# 摘 要

近年来，伴随着生活节奏的不断加快，人们对于餐饮的便捷性有了更高的要求，针对这种需求的提高，越来越多的餐饮企业将外卖系统引入了自身的发展经营中，而掌握外卖业务数据是餐饮企业立足于市场以及企业发展的重要条件。外卖企业的业务数据产生于现实中客户通过登录目标网站进行搜索，阅览外卖店铺简介及列表，进入店铺浏览菜单，选中目标产品并放入购物车，留存送货地址，生成产品订单等一系列基本数据。而互联网和O2O行业的发展，随之而来产生更多大量的外卖业务数据，传统数据库进行数据存储分部门抽取、整理、提交，综合性较弱、缺乏整合分析，缺乏指标的实时监控，难以即时发现和纠正问题。发现异常指标，如何进一步做追溯和排查，能否做到企业中不同部门领导监控不同的指标，等等都是现阶段外卖企业业务数据急于解决的问题。

如今以Hadoop分布式系统为基础,结合传统的数据处理技术以及基于Hive的日志分析系统的海量数据处理，并根据外卖企业业务需求对系统进行优化。在企业大数据背景下海量日志分析系统中的关键技术，包括Hadoop分布式文件系统HDFS、分布式计算框架MapReduce、数据仓库工具Hive与数据迁移工具Sqoop等，是常见的用于企业处理数据的大数据平台系统架构。此外在Hadoop平台中加入数据仓库技术，是当前对信息数据进行收集处理的核心环节。

此外我国线上的外卖行业也逐渐朝服务的智能化与商品的多元化方面进发。服务的智能化意思是指现如今的人工智能技术在外卖行业中起到的作用，而服务的多元化意思是在线上的外卖行业业务不再仅限于食品的配送，我们每日生活的各种各样需求也将加入到外卖的配送范围。在将来外卖的配送速率将进一步升级，线上平台也更加注重于平台的多维服务体系构建，逐渐趋向于更加优质、更加智能且高效的用户体验。本文主要是搭建了CDH的大数据平台，结合数据仓库技术，对数据进行抽取、转换、装载及利用OLAP（联机分析处理）进行辅助决策。

关键词：大数据；Hadoop；CDH；HDFS；MapReduce；Hive；数据仓库；OLAP

**ABSTRACT**

In recent years, with the constant acceleration of the pace of life, people have higher requirements for the convenience of catering. In response to the improvement of this demand, more and more catering enterprises have introduced the takeout system into their own development and operation, and mastering the takeout business data is an important condition for catering enterprises to be based on the market and enterprise development. In reality, the business data of the take out enterprise is generated by the customers who log in to the target website to search, read the introduction and list of the take out shop, enter the shop browse menu, select the target product and put it in the shopping cart, retain the delivery address, generate the product order and a series of basic data. With the development of the Internet and o2o industry, more and more take out business data will be generated. The traditional database is used to extract, sort and submit data by departments, which is weak in comprehensiveness, lack of integrated analysis, lack of real-time monitoring of indicators, and difficult to find and correct problems in real time. If abnormal indicators are found, how to further trace and check them? Can leaders of different departments monitor different indicators? And so on are the problems that take out enterprise business data is eager to solve at this stage.

Now, based on the Hadoop distributed system, combined with the traditional data processing technology and the massive data processing based on hive log analysis system, and optimized the system according to the business requirements of the takeout enterprise. In the context of enterprise big data, the key technologies in the massive log analysis system, including the Hadoop distributed file system HDFS, the distributed computing framework MapReduce, the data warehouse tool hive and the data migration tool sqoop, are common big data platform system architectures for enterprise data processing. In addition, adding data warehouse technology to Hadoop platform is the core link of information data collection and processing.

In addition, China's online take out industry is also gradually moving towards the intelligent service and diversification of goods. The intelligent meaning of service refers to the role of artificial intelligence technology in the takeout industry, while the diversified meaning of service means that the online takeout industry business is no longer limited to food distribution, and all kinds of needs of our daily life will also be added to the scope of delivery. In the future, the delivery rate of takeout will be further upgraded. Online platforms also pay more attention to the construction of multi-dimensional service system of the platform, and gradually tend to be more high-quality, intelligent and efficient user experience. In this paper, a big data platform of CDH is built. Combined with data warehouse technology, data is extracted, transformed, loaded and assisted by OLAP (online analytical processing).

Key words: Bigdata；Hadoop；CDH；HDFS；MapReduce；Hive；Data Warehouse；OLAP

# 

# **目 录**

[摘 要 I](#_Toc452001416)

[目 录 I](#_Toc452001417)

[1 绪论 1](#_Toc452001418)

[1.1 课题研究背景及意义 1](#_Toc452001419)

[1.1.1 大数据时代的来临 1](#_Toc452001420)

[1.1.2 数据仓库的由来与传统数据仓库 1](#_Toc452001421)

[1.1.3 外卖企业中基于Hadoop与数据仓库系统的运用 2](#_Toc452001422)

[1.2 国内外有关研究动态现况及应用 2](#_Toc452001424)

[1.2.1 国内外基于Hadoop的研究现状及应用 2](#_Toc452001425)

[1.2.2 国内外数据仓库的研究现状及应用 3](#_Toc452001426)

[1.3 论文结构 3](#_Toc452001428)

[2 大数据平台概述及CDH各组件搭建过程 5](#_Toc452001429)

[2.1 Hadoop介绍 5](#_Toc452001430)

[2.1.1 Hadoop的发展历程 5](#_Toc452001431)

[2.1.2 为什么要用Hadoop 5](#_Toc452001432)

[2.1.3 HDFS及MapReduce介绍 5](#_Toc452001433)

[2.2 CDH之Hive介绍及项目其他组件介绍与平台搭建 8](#_Toc452001435)

[2.2.1 Hive介绍及其他组件介绍 8](#_Toc452001436)

[2.2.2 CDH分布式集群的搭建 23](#_Toc452001437)

[3 外卖企业中基于Hadoop的数据仓库系统问题分析 29](#_Toc452001438)

[3.1 传统型数据仓库分析 29](#_Toc452001439)

[3.1.1 传统型数据仓库功能分析 29](#_Toc452001440)

[3.1.2 传统型数据仓库功能中的缺点 29](#_Toc452001441)

[3.2 外卖企业中基于Hadoop的数据仓库系统方案分析 30](#_Toc452001443)

[3.2.1 外卖企业业务数据处理新需求 30](#_Toc452001444)

[3.2.2 外卖企业中基于Hadoop的数据仓库系统新架构 30](#_Toc452001445)

[4 外卖企业业务系统中基于Hadoop的数据仓库详细设计 35](#_Toc452001449)

[4.1 系统分析外卖企业业务需求指标 35](#_Toc452001450)

[4.1.1 业务系统中功能性需求与非功能性需求分析 35](#_Toc452001450)

[4.1.2 外卖企业中基于Hadoop的数据仓库系统开发规范 35](#_Toc452001450)

[4.2 外卖企业业务系统中的数据仓库具体设计 36](#_Toc452001451)

[4.2.1 业务系统中的数据仓库功能模块设计与分层解析 35](#_Toc452001450)

[4.2.2 业务系统中数据仓库具体流程设计 35](#_Toc452001450)

[5 外卖企业系统中基于Hadoop与数据仓库的具体实现 40](#_Toc452001452)

[5.1 业务数据建模过程 40](#_Toc452001453)

[5.2 用户访问模型与用户订单模型的最终实现 42](#_Toc452001454)

[结论 46](#_Toc452001455)

[致 谢 47](#_Toc452001456)

[参考文献 48](#_Toc452001457)

基于Hadoop的外卖企业业务数据分析系统设计与实现

近些年随着网络的兴起以及计算机信息技术的高速发展，人们产生的数据量迅猛增加，从大量的数据中挖掘出有价值的数据变得十分关键。然而较早研究的数据库与数据处理方式无法处理现在甚至将来超海量的数据。Hadoop生态圈作为一个能很好地处理海量数据的平台，将数据仓库技术融入其中实施数据仓库的建设是如今企业应用中较为有效的方法，可以提升处理数据的性能和处理海量数据的能力。本文分析Hadoop生态圈自身特点，结合数据仓库分层处理的思想，提出了基于Hadoop大数据平台的外卖企业业务数据分析系统设计与实现，且在Hadoop大数据平台上实现，较为有效地解决传统数据库无法处理的超大型规模数据量级的问题并且整合外卖企业业务数据，建立统一的数据中心，生成用户的访问模型和用户的订单行为模型，用于辅助企业的管理与决策。

# 1 绪论

## 1.1 课题研究背景及意义

### 1.1.1 大数据时代的来临

现如今，数据已渗透到了各个行业和生产生活领域，并成为决策生产发展的重要因素。人们需要对海量数据进行储存、挖掘、分析以及运用，这也预示着生产的增长以及消费浪潮的到来。大数据行业的重要性就此体现出来，其行业在科技领域的发展不仅体现在互联网领域，还体现在金融、教育、医疗等诸多领域，尤其是现在的人工智能。

数据量的井喷式膨胀变大，也预示着企业的发展前景，如今越来越多的企业注意到了数据爆炸性的增长带来的隐患，越来越多的企业意识到数据对企业的重要性。

著名的未来学家托夫勒在作品《第三次浪潮》中就把“大数据”称为“第三次浪潮的华彩乐章”。《自然》杂志也在2008年9月推出了名为“大数据”的封面专栏。从2009年开始“大数据”成为互联网技术行业中的相当热门的词汇。

在外卖企业中业务数据的增长主要是由用户使用浏览器(PC端)和App对外卖平台进行访问时记录的用户注册信息，企业中通过调研平台与数据挖掘记录的用户的扩展信息，用户的每次点击和下单信息等。针对各种数据的增加，越来越多的餐饮企业将大数据技术引入到自身企业的发展经营中，从而掌握外卖业务数据以立足于市场。

### 1.1.2 数据仓库的由来与传统数据仓库

数据仓库的概念最早诞生于20世纪70年代，由MIT的研究员研究的一种优化的技术架构，后来DEC公司通过研究合并MIT的结论，创建了TA2（Technical Architecture2）规范，这种规范定义了分析系统的四个部分：数据获取、数据访问、目录及用户服务；为了实现解决企业的集成等问题，IBM公司首次提出了信息仓库（InformationWarehouse）的模型，称为VITAL规范（VirtuallyIntegrated Technical Architecture Lifecycle）；在1991年Bill Inmon（比尔·恩门）推出了他第一本关于数据仓库的书籍《Building the Data Warehouse》，象征着数据仓库的确立。

传统数据仓库具有面向主题、集成、非易失且随时间不断变化的四大特征。根据分析的具体要求来确定主题，这和按照数据处理及应用的要求来组织数据是不同的，而集成特性指数据仓库里的数据必须是一致的，数仓中的数据经过抽取转换加载形成的分析型数据，为企业提供决策分析，而执行的操作主要是以‘查询’为主，一般情况下较少进行‘更新’操作。数仓凭借维度的形式对数据进行较为有效组织，时间维度是数仓中十分重要的维度，且有些数仓中的数据时间维跨度大，从几年到几十年，称之历史数据。

### 1.1.3 外卖企业中基于Hadoop与数据仓库系统的运用

针对近年来人们对餐饮便捷性要求的提高，餐饮企业也逐渐将外卖系统引入自身的业务体系发展中，而掌握外卖业务数据是餐饮企业立足于市场以及企业发展的重要条件。如今以Hadoop平台为基础，结合传统的数据仓库处理技术以及基于Hive的日志分析系统的海量数据处理，并根据外卖企业业务需求对系统进行优化。大数据背景下，企业中海量日志的分析系统的关键技术，包括Hadoop分布式文件系统HDFS、分布式计算框架MapReduce和数据仓库工具Hive以及数据迁移工具Sqoop等，都是企业常见的用于处理大数据的系统架构。

本文通过分析数据仓库自身特点，结合Hadoop平台各大组件，提出基于Hadoop平台的数据仓库技术，最终利用数据仓库中分层的特点分析外卖企业的业务数据，使决策者在大量资料中快速有效的分析出具有商业有价值的讯息，以便用于决策规划及迅速地响应外在环境变化。

## 1.2 国内外有关研究动态现况及应用

### 1.2.1 国内外基于Hadoop的研究现状及应用

Hadoop最早发展源于2002年的Apache Nutch项目。在2004年Google的“操作系统设计及实现”（Operating System Design and Implementation）会议中公开发表名为MapReduce：Simplified Data Processing on Large Clusters（简化大规模集群中的数据处理）论文以后，深受启发的Doug Cutting等人开始试着搭建MapReduce的计算框架，并把它和NDFS（Nutch Distributed File System）融合，用来支持Nutch引擎的核心算法。因为MapReduce和NDFS在Nutch中有较好的作用，因此它们在2006年2月被分别提取出来，形成了一套完整且独立的架构，并且命名为Hadoop。

Yahoo是Hadoop平台的最大支持者，截止2012年Yahoo的Hadoop机器总节点数目超过42000个，并有超过10万的核心CPU运行Hadoop；Facebook也利用Hadoop储存企业内部日志和多维数据，作为报告、分析和机器学习的数据源；IBM蓝云则运用了Hadoop来搭建云基础设施，运用技术有：Xen、PowerVM虚拟化的Linux操作系统映像和Hadoop并行工作量调度，并且发布了自己的Hadoop发行版及大数据解决方案。除此之外还有许多的大型企业也投入到Hadoop的研究与应用中，如Adobe、Datagraph、LinkedIn等。

Hadoop在国内发展也紧跟国际脚步。华为就是对Hadoop发展作出重要贡献的公司之一，其在Hadoop的高可用（High Availability）方案中及HBASE的领域有着深入的研究，并且已向业界推出自己的基于Hadoop大数据解决方案。百度早在2006年就关注Hadoop并且开始研究与使用，截止到2012年，总集群规模超过了7个集群，单集群超过了2800个节点，Hadoop的节点机器总数超过了15000台，总共存储容量超过了100PB。除外国内还有众多名企也推出了基于Hadoop的大数据研究与应用，例如阿里巴巴就成立了阿里云公司，腾讯、奇虎360、中国移动等都有相应动作，包括中国电信、中兴等等企业也都转入了Hadoop大数据的研发运用当中。

Hadoop的应用范围极广，在现如今数据爆发增长的年代，掌握运用Hadoop分布式平台分析处理数据是企业中不可或缺的关键因素。大数据技术的战略意义在于掌握庞大的数据信息，且对这些含有意义的数据进行专业化处理。

### 1.2.2 国内外数据仓库的研究现状与应用

2010年全球财富杂志500个企业名单中，接近90％的企业都实行了企业级的数据仓库或者部门级别的数据集市，在这些企业中例如银行、证券、电信、航空、邮政、保险、大型零售等等行业都极其依赖数据仓库技术。在国际金融界也非常重视数据仓库的搭建，全球排名前100的大型银行都有自己的数据仓库，华尔街92％的证券、银行、保险等等机构利用数据仓库技术进行风险管理，包括著名的花旗银行、摩根·斯坦利、加皇银行、加拿大蒙特利尔银行等等。

随着国内信息化建设的逐步完善与企业中对数据仓库运用意识的提高，信息化程度较高的行业逐渐认识到数据仓库技术对于企业宏观发展带来的巨大经济效益。例如中国银行、中国建设银行、招商银行系统、中国移动等等都已经搭建了大型的数据仓库，这些公司也成为国内数据仓库应用的开拓者。随着我国加入WTO后，我国的企业也将面临更加严峻的挑战，企业的决策改变及旧的经营习惯，也将通过科学的手段降低经营风险、创造更高的利润。

综上，随着大数据时代的到来，建立数据仓库是企业变革与发展的必然要求，越来越多的企业意识到数据仓库技术的重要性并也将数据仓库技术运用到自身企业中。

## 1.3 论文结构

第一章重在介绍本文研究的背景及意义，国内外研究现状与应用，以及本论文的结构。

第二章介绍了关于Hadoop平台以及基于CDH平台的搭建。本章主要对Hadoop生态圈进行介绍与CDH中的各大组件体系结构及有关技术进行分析，最后简单介绍CDH分布式的搭建过程。

第三、第四、第五章为论文的核心部分，对企业中传统数据分析系统进行分析，总结出企业中传统数据数据分析系统的优缺点并提出当今外卖企业业务数据处理的新需求，设计出新型基于Hadoop平台的外卖企业业务数据分析系统的方案与架构，对基于Hadoop的外卖企业业务系统进行详细设计与系统实现。

最后结论，对本篇论文研究的主要内容进行概述和总结，分析指出本次工作中的不足，并提出改进的计划及下一步的实现目标。

# 2 大数据平台概述及CDH各组件搭建过程

## 2.1 Hadoop介绍

### 2.1.1 Hadoop的发展历程

Apache lucene是应用范围极广的基于文本搜索的系统库，其项目创始人道格·卡丁早在2002年带领团队开发该项目中的子项目Apache Nutch，并致力于重新打造网络搜索引擎，但在其开发的过程当中，发现了两个棘手问题：硬件的投入资金过高，难以存储大型文件。

在2003年到2004年间Google先后发表了《GFS》与《MapReduce》论文，为陷入困境中的该团队提供了灵感，并在道格·卡丁的带领下进行实现，NDFS(Nutch分布式文件系统)与MapReduce随之公之于众。2006年2月，相关开发人员把NDFS于MapReduce迁移出Nutch成立了独立的子项目，命名为Hadoop(据Doug Cutting所说，此次命名是借用了他孩子的毛绒玩具名字)。

Hadoop隶属于Apache基金会下的一个开源分布式储存以及分析运算平台，并使用java语言进行开发，具有良好的跨平台性，可以运行在廉价的硬件机器上。在用户不了解分布式底层实现细节的情况下，能够很好的开发分布式程序，充分地使用集群快速的计算与存储。

Hadoop是Apache的一个开源项目，有许多公司都开始向Hadoop方面发展，因此这些企业在原基础之上进行了二次开发转为商业化，根据自身企业特性加入了不同的特色。在Hadoop的发行版本中除去Apache社区提供的Hadoop以外，较为出名的企业有mapR、hortonworks、cloudera、华为、DKhadoop等等都开发出了各自的商业版本，但大型公司所提供的更为专业的技术支持，基本都收费。

### 2.1.2 为什么要用Hadoop

现如今我们生活在数据大爆发的时代。据相关专业人士预测在2020年，全球数据总量将超44ZB，也就是说，数据量最少在440亿TB以上，就算全球每人一块1TB的硬盘也依旧存储不下。

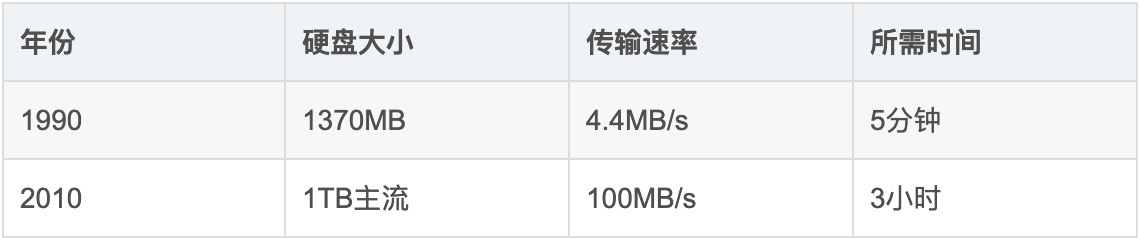
个别数据大小甚至远远超过1TB，解决数据存储问题已经成为当前社会的燃眉之急。于此同时硬件技术也面临到技术瓶颈，传统硬盘的传输速率的提升远远低于了硬盘容量的提升。

图2-1 磁盘容量提升与读写速率比较图

显而易见的是磁盘容量提升了近1000倍，但传输速率却只提升了20倍，完全读完硬盘所需的时间相对变得更长了，这完全违反了数据价值的即时性，读写数据都需要花费大量的时间。针对怎样提高读取效率，其实已经有了解决方法，可以将一个数据集储存到多个磁盘中，而后并行读取。例如1T的数据，平均1存储到100个1TB磁盘上，并行读取，读取整个数据集的时间不到两分钟。硬盘剩余的容量，则可用来存储其他数据集，不产生资源浪费。即解决了读取效率问题，也解决了大数据量的存储问题。

但同时对多个磁盘进行读写时，又出现了新的问题:

1、硬件中的故障问题。上述提出的多个硬件，对于个别硬件产生故障的几率就提高了许多，为了尽量避免数据的丢失，最常使用的办法就是复制(replication)：在文件系统中保存多个数据副本，一旦磁盘发生了故障，则使用其他的副本。

2、如何保证读取数据的正确性。在大数据时代里一个分析任务，就需要结合大部分的数据共同完成分析，所以从其中一个节点上读取的数据要和从其他99个节点上读取的数据结合使用。因此，在读取的过程当中，怎样才能保证数据正确无误是一个很大的挑战。

针对以上几个问题，Hadoop提供了可靠的并且可扩展的存储及分析平台。另外，因为Hadoop可以运行在商用的机器上并且是开源的，所以Hadoop的使用成本降低了许多，在许多用户的承受范围之内。

### 2.1.3 HDFS及MapReduce介绍

HDFS（Hadoop Distributed File System）的高容错特性可以很好的解决上一节中所提到的问题，存入平台中的数据将自动保存成多个副本，当副本丢失后，namenode会自动经行恢复,对于GB、TB、以至于PB级的数据、百万规模以上的文件数量，1000节点以上规模都适用，用户一次性写入可以进行多次读取，保证数据的一致性并且可以部署在廉价的机器之上。

手机屏幕截图

描述已自动生成

图2-2 HDFS体系结构

HDFS采用主从结构管理存储的数据，其结构主要由Namenode(名称节点)、Datanode(数据节点)、Client(客户端)和SecondaryNameNode。在全分布式的HDFS集群中包含一个Namenode节点以及若干个的Datanode节点。Namenode是集群中的一个中心服务器，它负责管理系统的命名空间(Namespace)以及响应客户端的请求。在集群中Datanode是一个单独的节点运行一个DataNode进程，它负责管理客户端的读写请求，听命于Namenode调度进行数据块的创建、复制及删除等等操作。而数据块的实际存储路径在Linux文件系统中。每一个Datanode默认三秒向Namenode发送数据，报告状态(心跳机制)。没有按规定时长发送心跳的Datanode会被Namenode标记为“死亡”，Namenode不会再给它分配I/O请求。Client进行I/O操作时，依旧像使用普通文件系统一样，使用文件名存储与访问文件，但在HDFS内部，超过规定大小文件会被切分成若干数据块，分布存储在若干个Datanode节点上。

而在海量数据在单机上处理，内存资源将受到极大限制，磁盘与运算能力也将受限，无法对海量的数据进行计算，并且将单机版程序应用到集群中进行分布式运行也极大的增加了编程的复杂程度与开发难度。分布式运算逻辑至少需要分成2个阶段：独立并发阶段，汇聚阶段。在程序开始前应该如何把文件切片，在运算过程中程序将如何分发，两阶段的程序应该如何启动，如何协调，整体程序运行过程如何做到有效监控中等等一系列问题都需要开发人员完成。

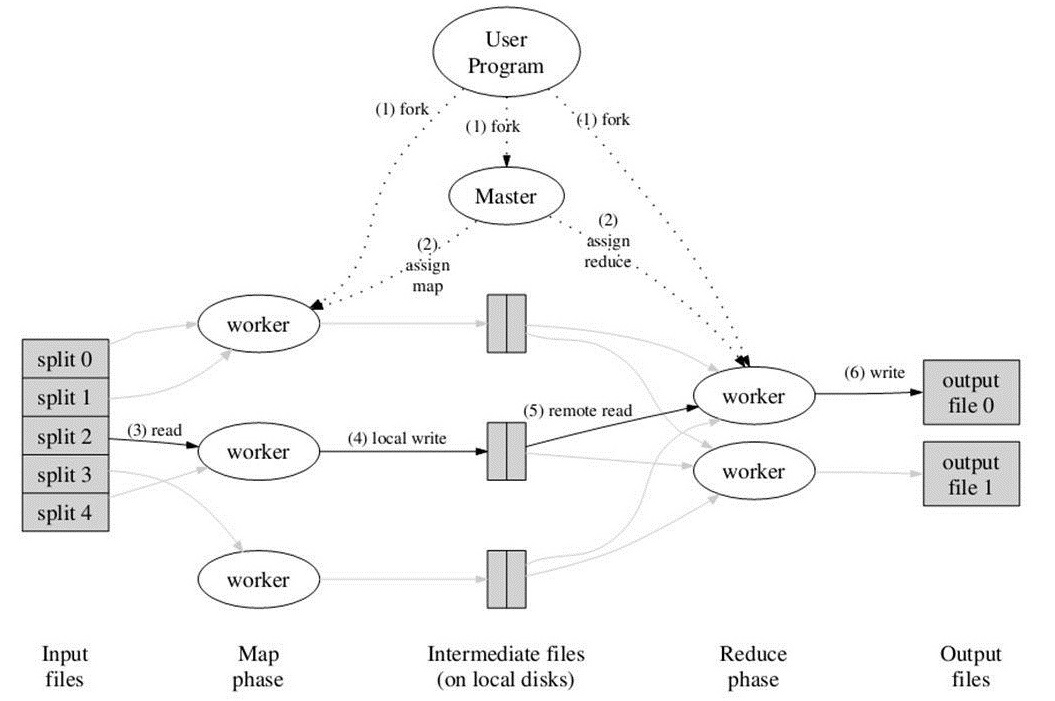
MapReduce框架随之诞生，在MapReduce框架下开发人员可以将大部分工作精力放在对业务逻辑的开发中，从而把分布式计算的复杂需求交由框架处理。

图2-3 MapReduce基本工作机制图

Mapreduce是一个分布式的运算框架，用于超大规模数据集的并行计算。一个完整的Mapreduce程序在运行时包含三类实例进程:

1、MRAppMaster：负责整个作业过程的调度与状态的协调。

2、MapTask：负责map阶段的数据流程处理。

3、ReduceTask：负责reduce阶段的数据处理流程。

Map(映射)与Reduce(归约)，是MapReduce的核心思想。其特性大多都是从函数式编程语言与矢量编程语言中借鉴来的。MapReduce极大地便利了开发人员在对分布式并行程序底层实现不了解的情况下，将自己的程序运行在分布式系统上。

## 2.2 CDH之Hive介绍及项目其他组件介绍与平台搭建

CDH是Cloudera公司的开源发行版，包括Apache Hadoop在内，专门为满足企业大数据需求而构建的大数据平台。CDH通过将Hadoop与相关的十几个开源项目集成在一起，创建了一个功能全面的系统，可帮助处理企业绝大部分大数据的工作流程。总之，CDH是一个拥有中心化管理、集群监控、自动化安装以及报警功能的软件，可使集群的安装从数天的时间缩短为数个小时，运维人数也将大幅下降，极大提高了集群管理效率。

### 2.2.1 Hive介绍及其他组件介绍

**1.Hive介绍**

Hive是Hadoop中的数据仓库基本工具，其主要功能是处理结构化数据。Hive依赖于Hadoop之上，使查询与分析更加方便，并且提供了简单的sql查询功能，可以通过sql语句将Job转换为MapReduce任务并运行。

Hive最初是由Facebook公司研发的，后由Apache基金会发布，并将它作为其名义下Apache Hive的一个开源项目。Hive并没有专用的数据格式。但在Thrift中Hive可以很好的工作，可以控制分隔符，并且允许用户指定数据格式。

Hive对于Hadoop属于强依赖关系，在Hadoop平台常常有较长的延迟且在作业提交与调度时需要大量的资源开销。所以，Hive 不能满足用户在超大数据集上实现低延迟的查询，Hive在百兆的数据集上运行查询基本都有分钟级的时间延迟，这也说明了Hive并不适合联机事务处理（OLTP）。其查询过程严格遵守MapReduce的作业执行，它将用户的HQL语句通过解释器转换为MapReduce的作业提交到Yarn集群上，而后返回结果给用户。Hive 并不是为OLTP（联机事务处理）而设计的，Hive不提供实时查询需求及基于行级的数据更新，Hive是为大数据集的批处理作业而生的。

**2.Sqoop介绍**

Sqoop作为一个开源的数据交互工具，其主要运用于Hive与Mysql、Postgresql等传统的数据库间的数据传递，它可以将关系型数据库 MySQL 、Oracle、Postgres等里的数据导入到HDFS里，也可反向导入到关系型数据库中。

Sqoop始于2009年，早期作为Hadoop的第三方模块，后为用户可以实现快速部署，并让相关技术人员进行快速开发，Sqoop转变成为Apache的一个独立的项目。

对于个别非关系型数据库Sqoop也提供了方便连接的连接器。与大多数ETL工具类似，Sqoop使用了元数据模型，用来判断数据类型并且在数据从数据源转移到Hadoop过程中保证数据安全的处理。同时Sqoop也专门为海量数据批量传输进行设计，它能够准确分割数据集并且建立Hadoop的任务用以处理每个数据块。

**3.Azkaban介绍**

Azkaban也是一个开源的任务调度工具，主要用于任务的调度执行，最常用于数据仓库中的调度，可替代linux中的计时器crontab。

Azkaban整体分为三部分：dbserver、webserver、executorserver。它由Linkedin开源的批量工作任务调度器在一个工作流内以用户指定的顺序运行提前设定好工作及流程，并使用Java作为开发语言。Azkaban定义了一种特定的KV文件格式用来连接任务间的依赖关系，并且提供了方便用户使用的web界面，可以通过此界面维护及跟踪工作流。

### 2.2.2 CDH分布式集群的搭建

**1.CentOS安装**

(1)安装环境：MacOS10.15.1，Parallels Desktop(PD)2014，CentOS 6.9

Parallels Desktop是一个可以虚拟出Person Computer的软件，利用现有操作系统虚拟化一个新的操作系统，相当于虚拟出了一台新的计算机，并以此实现在一台机器上同时运行两个操作系统。

其主要特点：本机的操作系统可以与虚拟操作系统进行网络通信。无需分区或者重启计算机就可在同一台计算机中使用两款以上的操作系统。可随时修改虚拟机中操作系统的硬件条件。

安装注意事项：如想卸载重装，可双击其安装文件，选择卸载软件，最好不要手动删除安装的目录。安装的路径下绝对不能输入中文、特殊符号以及空格。假如手动删除了Parallels Desktop的安装目录进行卸载，需要自行删除注册表中Parallels Desktop的有关安装信息。

(2)新建虚拟机，点击安装Windows或其它操作系统，选择事先准备好的CentOS-6.9-x86\_64-bin-DVD1.iso。

电脑萤幕的截图

描述已自动生成

图2-4 选择CentOS

(3)取消勾选快速安装，设置用户名密码，点击继续。

电脑萤幕的截图

描述已自动生成

图2-5 设置用户名密码

(4)选择第一项进行安装，而后点击skip跳过，进入正式安装CentOS再次点击下一步。

电脑萤幕的截图

描述已自动生成

图2-6 操作安装选项

(5)连续两次选择语言类型为英语，而后点击下一步。

手机屏幕截图

描述已自动生成

图2-7 语言类型设置

(6)选择基础储存设备。

手机屏幕截图

描述已自动生成

图2-8 选择存储设备

(7)继续选择不保存任何设备，设置主机名称时最好设置为之后需要使用的主机名称，此处设置成hadoop101，选择亚洲上海时区，为后期Azkaban调度时区做准备。

截图里有图片

描述已自动生成

图2-9 时区图

(8)设置root管理员密码，本次实验全部使用root用户进行操作，出于对实验操作的便捷密码设置为123456。

手机屏幕截图

描述已自动生成

图2-10 root管理员密码设置

(9)选择使用自定义磁盘。

手机屏幕截图

描述已自动生成

图2-11 自定义磁盘

(10)创建标准分区，建立初始化分区（/boot），一般设置为300M；建立交换分区（swap）一般设置为2048；最后将剩余资源全部留给根目录。设置完毕后格式化硬盘，按提示点击下一步，选择桌面版本，语言上选择支持中文。

手机屏幕截图

描述已自动生成

图2-12 自定义建立分区规划

(11)继续设置本人的用户名密码（此处也可不进行设置因为实验主要使用root用户进行操作）。虚拟机时间选择同步网络时间，为后期集群时间同步做准备。

手机屏幕截图

描述已自动生成

图2-13 同步网络时间

(12)最后以上述设置的root用户名密码进行登录完成虚拟机的安装。

手机屏幕截图

描述已自动生成

图2-14 root用户登陆

(13)将虚拟机关机，设置虚拟机网络模式为net模式，以便之后虚拟机与主机间的通信；内存大小设置6G；设置DVD镜像路径为虚拟机内建立本地yum源做准备。

截图里有图片

描述已自动生成

图2-15 虚拟机硬件设置图

**2.CDH各组件安装及搭建**

本次实验使用到了CDH5，及其组件hadoop2.6.0，hive1.1.0，sqoop1.4.6。其他组件有JDK1.8.0，mysql（client、server）5.6.26，及任务调度工具Azkaban（包括dbserver、webserver、executorserver）2.5.0；远程终端控制工具finalshell2.9.7。且因后期任务调度Azkaban极其消耗资源，介于本次实验使用到的是个人PC计算机，性能资源较为有限，所以本次实验采用搭建伪分布式的Hadoop平台。

（1）首先检查上一节中安装配置时设置的hadoop101主机名是否有误。

手机屏幕截图

描述已自动生成

图2-16 主机名设置图

（2）设置主机映射：vi /etc/hosts；本次实验因个人资源有限采用伪分布式单节点模式，因此只需要配置本主机映射条件，本机IP10.211.55.14，映射主机名为hadoop101。

手机屏幕截图

描述已自动生成

图2-17 IP映射设置图

（3）设置虚拟机内网络具体配置：vi /etc/sysconfig/network-scripts/ifcfg-eth0；BOOTPROTO=static，默认静态ip，HWADDR=00:0C:29:3D:B0:9B，电脑的mac地址,建议去掉；UUID=6f89eb5e-bff4-4f6b-a045-b7722bd188fc，电脑的UID建议去掉；ONBOOT=yes，是否开机自动使用，设置为yes；nm\_controlled=yes，设置network manager的参数，实时生效，修改后无需要重启网卡立即生效，建议删掉；IPADDR=10.211.55.14，IP设置为事先规划好的IP地址；子网掩码可设置为NETMASK=255.255.255.0或PREFIX=24；GATEWAY=10.211.55.1，网关设置为事先规划的网关；DNS可设置国外DNS1=114.114.114.114，国内的DNS2=8.8.8.8；defroute=yes是默认的路由，建议删掉；IPV4\_FAILURE\_FATAL=yes，如为yes，则ipv4配置失败禁用设备，建议删掉。

总体网络配置设置如图2-18：

手机屏幕截图

描述已自动生成

图2-18 network具体设置图

（4）设置完毕后，保存退出，重启网络服务：service network restart。

手机屏幕截图

描述已自动生成

图2-19 重启网络服务

（5）检查网络配置是否有误，可以尝试访问百度网址，成功设置如图2-20。

图片包含 文字, 游戏机

描述已自动生成

图2-20 测试网络访问

（6）使用远程终端控制工具finalshell控制虚拟机。配置finalshell连接如图2-21。名称主机名填写hadoop101，端口号填写为22，用户名密码填写上一节中设置好的root用户以及密码。

手机截图图社交软件的信息

描述已自动生成

图2-21 使用finalshell连接虚拟机

（7）在外部主机终端配置虚拟机IP主机名映射：vi /etc/hosts；如图2-22。

手机屏幕的截图

描述已自动生成

图2-22 外部主机映射

（8）上述设置了与远程终端控制器finalshell的连接，本次实验中全部以finalshell为虚拟机终端进行操作。将防火墙关闭：chkconfig iptables off；检查是否关闭成功如图2-23。



图2-23 关闭防火墙

（9）提前安装mysql为后期使用hive与sqoop做准备。本次实验安装mysql server与mysql client都是5.6.26版本。安装mysql前先卸载linux里自带的mysql：rpm -e mysql-libs-5.1.73-5.el6\_6.x86\_64 --nodeps；将准备好的mysql server与mysql client解压到指定目录下再使用安装命令分别安装server与client：rpm -ivh MySQL-server-5.6.26-1.linux\_glibc2.5.x86\_64.rpm；rpm -ivh MySQL-client-5.6.26-1.linux\_glibc2.5.x86\_64.rpm。而后使用命令：cat /root/.mysql\_secret获取mysql默认客户端登陆密码，执行文件/usr/bin/mysql\_secure\_installation将默认密码改为方便使用的123456。进入mysql客户端：mysql -uroot -p123456，分别执行如图2-24命令设置权限。

手机屏幕截图

描述已自动生成

图2-24 设置mysql权限

（10）安装JDK。将事先准备好的JDK解压到/opt/apps下建立软连接ln -s /opt/apps/jdk1.8.0\_221 jdk，配置并环境变量：vi /etc/profile。export JAVA\_HOME=/opt/apps/jdk，export PATH=$PATH:$JAVA\_HOME/bin。刷新环境变量source /etc/profile，测试java环境：java -version。

手机屏幕截图

描述已自动生成

图2-25 JDK环境变量

（11）安装Azkaban。将事先准备好的Azkaban解压到/opt/apps/azkaban下，建立软连接ln -s /opt/apps/Azkaban/azkaban-executor-2.5.0 executor；ln -s /opt/apps/Azkaban/azkaban-web-2.5.0 server；进入mysql客户端，创建azkaban数据库create database azkaban，并将解压的脚本导入到azkaban数据库source /opt/module/azkaban/azkaban-2.5.0/create-all-sql-2.5.0.sql。退出mysql在终端执行source /opt/module/azkaban/azkaban-2.5.0/create-all-sql-2.5.0.sql。

运行此命令后,会提示输入当前生成keystor的密码及相应信息，输入的密码请劳记，信息如下：

输入keystore密码，再次输入新密码，

您的名字与姓氏是什么?

[Unknown]:

您的组织单位名称是什么?

[Unknown]:

您的组织名称是什么?

[Unknown]:

您所在的城市或区域名称是什么?

[Unknown]:

您所在的州或省份名称是什么?

[Unknown]:

该单位的两字母国家代码是什么

[Unknown]:

CN

CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=CN 正确吗?

[否]: y

输入<jetty>的主密码(如果和 keystore 密码相同，按回车)，再次输入新密码。

完成上述工作后,将在当前目录生成 keystore 证书文件。

将keystore 拷贝到 azkaban web服务器根目录中mv keystore /opt/apps/azkaban/server/。

进入azkaban web服务器安装目录conf目录，打开azkaban.properties文件，配置如图2-26。

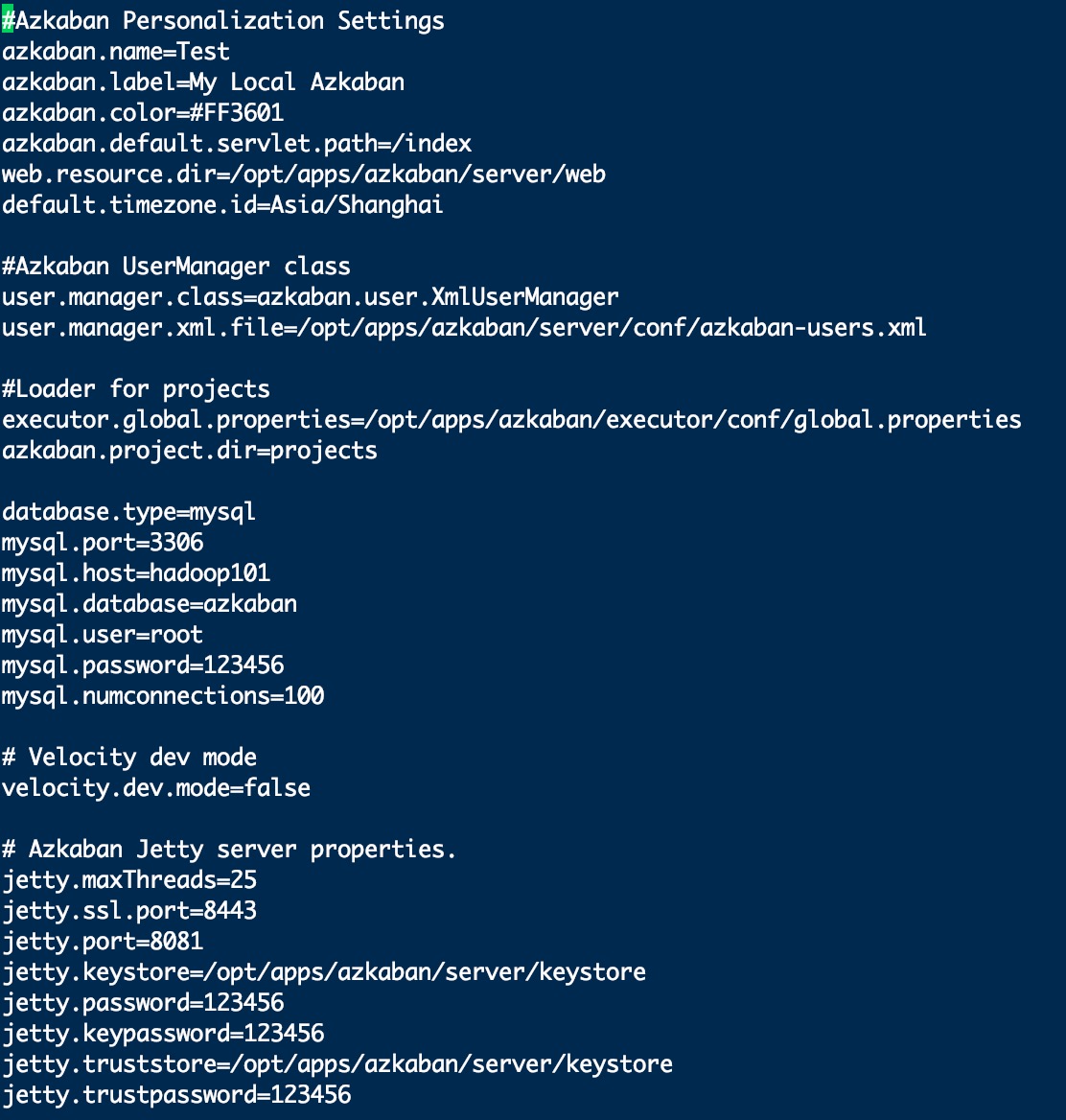


图2-26 配置azkaban.properties文件

（12）在azkaban web服务器的conf目录下，按照图2-27配置修改azkaban-users.xml文件，增加管理员用户。

手机屏幕截图

描述已自动生成

图2-27 配置azkaban-users.xml文件

（13）进入/opt/apps/azkaban/executor/conf执行服务器安装目录conf，打开azkaban.properties，配置如图2-28。

手机屏幕截图

描述已自动生成

图2-28 配置executor中的azkaban.properties文件

（14）启动Azkaban，执行bin/azkaban-executor-start.sh、bin/azkaban-web-start.sh。启动完成后，在浏览器(建议使用谷歌浏览器)中输入https://hadoop101:8443，即可访问azkaban服务了。在登录中输入刚才新的户用名及密码，点击login如图2-29。

手机屏幕截图

描述已自动生成

图2-29 登陆Azkaban webUI界面

（15）解压事先准备好的CDH5.13.2，本次实验使用CDH中的hadoop2.6.0、hive1.1.0、sqoop1.4.6。

解压hadoop，建立软连接ln -s /opt/apps/cdh-5.13.2/hadoop-2.6.0-cdh5.13.2进入./etc/hadoop中，

第一步：配置hadoop-env.sh

export JAVA\_HOME=/opt/apps/jdk

export HADOOP\_PID\_DIR=/opt/apps/cdh-5.13.2/hadoop/PIDresource，如图2-30与2-31。

男子的脸部特写与配字

描述已自动生成

图2-30 配置JAVA\_HONE

手机屏幕截图

描述已自动生成

图2-31 配置HAOOP\_PID\_DIR

第二步：配置mapred-env.sh如图2-32。

图片包含 游戏机

描述已自动生成

图2-32 配置mapred-env.sh

第三步：配置yarn-env.sh如图2-33。

男子的脸部特写与配字

描述已自动生成

图2-33 配置yarn-env.sh

第四步：配置core-site.xml文件如图2-34。

<property>

<name>fs.defaultFS</name>

<value>hdfs://hadoop101:8020</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>/opt/apps/cdh-5.13.2/hadoop/hdfs/tmp</value>

</property>

手机屏幕截图

描述已自动生成

图2-34 配置core-site.xml

第五步：配置hdfs-site.xml文件如图2-35。

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>/opt/apps/cdh-5.13.2/hadoop/hdfs/name</value>

</property>

<property>

<name>dfs.namenode.data.dir</name>

<value>/opt/apps/cdh-5.13.2/hadoop/hdfs/data</value>

</property>

<property>

<name>dfs.permissions.enabled</name>

<value>false</value>

</property>

</configuration>

手机屏幕的截图

描述已自动生成

图2-35 配置hdfs-site.xml

第六步：创建mapred-site.xml文件，执行命令cp mapred-site.xml.template mapred-site.xml。编辑mapred-site.xml文件如图2-36。

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

手机屏幕截图

描述已自动生成

图2-36 配置mapred-site.xml

第七步：配置yarn-site.xml文件如图2-37。

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

手机屏幕截图

描述已自动生成

图2-37 配置yarn-site.xml

第八步：配置slaves指定datanode节点，将localhost改成hadoop101。修改环境变量文件vi /etc/profile如图2-38，并使其生效。

图片包含 文字, 游戏机

描述已自动生成

图2-38 配置hadoop环境变量

（16）解压CDH中的hive1.1.0点到/opt/apps/cdh-5.13.2下，建立软连接ln -s /opt/apps/cdh-5.13.2/hive-1.1.0-cdh5.13.2 hive，进入conf目录下编辑hive-site.xml文件，

<configuration>

<property>

<name>hive.metastore.uris</name>

<value>thrift://hadoop101:9083</value>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/hive</value>

</property>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop101:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123456</value>

</property>

</configuration>

配置如图2-39。

手机屏幕截图

描述已自动生成

图2-39 配置hive-site.xml文件

（17）配置hive-env.sh文件如图2-40。

手机屏幕截图

描述已自动生成

图2-40 配置hive-env.sh文件

（18）将mysql驱动包放入到hive/lib目录下，hive的元数据需要存放在mysql中。配置hive的环境变量如图2-41，更新环境变量source /etc/profile。



图2-41 配置hive的环境变量

（19）解压CDH组件中的sqoop到/opt/apps/cdh-5.13.2下，建立软连接ln -s /opt/apps/cdh-5.13.2/sqoop-1.4.6-cdh5.13.2 sqoop。将事先准备好的java-json.jar包上传至sqoop的lib目录下，复制hive/lib中的hive-common-1.1.0-cdh5.13.2.jar、hive-exec-1.1.0-cdh5.13.2.jar与mysql的驱动包mysql-connector-java-5.1.28.jar至sqoop的lib目录下，拷贝hive中的hive-site.xml文件放入sqoop conf目录下。

配置sqoop的环境配置文件：mv ./sqoop-1.4.6-cdh5.13.2/conf/sqoop-env.template.sh ./sqoop-1.4.6-cdh5.13.2/conf/sqoop-env.sh。编辑：vi ./sqoop-1.4.6-cdh5.13.2/conf/sqoop-env.sh如图2-42。

图片包含 游戏机, 文字

描述已自动生成

图2-42 配置hive-env.xml文件

（20）配置sqoop的环境变量如图2-43，更新环境变量source /etc/profile。



图2-43 配置sqoop环境变量

（21）格式化hdfs：hdfs name -format；生成clusterID、blockpollID、namenode目录与datanode目录。启动hdfs：start-dfs.sh，启动yarn集群：start-yarn.sh，启动hive metastore服务以及hive hiveserver2服务，进入/opt/apps/azkaban/executor/下启动Azkaban executor ：bin/azkaban-executor-start.sh，进入/opt/apps/azkaban/server/下启动Azkaban webserver服务：bin/azkaban-web-start.sh。就此本实验所有进程全部搭建完毕且成功启动，如图2-44。

手机屏幕截图

描述已自动生成

图2-44 进程图

# 3 外卖企业中基于Hadoop的数据仓库系统问题分析

## 3.1 传统型数据仓库分析

### 3.1.1 传统型数据仓库功能分析

在创建之初数据仓库的目标是创建面向分析的集成的大数据平台，以数据的支持辅助企业决策与规划。实际上数据仓库自身并不产生任何数据，并且自身也不消耗任何数据。数据仓库中数据完全来自于外部环境，提供“可用的”数据供外部人员使用，因此人们不把它命名为“工厂”而命名为“仓库”。传统的数据仓库基础构建最主要的就是数据的流入与流出过程，可分为数据源层、数据仓库层、数据应用层自底向上如图3-1。

图片包含 游戏机, 桌子, 画

描述已自动生成

图3-1 传统数据仓库架构图

1、数据来源

如图可知基于传统数据仓库的架构，数据仓库的数据来源于不同路径的数据源且提供了各式各样的数据应用场合，以部分半结构化数据以及结构化数据为主，个别非结构化数据在转化之后以某一种形式存在于传统的数据仓库里，数据自底向上传入传统数据仓库，再在上层提供相关数据的应用。但传统的数据仓库只在中层集成化了数据管理的平台。

2、ETL过程

以各种数据源为目标从中获取数据与在数据仓库内部的数据转换等都是传统数据仓库的ETL (抽取Extra，转化Transfer，装载Load)过程，传统数据仓库中的流水线也是基于ETL生成的，因此ETL被人们称为传统数据仓库的血液，ETL保持了传统数据仓库里数据的诞生、清除等过程。在传统数据仓库中每日维护及管理工作的绝大部分时间就是为了维持ETL正常与稳定执行。

3、数据仓库层

经ETL过程后的结构化数据储存在传统数据仓库中，而传统的数据仓库中无需存储全部原始的数据。但是传统的数据仓库中却需要用到细节数据，且经整理和转换之后导入的数据必须是面向主题的。传统的数据仓库在基于保护细节数据的条件上对数据进行了处理，使数据能够真正地运用在分析中。传的统数据仓库主要作用有：

聚合数据。存在特殊需求的简单聚合或是多维数据的聚合，且体现在多维的数据模型当中，处理后的数据可直接展示在数据报表中。

多维的数据模型可以让决策人员有多种角度、多种层次地分析条件。例如地域维度、时间维度等等创建的企业销售星形模型与雪花模型，可以提供各种地域维度及时间维度的混合查询，以及对于地域维度于时间维度的细粒度划分。

企业业务模型就是用于对数据的分析以及决策的支持创建的数据模型。例如关联推荐模型、RFM分析模型、用户评价模型等等，亦或是为企业决策人员提供库存模型、线性规划模型等等，并且在数据挖掘的前中期对于数据的处理在这里也可以完成。

而从操作数据以及为专业的团队服务的数据源中收集数据的仓库称为数据集市。以范围来划分，从全企业的数据库到传统的数据仓库，或是其他数据源中提取出来的数据中心也可称为数据集市。数据中心的特点在于符合专业从业人员的业务需求，为相关人员提供数据内容、数据分析、数据表现及数据的易用性方面。

4、数据应用层

在维护细节数据的基础上传统的数据仓库对数据进行来处理，让数据能应用在企业的数据分析中。在数据仓库层完成了数据的处理及存储之后，数据也将被使用到数据的应用层，进行企业级系统分析或数据挖掘等。其核心任务有：

为报表系统提供数据支持与应用，使用聚合数据以及多维的分析数据，将数据展示到报表系统中，为企业管理人员提供最为简单与直观的数据展示。

传统的数据仓库中全部数据包括聚合数据、细节数据、多维数据以及分析数据等，都应开放即席的查询模式，有了即席查询就能提供各种活灵活现的数据获取途径，相关人员可根据自身所需查询数据，并且提供了可导入Excel等文件的能力。

数据的分析大多可为企业建立业务模型，也可运用聚合数据进行比较分析、趋势分析与其他相关分析等等。大部分维度数据模型都有多维分析的数据基础，且可以在细节数据当中获得个别的抽样数据进行分析也是常见的途径之一。

传统数据仓库可为数据挖掘工具例如SPSS、SAS等提供数据的接口，在个别较复杂的算法中在数据挖掘可使数据展示出意想不到的结果。并且数据挖掘可基于传统数据仓库中已搭建好的业务模型展开，许多时候数据挖掘也会直接从细节数据方面下手。

### 3.1.2 传统型数据仓库功能中的缺点

在图3-1中可得知传统的数据仓库结构大多数都以三层架构为主，最主要处理的数据源是结构化的数据，其能够处理的数据类型十分有限，这也是传统数据仓库面对海量的异构数据时处理的局限性，后果直接影响到企业在数据采集时存在对于企业有价值数据的缺失。

并且传统数据仓库层限制了数据库的类型，只能选择关系型数据库，不能支持非关系型数据库，在灵活性及拓展性等方面都存在部分的缺失。

而传统数据仓库的单节点服务器问题也是急于解决的。传统数据仓库常常使用盘阵加小型机的高性能一体式搭建，并且超过半数以上的传统数据仓库厂家为企业提供了一体式机器的搭建方案。一体式机器机聚集了存储、软件及服务器等设备，其管理、安装、配置较为简易，软硬件可做到协同均衡，也可提供高可用服务，企业可最大程度的得到厂商的支持。但是机器的硬件相对过于高昂，且时常遭遇I/O效率低下以及单点故障问题，它需要增加—台同样配置机器用来提供高可用性，但其存储能力与计算能力还是与集群相差甚远。

大数据的体系时各式各样的，相较于传统数据仓库来说现代社会数据量太过巨大。特别需要注意的是，人们偏向于阅读大量的数据来查询适合自己的信息。在得到信息之后也许会选择遗弃个别数据，在特殊情况下，大部分数据的存储时长也许会相对较短。假如人们选择保存所有数据，那就需要运用相比传统数据仓库性价比跟高的解决方案来存储非结构化数据，便于往后重新取出历史数据进行分析，这就是基于hadoop的数据仓库架构诞生的原因。

## 3.2 外卖企业中基于Hadoop的数据仓库系统方案分析

### 3.2.1 外卖企业业务数据处理新需求

在上一节中提出了基于传统的数据仓库在如今数据井喷式爆发的年代已没有办法为企业提供有效的数据分析与计算。虽然传统型数据仓库框架仍然在不停进化，例如TeraData就已经弥补了上一节中分析的多个缺点，但是还存在种种缺陷，传统型数据仓库在处理海量非结构化数据时的处理效率及方案未能达到人们的期望。现如今近半数以上的外卖企业都建立了各自的企业级数据仓库用以数据挖掘以及数据分析，相比运用传统型数据仓库的企业来说，它们在大数据时代中的优势就是增加了许多曾经没有办法获得的数据，就外卖企业而言，哪家外卖企业掌握的数据更多，哪家企业就有更大的概率挖掘或分析出对市场或对产品有价值的信息，从而掌握市场的主动权。

在外卖企业中如今以Hadoop为基础提出了数据仓库的新型搭建方式，将数据分析变得更加灵活且高效。所以各大外卖企业都开始构建自家企业的大数据平台来对应如今数据喷发年代的应用需求。

与此同时又一个问题出现在众多企业面前，企业花费了大量的财力物力人力资源来构建的传统型数据仓库在如今逐渐形成了孤立的模块，于是众多企业急于开发出一种共用的构建方式能够使传统型数据仓库与大数据平台结合在一起，构建统一集成的数据储存以及数据处理框架，用来继承传统型数据仓库的功能，并且能够应付大数据时代所带来的冲击。因此，本文提出基于Hadoop的外卖企业业务数据分析系统的设计与实现，本次方案在已有的传统型数据仓库基础上增添Hadoop平台，弥补传统型数据仓库在面对海量数据的存储、处理等方面的缺点，凭借Hadoop的大数据处理能力以及数据仓库自身优点打破传统型数据仓库在存储以及计算方面的瓶颈，为辅助外卖企业决策提供数据支撑。

### 3.2.2 外卖企业中基于Hadoop的数据仓库系统新架构

在2011年我国的线上外卖开始出现萌芽，到2012年各大企业才逐渐开始转攻线上外卖行业。饿了么、美团、到家美食会等线上外卖平台先后成立，在2013年线上外卖行业迎来高速发展时代。到了2017年我国的线上外卖市场进入整合兼并时期，同年8月饿了么与百度外卖正式合并，在市场逐渐形成饿了么与美团外卖的两极并立的格局。

而本次实验是外卖行业类的自营业务的运营分析平台的一部分，通过对业务数据的分析，获得用户主题和订单的相关分析情况，生成用户的访问模型和用户的订单行为模型，对访问时段和下单区域分布进行分析。

用户通过使用浏览器(PC端)和app对外卖平台进行访问，记录用户的注册信息，通过调研平台和数据挖掘记录用户的扩展信息，业务系统对用户的每次点击和下单信息记录到数据库中，并提供一台从库将主库的业务数据实时同步到从库中，同时分配对从库的只读访问权限，运营平台通过sqoop将数据抽取到运营分析平台，运营分析平台进行指标的计算，将结果存入到mysql中如图3-2。

截图里有图片

描述已自动生成

图3-2 新型数据仓库架构图

Hadoop与传统型数据仓库两者的分别，主要在于传统型数据仓库常常构建在一个关系型的数据库当中，属于单节点模式，此数据库仅仅起到了中央储存的意义。与之相比Hadoop 以及HDFS分布式文件系统跨跃了数个节点，并且采用处理海量异构数据的方法，而单台机器的数据库远远达不到这种存储运算能力，不仅如此Hadoop集群可搭建在廉价的服务器之上，构建成本也相对较低。

现如今外卖企业中的非结构化数据是Hadoop主要处理的数据类型，当然在外卖企业中的Hadoop也一样可处理结构化的数据以及半结构化的数据。其重在处理外卖网页的浏览情况、外卖订单日志、线上外卖的音频视频、外卖的文本文档以及网页中的商家食品图形图像等，这些数据大多是以尺寸不同的、结构不同的、超大型的文件等为特点，都是传统型的数据仓库没有办法应对的数据格式。这些数据凭借数据流的格式储存在HDFS分布式文件系统中，HDFS可以保证它可用性以及安全性，于此同时以Hadoop平台可以保持其良好的数据处理能力。这些超大型数据，已经远超于结构化的数据。而这些超大型的数据最后的使用途径是应用开发，通过结合CDH其它组件，在Hadoop平台上运行应用的开发，从这些海量数据中挖掘对外卖企业有价值的数据。

依据上述总结，线上外卖外卖数据主要产生于交易数据、业务的流水数据、客户反问等细节数据，通过进行ETL处理后将数据储存在基于Hadoop数据仓库之中，并最终应用于上层的数据集市。最为典型的方式是通过Hadoop将大数据预先进行处理，之后通过基于Hadoop的数据仓库工具hive主要通过hql语句实现对数据的分析，随后将这些处理过的数据导入到关系型数据库中，这种方法就是当下外卖企业中较为流行的处理业务数据的方式。

# 4 外卖企业业务系统中基于Hadoop的数据仓库详细设计

## 4.1 系统分析外卖企业业务需求指标

### 4.1.1 业务系统中功能性需求与非功能性需求分析

在上一章中对外卖企业基于Hadoop的数据仓库新架构做出了初步的介绍与分析，本节将以需求的视角分析外卖企业中基于Hadoop的数据仓库主体功能，包括数据的采集、存储、转换分析、应用等功能性需求。

1. 功能性需求

外卖企业中基于Hadoop数据仓库的数据来源必须是各式各样的，这样才可以保证数据的完整性以及数据的全面性，可以为企业管理人员提供各个角度的视角来辅助进行决策。其数据可以来自于业务系统数据库的数据，也可以是来自日志系统的数据，还可以是来自爬虫系统的数据等等，而该系统必须支持多种数据源及数据类型的获取。

超大型文件与数据可以存储在Hadoop集群中，并且Hadoop集群也可以存储多种类型的数据，但获取的数据并不一定适合直接存储在Hadoop集群中，个别源数据有可能需要预先进行处理，而后再存储到系统当中。

在数据仓库中进行分层规划，最终将用户的基本信息表和用户的扩展表进行合并生成用户基本宽表user\_basic，包括信息：用户id，登录名，用户性别，出生日期，年龄，星座，省份，城市，城市等级，邮箱，邮箱运营商，手机号，手机号段（前三位），手机运营商，注册时间，登录ip，登录来源，邀请人，会员积分，已使用积分，会员等级名称，是否黑名单，是否结婚，学历，月收入，职业，是否孕妇，是否有小孩，是否有车，使用手机品牌，使用手机等级，使用手机种类数量，更换手机数量，是否属于回购用户，用户账户数量，用户忠诚度，用户购物类型，体重，身高。

根据用户的访问日志信息，统计以下指标，结果存入用户访问模型表user\_visit。

表4.1 用户访问模型需求指标

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PC端最近访问时间 | PC端最近访问使用的session | PC端最近使用的cookie | PC端最近的pv量 | PC端最近访问的浏览器 | PC端最近访问的os | 第一次pc端访问的日期 | 第一次pc端访问的session |
| 第一次pc端访问的cookie | PC端第一次访问的pv | PC端第一次访问的浏览器 | PC端第一次访问的os | PC端连续7天访问次数 | PC端连续15天访问次数 | PC端连续30天访问次数 | PC端连续60天访问的次数 |
| PC端连续90天访问的次数 | 近30天pc端访问的天数 | 近30天pc端的pv | 近30天pc端每天的平均pv | 近30天0到5点pv数量 | 近30天6到7点pv数量 | 近30天8到9点pv数量 | 近30天10到11点pv数量 |
| 近30天12到13点pv量 | 近30天14到15点pv量 | 近30天16到17点pv量 | 近30天18到19点pv量 | 近30天20到21点pv量 | 近30天22到23点pv量 | 近30天使用不同ip量 | 近30天最常用的ip |
| 近30天使用的cookie的数量 | 近30使用最常用的cookie\_id | 近30pc最常用浏览器 | 近30天使用最常用系统 | 最近一次app端访问的日期 | 最近一次访问app端的名称 | 最近一次app端访问的os | 第一次app端访问日期 |
| 第一次app端访问名称 | 第一次app端访问os | app端第一次访问ip | app 端近7天访问pv数 | app端 近15天访问pv数 | app 端近30天访问pv数 | app端近60天访问pv数 | app端近90天访问pv数 |
| app端近30天0到5点访问pv数 | app端近30天6到7点访问pv数 | app端近30天8到9访问pv数 | app端近30天10到11访问pv数 | app端近30天12到13点访问pv数 | app端近30天14到15点访问pv数 | app端近30天16到17点访问pv数 | app端近30天18到19点访问pv数 |
| app端近30天20到21点访问pv数 | app端近30天22到23点访问pv数 | 最近一次访问的ip | 最近一次访问的城市 | 最近一次访问的省份 | 第一次访问的ip | 第一次访问的城市 | 第一次访问的省份 |

根据用户的购物订单信息，统计以下指标，结果存入用户订单模型表user\_order

表4.2 用户订单模型需求指标

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 首次下单时间 | 近一次下单时间 | 首单距今时间 | 尾单距今时间 | 近30天订单数（不含退拒） | 近30天订单金额（不含退拒） |
| 近60天订单数（不含退拒） | 近60天订单金额（不含退拒） | 近90天订单数（不含退拒） | 近90天订单金额（不含退拒） | 近30天订单数（含退拒） | 近30天订单金额（含退拒） |
| 近60天购买次数（含退拒） | 近60天购买金额（含退拒） | 近90天购买次数（含退拒） | 近90天购买金额（含退拒） | 近90天的客单价（含退拒） | 最大消费金额 |
| 最小消费金额 | 累计消费次数（不含退拒） | 累计消费金额（不含退拒） | 客单价（不含退拒） | 最常用收货地址 | 最常用支付方式 |
| 退货商品量 | 退货总金额 | 拒收总数量 | 拒收总金额 | 最近一次退货时间 | 学校下单总数 |
| 公司下单总数 | 家里下单总数 | 凌晨下单总数 | 上午下单总数 | 中午下单总数 | 下午下单总数 |

注：除上述指标外还包括晚上下单总数，指标口径：凌晨:0-05 上午:06-12 中午:13-15 下午16-20 晚上：21-24。

1. 非功能性需求

本系统运用外卖企业中较为流行巧妙地将Hadoop平台与数据仓库结合的架构，其目的就是为了创立集中的数据存储与数据处理架构。因此本系统之后可能真正被运用到外卖企业的商业体系之中，所以还需要定义非功能性需求：

（1）共通性：本次实验设计的外卖企业业务分析系统可以应用于市面上各式各样不同版本的Linux系统。用过运用CDH5基于Hadoop2.0版本，降低了集群规模庞大时搭建Hadoop集群的复杂度，减小了工作量。而Hadoop 1.0整体稳定性较弱，还可能会出现个别API失败调用等问题。

（2）便捷性：本次实验所设计的系统旨在为相关开发人员与系统的管理人员构建的，可通过友好的交互界面，便于相关开发人员直观了解任务执行流程，同时可通过使用大致理解系统的功能模块与架构。

（3）安全的：本次实验设计的系统必须保障企业使用的安全性，其中重在维护系统数据的安全性，设定严格的使用执行权限与读写规则，确保数据不易丢失与泄露。

（4）易扩展性：本次实验设计的系统必须建立在CDH的Hadoop平台上与Linux系统，而它们两者也在发展与变化，因此本系统需要达到基本要求的同时也要尽可能的跟随时代发展脚步，为各式各样的变化后的Hadoop平台与Linux系统提供支持，可在本系统内部预留接口以此应对跟新换代后的新型版本。

### 4.1.2 外卖企业中基于Hadoop的数据仓库系统开发规范

1、数据库命名

命名规则：数仓对应分层\_{业务线|业务项目} ；

命名示例：ods\_nshop、dwd\_nshop、dws\_nshop、dim\_nshop、ads\_nshop。

2、数仓各层对应的数据库

ods/sda层 => sda/ods\_{业务线|业务项目}（原始数据）；

dw层 => dwd\_{业务线|业务项目}（主题）+ dws\_{业务线|业务项目}(基于主题宽表汇总）；

dim层 => dim\_维度 （维表库）；

ads层 => ads\_{业务线|业务项目}（应用统计指标等）；

middle层 => mid\_{业务线|业务项目}（中间库）；

临时数据 => temp\_{业务线|业务项目}（临时库）。

3、表命名

（1）原始数据层：

ods\_{业务线|业务项目}\_{数据来源类型}\_{业务}；

ods\_{业务线|业务项目}\_{数据来源类型}\_{业务}\_{时间粒度}\_delta （delta代表增量，主要用于数据同步方向产生的原始数据表）；

命名示例：

ods\_nshop\_01\_useractlog XX用户日志原始数据表；

ods\_nshop\_02\_user XX用户表（全量）；

ods\_nshop\_02\_user\_delta XX用户表（增量）；

ods\_nshop\_02\_user\_hh\_delta XX用户表（小时级增量dd天）。

（2）主题或事实数据层：

命名规则：dwd\_{业务线|业务项目}\_{主题域}\_{子业务}；

命名示例：

dwd\_nshop\_user\_logproview XX用户产品浏览日志事实表；

dwd\_nshop\_user\_comment XX用户关注事实表。

（3）主题或事实汇总层：

命名规则：dws\_{业务线|业务项目}\_{主题域}\_{汇总相关粒度}\_{汇总时间周期}；

命名示例：

dws\_nshop\_user\_order\_nd XX用户订单汇总N天统计表；

dws\_nshop\_user\_cmtpro\_nd XX用户产品关注汇总N天统计表。

（4）维表层：

命名规则：dim\_{业务线|业务项目|pub公共}\_{维度}；

命名示例：

pub\_date 时间维表；

dim\_pub\_area 地区维表；

dim\_pub\_category 商品分类。

（5）集市层：

命名规则：ads\_{业务线|业务项目}\_{统计业务}\_{报表form|热门排序topN}；

命名示例：

ads\_nshop\_order\_form（订单统计表）；

ads\_nshop\_orderpay\_form 订单支付统计。

注释：如果业务名称较长可以简写 如 ods\_01\_useractlog。

（6）数据来源代码(ods层)

01 => hdfs数据ods\_nshop.ods\_01\_action\_log 行为日志表；

02 => mysql数据ods\_nshop.ods\_02\_user 用户表；

03 => redis数据；

04 => mongodb数据；

05 => tidb数据。

可参考图4-1。

地图的截图

描述已自动生成

图4-1 数据仓库开发规范流程图

## 4.2 外卖企业业务系统中的数据仓库具体设计

本小节主要介绍外卖企业业务系统具体设计，数据仓库功能模块的详解，数据仓库中分层的划分以及系统的执行流程。

### 4.2.1 业务系统中的数据仓库功能模块设计与分层解析

根据上节中提出的的需求，分析之后将系统分成了如图4-2的功能系统模块。本系统分成采集数据模块、存储数据模块以及应用数据的三大模块，而应用数据模块包含数据集市与大数据平台的应用，数据集市可为各个部门提供不同的数据需求，以及为企业中业务系统数据库中补充数据，大数据平台的应用可以提供海量的数据存储以及满足各种并发的数据查询。

手机屏幕截图

描述已自动生成

图4-2 系统中的模块功能图

如图4.2所示，本系统分为三大模块：采集数据、存储数据以及应用数据。采集数据模块主要由三部分组成：Web数据采集、企业内部业务数据库数据导入以及爬虫系统等数据源。

### 4.2.2 业务系统中的数据仓库具体流程设计

# 5 基于Hadoop平台决策树C4.5算法的实现

基于Hadoop平台的决策树C4.5算法可以解决传统C4.5算法不能处理大规模数据的问题。

## 5.1 决策树算法在Hadoop平台上的实现

通过hadoop jar dt.jar program.DecisionTreeDriver /test-data/names /test-data/data /tmp /test-data/model 指令指明使用的jar包（dt.jar）、程序的入口类（program.DecisionTreeDriver）、存储数据属性元信息文件的路径（/test-data/names）、训练集数据文件路径（/test-data/data）、当前统计结果所在文件夹路径（/tmp）以及模型文件存储路径（/test-data/model）。

该决策树算法主要设计DecisionTreeDriver类、DecisionTreeMapper类和DecisionTreeReducer类来实现算法的主要过程。

**1. DecisionTreeDriver类**

该类为程序的入口，初始化JobTracker，通过setMapperClass()、setReducerClass()方法分别设置DecisionTreeMapper类和DecisionTreeReducer类作为Map、Reduce函数的实现类。具体步骤如下：

（1）通过指令得到各种路径；

（2）loadAttributeRange(attributesMetaInfoPath)：载入属性元信息；

（3）Queue<datatype.Rule>：当前层及下一层的划分规则队列；

（4）判断是否包含Root节点，用于甄别是否是初始执行；

（5）对每一层进行迭代，直到无法继续构建为止，步骤如下：

① 准备当前轮迭代的环境变量数据；

② outputNodeRuleQueueToFile()：将下一层的划分规则队列中保存的当前轮迭代所在的层的节点信息写入文件；

③ 将当前层的队列指针指向上一轮迭代的输出；

④ 判断是否有根节点，如果有，则对于根节点需要单独处理，向队列中插入一个空白的节点，作为根节点；

⑤ 判断一下当前层数上是否有新的节点可以生长，如果已经没有新的节点供生长了，决策树模型已经构造完成，退出构造while循环；

⑥ runMapReduceJob()：然后继续运行，说明当前层还有节点可供生长，运行Mapreduce作业，对当前层节点分裂信息进行统计；

⑦ loadStatisticInfo()：从输出结果中读取统计好的信息；

⑧ 对于当前层的每个节点进行处理，节点统计信息里应该包含1到|Q|这|Q|个规则。

i. 首先获取一些当前节点的统计信息备用；

ii. findBestSplit()：然后查找最佳分裂属性splitAid，如果无法找到最佳分裂属性，即属性分裂不能够提供信息增益了，那么就停止构建新的子节点。如果最佳分裂属性“splitAid=0” 则将“当前规则->当前多数标签”这个规则加入模型中：model.add()，然后继续处理下一个当前层的节点；

iii. satisfyLeafNodeCondition()：分裂当前节点，判断是否满足终止条件，增加规则。如果是叶子节点，则将叶节点加入模型，否则是中间节点，把这个规则加入到新的Queue中；

（6）不断的向深层扩展决策树，直到无法继续构建为止 ；

**2. DecisionTreeMapper类**

该类继承Mapper类，用来实现Map过程。输入训练数据集D以及条件集队列文件Q（通过DistributedCache方式传入），然后从HDFS中读入JobTracker的输出结果，将<key,value>对，其中key是一个复合类型，key=<条件号#属性号，属性值，元素类标号>，value=1，即<<条件号#属性号，属性值，元素类标号>,1>作为数据对象读入。计算该数据对象的属性的增益率，并将结果形成的<key,value>对输出，其中key是一个复合类型，key=<条件号#属性号，属性值，元素类标号>，value为输入的value列表中所有值的和。根据value的值才可以计算信息增益率。

决策树算法的Mapper类：

输入：训练数据集D（D是文本文件，每一行是一个训练样本，各个属性字段之间用

逗号分隔）；条件集队列文件Q（通过DistributedCache方式传入）。

输出：<key,value>对，其中key是一个复合类型， 具体为key =<条件号#属性号，属性值，元素类标号> value = 1。

（1）setup(Context context)：配置函数，主要负责从DistributedCache中读取条件集队列，然后载入条件队列信息：loadQueueFile()；

（2）loadQueueFile()：从filePath中读取Queue文件，判断ruleQueue文件是否为空，如果为空，那么此时应该正在处理根节点，然后为根节点生成一个空白的规则，加入队列，确保队列非空；

（3）Map函数：

输入：<key1,value1>

输出：<key2,value2>

① 将读入的行数据解析成记录，且最后一个属性为类标号，并读取value1的值；

② 如果训练样本符合规则，则遍历每一个属性，values2为输入的value1列表中所有值的和；

③ context.write(key2,value2)；

④ 结束。

（4）判断一个样本记录是否符合规则要求：isFitRule()；

**3. DecisionTreeReducer类**

该类继承Reducer类，用来实现Reduce过程。获取各节点Map处理过程中的局部结果，然后对该结果进行汇总、排序，选出value的最大值，得到最佳属性分裂点，重新分割数据集并写回结果。

决策树算法的Reducer类：

输入：<key,value>对，其中key是一个复合类型， 具体为key = <条件号#属性号，属性值，元素类标号>，value = 1。 功能：将同一个key下的所有value值相加，即进行求和操作。

输出：key仍保持输入的Key，value是输入的value列表中所有值的求和。

该Reducer是MapReduce应用中常见的一种Reducer的类型，其功能是将同Key的所有Int类型的Value进行求和，作为该Key的新的Value。Hadoop为这种类型的应用提供了一个标准的“IntSumReducer.class”，在实现时，可以直接在配置中使用该Reducer，而不再需要单独编写Reducer代码。

Reduce操作简述：

输入：<key2,value2>，即与Map的输出相同

输出：<key3,value3>

① 利用输入的键值对，计算各属性的增益率；

② 创建节点A；

③ for(value2){求最大增益率，使节点A为最大增益率的属性；}

④ 对非叶子节点的分支做分割处理；

⑤ context.write(key3,value3)，key3为当前节点A的各个分支节点；

⑥ 结束。

## 5.2 实验结果分析

本次实验数据选自UCI（http://archive.ics.uci.edu/ml）数据库中的Breast Cancer Wisconsin（Original）数据集，该样本有10个属性，共有699个样本，共15KB，用于乳腺细胞学检查的医学诊断（乳腺癌的类别：良性或恶性）。对Breast Cancer Wisconsin（Original）数据集的描述如下表5.1所示。

表5.1 Breast Cancer Wisconsin（Original）数据集

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据集特征： | 多变量 | 元组数： | 699 | 领域： | 生活 |
| 属性特征： | 整型 | 属性个数： | 10 | 捐献日期 | 1992-07-15 |
| 相关任务： | 分类 | 缺失值： | 无 | 网络点击量： | 206105 |

对Breast Cancer Wisconsin（Original）数据集每个属性的说明如表5.2所示。

表5.2 Breast Cancer Wisconsin（Original）数据集属性说明

|  |  |  |
| --- | --- | --- |
| 属性名 | 属性取值 | 说明 |
| ClumpThickness | 1~10 | 肿块厚度 |
| CellSize | 1~10 | 细胞大小均匀度 |
| CellShape | 1~10 | 细胞形状的均匀性 |
| MarginalAdhesion | 1~10 | 边缘粘 |
| SingleSize | 1~10 | 单上皮细胞的大小 |
| BareNuclei | 1~10 | 裸核 |
| BlandChromatin | 1~10 | 乏味染色体 |
| NormalNucleoli | 1~10 | 正常核 |
| Mitoses | 1~10 | 有丝分裂 |
| Class | 2,4 | 类：（良性2级，恶性4级） |

数据集中类分布为良性65.5%,恶性为34.5%。

（1）时间对比

将Breast Cancer Wisconsin（Original）数据集共15KB，使用复制的方式将其分别扩大为104KB、312KB、934KB和2.13M大小的数据集。测试比较不同大小的数据集的运行时间，结果统计如图5-1所示。

处理时间（s）

20

22

24

26

28

30

32

34

36

38

40

42

44

46

15

104

311

934

2180

数据量（KB）

1台

3台

2台

图5-1 不同数据量处理的运行结果

由图5-1可知，在相同集群下，数据的处理时间随着数据集的增大而增大，不同集群相同数据量下，处理时间随着集群的增大而有效缩短了。

（2）准确率

将Breast Cancer Wisconsin（Original）数据集进行划分80%（559条记录）为训练数据集，20%（140条记录）为测试数据集，最终发现当有一个节点时，这140条记录中有29条记录与所得的的决策树不符；当有有两个节点时，这140条记录中有30条记录与所得的的决策树不符；当有有三个节点时，这140条记录中有33条记录与所得的的决策树不符。实验结果如表5.3所示。

表5.3 算法的运行结果

|  |  |
| --- | --- |
| 节点数 | 准确率（%） |
| 1 | 94.49 |
| 2 | 94.22 |
| 3 | 94.09 |

由表5.3可知，随着节点的增加，同一数据量，算法的处理时间减少，但是算法的准确率维持在一定的水平，变化不大。

其他算法也有用到该数据集做研究分析的，有实验研究表明，如用聚类算法或运用RelieefF算法（求个特性的特征权重）和K-means聚类算法（求满足目标函数的最小的K个聚类）相结合的算法对该乳腺癌数据集进行实验，其中只用聚类算法得到的实验准确率为91.8% ，而运用RelieefF算法和K-means聚类算法相结合结果的准确率达到94.11%。

然后利用决策树C4.5算法从训练集中得到诊断模型，如图5-2至图5-3所示。

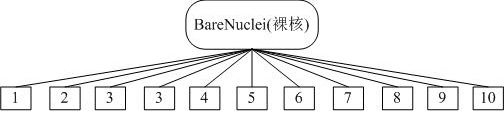


图5-2 决策树的总结构图



图5-3 决策树属性6=1的分支

上图5-2是先判断出训练集中所有属性的信息增益率的最大值，并对所有属性值进行分析运算。图5-3是当BareNuclei（裸核）为1时，如果不是叶节点，求剩余属性的最大增益率，然后在对其最大增益率属性的各属性值进行分析运算，以此类推。如果是叶节点，则判断决策属性值。以此类推得到关于乳腺癌诊断的分类决策树。

由决策树C4.5算法提取出来的诊断规则描述简单，应用方便。决策树方法在辅助FNAC(细针穿刺细胞学检查简称诊断细胞学)诊断乳腺癌的同时，还可判断各参数对乳腺癌诊断贡献的大小。从形成的决策树模型可见，裸核对乳腺癌诊断起决定性作用，细胞大小均匀性和有丝分裂则可作为诊断的重要指标，因此，决策树方法是一种简单便捷、可行性高的计算机辅助诊断方法，可从病例中或者检查报告中自动提取诊断规则，来判断疾病的类别或属性。这种方法具有较广泛的实用价值，也可以应用在其它疾病的诊断与研究上。

# 结论

突出有价值的工作、意义

提出了新的观点、实现的功能

本设计可以解决xx问题、为xx提供企业管理、决策的依据，提高企业的效益，

我的工作内容，意义

这篇论文做了什么

总结、评价

本文主要研究了大数据处理领域的热门技术Hadoop，实现了基于Hadoop平台的决策树C4.5算法的实现，搭建了Hadoop环境，对决策树C4.5算法进行了分析，并将大数据与开源平台Hadoop相结合，从而实现了数据的有效存储和高效处理，解决的传统决策树算法不能处理海量数据的问题。

本文的主要工作总结为以下几点：

（1）分析并简单介绍了本课题的背景和意义，总结了课题在国内外的研究现状。

（2）简单介绍了Hadoop技术以及Hadoop的核心HDFS和MapReduce的体系结构及相关技术，最后对Hadoop平台的两种搭建过程及搭建过程中可能遇到的问题及解决方法进行了简单的介绍。

（3）对数据挖掘进行了了解与分析，然后了解各分类算法，最后对传统ID3和C4.5算法进行介绍和分析，总结C4.5算法、MapReduce框架以及Hadoop平台的特点，提出了基于Hadoop平台的决策树C4.5算法的实现方案。通过算法的实现，以及实验分析，验证了基于Hadoop平台的决策树C4.5算法可以高效处理大规模数据集。

本文未完成的工作：

1. 进一步优化完善Hadoop平台，实现更多的数据挖掘算法，并提高算法在Hadoop环境下的效率、准确度及性能。
2. 进一步完善C4.5算法，对决策树进行修剪操作，对数据集中的噪声和离群点进行进一步的处理。
3. 在实际应用中，Hadoop平台一般是个庞大的异构集群和异地集群，之后可以深入研究异构环境下的数据集的处理，来提高算法的适用性。

# 致 谢

此刻，我要向大学五年的时间里给予我很多帮助和鼓励的各位老师、朋友、同学和家人表达最真挚的感谢。因为有了他们的帮助，我学到了很多的知识，并顺利的度过了这五年的大学生活。

首先我要感谢我的指导老师曲文龙老师。他在毕业设计的选题、开题、设计到论文的撰写、修改以及定稿都给予了深刻而细致地指导。曲老师还有耐心，态度严谨，在论文撰写阶段，一次次的指出我论文中的错误和不足之处，在老师的指导下，我将这些问题一一改正。

另外，还要感谢我的舍友和我的同组同学还有不认识的Hadoop交流群里的朋友们，在设计中遇到一些问题和困难，你们都耐心帮我解答，遇到一些相似的问题，我们之间进行不断的讨论与试验，学习各自的长处，使我更快地将本次设计完成。

最后诚挚的感谢各位评审老师，感谢您们的指导和提出的宝贵意见！

# 参考文献

[1] 孙媛，黄刚．基于Hadoop平台的C4.5算法的分析与研究[N] .计算机技术与发展，2014，24(11).

[2] 范东来.Hadoop海量数据处理：技术详解与项目实战[M] .北京：人民邮电出版社，

2015，3:23-45．

[3] 赵书兰.典型Hadoop云计算[M] .北京：电子工业出版社，2013:25-43.

[4] 张鑫.深入云计算Hadoop源代码分析[M].北京：中国铁道出版社，2014.8:17-61.

[5] 陆秋，程小辉.基于MapReduce的决策树算法并行化[J].桂林：计算机应用，2012，

32(9):2464-2465.

[6] 林树地.基于Hadoop的决策树分类算法研究[A].福建:华侨大学，2013:35-39.

[7] 赵振崇.基于Hadoop的决策树挖掘算法的研究[A] .兰州：兰州大学，2015.4:34-46.

[8] 刘鹏.云计算[M].北京：电子工业出版社，2010.

[9] 李力.基于Hadoop的决策树分类算法的并行化研究[A].四川：四川师范大学，2015:39-45.

[10] 卢东标.基于决策树的数据挖掘算法研究与应用[A].武汉：武汉理工大学，2008:20-30.

[11] 陈沛玲.决策树分类算法优化研究[A].长沙：中南大学，2007：17-27.

[12] 房祥飞.基于决策树的分类算法的并行化研究及应用[D]. 济南：山东师范大学，2007.

[13] 朱敏，万剑怡，王明文.基于MR的并行决策树分类算法的设计与实现[J] .广西师范大学学报：自然科学版，2011，29(1) :82-84.

[14] 杨宸铸.基于HADOOP的数据挖掘研究[D].重庆：重庆大学，2010:45-46.

[15] 累万云.云计算技术、平台及应用案例[M] .北京：清华大学出版社，2011:22-224.

[16] 唐华松，姚耀光.数据挖掘中决策树算法的探讨[M].1011-3695(2001)08-0018-02.

[17] 朱玉全，杨鹤标，孙蕾．数据挖掘技术[M].南京：东南大学出版社.

[18] 刘平，陈亚楠.数据挖掘中关联规则算法的研究[J].期刊论文: 2012-11-28.

[19] Fayyad. Data mining and knowledge diseovery on making set out of data[J].ComPutering APPlied，1996，12:20-25.

[20] 陆嘉恒.Hadoop实战[M].北京：机械工业出版社，2011.

[21] T. White. Hadoop: the definitive guide[A]. O'Reilly, (2012).

[22] 田金兰，赵庆玉.并行决策树算法的研究[J].计算机工程与应用，2001，16(5)：17-20.