于数据的挖掘分析。通过处理过后的数据与百度地图的方式叠加进行精度测试，基于出租车营运源数据进行了三组的关联分析测试，

得出表 5-7 数据结果数据。

表 5-7 时间、精度、维度、载客关联分析

系统平台	可信度	平均正确率
单机系统	0.7/0.8/0.9	80%
Hadoop 云平台	0.7/0.8/0.9	80%

通过对关联分析结果数据观察，通过对置信度分别设置为 0.7、0.8、0.9，结果数据的精度始终维持在 80%左右。实验表明，基于 Hadoop 云平台的 MapReduce 并行挖掘分析并不会对数据的精度造成影响，只会在数据的挖掘的速度上有很大的提升。

5.8 本章小结

本章节首先介绍了系统评价指标，相对于传统系统在处理数据时的优势。然后对实验数据进行了介绍，对样本数据的格式选择、数据提取进行了介绍。接着对系统的前后台进行了介绍，包括前台报表关键代码实现、Hadoop 平台的搭建等。最后针对南京市 7726 辆出租车的共 33042225 条数据记录，采用 MapReduce 分类统计方法进行载客热点分析、载客峰值统计分析，通过 MapReduce+Apriori 对出租车营运规则分析，验证了基于 Hadoop 云平台的 Apriori 并行关联算法的时效性，将关联分析计算后的结果通过百度地图和简数 CDN 图表进行了可视化展示。

6 总结与展望

6.1 总结

本文以大数据、数据挖掘和出租车营运数据挖掘为背景，介绍了国内外的相关研究背景；在基于关联规则算法 Apriori 基础之上提出了基于 Hadoop 云平台的并行化 Apriori 关联规则算法；提出了基于 Hadoop 的出租车营运数据挖掘系统以及基于该系统的载客热点分析应用、载客峰值统计应用和出租车营运规则推荐应用。具体工作如下：

（1）了解了大数据、Hadoop、数据挖掘之间的关系和基本概念，懂的了它们的历史发展与服务形式。探讨 Hadoop 云平台的两个组件 HDFS 和 MapReduce，简单说明了它们的体系结构与作业生命周期，为本系统后台的挖掘分析打下良好基础；另外还阐述了 JSP 技术和 MySQL 技术，为本系统的前台设计打下良好基础。

（2）学习了关联规则算法的基本知识，详细了解 Apriori 关联算法的实现原理。针对海量的数据关联分析，利用 Hadoop 云平台对数据并行化处理的思想，提出了基于 MapReduce+Apriori 的关联分析算法的并行实现方案，最后通过 Hadoop 云平台和单机系统对 Apriori 关联规则算法的时效性和准确性进行了结果分析验证。

（3）结合关联规则算法 Apriori 和营运数据的特征，实现了对出租车载客热点、载客峰值和营运规则的挖掘分析。选择南京市出租车的营运数据，经过营运数据的筛选、统计、排序、关联规则分析，证实了对以上三点挖掘的可行性。

6.2 展望

本文通过 Hadoop 云计算平台与 Apriori 关联分析算法相结合的方式，研究出一种基于 Hadoop 的出租车营运数据挖掘系统，有效地提高出租车营运大数据分析的效率。但由于研究时间和水平有限，有许多功能实现的不够完善，需要在未来的工作与学习中继续研究与改进：

（1）关联规则算法 Apriori 算法还存在优化问题。本文使用的 Apriori 算法进行关联分析时，需要手动输入 minSup、minConf、maxNivel 三个参数，参数不同时数据关联分析后的结果会存在差异，所以在 Apriori 算法参数优化问题尚有待研究；另外参数的设置可以通过对数据的类型、大小、维度等动态设置，这也是日后进一步研究的方向。

（2）关联分析数据显示的问题。由于系统研究没有自己的地图系统，所使

用的是百度的地图的接口进行展现，在数据展现方面还无法具体的实现。

（3）数据深度不足。本文使用的南京市出租车的数据仅仅只有两天，规模不够大，所以在关联分析结果的正确性方面还有待进一步验证。当然，数据量越大，且数据深度越大，关联分析结果的正确性就越可靠。另外，本研究中集群中使用的节点数较少，只是用作学术研究，而无法和真实的生产环境相比。

（4）本文仅仅提出了基于 Hadoop 的 Apriori 算法并行挖掘系统，对实现的功能业务都是离线的，下一步工作可以加入对出租车产生的实时数据进行关联分析，如对实现出租车路线的优化、出租车的异常检测等。

参考文献

[1] Zheng Y.Urban computing:enabling urban intelligence with big data[J].Frontiers of Computer Science, 2017,11(1):1-3.

[2] Zhang J,Meng W,Liu Q Q,etal.Efficient vehicles path planning algorithm based on taxi GPS big data[J] .Optik-International Journal for Light and Electron Optics,2016,127(5):2579-2585.

[3] 陈盛,陆建.出租车交通调查分析及对策[J].交通标准化,2003(5):41-44.

[4] 张爽.城市出租车拥有量的确定方法研究[D].成都:西南交通大学交通运输与管理学院交通工程系,2009.

[5] Usama Fayyad, Piatesky-Shapiro G, Smyth P. The KDD process for extracting useful knowledgeform volumes of data[J]. Communications of the ACM,1996,39(11):27-34.

[6] 李强.数据挖掘中关联分析算法研究[D]. 哈尔滨:哈尔滨工程大学计算机科学与技术学院,2010.

[7] 李晓.运用大数据提升交通管理能力[J].企业技术开发,2015, 34(21):127-129.

[8] 邓倩妮,陈全.云计算及其关键技术[J].高性能计算发展与应用,2009(26):2-6.

[9] 廖宝魁孙隽枫.基于 MapReduce 的增量数据挖掘研究[J].微型机与应用,2014,1(33):67-70.

[10] 秘中凯,姜晓红,雷蕾.一种稳定的并行分布式频繁集挖掘算法及其应用[J].计算机应用与软件,2011,3(28):84-86.

[11] 魏玲,魏永江,高长元.基于 Bigtable 与 MapReduce 的 Apriori 算法改进[J].计算机科学,2015,10(42):2 08-211.

[12] 郝晓飞.谭跃生,王静宇.Hadoop 平台上 Apriori 算法并行化研究与实现[J].计算机与现代化,2013 (13):1-5.

[13] 毛卫俊.基于云平台的并行关联规则挖掘算法研究[D].上海:华东理工大学计算机科学与工程系,2013.

[14] 崔日新.大规模数据挖掘聚类算法的研究与实现[D]. 西安:西安电子科技大学计算机科学系,2013.