学校代码：10004

密级：公 开

北京交通大學

BEIJING JIAOTONG UNIVERSITY

硕士专业学位论文

基于 HDFS 的百度多酷移动游戏数据平台的设计与实现

作者姓名 许仕霖

工程领域 软件工程

指导教师 卢 苇 教授

培养院系 软件学院

二零一八年五月

万方数据

Y3496764

北京交通大学

硕士专业学位论文

基于 HDFS 的百度多酷移动游戏数据平台的设计与实现

Design and Implementation of Baidu Duoku Mobile Game Data Platform Based on HDFS

作者： 许仕霖

导师：卢苇

北京交通大学

2018 年 5 月

万方数据

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：许仕豪　　导师签名：

签字日期：2018 年5 月29日　　签字日期：2018 年5 月

万方数据

学校代码：10004 密级：公开

北京交通大学

硕士专业学位论文

基于 HDFS 的百度多酷移动游戏数据平台的设计与实现

Design and Implementation of Baidu Duoku Mobile Game Data Platform Based on HDFS

作者姓名：许仕霖 学　　号：16126229

导师姓名：卢苇 职　　称：教授

工程硕士专业领域：软件工程 学位级别：硕士

北京交通大学

2018 年 5 月

万方数据

致谢

本次毕业论文的所有过程都是在我的指导老师卢苇老还有邢薇薇老师的全力帮助和耐心指导下完成的。从开题报告的选题到论文的完成，卢苇老师的关心和教导给予了我前进的方向和对完美的要求，在卢苇老师和邢薇薇老师悉心教导与关怀下，我在设计和实现的过程中把握住了正确的方向，没有失去动力去前进。卢苇教授的一丝不苟的科研态度和精益求精的治学理念，还有认真学习的学习理念，这些都是我需要学习的地方，也是我的不足之处。所以，在这里非常感谢卢苇教授给予了我一个非常难得的学习的机会，同时也非常感谢邢薇薇教授在论文的撰写过程中给予的指导和关心，邢薇薇教授也是我学习的榜样！

感谢给予我学习机会和给予我最大支持的父母，感谢在学校里和生活中帮助过我的朋友们，感谢两年来为我讲课和解惑的老师们；最后，感谢北京交通大学软件学院，这是我生活、学习、进步的地方，谢谢！

万方数据

摘要

多酷游戏是百度游戏事业部独立出来的公司，由于先前的业务已经不满足于日益增长的需求，公司领导决定重新梳理公司的业务，并且重建更清晰简洁的数据调度平台与数据展示平台。这次任务主要实现先前百度游戏事业部各种渠道游戏的基础数据的 ETL 以及数据展示功能。先前百度游戏事业部的业务不断扩张，现有的集群以及技术不足以处理越来越多的 T 级数据，尤其独立公司以后，业务更要与之前相比独立、解耦。随着公司各种渠道的游戏不断增加、业务不断改进、与外部的合作不断增加、数据量越来越大、数据的维度不断扩展，对于数据的获取和处理速度，以及对于数据的利用包括分析和预测等的需求，尤其是对于数据存储的需求不断增加。

因此，我们决定用新的数据处理技术来创建新的数据平台。项目希望此平台能够满足各渠道游戏的运营人员的各种数据分析需求。在数据处理和存储、展示部分，项目采用主流的 Hadoop 和 Spark 来做数据的处理和存储，最终的报表采用轻便的 MySQL 数据库来存储，采用 PHP 语言和 CI 框架来快速实现前期的数据展示功能，在这一部分，本文通过比较有代表性的数据表：今日概况和今日实时概况来阐述。在数据的实时查询和 HDFS 以及 Hive 的可视化操作部分，我们考虑到以后的需求和技术的提升会改动前端与后台，所以在设计这一功能时，需要将前端，即查询平台 Web 端，和后台，即查询平台的访问后台，与实时查询提供者 SparkSQL 的耦合性达到最低，因此项目采用 Thrift 来实现这一需求，在 Thrift Server 中定义查询平台以及可视化操作的所有服务，再由平台后端来调用这些服务，在平台的搭建上，项目采用 Tomcat 容器。同时由于集群的计算资源有限，我们需要将 ETL 流程进行一个优化步骤，使得 ETL 的整体计算量平均分布到凌晨一点到上午九点的所有时间段，避免集群拥堵的情况，节省计算资源，在这一部分项目采用了调度问题常用的算法：遗传算法，并且在需要实现每天定时启动算法检查 ETL 流程是否更新，再决定是否重新计算新的流程。

目前系统已上线并且正常运行，为每日的运营人员分析、公司领导决策提供了所有游戏的概况以及更细维度的分析，得到了同事的广泛好评。实时查询功能极大提高数据开发效率，ETL 优化显著提高集群的利用率。

关键词：移动游戏；数据平台；HDFS；ETL 优化

万方数据

ABSTRACT

The Duoku Game has been a department of Baidu. Since the previous business is not satisfied with the growing demand, leadership of Duoku Game decided to rearrange the company's business and rebuild a clearer and succinct work scheduling platform and data display platform. This thesis mainly realizes the ETL and the function that display data of the current or previous business from Baidu Game Department's various channels. Since the expansion of Baidu's gaming business, the existing clusters and technologies are not enough to deal with more and more data which is in TB, especially after the independence. And the business of company should be more independent from each other.With the increasing games from various channels, the improving business, the cooperation with the outside, the more data, and the more division of data, the requirement of the speed of dealing with data and the obtaining of data is increasing.

Therefore, we plan to set up a new data planform using new technology. We hope the system could meet the operators of various channel games and variety of data analysis requirements. In the part of data processing, data storing, data displaying, we decide to use Hadoop and Spark which are the most popular technology. The final report will be stored in MySQL. And we decide to use PHP and CI framework to build the front-end and back-end for building quickly. For this part, this thesis pick up two module to represent it which are the function Today Overview and Real-Time Overview. In the module Real-Time Searching and Visual Operation of HDFS and Hive, we use thrift framework to make the realization of front-end and back-end doesn't rely on the SparkSQL. We define all of the servers which this module need in thrift server. So the back-end just need to call those servers. For this module, we decide to use Tomcat to build the front-end and back-end. Because the computing resources are not infinite, so we need to optimize the etl flow to prevent blocking of clusters. We want to make an average distribution of the amount of calculation of all working unit during 1 o'clock to 9 o'clock. So we decide to use Genetic Algorithm to deal with it. We want to calculate the best flow every day if there is any different from the yeasterday's etl flow.

Now the platform is in use. Platform provide the basic data of all games and much analysis in different dimisions for the operators and leaders. And platform got a good notice. The real-time searching module improving much efficiency of data development. And the ETL optimizing module improving the using rate of clusters.

KEYWORDS: Mobile Game; Data Platform; HDFS; ETL Improvement

万方数据

序言

随着公司的不断发展，各运营部门对数据的要求，包括实时性和多样性以及可观性的要求越来越高，原有的系统已经不足以支撑不断扩张的业务，因此公司领导决定重新搭建数据平台。

主要的目标包括，将原先不合理的业务集中管理，将原先零散的业务用新的流程集中起来，并且制定新的数据规范。应用新的技术，改进前端的数据展示，并且提高数据流的速度；应用现有的框架搭建轻便简洁直观的调度平台；应用合理的算法，将 ETL 流程合理分配。

本文的主要内容分为两大部分，第一部分包含平台的概要设计、详细设计以及具体实现三个小部分，第二部分包含 ETL 优化的设计以及算法实现。

万方数据

目录

万方数据

万方数据

万方数据

1 引言

本项目的主要目的是在原有业务的基础上，对平台的性能进行提升，希望能够将每天的 ETL 时间控制到 9 点前完成，能够将零散业务集中起来，并且制定新的数据规范，进而减少每天维护数据的工作量，增强数据带来的收益。

1.1 项目背景

据文献，2017 年里中国游戏行业整体营业收入约为 2189.6 亿元，同比增长 23.1%。其中，网络游戏对行业营业收入贡献较大（前三季度营业收入达到 1513.2 亿元），预计全年营业收入约为 2011.0 亿元，同比增长 23.1%；家用游戏机相关营收约为 38.8 亿元，同比增长 15.1%。具体来看：

随着硬件技术的提升，以及用户游戏习惯的转变，网络游戏内部结构有较大分化：移动游戏以全年约 1122.1 亿元的营业收入领先，同比增长 38.5%，占网络游戏的市场份额达 55.8%；客户端游戏营业收入约为 696.6 亿元，同比上升 18.2%，占网络游戏市场比重为 34.6%；网页游戏营业收入约为 192.3 亿元，同比下降 14.7%，占网络游戏市场总份额的 9.6%。

2017 年，自研网络游戏收入稳健提升，约为 1420.7 亿元，同比增长约 14.5%。家用游戏机市场仍处于成长期，用户付费习惯逐渐养成，全年实现营业收入约为 38.8 亿元，同比增长 15.1%；游戏游艺行业进入高速发展阶段，游戏游艺机销售收入约为 135.8 亿元，同比增长 24.7%[1]。

随着公司日益增长的业务和数据，以及用户数量，急需重新建立数据平台[8]以提高数据处理速度，同时理清先前的业务，将不合理的业务集中处理，同时将业务模块重新划分，并且制定数据规范，做到能够将新需求统一处理。

据《2016 年中国游戏产业报告》报道：二零一六年我国在网游用户中使用客户端进行游戏的人数猛降到 64.5%，同时我国游戏用户中，选择手机游戏和页游的玩家人数占比分别猛升到 14.9%和 12.3%，这里面上涨比率最为明显的是手机游戏。在网游用户中，二零一五年的选择手机游戏的玩家仅仅占比 5.4%，这一数据在二零一六年增长了 1.5 倍，并且按照一六年的上涨趋势估计选择手机游戏的玩家在未来 5 年的上涨幅度会保持不变。

对于游戏收入这一模块，在二零一六年端游收入为 664.7 亿元，在游戏市场收入中占比很高，选择端游的玩家占比增长 15%，同比增长 18.9%，在这一领域的市场规模增长的势头未消，但是这一领域的增速却是在逐步放缓，在二零一五年这

万方数据

一领域的用户数和市场规模的较去年增长率分别是 21.2%，11.4%；二零一三年在端游市场的收入和选择端游的玩家数同比增长较二零一二年同比增长下滑 14.1%和 30.1%。随着新型的网络平台进入游戏市场，会对优势市场做出冲击，预计这种现象会进一步冲击端游的市场规模[1]。

在此背景下，公司同时有着广泛的合作关系优势，有着很多优秀的游戏以及数据的优势，所以，要将数据应用起来。将每天的数据定时展示到前端，将重点游戏的实施状况展示到前端，让公司的运营人员及时作出决策。同时我们可以根据数据来判断游戏走向。争取利用数据，在这个市场中取得更大的利润。

1.2 项目意义

先前百度游戏有多个数据流从不同的渠道引入数据到集群，为不同的平台提供不同的数据服务。而由于基础数据数量较多、数据源差异性大，导致数据多样化、统计维度单一混乱等问题。进而，当新需求出现，就必须重新考虑基础数据及报表，而在项目开展过程中，不可避免的需求更迭就会导致资源浪费。并且随着业务的增长，老调度系统无法直观的查看所有的数据走向，必须通过点击依赖任务来一步步手动画出所有的数据流，这给 ETL 的维护造成了很大不便。

基于原数据平台[28]现状，本人提出一种全新的面向维度报表设计思路——多角度、多维度建立报表及相关统计项，并将数据集中存储，减少计算工作量。通过建立模板报表及工具包，实现数据的定制化呈现。通过对现有 ETL 系统优化，实现对易扩展的便捷数据进行处理，从而满足百度游戏事业部自定义报表多维度分析的需求。建设内容主要包括 ETL 系统优化及维度建模，并提供查询、监控、管理、恢复等功能。新的数据平台可以优化整理并深入挖掘现存数据的价值，为将来大数据量的存储和计算打下基础，并提供高精度数据，便于市场、运营部门快速检索，从而提高项目盈利及工作效率，减少不必要的人力投入。

本项目的大数据平台[17]主要包括一平台、三中心，即应用平台、采集中心、数据存储中心和计算中心。数据采集中心主要采集代理游戏数据、独家代理游戏数据、自研游戏数据等相关数据。数据存储中心主要存储关系型数据库、文件、非关系型数据库及数据索引。计算中心则支持实时[5]在线与离线两种计算方式。应用平台除提供应用外，也提供了包括检索服务、数据展示及采集服务等。笔者主要研究实现的包括计算中心的数据清洗和统计工作，如统计活跃数据与付费数据等。还有应用平台的部分工作，主要是数据展示以及实时[2]搜索的工作。

在应用平台的搭建过程中，我会负责优化数据流计算的 ETL 优化工作，由于业务线多，每条业务又包含几十个子任务，所以很难将这些任务平均分配到每一

万方数据

个时间段，所以需要用专门的算法来解决这个问题，进而减少任务的并发量，提高集群在每个时间段的利用率，防止任务堵塞。

1.3 国内外发展现状

事实上，大数据并非新鲜事物，早在上世纪90年代，伴随全球经济的飞速发展，人们对数据积累和分析的需要就已经开始急剧增长，其应用领域开始逐步从天文、气象、军事、基因生物，逐步拓展到搜索、互联网乃至电子游戏等民用范畴。伴随应用商店和社交网络的兴起，游戏市场规模空前扩大，大数据对于游戏运营特别是延长产品寿命的积极作用越发明显，但什么样的数据有价值，如何利用数据的价值，却依然是个难题。

百度总裁李彦宏在百度联盟峰会上曾表示看好大数据和企业级软件，对数据的积累与识别提出了一系列设想与问题。游戏平台与大数据分析平台不同的地方在于，游戏平台的数据量相对较低，且用于数据分析的服务器数量有限。因此，对于游戏大数据平台而言，数据的分析和处理就要同时考虑结构化数据的时效性及非结构化数据的分布式处理能力。而多元化的数据处理技术和兼容分布式计算使得游戏大数据平台的数据处理能力及规模有了大幅度提升。其中，从架构角度讲，游戏大数据平台[3]主要包括数据源采集、数据存储、数据处理及数据可视化与数据服务四层内容。

腾讯游戏平台内游戏急速增长，游戏个数多；数据源异构化突出，数据量大；一游戏一世界，数据抽象模型困难。如何利用大数据技术帮助产品快速、有效的实现游戏数据精细化运营。这些都是摆在游戏大数据应用面前的问题。

据周东祥介绍，大数据的架构大同小异，基本上都包含采集、传输、存储、计算、应用，还有相关数据挖掘。在游戏特定场景下有一些应用的服务痛点和想要突破的问题。在游戏大数据应用服务高频营销场景中，需要用户行为的实时[15]规则。

在王者荣耀等游戏中，都会看到游戏内实时任务奖励体系，例如，玩几局排位赛可以立刻领钻石，同样的累计登录、连胜、杀人、推塔等任务完成也会给到相应的奖励。在以前没有大数据技术与游戏运营结合情况下，游戏后台开发更多是以事务性DB来做核心方案实施。

1.4 项目目标

本项目在于搭建一个新的数据平台[16]、调度平台和集群来适应新的需求。首

万方数据

先由于先前业务很杂乱，没有一个完整的规范，所以急需一个清晰明了的任务调度系统，将原来的数据流迁移的新的系统上，使得所有的数据流能够清晰的展示。

第二就是由于原先的业务并没有真正利用集群的性质，只是把集群当做一个分布式存储工具，主要的任务通过 HIVE[31]来实现，所以，需要搭建一个最新版本的 Hadoop 集群，来进行每天的任务。

第三，先前的数据调度平台与数据展示平台[16]是整合到一起的大系统，而业务人员并不需要登录调度平台，只需要看到每天的数据。所以我们需要单独搭建一个数据展示平台，将每天的数据分用户组展示给不同的运营组。这样可以保证两个平台便于维护，同时也保证了数据流后台的安全。

第四，由于数据流程过多，而公司独立出来后，集群的数量有限，所以，我们需要用算法来保证任务流的分配是合理的，也就是每个时间段内并发适量的任务似的集群的资源能够充分利用。

1.5 关键问题与技术难点

本数据平台[7]系统定位为大数据多维分析系统。平台旨在解决游戏报表的每日产出，并实现对游戏数据的分析挖掘功能和实现新业务报表需求。平台目标是汇总基础数据，多维度计算分析数据，根据不同任务优化平台性能，提高平台运行效率及可靠性。

由于本平台是在原平台基础上进行优化，所以既需要优化原平台的功能，又要实现新功能以满足新需求的不断提出。本平台的主要实现的功能包括定制报表、多层级的权限控制以及实时查询以及 HDFS 可视化操作等，这些功能模块的实现本人将重点阐述，权限控制功能本人没有参与开发，所以只做简单介绍。

1.6 论文结构

第一章引言。主要介绍了项目的背景、明确了选题的研究内容与意义，并提出了系统的建设目标，为后文的设计与实现做了铺垫。

第二章多酷游戏数据平台相关技术综述。阐述了数据平台[19]应用到的各项技术。

第三章多酷游戏数据平台需求分析。从各个模块出发，阐述了平台的功能性能需求，以及数据流程的优化需求。

第四章多酷游戏数据平台概要设计。阐述了平台总体架构、优化的算法选择和实现的概要设计，还有数据表的设计。

万方数据

第五章多酷游戏数据平台详细设计及实现。论述了平台各功能模块的详细设计和实现方式，并阐述了算法的具体实现和实施步骤。

第六章多酷游戏数据平台的测试与效果验证。阐述了平台测试设计和测试验证结果。

第七章结论。是对全文的总结和展望。

1.7 本章小结

本章主要介绍了项目的实施背景和项目需要达到的目的，并且研究了国内外同类项目的发展现状，同时确立了明确的目标，指出了项目中会遇到的问题以及技术难点，最后本章阐述了论文的结构。

万方数据

2 多酷游戏数据平台相关技术综述

本章主要介绍平台应用的关键技术，介绍从基础数据计算到平台功能实现所需要用到的关键技术点，包括 Hadoop、Hive、Thrift、Azkaban、Spark、Kafka 和遗传算法[37]。

2.1 Hadoop

本项目在存储和离线计算时用到了 Hadoop[9]和 HDFS（Hadoop Distributed File System，Hadoop 分布式文件系统），它是一个容错率非常高的系统，对服务器要求较低。通过 HDFS 能实现高频次的数据访问，适用于数据量较大的应用程序。

HDFS 的设计特点包括：

(1) 存储空间大，非常适合以 T 单位技术的大文件存储。

(2) 模块化存储，HDFS 通过将文件平均分模块存储到不同计算器上实现读取文件时的模块化读取，提高读取效率。

(3) 流式数据访问，与传统数据访问不同，流式数据访问一旦文件写入，就不支持再次修改，如要变化，只能在文件末尾进行添加。

(4) 硬件故障，HDFS 理论中，为了防止计算机故障对数据造成恶劣影响，会将文件块的副本分配到其他主机上，如果其中一台主机出现问题，可以通过其他主机读取数据。

Hadoop[12]生态圈大概如图 2-1 所示：

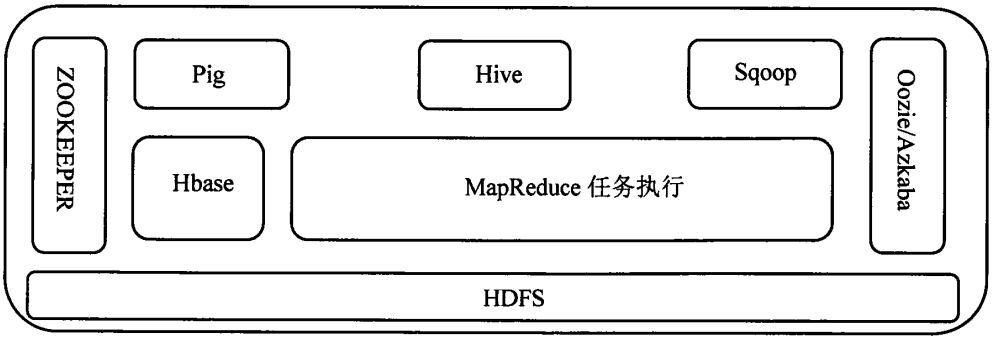

图 2-1 Hadoop 生态圈

Figure 2-1 Hadoop Ecosphere

万方数据

2.2 Hive

本项目在ETL部分用到了Hive，Hive是一种基于MapReduce的并行计算框架，它主要功能是用来接收SQL并转换为MapReduce，并向数据处理人员提供一种方便的SQL接口，使数据处理人员并不需要非常精通MapReduce就可以使用Hadoop[18]。

Hive有如下特点：

(1) Hive 最大的特点是 Hive 通过类 SQL 来分析大数据，而避免了写 MapReduce 程序来分析数据，这样使得分析数据更容易；

(1) Hive 是将数据映射成数据库和一张张的表，库和表的元数据信息一般存在关系型数据库上（比如 MySQL）；

(2) Hive 本身并不提供数据的存储功能，数据一般都是存储在 HDFS 上的（对数据完整性、格式要求并不严格）；

(3) Hive 很容易扩展自己的存储能力和计算能力，这个是继承自 Hadoop 的（适用于大规模的并行计算）；

(4) Hive 是专为 OLAP 设计，不支持事务。

2.3 Thrift

本项目在实现实时SQL查询和HDFS的操作时用到了Thrift，Thrift是Apache下的一个子项目，最早是Facebook的项目，后来Facebook提供给Apache作为开源项目，在官网上，Thrift是一种跨语言服务接口，有一套自己的依赖库来实现跨机器和语言的通信同能。

Thrift内置一个代码生成器，可以按照Thrift的规则来生成接口，生成的接口用来实现跨编程语言和机器通讯。Thrift是一个服务框架，最初在二零零七年由脸书初步开发，旨在实现跨语言通讯。Thrift项目在二零零八年进入Apache，成为Apache的开源项目之一。Thrift内部的代码生成器根据Thrift的中间语言，来定义远程调用的参数类型和接口，通过一个中间语言(IDL，接口定义语言)来定义RPC的接口和数据类型，然后通过该生成器生成统一的代码（目前代码生成器主要支持C++、Java、Python、PHP等），而且通过生成的接口来实现远程控制的协议和实现。

Thrift通过最常见的C/S架构来提供跨语言服务的服务，所有的公共服务都可以配置于Thrift中，所有调用服务的都是客户端。Thrift使用其内置的代码生成器，为客户端和服务端各生成与其编程语言相匹配的接口，再根据接口实现客户端和

万方数据

服务端的跨语言通讯。首先用户创建一个.thrift 为后缀的文件，并在文件中声明自己的服务，声明方式要根据 Thrift 的声明规则，代码生成器会根据.thrift 的声明自动生成接口代码，用户只需要实现服务的方法即可。其中协议层（数据传输的默认格式，如 Json 或者 XML）和传输层（数据传输的默认方式，如 TCP/IP、UDP 等）是运行时库。

一般情况下的跨机器的通信框架都是跨软件平台的(Linux，windows)，而 Thrift 最特别之处在于它是跨语言的：目前其可以支持几乎所有流行语言（C++、Java、Python、PHP、Ruby、Erlang、Perl、Haskell、C#、Cocoa、JavaScript 等等）来实现通讯过程，可以将公共的功能做成服务，比如本人的 HDFS 操作和实时查询功能，为了方便 PHP 端也能够便捷的访问这两个功能，可以配置好 Thrift 服务器，服务器中包含这两种服务，客户端则可以使 PHP 也可以是 Java，不必拘泥一种语言选择；如果编写一个服务器端程序，首先定义好通讯规则（在 Thrift 中是.thrift 文件），即包含哪些服务，再用自己熟练的语言在 Thrift Server 中实现这些服务，其余客户端通过 Thrift 生成同样的接口，调用 Thrift 里的服务。

与 Thrift 相类似的开源项目是 Google 的 Protocol Buffer(Protobuf)，Protobuf 目前提供了 C++、Java、Python 三种语言的 API，比 Thrift 简单一些，应用也不如 Thrift 广泛，大概工作流程如图 2-2 所示：

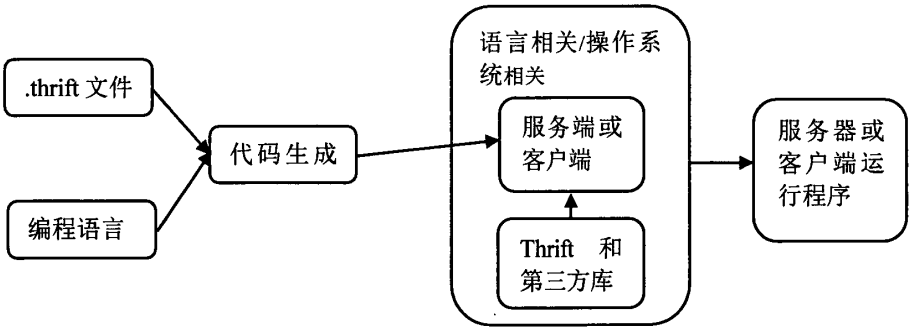

图 2-2 Thrift 流程

Figure 2-2 Flow Chart of Thrift

2.4 Spark

本文在实时部分用到了 Spark[10]，Spark 是一种通用内存并行计算框架，它在 2013 年 6 月形成雏形，并在八个月后一跃成为炙手可热的顶级项目，这一定程度也证明了 Spark 的优秀。后期，随着 Spark SQL、Spark Streaming、MLLib 和 GraphX 等组的问世，大数据处理一站式解决平台逐渐形成并广泛应用。从各方面报道来看 Spark 抱负并非池鱼，而是希望替代 Hadoop 在大数据中的地位，成为大数据处

万方数据

理的主流标准，不过 Spark 还没有太多大项目的检验，离这个目标还有很大路要走。

Spark[24]的实现语言是 Scala，Scala 是一种新型函数式语言，面向对象，并且能够使用函数进行操纵分布式数据集，语法与操作本地集合对象一样简单方便。而且 Scala 提供一个并行的框架，框架通过接口只用来发送和接受消息，而不与去其他框架共享消息。由于 Spark 基于内存，并且 Scala 基于 Java，所以只要有 Java 环境，便可以安装 Scala，进而运行 Spark，而且基于内存的 Spark 相比较于 Hadoop 更快。Spark[29]的生态圈如图 2-3 所示：

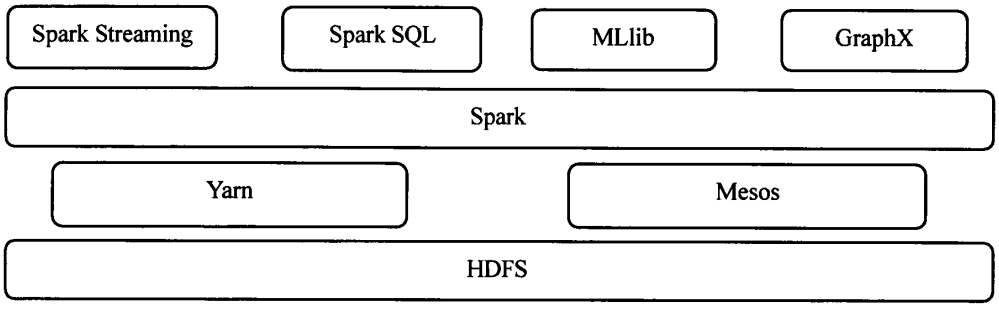

图 2-3 Spark 生态圈

Figure 2-3 Spark Ecosphere

2.3.1 SparkSQL

Shark 是 SparkSQL 的前身，它发布于 3 年前，那个时候 Hive 可以说是 SQL on Hadoop 的唯一选择，负责将 SQL 编译成可扩展的 MapReduce 作业，鉴于 Hive 的性能以及与 Spark 的兼容，Shark 项目由此而生。Shark 即 Hive on Spark，大致原理是通过 Hive 本身的解析 SQL 功能，将 SQL 转换成 RDD 操作，在运行时需要用到的 Hive 信息则是通过访问 Hive 的 MySQL 表来获取 Hive 的元数据信息，Shark 通过这种方式获取到 HDFS 上的文件后，再放到 Spark 上并行计算。Hive 可以和 Shark 完美兼容，并且可以再命令输入模式下，使用相应的函数来调用 Hive，获得 HiveSQL 的结果集，拿到 Scala 下计算，并且 Shark 支持自定义的简单的函数。在 2014 年 7 月 1 日的 Spark Summit 上，Databricks 宣布终止对 Shark 的开发，将重点放到 Spark SQL 上。Databricks 表示，Spark SQL 将涵盖 Shark 的所有特性，用户可以从 Shark 0.9 进行无缝的升级。在会议上，Databricks 表示，Shark 更多是对 Hive 的改造，替换了 Hive 的物理执行引擎，因此会有一个很快的速度。然而，不容忽视的是，Shark 继承了大量的 Hive 代码，因此给优化和维护带来了大量的麻烦。在性能不断优化和计算复杂的加深，以及对数据整合的需求，基于 MapReduce 设计的部分无疑成为了整个项目的瓶颈。因此，为了更好的发展，给用户提供一

万方数据

个更好的体验，Databricks 宣布终止 Shark 项目，从而将更多的精力放到 Spark SQL 上。数据处理人员可以使用 SparkSQL 的时候直接处理 RDD 数据集，也可以查存在 Hive 上的其他数据。SparkSQL 可以用相同的规则处理各种表和 RDD 数据集，这种特性使得数据处理人员能够随意的编写 SQL 来做外部查询，并且做更为复杂的多维度分析。除了 Spark SQL 外，Michael 还谈到 Catalyst 优化框架，它允许 Spark SQL 自动修改查询方案，使 SQL 更有效地执行。

Spark SQL 的特点：

引入了新的 RDD 类型 SchemaRDD，可以象传统数据库定义表一样来定义 SchemaRDD，SchemaRDD 由定义了列数据类型的行对象构成。SchemaRDD 可以从 RDD 转换过来，也可以从 Parquet 文件读入，也可以使用 HiveQL 从 Hive 中获取。

内嵌了 Catalyst 查询优化框架，在把 SQL 解析成逻辑执行计划之后，利用 Catalyst 包里的一些类和接口，执行了一些简单的执行计划优化，最后变成 RDD 的计算。在应用程序中可以混合使用不同来源的数据，如可以将来自 HiveQL 的数据和来自 SQL 的数据进行 Join 操作。

2.3.2 Spark-Streaming

Spark Streaming 是由 Spark 的 API 扩展出来的实时处理框架，Spark Streaming 有吞吐量高，可容错的针对于实时流式数据处理的计算框架。Spark Streaming 目前支持的数据源有多种，包括 Kafka、Flume 等分布式消息分发框架以及 TCP 的 Socket。使用 Spark Streaming 可以使用映射、聚合、连接、等复杂的高级关系函数来处理获取到的数据源。在计算完成后，可以将计算得到的结果储存到分布式存储系统如 HDFS。在“一栈式数据处理”原则的基础上，使用 Spark 的同时，可以使用 Spark 的其他子框架。

Spark 生态圈的每个部件，都基于 Spark 的核心 API 开发，Spark Streaming 也是如此，其处理流程是，实时地接受流式数据，然后将其根据时间间隔分批，最终得到一批一批的批数据，然后交给 Spark 的 Engine 来处理这一批批的数据，最后得到这一批批数据的对应处理结果。这些批数据在 Spark 的内存中都有一个对应的 RDD 对象。所以，流式数据可以看做是一组 RDD，多个 RDD 组成的数组。通俗的说，是将流式数据分批后，生成一个 FIFO 规则的队列，这些分批后的 RDD 依次通过这个队列，SparkEngine 同时去除队列里面一个个出队的批数据，最后将这些批数据封装成 RDD。RDD 形成后即可进行各种复杂的函数逻辑运算。FIFO 的队列连接首尾两个部分，形成一个典型的消费者-生产者模型，同时可以通过操

万方数据

控队列来调控生产效率和消费效率。

Spark-Streaming 在项目中主要用于实时数据的处理，首先获取从 Kafka 的不同主题的数据流，再将数据清洗，做一部分聚合工作，最终再次存储到 HDFS 中。再由 SparkSQL 来做剩余的统计工作，生成数据报表，更新到 MySQL 中以供前端的查询和更新。

2.4 Azkaban

本项目在任务调度时用到了 Azkaban，Azkaban 是 twitter 出的一个任务调度系统，操作比 Oozie 要简单很多而且非常直观，提供的功能比较简单。Azkaban 以 Flow 为执行单元进行定时调度，Flow 就是预定义好的由一个或多个可存在依赖关系的 Job 组成的工作流。它的的主要特点有如下几条：

(1) 兼容所有 Hadoop 版本包括 1.x，2.x，CDH

(2) 可以通过 WebUI 进行管理配置，操作方便

(3) 可以通过 UI 配置定时调度

(4) 扩展性好，可针对某一问题开发组件(目前有三个插件 HDFSBrowser，JobtypePlugins 和 HadoopSecurityManager)

(5) 有权限管理模块

(6) 可以通过 WebUI 跟踪 Flow 或者 Job 的执行情况

(7) 可以设置邮件提醒

(8) 可以为定时 Flow 或者 Flow 中的某个 Job 配置执行时间长度的控制，如果执行时间超过了所设的时间，可以发送警告邮件给相关人员或者 Kill 掉相应设置的 Flow 或 Job

(9) 可以重试失败 Job

(10)Azkaban 也有一些局限性（尚待挖掘），例如任务之间的依赖，不能够指定部分完成（比如我们希望任务 A 依赖于 B，但是并不是 B 完全执行完成 A 才可以启动，而是 B 的某个阶段完成的话就可以启动 A）

Azkaban 是一种类似于 Oozie 的工作流控制引擎，可以用来解决多个 Hadoop 计算任务之间的依赖关系问题。Azkaban 由 Azkaban Web 服务器、Azkaban Executor 服务器和 MySQL 这 3 个组件构成。

(1) MySQL 数据库

Azkaban 使用 MySQL 来存储它的状态信息，Azkaban Executor Server 和 Azkaban Web Server 均使用到了 MySQL 数据库。AzkabanExecutorServer 在如下几个方面使用到了数据库：获取 Project 的信息、执行工作流、存储工作流运行日志；

万方数据

AzkabanWebServer 在如下几个方面使用到了数据库：Project 管理、跟踪工作流执行进度、访问历史工作流的运行信息、定时执行工作流任务

(2) AzkabanWebServer

AzkabanWebserver 是整个 Azkaban 工作流系统的主要管理者，它负责 Project 管理、用户登录认证、定时执行工作流、跟踪工作流执行进度等一系列任务。同时，它还提供 Web 服务操作的接口，利用该接口，用户可以使用 Curl 或其他 Ajax 的方式，来执行 Azkaban 的相关操作。操作包括：用户登录、创建 Project、上传 Workflow、执行 Workflow、查询 Workflow 的执行进度、杀掉 Workflow 等一系列操作，且这些操作的返回结果均是 Json 的格式。

(3) AzkabanExecutorServer

之所以将 AzkabanWebServer 和 AzkabanExecutorServer 分开，主要是因为在某个任务流失败后，可以更方便的将重新执行。而且也更有利于 Azkaban 系统的升级。两种不同类型的 Azkaban，Solo Server Mode 模式使用 H2 数据库，且 WebServer 和 ExecutorServer 运行在同一个进程中，没有单独分开。该模式适用于小规模的使用。Two Server Mode 模式使用 MySQL 数据库，WebServer 和 ExecutorServer 运行在不同进程中，该模式适用于大规模应用。

2.5 Kafka

本项目在实时消息分发部分用到了 Kafka，Kafka 是一个开源的，分布式的，高吞吐量的消息系统。随着 Kafka 的版本迭代，日趋成熟。大家对它的使用也逐步从日志系统衍生到其他关键业务领域。特别是其超高吞吐量的特性，在互联网领域，使用越来越广泛，生态系统也越来的完善。同时，其设计思路也是其他消息中间件重要的设计参考。

Kafka 原先的开发初衷是构建一个处理海量日志的框架，基于高吞吐量为第一原则，所以它对消息的可靠性以及消息的持久化机制考虑的并不是特别的完善。0.8 版本后，陆续加入了一些复制、应答和故障转移等相关机制以后，才可以让我们在其他关键性业务中使用。

Kafka 的各组件之间通过 TCP 协议通信，下面介绍 Kafka 的具体术语和组件：

(1) Topic：主题，或者说是一类消息。类似于 RabbitMQ 中的 Queue。可以理解为一个队列。

(2) Broker：一个 Kafka 服务称之为 Broker。Kafka 可以集群部署，每一个 Kafka 部署就是一个 Broker。

(3) Producer & Consumer：生产者和消费者。一般消息系统都有生产者和消费

万方数据

者的概念。生产者产生消息，即把消息放入 Topic 中，而消费者则从 Topic 中获取消息处理。一个生产者可以向多个 Topic 发送消息，一个消费者也可以同时从几个 Topic 接收消息。同样的，一个 Topic 也可以被多个消费者来接收消息。

(4) Partition：分区，或者说分组。分组是 Kafka 提升吞吐量的一个关键设计。这样可以让消费者多线程并行接收消息。创建 Topic 时可指定 Parition 数量。一个 Topic 可以分为多个 Partition，也可以只有一个 Partition。每一个 Partition 是一个有序的，不可变的消息序列。每一个消息在各自的 Partition 中有唯一的 ID。这些 ID 是有序的。称之为 offset，offset 在不同的 Partition 中是可以重复的，但是在一个 Partition 中是不可能重复的。越大的 offset 的消息是最新的。Kafka 只保证在每个 Partition 中的消息是有序的，就会带来一个问题，即如果一个 Consumer 在不同的 Partition 中获取消息，那么消息的顺序也许是和 Producer 发送到 Kafka 中的消息的顺序是不一致的。这个在后续会讨论。

2.6 遗传算法

本文在 ETL 优化部分用到了遗传算法，遗传算法顾名思义是来模拟生物种群优胜劣汰、适者生存的迭代过程，这一理论基于自然选择学说，旨在用代码模拟群体的繁衍过程，最终留下的优良个体便是我们寻求的最优解。首先遗传算法规定用编码来表示某种问题，这是最关键的一步，编码的基础决定了后续的遗传操作的效率和能力，在编码后通过对一组编码进行简单的操作如遗传、变异最后优胜劣汰来确定搜索的方向。遗传算法的基本单元是一个染色体，也就是一个个体，即一个二维数组或者二进制串，多个染色体即多个个体组成的群体即种群。在遗传算法里，问题被编码成染色体。最开始随即产生多个个体组成群体，然后对群体半随机选取个体进行交叉遗传，再随机选择个体进行变异，最终计算每个个体的适应度，通过适应度筛选优良个体。算法不断循环，就像种群不断繁衍一般，一直到代码中设定的适应度或者到达最多循环次数，则我们获取到最终的优良个体。

遗传算法应用情景一般是寻找最优解，所以对于问题来说是拥有很多答案，每个答案都是合理的，并且如果穷举这些答案，那么会造成很庞大的计算量，那么在这种情况下，遗传算法[6]是最好的选择。

遗传算法基于进化论和物竞天择适者生存的规则，模拟种群的繁衍更替，将繁衍、淘汰、遗传、变异等概念通过代码实现，通过不断更替一组初始解即初始集群，并且对初始解进行打乱重新组合，可以改变在寻找接的过程中的走向，减少了找到局部最优解的可能，最终走向最优解。正如此遗传算法减少了局部最优解

万方数据

的出现，使得寻找方向趋向最优解。遗传算法的大致步骤如下所述：

(1) 找出问题的量化方法，以及计算个体适应度的方法；

(2) 生成初始群体，可以由 01 编码，可以使二维数组编码，可以使二进制数组，并且规定个体的合法规则，每个个体代表一个解，多个群体代表一个种群；

(3) 分别计算群体中每个个体的适应度，适应度是进行优胜劣汰的基础，然后按照适应度的大小倒序排列，淘汰适应度低的个体，也可以进行意外干预，使得适应度低的个体留下，淘汰个别适应度更高的个体；

(4) 对新的集群进行适应度排序，随即选择适应度较高的两个个体，将两个个体进行交叉遗传生成两个新的个体；

(5) 对新的集群进行适应度排序，随即选择适应度较低的一个个体，将该个体随机选择变异点进行变异；

(6) 重复步骤(3)到步骤(5)的操作，直到所有个体都相同，即所有个体的适应度相同，或者到了最多迭代次数，结束算法，并得出最优解。

2.7 本章小结

在这一章介绍了 Hadoop、Hive、Spark、Azkaban 和遗传算法五种技术，这五种技术都与平台有关，是构建平台的关键技术，平台使用 Hadoop 的 MapReduce 的计算框架进行数据清洗，利用 HDFS 作为数据平台的存储方式，利用 Spark 作为实时计算的关键技术，利用 Azkaban 来进行任务调度，同时最后使用遗传算法对数据流程进行优化，为后面的项目概要设计和详细设计做了充分准备。

万方数据

3 多酷游戏数据平台需求分析

多酷游戏数据平台从工作模式上分为离线处理和实时处理两大部分，数据来源上可划分为三大类：独代游戏、自研游戏、渠道游戏，从数据类型上课分为活跃数据、付费数据、下载数据三类，同时在前段展示每类数据的今日概况、分维度统计以及实时数据[25]展示，同时数据展示平台可以设置用户组，为每组用户设置权限以及开放不同板块的数据报表。各模块都会根据功能进而拆分成各子模块，每个子模块组成一个模块，每个模块组成数据平台，各模块在功能上协同工作，互相配合。

3.1 功能需求分析

本小节主要分析平台所需要的功能，主要涉及本文负责的部分。并且在这里本文将会给出每个模块的用例图。在今日概况中，主要介绍了离线数据的数据表，在实时概况中介绍实时数据的数据表。

3.1.1 今日概况

用户登录后，首先就是今日概况，每天准时更新昨日注册用户数、昨日新增付费用户数、 纯新增付费用户数、活跃用户、付费情况和分渠道付费信息，并以表格的方式展示。今日概况的用例图如图 3-1 所示。

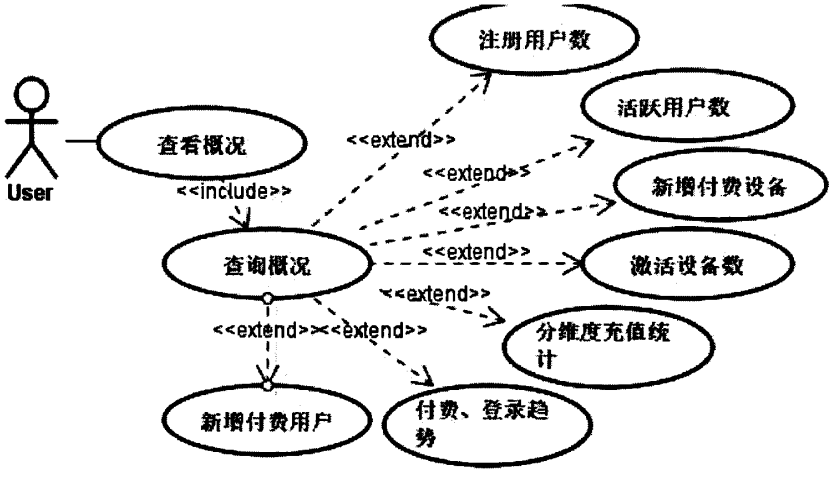

图 3-1 今日概况用例图

Figure 3-1 Use Case Diagram of Function Today Overview

万方数据

查询条件：添加时间维度：昨天(默认值，今日流程更新后的计算值)、选择天(选择天的流程更新后的统计值)、前日环比(昨日数据和前日数据的环比)、七天数据(昨日数据以及七日前数据之和)、本月数据(本月数据之和)、上月数据(上月数据之和)；选择平台；选择渠道；选择游戏；选择大区；

选择维度后，将会根据选择的维度展示；日期、渠道、大区、活跃设备、首次活跃设备（激活设备）、新增设备、平均登录次数、付费设备、付费金额、ARPU、ARPPU、付费率、新增付费设备、新增设备付费金额；

3.1.2 今日实时概况

用户登录后，进入实时模块的首页就是今日实时概况，实时更新核心游戏实时的在线人数、注册用户、新增付费用户数、纯新增付费用户数、分渠道付费金额和分游戏渠道付费金额统计数，并主要以表格的方式展示。由于对于每个游戏，实时数据起到的作用并不是很大，最多只是用来监控某个游戏。所以在本项目中，重点在于离线模块，而实时模块则主要用来做实时查询的功能。况且每个游戏做到拥有实时界面，加之每个游戏的内容都不一样，所以工作量未免太大。所以在实时这方面，本项目只是做了一个简单的所有游戏的分维度统计实时概况。实时概况模块用例图如图 3-2 所示。

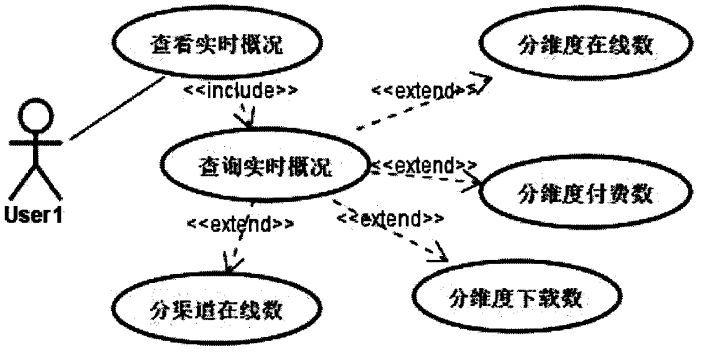

图 3-2 实时概况用例图

Figure 3-2 Use Case Diagram of Function Real-Time Overview

3.1.3 实时查询

定制报表无法完全满足随时变化的报表需求，并且实时功能主要用来实现实时搜索和查询，所以提供实时的查询接口，不仅可以为 ETL 提供一个快速接口，而且也可以使运营人员可以随时查询需要的临时报表。

万方数据

借助这个功能，用户无需掌握复杂的大数据开发技术，只要熟悉 SQL 语法，就可以快速对海量数据进行类似关系型数据库的查询操作，并以可视化的方式获得直观的查询结果。

功能和特性描述如下：

(1)数据链接管理：支持 Hive、Spark 数据源，支持查询时选择 Hive 或者 Spark 进行查询；

(2)查询结果可视化：查询结果可以快速的形成结果表，结果支持下载；

(3)安装配置管理：提供可视化及命令行的配置，也可在 LEAP 系统管理进行统一的安装配置；

(4）查询管理服务：可以查看表结构，字段功能，支持查询过程可视化；

查询多窗口化：可以新建多个窗口，每个窗口进行不同的 SQL 语句查询，方便数据对比和查看。

(5)对 HDFS 操作；提供了 HDFS 的文件浏览功能，并且可以对 HDFS 的文件系统直接进行操作；

(6)数据表预览：支持数据预览，了解相关的数据，并对想要的数据进行计算；

其用例图如图 3-3 所示：

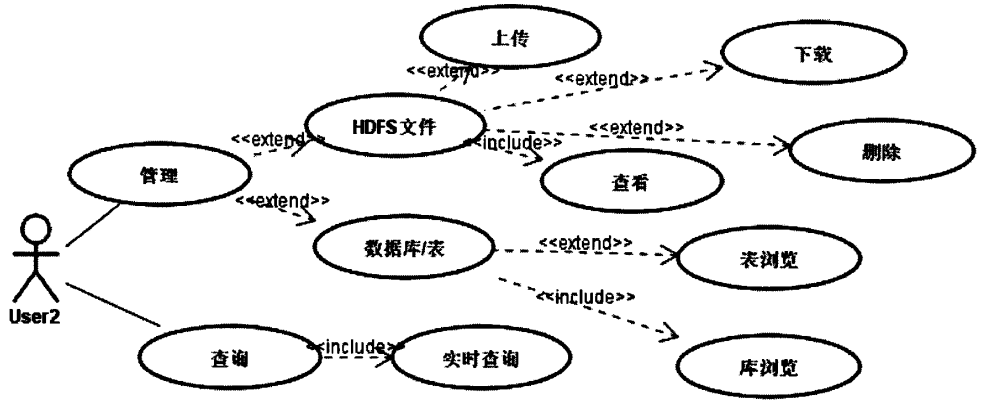

图 3-3 实时查询功能用例图

Figure 3-3 Use Case Diagram of Function Real-Time Query

可以浏览所有的 HDFS 数据库和所有的数据表，双击产生查询页面会自动选择某个库或者自动选择某个表，在点击执行是会检测是否有参数，如果有则会提示输入参数。

可以在此功能里，对 HDFS 文件进行直接操作，主要包含上传、删除和下载操作。最后，用户可以不局限于生成报表的形式，支持在线转化成柱形图或饼状图，以后会不断添加更多样化的图形。

万方数据

3.1.4 配置管理

每个系统都必须有登录和权限，本项目也有相应的模块来做到用户管理、游戏管理两部分，但由于该模块不在本人的负责范围内，所以这个模块只在此做相应的需求分析，不做后续的详细介绍。配置管理分为用户管理和游戏管理两部分，用户管理分为用户管理和用户权限管理两个子功能，游戏管理主要作用于游戏分组管理。下面一一介绍各功能。

（1） 用户管理用例图如图 3-4 所示

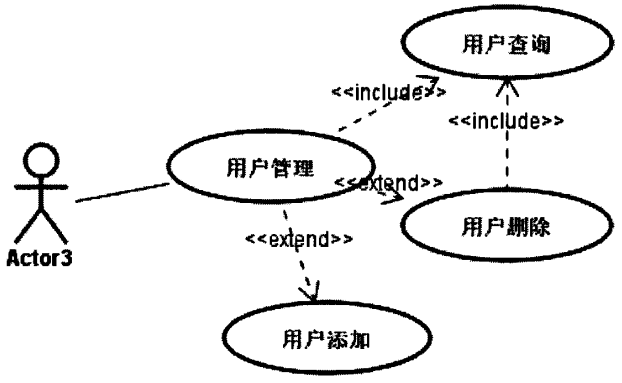

图 3-4 用户管理用例图

Figure 3-4 Use Case Diagram of Function User Management

用户管理主要包括用户查询，用户删除，和添加用户三个功能，能够满足所有的用户操作需求。而用户无法编辑，只能通过后台修改职工信息来修改用户，增加了信息的安全性。

（2） 用户权限管理用例图如图 3-5 所示

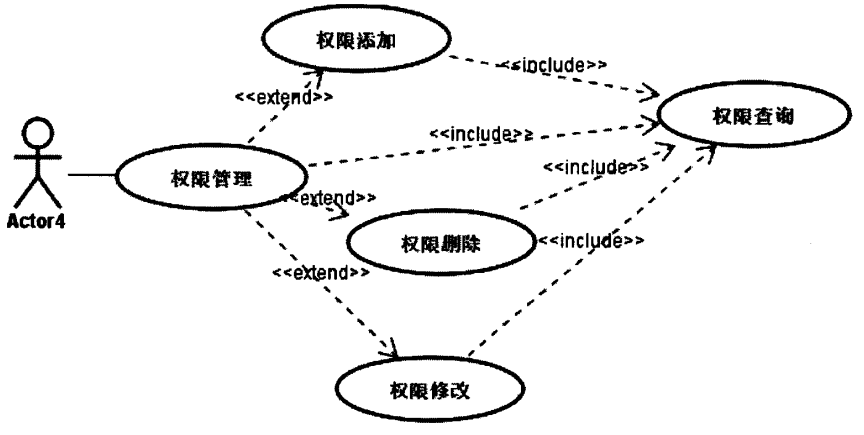

图 3-5 用户权限管理用例图

Figure 3-5 Use Case Diagram of Function User Powers Manager

万方数据

管理员可以操作用户的权限，增删改查能够满足所有操作用户权限的需求。权限主要包括对某些页面，也就是某些游戏的浏览和查询权限。

（3） 游戏管理用例图如图 3-6 所示

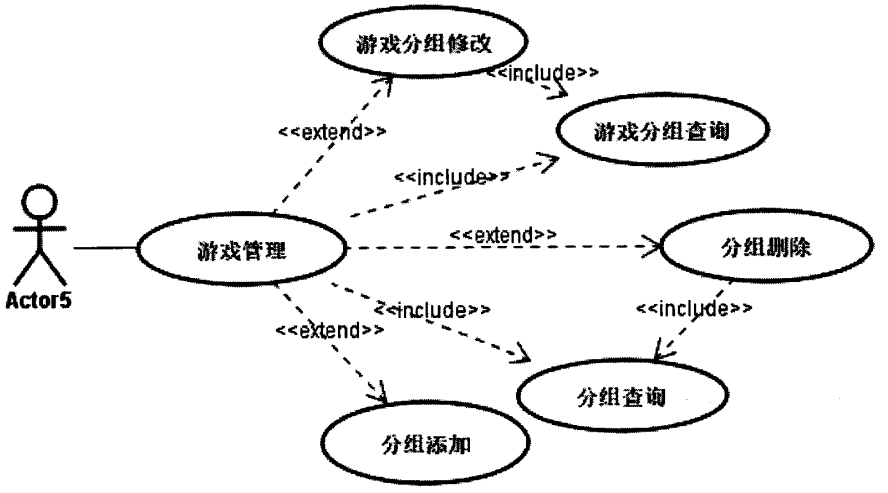

图 3-6 游戏管理用例图

Figure 3-6 Use Case Diagram of Function Game Management

游戏管理主要是游戏分组管理，包括代理游戏分组，独家代理游戏分组，自研游戏分组等。

3.1.5 ETL 流程优化

在通常的任务流程中，对任务流影响最大之一便是如何精准的启动合适的任务，这就产生了问题：如何在有限的资源里分配过多的任务使之得到充分利用。

当前这种调度[21]问题的常用解法其一是基于知识的解法，其二是基于运筹学的方法。后者是将问题抽象成一个模型，然后用一些规划的方法来取得调度结果，最简单、最极端的如穷举法，但是这种方法的计算量太大，而且很多问题很难被抽象成某个模型、很难量化。这种方法实际应用中会有很多的困难。前者基于知识的方法对解决这类问题有比较好的方法。但是由于此类调度生产问题的负责，很多问题的知识表示还没得到很好地解决，人们的知识储备不完整，再加上每个人对问题的理解不一，使得这种解法也不够好。

而 ETL 流程则对应于生产控制系统中的作业流，在资源总量有限的情况下，不仅要为负责每天定时的数据清洗和处理，还要负责大数据量的存储，尤其有时还要额外承担一部分查询工作。因此各任务在何时启动，即资源如何分配成了一个关键问题。本文在这里选择了遗传算法[20]作为任务调度[11]的优化算法。遗产算法

万方数据

将问题用编码表示，通过读取数据库获取到每个任务间的依赖关系，并且按照规则将种群中的非法接排除。通过一系列的遗产操作如交叉、遗传、变异和优胜劣汰来寻找最优解，得到最优解后便可以更新任务流，使任务流按照最优化的方式启动，进而解决集群的堵塞现象，进而提高资源利用率。

于是本项目决定设计一个离线任务的优化算法[4]来充分利用集群资源，经过多方面的商讨，最终本项目决定采用遗传算法来解决这个问题。

ETL 优化部分的用例图如图 3-7 所示：

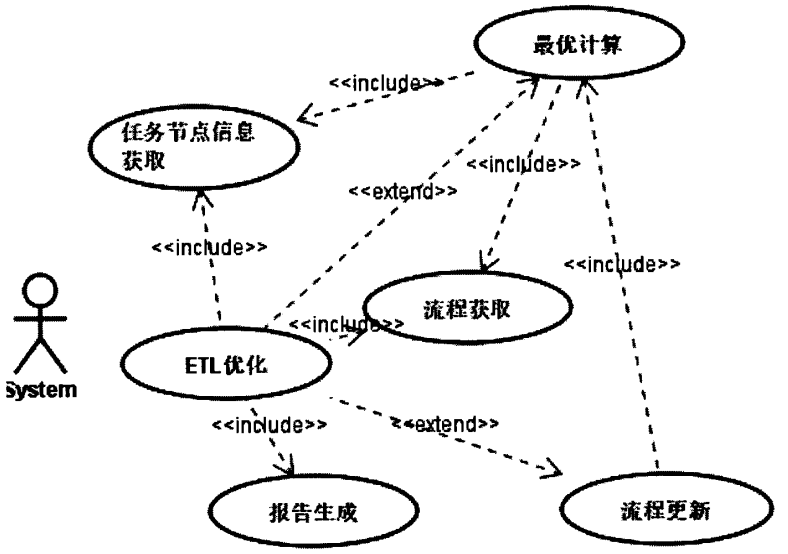

图 3-7 ETL 优化用例图

Figure 3-7 Use Case Diagram of Function ETL Optimising

算法需要每天自动执行，检测是否有流程修改，有则计算最优流程，无则直接退出。

3.2 非功能需求分析

本项目要设计一个大数据平台，涉及到数据的清洗、转换、存储和展示，因此要满足以下几条非功能性需求：

(1) 数据安全性：没有数据安全性，那么就是一个不合格的数据平台。在这里的安全性不仅仅指数据访问的安全，还有数据存储的安全，解决这两点的关键就是访问控制和数据备份。

(2) 时间需求：在使用离线查询时，往往需要花费很长时间，这极大影响工作效率，因此时间也是一个很重要的非功能需求。一般在离线查询，查询时间不能

万方数据

超过三十分钟，实时查询时间不能超过二十秒。

(3) 业务可拓展性：公司的业务会随着时间不断更新，新的数据新的业务会出现，我们要保证当新的数据出现时，目前的 ETL 流程能够完美处理，当新的业务出现时，根据定制的规范，我们能够很快的做出新业务的 ETL 流程。

(4) 准确性：本数据平台是为公司的运营提供决策方向，更重要的是带来利润，因此在处理数据是要非常的准确和小心，尤其是关系到玩家或者角色的基本信息时，每个失误都可能会造成很严重的后果。

(5) 易用性：由于本项目面对的一部分用户是运营人员，所以对用户界面的设计要求是通俗易懂。同时，对于平台的每个界面来说，要设计合理、统一风格。

3.3 本章小结

本章详细描述了多酷数据平台今日概况、今日实时概况、实时查询三个部分的各自的需求分析，指定了每个模块的功能。其中，今日概况、今日实时概况为项目的数据展示重点部分，是数据平台展示数据的重要模块，实时查询等模块力求满足用户的多样化的需求；其余的随着业务的增多会不断添加新的模块如市场推广和运营统计，这两个部分会在以后逐渐承接先前业务并且建立起来并完善以承接后续业务。

另外本章还明确了 ETL 优化的功能和优化使用的算法。

最后，本章还指明了本平台的非功能性需求。上述需求分析提出了功能，为下文平台的架构设计、详细设计以及最终实现明确了目标和方向，为开发和设计做了坚实的基础。

万方数据

4 多酷游戏数据平台概要设计

本系统的数据来源多处，包括客户端日志、服务端日志、代理游戏的数据拉取等。另外先前业务的数据量巨大，数据内容和来源复杂多样，需要统计的维度很多。并且先前的系统已经不满足于当前的需求，为了解决这些问题，提出重新创建多酷数据平台的方案。在本章将会讲述本平台的概要设计，平台包括架构设计和基础流程设计两部分。

4.1 系统整体架构设计

数据流作为承载多源数据的清洗、转化和存储的主要部分；采用了 Hadoop[33] 和 Spark[36]作为主要计算框架，采用 Azkaban 作为任务调度[32]系统，采用 MySQL 作为汇总数据库，前端和后台均采用 PHP 语言编写，平台的总体架构设计如图 4-1 所示。

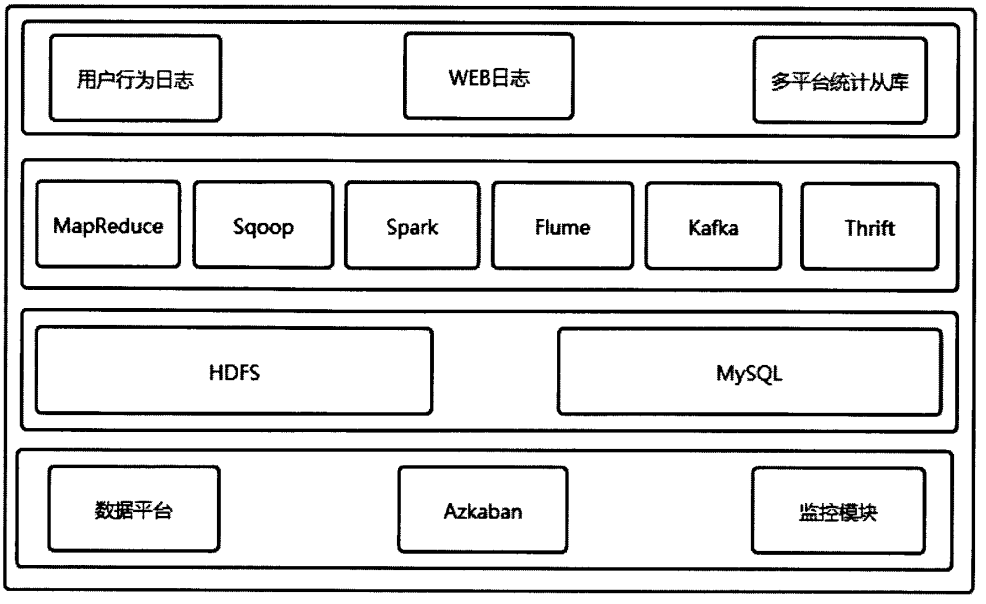

图 4-1 系统架构图

Figure 4-1 System Architecture

平台主要由后台的数据获取模块，数据处理流，数据存储模块还有前端的数据展示模块组成。数据获取模块主要由 Sqoop 和 Flume 组成，负责将多源的数据拉取到集群上，为数据处理流坐下基础；数据处理流采用 Hadoop 和 Spark 作为计算框架，采用 Kafka 作为实时数据的分发框架，其中 Hadoop 的 MapReduce[34]和 Hive

万方数据

作为离线计算框架，Spark 作为实时计算框架；数据存储模块主要有两部分：元数据存储和汇总数据存储，元数据采用 HDFS 分布式存储，因为其数据量大，汇总后则可以使用 MySQL 存储，因为 MySQL 操作简单，方便使用。

前端和后台均采用 PHP 语言，并且用 CI 框架实现 MVC 架构。使用 Java 编写 MapReduce 实现数据清洗，并且使用 Hive 实现数据的统计和聚合。

4.2 系统数据流程设计

数据流程分为离线处理和实时处理，两种处理方式在非特殊情况下交替执行，分别服务不同的业务。离线模块主要处理大规模的数据统计和数据清洗，并且将各种维度的聚合结果导入到MySQL，每天的处理时间为十二点到第二天上午九点。实时处理的处理时间为上午九点到晚上十二点。

4.2.1 离线模块数据流设计

离线模块数据流的整体思路是从各数据源拉取数据，存储到 HDFS，再经过一系列离线的计算、清洗等，存储汇总结果到 MySQL，再由前端拿来展示。

解决方案的核心是数据。数据平台的数据源多样，来自于客户端数据、后台数据和其他运营商数据，经过 Flume 拉取到 HDFS 集群，然后由 MapReduce 来进行数据清洗，有 Hive 进行离线的维度计算，最终将汇总结果存储到 MySQL，前端和后台使用 PHP 的 CI 框架，向 MySQL 请求数据来展示，数据流如图 4-2 所示。

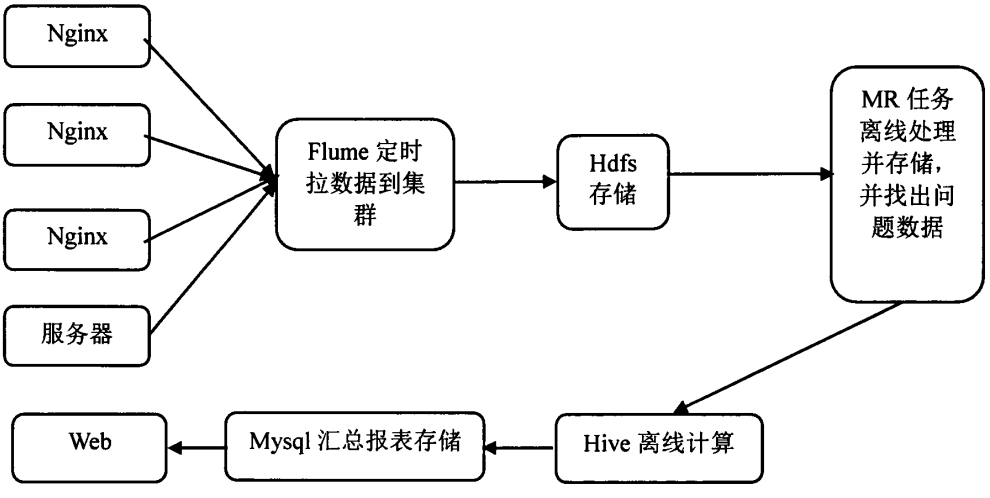

图 4-2 离线模块数据流图

Figure 4-2 Data Flow Diagram of Offline Module

万方数据

多酷游戏数据平台可划分为以下几个模块：

数据抓取模块：通过 Flume 将服务器上的日志定时拉取到 HDFS 上存储，这部分需做到实时拉取。

数据清洗模块：由 MapReduce 组成，负责数据的清洗工作，并将清洗后的数据存储到 Hive 表；

数据转换统计模块：由 Hive 和 MapReduce 组成，负责数据的转换和统计工作，并将统计后的汇总数据放入 Hive，再导入 MySQL；

数据平台展示模块：负责前端的各种维度展示数据以及实时的查询数据，用户可以选择各种维度查询，或者根据自身需求编写 SQL 查询；

4.2.2 实时模块数据流设计

本模块主要用来实时展示重要游戏的数据，从 Flume 的实时拉取、Kafka 的实时分发，到 Spark 的实时统计，到前端的展示，数据流是实时模块的核心。数据平台的数据来源于商定好的核心游戏的用户产生的行为日志[1]和游戏平台的基础数据，经由 Spark 集群的数据清洗处理，再经过 Spark 集群的数据统计处理，最终存储到 HDFS，再将汇总表存储到 MySQL，最终数据平台访问后台表格展示今日概况或者查询结果，呈现给拥有权限的用户。

由于实时数据模块的报表主要作用于核心游戏，而这些报表与离线数据设计相仿，所以此处不再赘述，另外实时模块还主要用来实时查询，所以在这里主要介绍前端实时查询、HDFS 操作和实时数据处理的设计，数据流如图 4-3 所示。

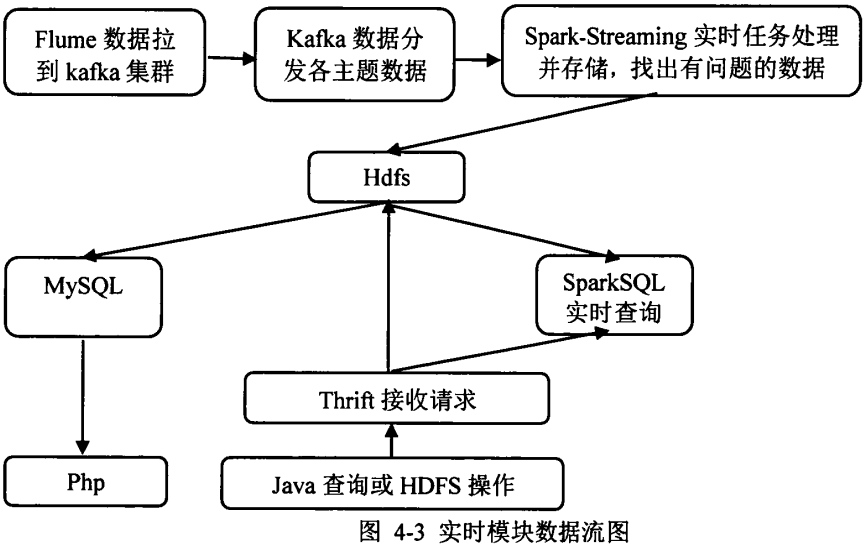

图 4-3 实时模块数据流图

Figure 4-3 Data Flow Diagram of Real-Time Module

万方数据

流程用 Flume 随时从机器上拉取数据到 Kafka 集群，并且对应不同的主题，Kafka 发布这些主题的流数据，Spark-Streaming 实时从 Kafka 集群里消费这些最新的数据，然后对数据进行清洗、转换和统计，最终存入 HDFS 里供前端通过 SparkSQL 查询。

前端和 Thrift 服务器通过 Thrift 脚本生成接口代码，使得不管前端的任何改动，SparkSQL 都能够收到能够执行的代码，使得前后两端解耦。

同时，本项目在设计的时候主要考虑到下面几个特性：

作为一个实时系统，实时性是首要指标。线上系统面对着各种异常情况。例如如下几种情况：

(1) 突发流量洪峰，怎么应对；

(2) 出现失败数据或故障模块，如何保证失败数据重试并同时保证新数据的处理；

(3) 环境问题或 Bug 导致数据积压，如何快速消解；

(4) 程序 Bug，旧数据需要重新处理，如何快速处理同时保证新数据；

作为基础服务，对可用性的要求比一般的服务要高得多，因为下游依赖的服务多，一旦出现故障，有可能会引起级联反应影响大量业务。项目从设计上对以下问题做了处理，保障系统的可用性：

(1) 系统是否有单点？

(2) DB 扩容/维护/故障怎么办？

(3) 系统维护/升级补丁怎么办？

(4) 服务万一挂了如何快速恢复？如何尽量不影响下游应用？

首先是系统层面上做了全栈集群化。Kafka 和 Spark 本身比较成熟地支持集群化运维；Web 服务支持了无状态处理并且通过负载均衡实现集群化；DB 方面携程已经支持主备部署，使用过程中如果主机发生故障，备机会自动接管服务；通过全栈集群化保障系统没有单点。

另外，业务和引进的游戏会越来越多，前端也会随着开发而不断改善，那么如何提高架构的扩展性也是需要考虑的一个问题：

(1) 新引进游戏能否快速加入到数据流里？

(2) 前端的大规模改动是否影响数据后台？

首先 Kafka 可以通过发布 Topic 的方式来添加新游戏，使各个游戏之间的耦合最低，同时 Thrift 的存在，使得前端与数据后台的耦合达到最低，从而前端的改动不会影响数据流程。

万方数据

4.3 功能流程设计

这部分主要讲述了今日概况、今日实时概况、实时搜索、ETL 优化三个部分的功能主要流程设计。

4.3.1 今日概况流程设计

今日概况的流程图如图 4-4 所示：

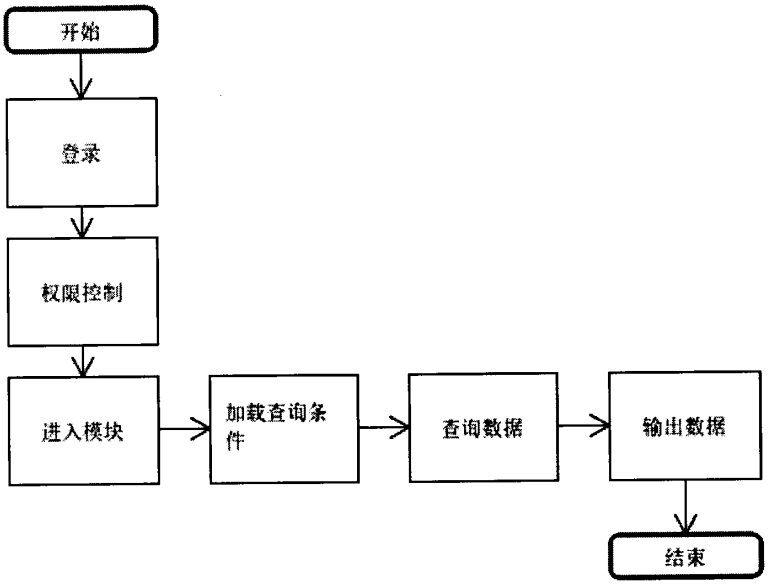

图 4-4 今日概况流程图

Figure 4-4 Flow Chart of Function Today Overview

首先用户进入登陆界面，登录系统后系统会记性权限判断，符合权限则进入系统，同时展示该用户可见的页面，然后加载查询条件查询数据库，最终将数据展示出来。用户没有权限则无法看到该页面的数据，用户可以通过申请来获得某个页面的权限，由于权限控制模块不在本人的负责内，所以本文并没有详细阐述权限控制模块。

4.3.2 今日实时概况流程设计

今日实时概况由于前端和后台采用的 PHP 语言，所以应是前端使用 Ajax 技术

万方数据

定时向后台发送请求，后台根据请求向关系型数据库进行查询，从而将最新数据展示到界面上。考虑到前端轮询的方式简洁且容易实现，所以在这里我们没有使用后台通过 WebSocket 向前端推送数据的模式。

今日实时概况的流程图如图 4-5 所示：

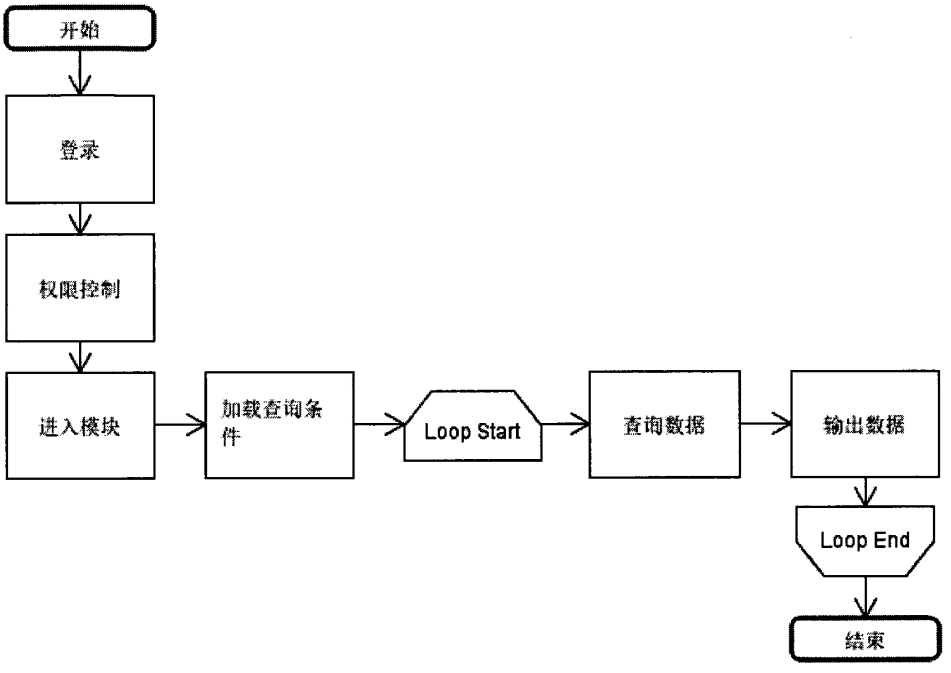

图 4-5 今日实时概况流程图

Figure 4-5 Flow Chart of Function Real-Time Overview

首先用户进入登录页面，登录后后台进行权限判断，用户有实时权限则跳转到实时概况界面，没有则跳转到警告界面。进入界面后，前端直接进行查询最新数据，并将数据展示到前端。

4.3.3 实时搜索流程设计

实时搜索功能主要是将 SQL 查询展示到页面上，并且将集群的信息也展示到页面上，方便进行查询，也方便运营人员进行查询。同时可以将 HDFS 的文件目录展示出来，以便更方便的管理文件目录。为了数据开发人员的方便，未来实时搜索功能还可以选择搜索引擎，不仅限于 SparkSQL，开发 Hive 的 ETL 流程时，还可以选择用 MapReduce 引擎。流程图如图 4-6 所示：

万方数据

具体流程图如下：

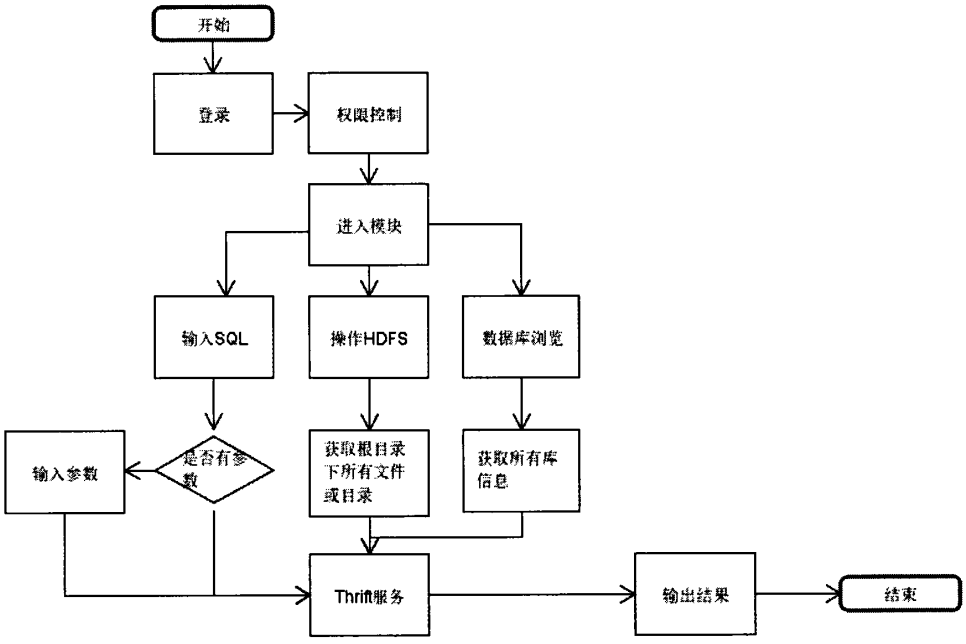

图 4-6 实时搜索流程图

Figure 4-6 Flow Chart of Function Real-Time Query

首先用户进入登录界面，然后后台进行权限查询，符合权限则进入界面。用户可以进行 HDFS 文件的操作；可以进行简单查看数据仓库中的库和表；输入 SQL，如果 SQL 中含有参数，则弹出输入参数提示框，最终将 SQL 发送到 Thrift Server，然后由 Thrift Server 执行相应的 SQL，最后返还搜索结果，展示到界面上。

4.3.4 ETL 优化设计

在需求分析中已经说明，本项目将会采用遗传算法对 ETL 中的数据流优化，使集群中运行的任务的任务强度始终处在一个合适的数值，不至于同时并发太多任务使得集群卡住。

但是实现算法只是这个功能的一部分，在本项目中，我们会初步将优化任务作为定时任务，在所有的任务流之前开启，使得集群中运行的是优化后的任务流。

ETL 优化部分的流程图如图 4-7 所示：

万方数据

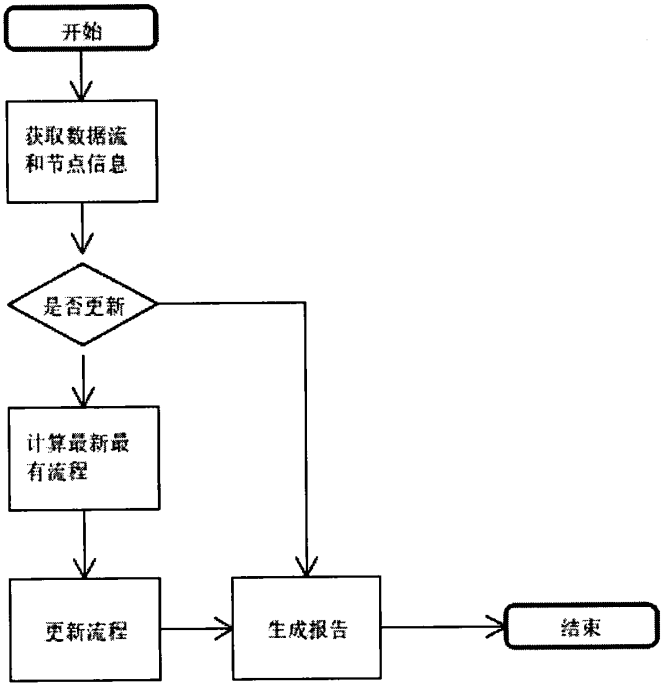

图 4-7 ETL 优化流程图

Figure 4-7 Flow Chart of Function ETL Optimising

每天十二点开始，首先获取存在 MySQL 中的数据流，与昨天的数据流比较，如果有改动则计算最新结果，并且将结果转化成数据流存入 MySQL，并且声称优化报告到本地，如果没有改动则直接生成优化报告文件到本地。

4.4 多酷游戏数据平台数据表设计

在数据平台[26]的设计过程中，最重要的问题之一就是数据的存储，好的数据仓库不仅能加快开发效率，降低表之间的关系依赖，降低事务复杂度，还能减少数据冗余，使各数据表之间的耦合达到最低，减少数据表和 ETL 流程的维护工作，使代码统一、具有可读性，并且加速 ETL 流程的速度。本小节将介绍主要的 Hive 和 MySQL 表结构。

4.4.1 HIVE 表设计

Hive 中存储了数据平台里所有的元数据，包括清洗后的基础数据，维度表数据，和从其他代理商抓取的数据，以及前端后台的日志等。本节主要介绍 Hive 中

万方数据

最重要的用户活跃表、用户注册表和用户付费表的表结构。

(1) 用户活跃表(BASE_USER_PV)

用户活跃表 BASE_USER_PV 里存储所有用户的活跃数据，包括各种维度如机型、渠道号、什么游戏等维度。活跃表是数据流最大的表，代理游戏根据日期分区，独家代理和自研游戏根据日期和游戏分区；由此表可以计算用户的留存，新增用户等统计。表结构如表 4-1 所示。

表 4-1 用户活跃表

Table 4-1 User PV

类型	字段名	注释
string	pv_type	数据类型
string	date	日期
string	ip	ip 地址
string	uid_udid	百度 id
string	imei	设备号
string	ua	机型
string	appid	游戏 id
string	channel	渠道
string	sdkv	sdk 版本
string	ifLogin	是否是登录
string	ifLoginSuc	是否登陆成功
string	count	-
string	standalone_tag	游戏阶段
string	duokuid	多酷 id
string	cuid	-
string	resultCode	状态码

(2) 用户注册表(BASE_USER_REGISTER)

用户注册表 BASE_USER_REGISTER 保存一般是自研游戏或者独家代理游戏用户的注册数据，包括游戏平台、渠道、用户 ID 等字段，根据年月日和游戏分区。表结构如表 4-2 所示。

万方数据

表 4-2 用户注册表

Table 4-2 User Register

类型	字段名	注释
string	pv_type	数据类型
string	date	日期
string	ip	ip 地址
string	uid_udid	百度 id
string	imei	设备号
string	ua	机型
string	appid	游戏 id
string	channel	渠道
string	sdkv	sdk 版本
string	duokuid	多酷 id
string	cuid	-
string	resultCode	状态码

(3) 用户付费表(BASE_USER_PAY)

用户付费表 BASE_USER_PAY 表保存各种维度的用户的付费信息，是运营人员最重视的数据。包括订单号、用户 ID、渠道号等字段，根据日期和游戏分区。表内数据量相对活跃表较小，付费表在 Hive 中以全量的形式存储，分区为年月日，部分游戏如独家代理游戏单独存放于 Hive 目录，并且增加游戏分区，不同游戏在不同的目录下，其余游戏则统一存放在大目录下。定时任务每隔一周清除上一周数据，垃圾箱一个月清理一次，全量存放便于计算新增付费用户和新增付费设备等比较重要的统计项，因此也便于计算 ARPPU、ARPU 等统计项，这些统计项能够非常显著的代表一个游戏的用户质量和营收效果。表结构如表 4-3 所示。

表 4-3 用户付费表

Table 4-3 User Pay

类型	字段名	注释
string	date	日期
string	appid	游戏 id
string	uid	用户 id
string	imei	设备号
string	channel	渠道

万方数据

表 4-3（续表）

Table 4-3 (Continued)

类型	字段名	注释
string	server	服务器
string	player	角色号
string	pay_type	付费方式
string	sdk_order	sdk 订单号
string	pay_money	付费金额
string	pay_result	付费结果
string	pay_actions	付费状态
string	commodity_id	道具
string	order_channel	订单渠道
string	cp_order	cp 订单

4.4.2 MySQL 表设计

MySQL 中保存了统计过后的汇总表，前端展示的数据需请求后台来请求 MySQL 的数据。本节主要介绍今日概况数据表，介绍表结构设计和信息。

(1) 今日概况表(Report_Game_Rank)

今日概况表 Report_Game_Rank 是今日概况模块需要用到的结果数据表，每天由定时任务流开启从 Hive 中的活跃、注册、付费维度表计算结果数据并存入表中。表结构如表 4-4 所示。

表 4-4 今日概况表

Table 4-4 Today Overview

类型	字段名	注释
varchar(60)	date	日期
varchar(60)	server	大区
varchar(60)	channel	渠道
varchar(60)	pv_device	活跃设备
varchar(60)	new_device	新增设备
varchar(60)	pure_device	激活设备
varchar(60)	aver_login_num	平均登陆次数

万方数据

表 4-4（续表）

Table 4-4 (Continued)

类型	字段名	注释
varchar(60)	pay_money	付费金额
varchar(60)	pay_device	付费设备
varchar(60)	arpu	-
varchar(60)	arppu	-
varchar(60)	pay_rate	付费率
类型	字段名	注释
varchar(60)	new_pay_device	新增付费设备
varchar(60)	new_pay_device_money	新增付费设备金额

4.5 本章小结

本章对项目进行了概要设计，将项目的大体框架展示出来，其中还包括了本文没有负责的配置管理部分，为后文的详细设计奠定了基础。

万方数据

5 多酷游戏数据平台详细设计

多酷游戏数据平台分为数据处理和平台展示两大系统，两个系统又根据需求划分成几个不同的子模块，几个子模块协同工作组成数据平台。

本章将重点讲述本人参与到的今日概况和实施概况以及实时查询和 ETL 流程优化四个模块，其余模块不在本章讲述。

5.1 多酷游戏数据平台今日概况模块

用户登录数据平台后首先进入的页面就是今日概况界面，也是每天数据流结果的展示界面，辅佐运营人员决策，多维度的展示付费、活跃等数据。

本小节主要展示今日概况模块的类设计和时序设计，展示方式是类图和书序图。类设计讲述了用到的主要类以及主要类里面的主要方法，以及这些类和方法的用途，时序设计展示了各个类之间的调用过程。

5.1.1 今日概况类设计

今日概况模块的核心类由 SummaryPV、SummaryPay、SummaryDown、UserPVAttribute、GameRankView、GameRankModel、GameRankController 构成。其中 SummaryPV、SummaryPay、SummaryDown 是 Java 实现的 MapReduce，SummaryPV 类负责每天凌晨定时从将原始 Log 数据仓库清洗成活跃数据，并且存放到 Hive 表里，存放增量数据；SummaryPay 类负责每天将以前的历史数据以及新的付费 Log 数据清洗合并，然后存入到 Hive 表里，存放全量数据；SummaryDown 类负责每天将以前的历史数据以及新的下载 Log 数据清洗合并，然后存入到 Hive 表里，存放全量数据。GameRankView 是前端的页面，负责展示前端数据。GameRankModel 负责处理数据，负责查询数据库。GameRankController 负责接收前端请求，调用 GameRankModel，并且将数据返还给 GameRankView 来展示。其余涉及的 HiveSQL 在这里不再赘述。下面逐个介绍每个类。

类关系描述：其中 SummaryPV、SummaryPay、SummaryDown、UserPVAttribute 为 MapReduce 类，在这里没有关系描述，GameRankView、GameRankModel、GameRankController 三个类采用 MVC 架构，互相依赖，互相调用。

SummaryPV：

(1) 类功能描述

万方数据

SummaryPV 类主要实现一个功能：清洗昨天的活跃数据并作为基础表存入 Hive。以增量的形式存入 Hive，以供后面的活跃人数统计，新增玩家统计，留存统计等。

(2) 类主要函数功能描述

1) StandaloneTag101：处理单机游戏里 Tag 为 101 的数据；
2) StandaloneTag45：处理单机游戏里 Tag 为 45 的数据；
3) JudgeEntire：检查数据是否完整；
4) GetFields：获取一条日志中所需要的数据字段；
5) HandleFields：将字段处理成最终的格式；
6) GenerateKey：生成 Map 和 Reduce 的 Key；
7) HandleCounter：操作 MapReduce 的 Counter，实现记数。

SummaryPay 与 SummaryDown 类相似：

(1) 类功能描述

SummaryPay 类主要实现将历史付费数据和最新的付费日志数据清洗并合并，然后存入 Hive 作为付费的基础表，以全量的方式进行存储，可以用来统计各种维度的付费，以及新增付费用户和付费设备等。

SummaryDown 类主要实现将历史下载数据和最新的下载日志数据清洗后合并，存入 Hive 的基础下载表中，全量存储，可以用来分维度统计下载量。

(2) 类主要函数功能描述

1) HandleHistoryData：处理历史数据，生成指定格式的数据；
2) JudgeEntire：检查数据的完整性，关键字段是否存在；
3) HandleData：处理新数据成为指定格式的数据；
4) FillUpData：将数据空缺的字段填充成指定字段；
5) JudgeLegal：判断数据是否合法；
6) GenerateKey：生成 Map 和 Reduce 的 Key；

UserPVAttribute ：

(1) 类功能描述

UserPVAttribute 类主要实现计算每个维度的用户的留存信息，最多纪录至三十天留存，将历史用户留存数据与最新的用户活跃数据聚合到一起，如果有新的活跃，那么就将该用户的最新活跃时间改成今日，最新活跃渠道改为最近一次活跃的渠道。

(2) 类主要函数功能描述

1) HandleHistoryData：处理历史数据，生成指定格式的数据；
2) JudgeEntire：检查新增数据的完整性，关键字段是否存在；

万方数据

3) HandleData：处理新数据，将新数据转化成指定格式；

4) FillUpData：将数据空缺的字段填充成指定字段；

5) JudgeLegal：判断数据是否合法；

6) GenerateKey：生成 Map 和 Reduce 的 Key；

7) UpdateData：将该用户的最新活跃时间、活跃渠道等字段更新；

GameRankModel ：

(1) 类功能描述

GameRankModel 类主要实现访问后台的数据库，并且将返回值处理成指定格式的数据。

(2) 类主要函数功能描述

1) GetData：生成 SQL，并且访问后台数据库，获取数据；

2) ReformData：将数据转化成指定格式数据，并返还给 Controller；

GameRankController ：

(1) 类功能描述

GameRankController 类主要实现接收前端的请求，并且调用 GameRankModel 中访问数据库的方法，并且将返回值处理成指定格式的数据。

(2) 类主要函数功能描述

1) GameRank：生成初始标题，获取当前日期，初始化数据；

2) AjaxGameRank：调用 GameRankModel，使之查询不分维度的聚合数据，并且将返回的数据生成 JSON 格式的数据；

3) AjaxGameByChannelRank：调用 GameRankModel，使之查询不同渠道维度下的聚合数据，并且将返回的数据生成 JSON 格式的数据；

4) AjaxGameByChannelGameRank：调用 GameRankModel，使之查询不同渠道和游戏维度下的聚合数据，并且将返回的数据生成 JSON 格式的数据；

今日概况模块的后台类图如图 5-1 所示：

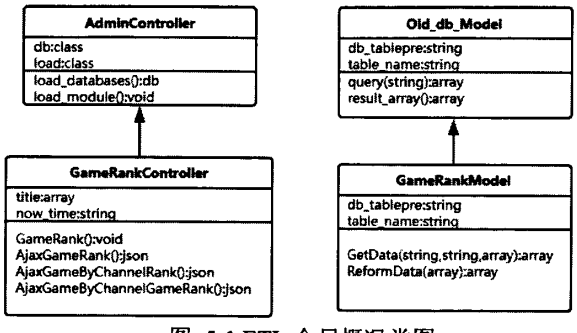

图 5-1 ETL 今日概况类图

Figure 5-1 Class Diagram of Function Today Overview

万方数据

5.1.2 今日概况时序设计

今日概况由于本文负责数据流的清洗和 ETL 部分，以及前端后台的代码工作，所以在时序中，会有额外的模块表示 MapReduce 和 Hive 部分。时序图如下图 5-2 所示：

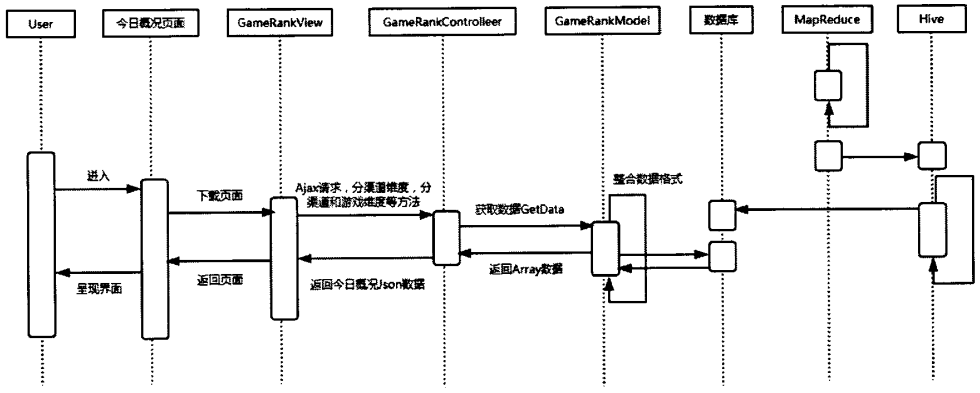

图 5-2 今日概况时序图

Figure 5-2 Sequence Diagram of Function Today Overview

首先用户进入界面，界面根据 GameRankView 生成空页面，再向 GameRankController 请求不同纬度的数据，GameRankController 会调用 GameRankModel 类里的访问数据库方法，并且 GameRankModel 会将数据格式化后返回给 GameRankController，最终 GameRankController 将数据整合并且转换成 Json 格式，返回给 Ajax，进而返回给 GameRankView 来填充界面。

5.2 多酷游戏数据平台今日实时概况模块

用户进入数据平台[13]的实时部分后看到的第一个页面是今日实时概况界面，实时概况模块集中展示了核心游戏的包括登录、注册、付费在内的重要数据。 实时模块展示的都是重点游戏的游戏内数据和重点渠道的游戏概况数据。

本小节主要展示今日概况模块的类设计和时序设计，展示方式是类图和书序图。类设计讲述了用到的主要类以及主要类里面的主要方法，以及这些类和方法的用途，时序设计展示了各个类之间的调用过程。

5.2.1 今日实时概况类设计

今日概况模块的核心类由 OnTimePV、OnTimePay、OnTimeDown、

万方数据

GameOnTimeView、GameOnTimeModel、GameOnTimeController 构成。其中 OnTimePV、OnTimePay、OnTimeDown 是 Java 实现的 Spark 程序，OnTimePV 类负责每天每时每刻定时处理 Flume 从服务器后台拉取到 Kafka 集群的活跃数据，将数据清洗后进行维度统计最终存入到 HDFS 里，最终导入到 MySQL；OnTimePay 类负责每天每时每刻定时处理 Flume 从服务器后台拉取到 Kafka 集群的付费数据，将数据清洗后最终存入到 HDFS 里；OnTimeDown 类负责每天每时每刻定时处理 Flume 从服务器后台拉取到 Kafka 集群的活跃数据，将数据清洗后进行维度统计最终存入到 HDFS 里。GameOnTimeView 是前端的页面，负责展示前端数据。GameOnTimeModel 负责处理数据，负责查询数据库。GameOnTimeController 负责接收前端请求，调用 GameOnTimeModel，并且将数据返还给 GameOnTimeView 来展示。其余涉及的 SparkSQL 在这里不再赘述。下面详细介绍上述各个类的具体情况。

类关系描述：其中 OnTimePV、OnTimePay、OnTimeDown 三个类为 Spark 数据流处理类，在这里没有关系描述，其中 GameOnTimeView、GameOnTimeModel、GameOnTimeController 同样适用 MVC 架构，三个类互相依赖，互相调用。

OnTimePV、OnTimePay、OnTimeDown 三个类相似：

(1) 类功能描述

OnTimePV 类主要实现一个功能：清洗实时的活跃数据并作为基础表存入 Hive。以增量的形式存入 Hive，以供后面通过 SparkSQL 进行实时活跃人数的统计，实时新增玩家的统计等。

OnTimePay 类主要实现一个功能：清洗实时的付费数据并作为基础表存入 Hive。以全量的形式存入 Hive，以供后面进行实时付费的统计等。

OnTimeDown 类主要实现一个功能：清洗实时的下载数据并作为基础表存入 Hive。以全量的形式存入 Hive，以供后面的实时下载设备数的统计，实时新增玩家统计等。

(2) 类主要函数功能描述

1) Standalone：处理单机游戏的数据；
2) Online：处理联网游戏的数据；
3) JudgeEntire：检查数据是否完整；
4) GetFields：获取一条日志中所需要的数据字段；
5) HandleFields：将字段处理成最终的格式；
6) FillUpFields：将空缺字段补成默认值；

GameOnTimeModel ：

(1) 类功能描述

万方数据

GameOnTimeModel 类主要实现访问后台的数据库，并且将返回值处理成指定格式的数据。

1) GetData：生成 SQL，并且访问后台数据库，获取数据；

2) ReformData：将数据转化成指定格式数据，并返还给 Controller；

GameOnTimeController ：

(1) 类功能描述

GameOnTimeController 类主要实现接受前端请求，然后调用 GameOnTimeModel 中的访问数据库功能，并且将返回值处理成指定格式的数据。

(2) 类主要函数功能描述

1) GameRank：生成初始标题，获取当前日期，初始化数据；

2) AjaxGameOnTime：调用 GameOnTimeModel，使之查询所有维度下的聚合数据，并且将返回的数据生成 JSON 格式的数据；

今日实时概况模块的后台类图如图 5-3 所示：

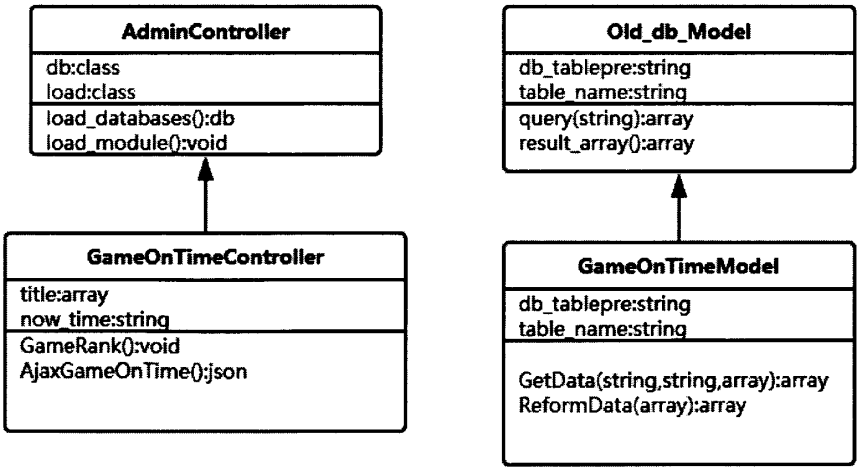

图 5-3 今日实时概况类图

Figure 5-3 Class Diagram of Function Real-Time Overview

5.2.2 今日实时概况时序设计

今日概况由于本文负责数据流的清洗和 ETL 部分，以及前端后台的代码工作，所以在时序中，会有额外的模块表示 Spark 和 SparkSQL 部分。时序图如图 5-4 所

万方数据

示：

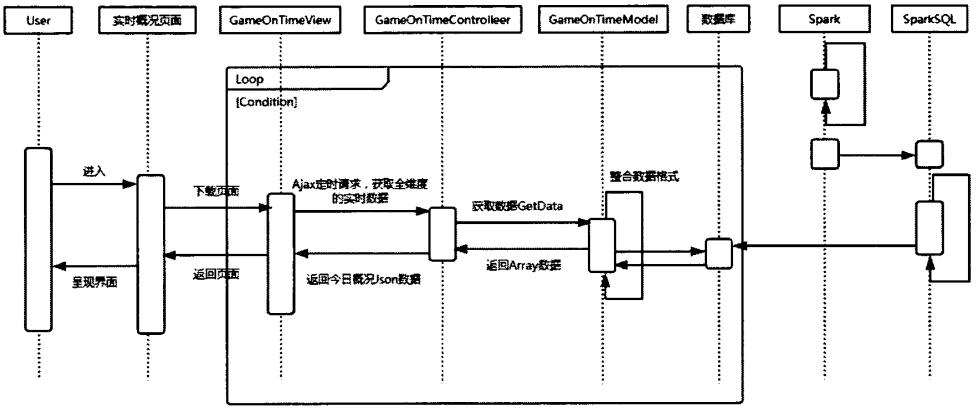

图 5-4 今日实时概况时序图

Figure 5-4 Sequence Diagram of Function Real-Time Overview

首先用户进入界面，界面根据 GameOnTimeView 生成空页面，再向 GameOnTimeController 请求全纬度的数据，GameOnTimeController 会调用 GameOnTimeModel 类里的访问数据库方法，并且 GameOnTimeModel 会将数据格式化后返回给 GameOnTimeController，最终 GameOnTimeController 将数据整合并且转换成 Json 格式，返回给 Ajax，进而返回给 GameOnTimeView 来填充界面。实时效果由前端的 Ajax 定时请求 GameOnTimeController 获取最新数据，并且更新界面，这种方式简单容易实现而且减少后台的负担。

5.3 多酷游戏数据平台实时查询模块

实时查询模块主要分为配置模块，HDFS 管理模块，和查询模块。配置模块主要用来读取配置，目前支持的为 HDFS，和读取 Hive 配置，读取配置后，获取到相应的元数据。HDFS 管理模块是在读取到配置文件和元数据后，将 HDFS 的结构展示到前端，并且前端可以上传和删除、浏览相应的文件，通过前端点击操作的按钮，后台生成相应的命令，再 Java 的 API 来对 HDFS 文件进行相应的操作。查询模块则主要依赖于 Thrift Server，前端输入 SQL 并且检测完参数后，会将完整 SQL 传入后台，后台调用 Thrift 的接口方法，将 SQL 和用户选择的查询计算框架传给 Thrift Server，由 Thrift Server 来执行相应的 SQL 并返回原始查询结果。

下面首先分子模块介绍相应的主要类和关键方法：

万方数据

5.3.1 配置模块

配置模块主要类有 LoadHive 和 LoadHDFS，其中 LoadHive 主要用来读取 Hive 的配置，并将 Hive 的配置读取为静态变量，主要读取 Hive 里面的数据库和数据表。LoadHDFS 主要用来读取 HDFS 的配置，主要用来读取 HDFS 文件系统的结构和文件信息。两个类互为依赖关系。

LoadHive：

(1) 类功能描述

该类用来读取 Hive 配置，读取数据库和数据表的信息，并将之格式化为对象类型变量。

(2) 类主要函数功能描述

1) CheckFileExist：检查配置文件是否存在；
2) JudgeVersion：检查 Hive 版本是否符合要求；
3) LoadHiveMetaData：加载 Hive 元数据；
4) TransformConfig：将 Hive 元数据转化成对象类型变量；

LoadHDFS：

(1) 类功能描述

该类用来读取 HDFS 配置，读取文件目录和文件的信息，并将之格式化为对象类型变量。

(2) 类主要函数功能描述

1) CheckFileExist：检查配置文件是否存在；
2) JudgeVersion：检查 Hadoop[23]版本是否符合要求；
3) LoadHDFSFileTree：加载 HDFS 的文件目录；
4) TransformConfig：将 HDFS 目录转化成树形结构变量；

5.3.2 HDFS 管理模块

HDFS 管理模块主要用到的类是 HdfsDoController，HdfsDoModel。其中 HdfsDoController 主要用来相应前端的请求，并且调用 HdfsDoModel 相应的方法。HdfsDoModel 来实现 HDFS 的具体操作，比如获取用户登录时获取 HDFS 的文件目录；用户点击某个文件后，需要获取到该文件的前几行；用户上传某个文件后，需要将文件上传到 Hadoop。所有的 HDFS 操作通过脚本完成。

HdfsDoController：

(1) 类功能描述

万方数据

该类用来接收前端的 HDFS 操作请求，并且获取到相应的信息如文件的归属用户，文件的权限等信息，最终将参数对象传递给 HdfsDoModel，并等待 HdfsDoModel 的结果，最终将结果返回给前端。

(2) 类主要函数功能描述

1) GetFileInfo：获取文件的信息；

2) HdfsDo：通过前端传来的操作符来进行相应的 HDFS 文件操作，也就是调用 HdfsDoModel 相应的方法；

3) GetFiles：GetFiles 是用来在读取好的树形文件结构中，找到对应的文件的所有子目录，目前并没有做任何的权限控制，所以对文件的操作只需要文件的路径就够了；

4) GetHiveInfo：获取 Hive 的元数据信息。

HdfsDoModel：

(1) 类功能描述

来实现 HDFS 的具体操作，比如获取用户登录时获取 HDFS 的文件目录；用户点击某个文件后，需要获取到该文件的前几行；用户上传某个文件后，需要将文件上传到 Hadoop。

(2) 类主要函数功能描述

1) DecideWhichFunction：决定进行哪个 HDFS 操作；

2) UpdateHDFSConfig：UpdateHDFSConfig 主要用来实现更新 HDFS 的配置，因为用户可能或选择上传文件和删除文件，那么 HDFS 的文件目录就会发生变换，此时就需要改动 HDFS 的树形对象变量；

3) GetHiveInfo：获取 Hive 的元数据信息。

5.3.3 查询模块

查询模块主要用到的类是 SearchController，SearchModel。其中 SearchController 主要用来相应前端的请求，并且调用 SearchModel 相应的方法。SearchModel 来实现生成 SQL，将 SQL 发送到 Thrift Server 的具体操作。

SearchController：

(1) 类功能描述

该类用来接收前端的 SQL 实时查询操作请求，并且获取到相应的参数信息以及选择的计算框架，比如用户在 SQL 里写了${}的字段，那么前端就会提示填写相应的参数，并将参数一并发送给 Controller，最终 SearchController 将参数转化成键值对，并且将 SQL 和参数一起传递给 SearchModel，并等待

万方数据

SearchModel 的结果，最终将结果返回给前端。

(2) 类主要函数功能描述

1) ReformArgs：转换参数为键值对；

2) Search：将 SQL 和参数以及选择的计算框架标志一并传递给 SearchModel，并且等待结果；

3) JudgeResult：判断返回的结果，比如查询成功还是失败，或者创建成功还是失败，并且将结果转化成 Json，传递给前端。

SearchModel：

(1) 类功能描述

将 SQL 和参数合并生成完整 SQL，并且将完整的 SQL 与计算框架标志合并生成 Json，发送到远程 Thrift Server，等待结果。

(2) 类主要函数功能描述

1) GenerateSQL：生成完整 SQL；

2) GenerateArgs：生成发送给 Thrift Server 的参数，包含 SQL，和计算框架标志；

3) SendSQL：将完整参数发送给 Thrift Server，并等待结果，将结果直接返回给 Model；

5.3.4 实时查询类设计

类图中我会适当画出上文没有提到的类，通常是父类或者接口，实时查询类的设计在下面会展示。

有些实体类都是在上面的关键类中常用的类，包括 Hive 类、HIVEDB 类，HIVETable 类，还有 HDFS 类。其中 HIVE 类是存放 Hive 元数据的类，目前只需要存放有多少数据库即可和 Hive 版本即可；HIVEDB 类是 Hive 的数据库数据，主要包含数据库名，数据库含有的表，数据库的创建者；HIVETable 主要包含某个数据库下 Hive 表的信息，主要有 HDFS 路径，表名等信息。

HDFS 类主要以树形结构存放 HDFS 的文件目录，主要内容包含当前文件名，当前文件下的子文件，当前文件的拥有者，当前文件在哪些 Block 的 URL 上，当前文件的大小是多少。类之间互相依赖，互相调用

SqlResultDict 类主要存放 SQL 的各种可能返回结果，包括查询成功，查询失败，查询异常，创建表成功或失败，删除表成功或失败等。将这些可能返回的结果对应成容易辨别和前端展示的字段，并且以键值对的 Map 形式存放。

主要类类图如图 5-5 所示：

万方数据

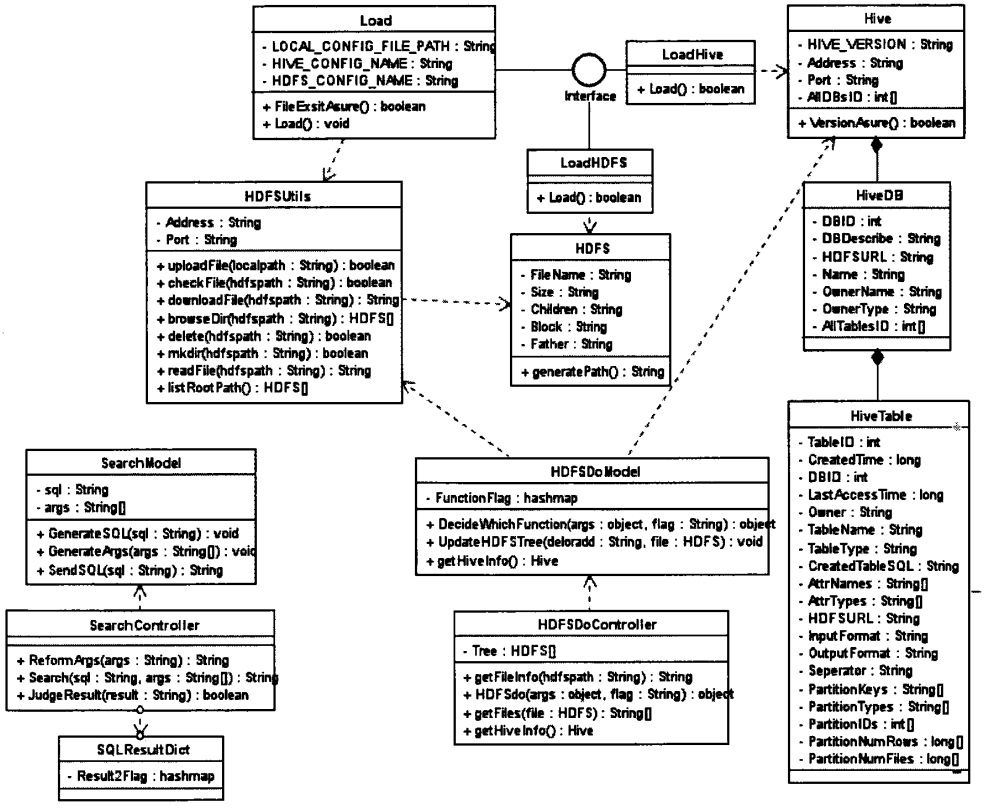

图 5-5 实时查询主要类类图

Figure 5-5 Class Diagram of Function Real-Time Query

5.3.5 实时查询时序设计

实时查询的主要时序图由上述的几个主要类完成。配置管理时序图如图5-6所示：

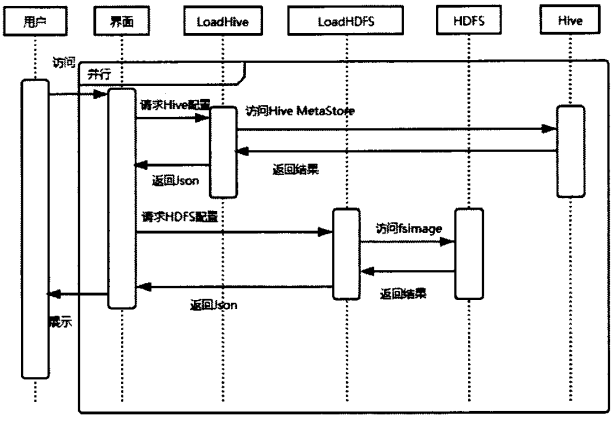

图 5-6 实时查询配置管理时序图

Figure 5-6 Sequence Diagram of Function Configuration Module of Real-Time Query

万方数据

首先用户登录后进入实时查询界面，接着前端 Ajax 会访问 Hive 和 HDFS 的元数据，LoadHive 和 LoadHDFS 会并行访问元数据，并且得到数据后转化成相应格式，返回给前端，前端展示。

HDFS 管理模块时序图如图 5-7 所示：

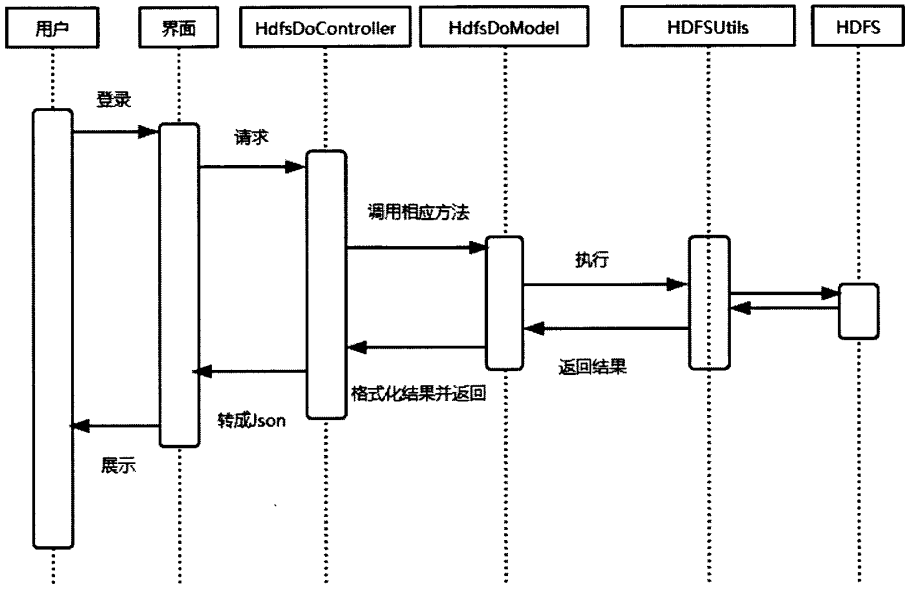

图 5-7 实时查询配置管理时序图

Figure 5-7 Sequence Diagram of Function Configuration Module of Real-Time Query

首先用户登录后进入实时查询界面，接着前端 Ajax 会访问 Controller，Controller 会调用相应的 Model 方法并传入参数，Model 方法会生成命令，通过命令行操作 HDFS，并且等待结果，Model 得到结果后会将结果格式化并返回给 Controller，Controller 转换成 Json 格式返回给 Ajax，进而在前端展示结果。目前项目时使用 Java 运行 HDFS 的命令来实现对文件系统的操控，其中对 HDFS 的一切操作被封装在不同的脚本里，Java 不会直接运行 HDFS 的命令，而是通过运行脚本的方式来运行 HDFS 命令，在脚本中本项目会规范好执行成功的标识和执行失败的标识并且设定好浏览文件的前多少行和后多少行，浏览文件内容并不能浏览全部文件内容，本项目中会限制可浏览的文件大小，超过一定大小的文件会只能浏览前一百行，不超过大小的文件会可以浏览全部内容，以后会逐渐改版为使用 HDFS 的 Java 的 API 来实现对文件系统的操控。

查询模块的时序图如图 5-8 所示：

万方数据

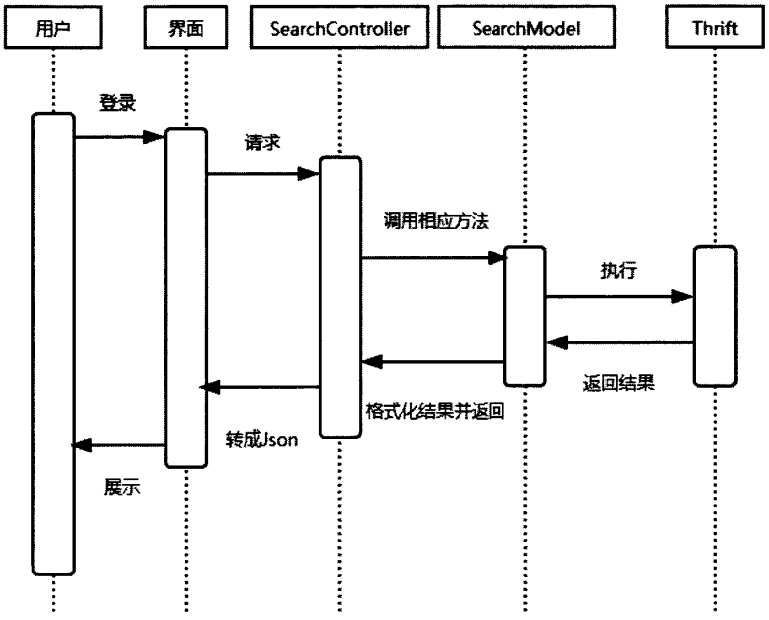

图 5-8 实时查询模块时序图

Figure 5-8 Sequence Diagram of Real-Time Query Module

首先用户登录实时查询界面后，会输入 SQL，首先需要选择查询框架，当前只有 SparkSQL，未来可以选择 MapReduce[40]，当选择完毕并且输入 SQL 后，点击查询。首先前端会检查 SQL 中是否含有参数，当前默认格式为${}，如果有则弹出提示框输入参数，在输入参数后，前端会将选择的查询框架标识、SQL 和参数一起传给 Controller，Controller 调用 Model 的相应方法，并且将参数传给 Model，Model 首先生成完整的 SQL，然后根据选择的查询框架，将 SQL 和标识生成一个 Json，将 Json 对象作为参数调用 Thrift Server 的相应方法，Thrift Server 会进行查询。在查询结束后，Thrift Server 会将原始结果发送给 Model，Model 将结果格式化，由 Controller 转化成相应格式的数据，并最终传递给前端，前端将数据展示展示成报表的形式。

5.4 多酷游戏数据平台 ETL 优化模块

ETL 优化是本项目的一个尝试创新的地方，ETL 优化要做到将数据流任务合理分配，避免同一时间并发过多任务流，将各任务的开启时间平均分配到每个小时，最终达到合理利用集群资源的目的。

万方数据

5.4.1 ETL 优化准备工作

在进行遗传算法[14]之前，我们需要提前测试出：

(1) 当提交任务量的计算总量到达多少时，会使集群超过负载；

(2) 随着任务的提交，执行完所有任务的时间 T 与任务所计算的数据大小 G 和当前集群中所有任务所计算的数据大小 AG 的关系函数（F(T,G,AG)）。

5.4.2 遗传算法流程设计

在本项目中，流程优化选择了启发式算法，遗传算法。因为流程调度[22]属于多种最优解的路径优化问题，所以选择遗传算法来找到最接近最优解的解。

遗传算法[27]主要目标是利用 GA 和启发式调度个子有点，解决生产调度的效率，达到生产调度的全局优化。该算法主要由遗传编码，遗传操作，调度[42]计算，优化评价模块构成，每个模块又对应于一种算法。算法的总体流程如图 5-9 所示：

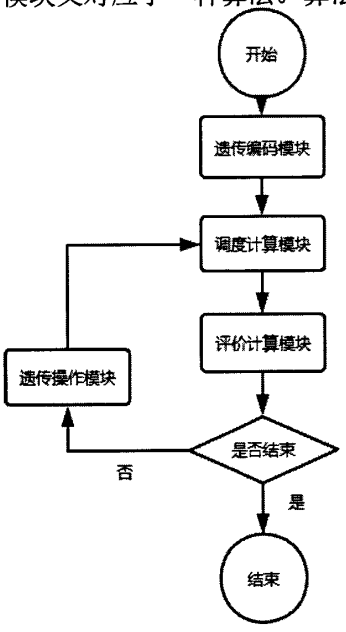

图 5-9 ETL 优化模块算法流程图

Figure 5-9 Flow Chart of Function ETL Optimising

本项目中由于需要优化的对象是分布式任务，可以跨依赖执行，其余特性与车间加工零件无异。所以总结本项目每个任务（流程）的特点为：

(1) 拥有直接依赖流程 A，并且可以依赖所有依赖于 A 的流程；

(2) 在依赖流程没有做完时，流程不会开启；

(3) 每个流程拥有运行时间，计算的数据大小，依赖流程，所属业务四种基本

万方数据

属性；

(4) 当前所有运行的流程所需计算量不应超出集群的负载。

其中，有集群内存大小、集群存储空间等全局属性。

流程之间依赖如图 5-10 所示：

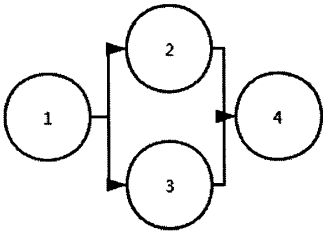

图 5-10 流程示例 1

Figure 5-10 Flow Example 1

则如果任务 2 和任务 3 的计算量大于集群的上限，则集群任务可能会堵塞，则该流程可以改为如图 5-11 所示流程：

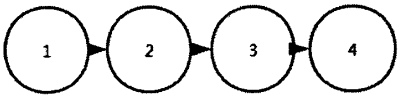

图 5-11 流程示例 2

Figure 5-11 Flow Example 2

5.4.3 遗传算法编码设计

由于本项目要使用遗传算法解决优化问题，所以如何将问题用编码的方式展示时首要要考虑的，即将如何将任务流用 Java 里面的各种数据类型表示出来，目前可以编码成 01 编码，二维数组编码，二进制数组编码。由于任务流的表述太过复杂，很难将其表示成 01 编码，所以在这里本项目初步决定采用直接树形结构表示任务流，外加一个业务维度，即每个业务都有自己的任务流，业务之间互不干扰。这样便能够极大的简化编码方法，但也同时带来其他问题，最典型的就是遗传操作麻烦，找到某个任务节点麻烦，所以本项目将用二维数组的方式来表示树形结构，二维数组里存放的是任务节点的编号，同时每个任务实体类里额外存储该任务在二维数组里的位置，对二维数组进行遗传操作比对树形结构进行遗传操作要简单很多。所以最终决定项目采用二维数组的方式编码一个任务流。同时我们应知道每个任务都有自己的规则，即依赖于谁，任务属于哪个业务。每个业务就是一个任务流，多个业务即多个任务流来组成完整 ETL 流程，多个完整的 ETL 流程组成一个群体，而群体便是遗传算法更新迭代的主体。

(1) 如果每个任务在所有依赖任务完成后开启，则该任务流为合法任务流；

万方数据

(2) 所有合法的任务流为一个群体，即一个解集。

利用上述概念，产生编码链的算法如下

(1) 初始化，给各任务编号，形成初始调度[35]任务集；

(2) 处理任务约束，形成可编码任务集（初始为空集）；

(3) 随机从可编码任务集中挑选一个任务，放入二维编码链；

(4) 重新构造可编码任务集。

若为空，则结束，否则，转向(3)。

在算法的实际实现过程中，如果使用树形结构来表示任务流，太过麻烦，所以在本项目中使用二维数组的方式来表示每个业务的任务流，所以考虑到业务这个维度，每个个体的实际表示使用三维数组，第三位即业务维；

假设有 M 条业务，每条业务的任务数量为 Ni，最多的任务数量表示为 Max(Ni)，则三位数组的每一维分别是[Max(Ni)]* [2*Max(Ni)]*M，第三维为业务维度。

假设不考虑业务维度，二维数组的每一层代表树形结构的每一层，最差情况是所有任务成竖状直线排列，所以最多有 Max(Ni)行。而本项目为了在二维数组中额外存储节点之间的父子依赖关系，每一行的任务排列按照严格顺序排列，同时，非同一父节点之间的任务流，要额外填充一个分隔符，在这里假设分隔符为 X。

下面举例说明，如图 5-12：

该图表示一个 9 个任务的任务流，其中任务 2、3 依赖于 1，其中任务 4、5、6 依赖于 2，任务 7 依赖于 3，任务 8、9 依赖于 7

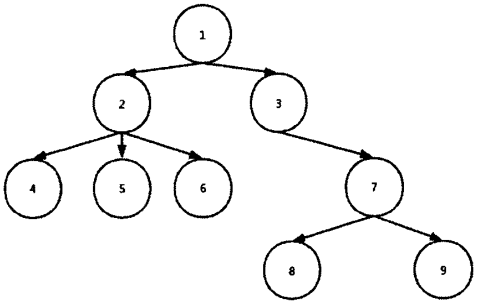

图 5-12 流程示例 3

Figure 5-12 Flow Example 3

图中的圆圈代表一个任务，数字代表任务的编号，箭头代表依赖关系，则该任务流可以表示为二维数组如表 5-1 所示：

万方数据

表 5-1 二维数组示例

Table 5-1 Two-Dimensional Array Example

节点	节点	节点	节点	节点
1				
2	3			
4	5	6	分隔符	7
分隔符	8			

每一层代表任务树的每一层，每一层以分隔符分割，每一层的小组的顺序与上一层的所有元素整体顺序对应，对应相同位置的为上一层的子流程，意味着小组内所有流程依赖于上一层对应位置的流程。

在判断任务流是否合法时，遵循以下步骤：

(1) 遍历所有任务，假设当前循环到任务 Task；

(2) 对于每个任务，顺序循环二维数组，当循环到该任务的依赖任务时，将其记录到一个数组 A 里；

(3) 继续顺序循环二维数组，对于在 A 中存在的所有任务，将安排在其后运行的任务代替其在 A 中的位置；

(4) 当 A 中包含 Task 时，或者遍历完整个数组后，跳出循环；

(5) 如果 A 中包含 Task，且个数等于 Task 的依赖任务个数时，则合法，否则不合法。

在生成出初始群体时，只需要生成每个业务的任务流即可，每个业务的任务流组成三维数组，形成一个个体，所有个体组成初代群体。

在生成初始解时，按照如下规则进行生成：

(1) 随即选择一个任务；

(2) 遍历该个体的依赖流程，并遍历依赖流程所依赖的流程，直至没有依赖流程，则该流程可以当做根节点，并生成树；

(3) 随机选择步骤(2)没有选择的任务，进行步骤(2)，并将生成树合并；

(4) 当没有剩余任务时，初始个体创建完成；

(5) 初始群体创建完成。

5.4.4 遗传算法操作设计

遗传算法[30]通过一系列的交叉、变异、迭代、倒立来实现遗传过程。

在本项目中，采用迭代、较差、变异三种操作来实现遗传算法的流程。

(1) 迭代：迭代是指将整个群体按照适应度高低排序，然后将适应度低的几个个体淘汰掉，即将效果较差的任务流淘汰掉，其他的个体重新组成群体，进行新

万方数据

一轮的过程；

(2) 交叉：交换是随即选择两个适应度较高的染色体，即较好的任务流，对两个配对染色体随机地选择一个交换点，然后对单交换点后的部分进行交换，建立起新串的操作，其目的是在遗传过程中，保留优秀编码子段，并能够杂交优化；

(3) 变异：变异是指对某个适应度较低的染色体，即较差的任务流，随机选择某两个任务接点，将某个任务节点移动到另一个任务节点下；

(4) 倒位：倒立是指对随机的某个染色体，进行某两个点的交换，它保证遗传过程的随机扰动性。上述遗传操作的关键是其随机性，所有操作都是按概率来确定的。一般来讲，交叉操作是必然发生的，两个染色体繁衍的过程较多，而倒立应用较少，同时考虑到本项目中倒立基本会生成一个非法接，因此在本项目不采用倒立操作。

对于调度[38]问题，简单的交换和倒位容易产生违反任务约束的非法解。采用强制非法的方法可解决这一问题，即当且仅当个体 1 的任务 A 和个体 2 的任务 B 交换后合法时，才允许 A 与 B 的位置交换。变异同理，当且仅当同一个体的任务 A 和任务 B 交换后，该个体合法，才允许 A 和 B 进行互换变异。对于倒立则在本项目中没有使用。综上所述对遗传操作算法描述如下：

(1) 遗传：选择适应度排名最靠前的两个个体，随即选择交叉点，两个个任务序列交叉产生两个新的序列，如果新的个体违反约束则重复此流程直至出现合法新个体；

(2) 变异：随即选择一个个体，随即选择两个变异任务点进行变异（将选中的任务点依赖到另一个任务点下），如果新个体违反约束则重复此流程直至出现合法新个体；

(3) 群体繁衍更新：对所有的新个体进行适应度计算然后根据计算结果排序，将末位三个个体淘汰。

5.4.5 遗传算法个体评价设计

遗传算法的最终目的是得到问题的最优解，而如何评价量化染色体即个体的好坏，就是在本小节要说明的问题。每个个体就是每个任务流，遗传算法的效果好坏与适应度计算方法也有很大关系。

在此项目里，性能指标只有所有任务完成的时间，所以最终计算由时间指标来决定。在这里我们要借助准备工作中，得到的该任务的计算量 G 以及当前总计算量 AG 和当前任务执行完所需要的时间 T 的关系函数。

(1) 循环所有业务，取每个业务流的第一个任务，取出任务的所计算数据大小，

万方数据

带入 T(T, G, AG)，得到所有第一个任务的计算所需时间，并根据时间从小到达排列，这些任务组成当前执行任务初始集 S；

(2)取出最先完成的那个，踢出 S，然后把这个任务的子任务，加入到 S，重新计算每个任务的完成所需时间，更新每个任务的状态；

(3)取出最先完成的那个，踢出 S，再重新计算；

……

直至所有任务都完成，并且在此过程中维护一个总时间 T，最终得到 T。

以上步骤得到一个个体的运行时间，循环上述所有步骤，可以得到所有个体的运行时间 T，本项目中完全根据 T 的倒序排序来决定某个个体是否适应度高。

5.4.6 ETL 优化实现类设计

首先 ETL 的优化过程是每天都执行的，而且该模块依赖于 Azkaban 任务调度[39]系统，所以该模块不仅仅需要有算法实现类，而且需要有与 Azkaban 接口的业务类。

其中用到的主要类有 AzkabanHandler，LoadFiles，GAProcess，Unity。AzkabanHandler 类主要用来读取和存储 Azkaban 在 MySQL 里存储的任务流数据，GAProcess 类主要用来实现遗传算法的一系列操作，比如生成初代群体、遗传、变异、计算个体适应度以及群体迭代等操作。Unity 是实体类，代表一个任务。LoadFiles 类主要用来读取配置，如任务信息文件，Azkaban 的配置，尤其读取是否有强制优化的文件。四个类的描述如下：

AzkabanHandler：

(1) 类功能描述

AzkabanHandler 类主要用来读取和存储 Azkaban 在 MySQL 里存储的任务流数据，实际上是对 MySQL 的操作类，比较今日流程和昨日流程是否有变化等。

(2) 类主要函数功能描述

1) GetTodayFlow：从 Azkaban 的数据库里获取今日最新的任务流信息；

2) UpdateFlow：将计算后的任务流更新至数据库；

3) JudgeDifference：比较今日流程和昨日流程是否有变化；

LoadFiles：

(1) 类功能描述

LoadFiles 类主要用来读取存储在本地的 Azkaban 配置文件以及每个任务计算量大小的文件，也就是任务信息文件，同时检查本地是否有强制优化的名为

万方数据

Force 的文件，如果有该文件则应当必须进行优化，同时还主要用来负责存储上次的任务的报告到本地。

(2) 类主要函数功能描述

1) LoadProperties：读取配置好的 Azkaban 信息；

2) LoadFiles：读取配置好的任务信息文件，并且检查是否有强制执行文件；

3) OutputReport：生成本次的优化报告；

GAProcess：

(1) 类功能描述

GAProcess 类主要用来实现算法的每个步骤，包括生成初始群体、遗传、变异、计算个体适应度以及群体迭代等操作，等同于遗传算法的实现类。

(2) 类主要函数功能描述

1) InitializeCluster：随机生成初代群体；

2) SelectInheritance：不完全随机选择个体进行遗传；

3) Inherit：进行遗传操作；

4) SelectVariation：不完全随机选择个体进行变异；

5) Vary：进行遗传变异操作：

6) CalculateFitness：计算每个个体的适应度；

7) Multiplication：种群根据适应度进行迭代；

8) UpdateFlow：每轮迭代完毕都将更新群体。

Unity：

(1) 类功能描述

Unity 类是实体类，主要用来表示一个单独的任务，里面有每个任务的所有基本信息以及一些转换的方法。

(2) 类主要成员变量以及函数的功能描述

1) ID：该任务的编号；

2) Fathers：该任务所依赖的所有任务；

3) Children：该任务目前在目前任务流中的直接后续任务；

4) CalculationScale：该任务的计算量；

5) IdInArray：该任务在二维数组里的位置；

6) Business：该任务所属于的业务 ID：

7) GenerateArray：根据树生成二维数组；

8) UpdateUnity：更新任务节点；

还有群体类 Cluster，里面主要包含 Unity 任务节点，以及生成的二维数组，还有更新群体的方法，由于不是常用类，所以在此不再赘述。

万方数据

根据上述主要类的表述，ETL 优化模块的类图如图 5-13 所示：

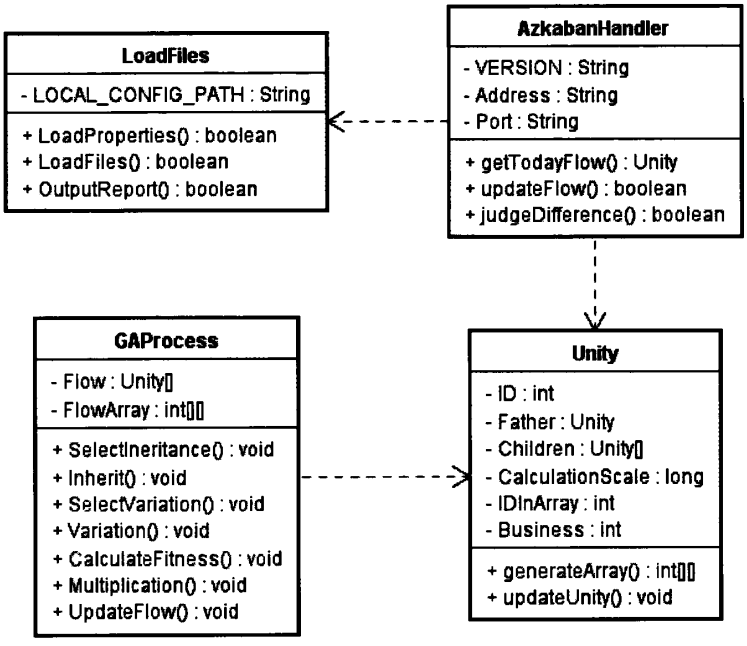

图 5-13 ETL 优化类图

Figure 5-13 Class Diagram of Function ETL Optimising

5.4.7 ETL 优化时序设计

该模块的时序图如图 5-14 所示：

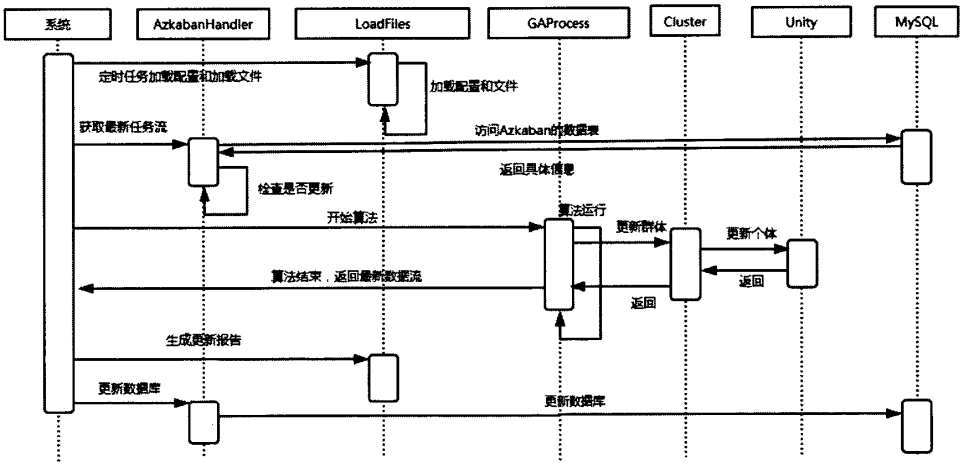

图 5-14 ETL 优化类图

Figure 5-14 Sequence Diagram of Function ETL Optimising

万方数据

首先 Azkaban 定时任务开启，读取配置文件和任务信息文件，然后代码获取到 MySQL 里的最新任务流，并且格式化。判断是否更新后，接着进行算法模块。算法模块结束后，会返回最新任务流，系统会调用相应方法生成本地报告和更新远程数据库。

5.6 本章小结

本章在前面概要设计的基础上，对每个功能模块和每个页面的实现进行了详细设计讲述，并且将数据库的表结构也展示出来，使得整个项目的设计过程一目了然。

万方数据

6　多酷游戏数据平台测试与验收

本章主要介绍了项目验收阶段数据平台的测试与使用。项目总体分三个阶段上线，首先是离线部分的上线，率先验收并且投入使用，满足了绝大部分的离线数据需求。然后是实时部分上线，提高了数据开发效率。最后 ETL 优化部分上线，提高了集群利用率。离线部分测试主要有业务运营人员来检查数据的完整性，实时部分主要测试实时查询的效率以及稳定性，以及功能的友好性。最后 ETL 优化部分测试是一个持久的过程，最终该部分会不断优化遗传算法中的各种参数如迭代次数、繁衍次数、适应度合格线等。

6.1　效果展示

由于 ETL 优化模块没有界面，并且今日概况和今日实时概况界面相同，所以在这里本文只给出今日概况和实时查询模块的效果展示。

(1) 今日概况、今日实时概况如图 6-1 所示

在实时概况中表格中的数据将会实时更新。

日期	区服	渠道	活跃设备	首次激活设备	新增设备	平均登录次数	付费设备	付费金额	ARPU	ARPPU	付费率	新设备付费设备	新设备付费金额
2017-11-24	测试服	-1|全部渠道	0	0	0	0.0	0	0	0.00	0.00	0%	0	0
2017-11-23	测试服	-1|全部渠道	0	0	0	0.0	0	0	0.00	0.00	0%	0	0
2017-11-22	测试服	-1|全部渠道	0	0	0	0.0	0	0	0.00	0.00	0%	0	0
2017-11-21	测试服	-1|全部渠道	0	0	0	0.0	0	0	0.00	0.00	0%	0	0

图 6-1 ETL 今日概况效果图 1

Figure 6-1 Rendering of Function Today Overview1

新设备付费ARPPU	新设备付费率	老设备付费设备	老设备付费金额	老设备付费ARPPU	老设备付费率	首次付费设备	首次付费金额	首次付费ARPPU
0.00	0%	0	0	0.00	0%	0	0	0.00
0.00	0%	0	0	0.00	0%	0	0	0.00
0.00	0%	0	0	0.00	0%	0	0	0.00
0.00	0%	0	0	0.00	0%	0	0	0.00

图 6-2 ETL 今日概况效果图 2

Figure 6-2 Rendering of Function Today Overview2

(2) 实时查询效果图如 6-3 所示

万方数据

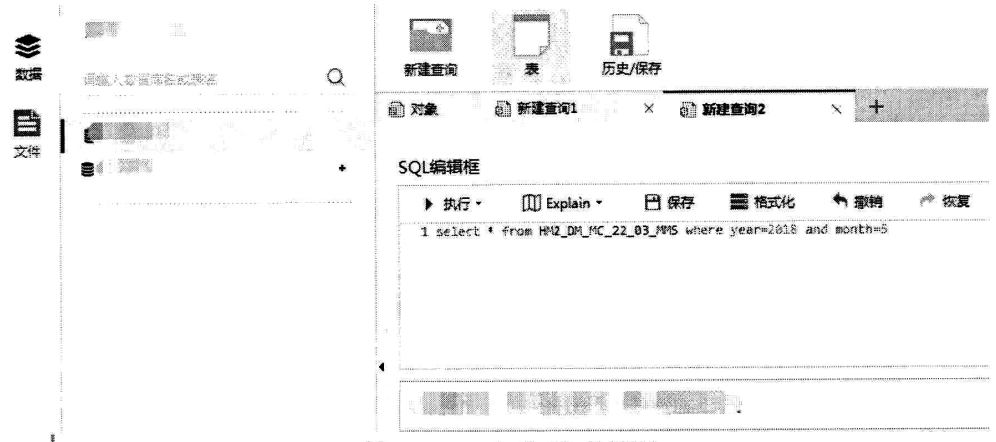

图 6-3 ETL 实时查询效果图 1

Figure 6-3 Rendering 1 of Function Real-Time Query

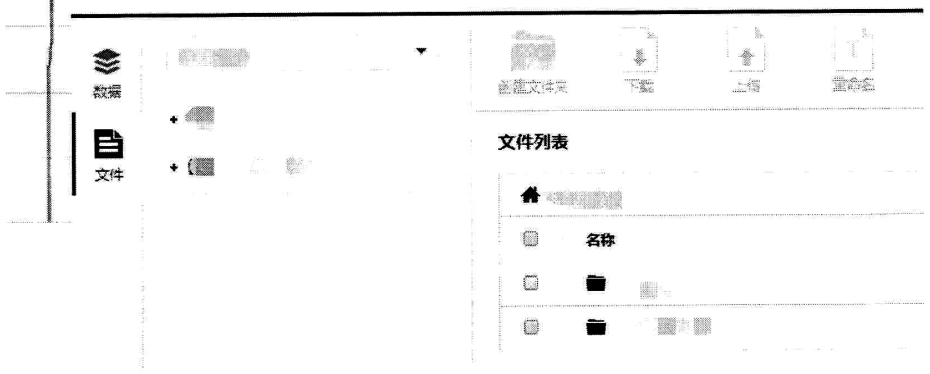

图 6-4 ETL 实时查询效果图 2

Figure 6-4 Rendering 2 of Function Real-Time Query

6.2 系统测试

由于本项目的今日实时概况模块前端没有任何操作，完全通过代码的实时更新完成实时效果，而且 ETL 优化部分主要效果在于对集群资源利用率的提升，并没有任何用户界面，所以下面将重点介绍今日概况模块、实时查询模块的主要测试用例，如表 6-1 和表 6-2 所示。实时概况暂无测试，实时查询测试用例如表 6-3 所示，ETL 优化测试用例如表 6-4 所示。

测试环境描述：

万方数据

表 6-1 测试环境表

Table 6-1 Envoriment of Test

项目	版本	规模
Hadoop	2.7.0	40 台机器集群
Spark	2.2.1	40 台机器集群
Azkaban	3.0	1 台 WebServer，1 台 ExectorServer
Hive	2.3.0	40 台机器集群
Thrift	0.11.0	1 台机器
机器配置	Centos7 64 位	1T 硬盘，36G 内存

表 6-2 今日概况测试用例表

Table 6-2 Test Case of Function Today Overview

用例名称	输入及操作步骤	预计输出
查询今日概况	用户登录界面	输入今日的全维度概况
查询昨日概况	用户登录界面并选择昨天日期	输出昨日的全维度概况
查询分渠道概况	用户登录界面并选择某一渠道查询	输出今日该渠道的概况

表 6-3 实时查询测试用例表

Table 6-3 Test Case of Function Real-Time Query

用例名称	输入及操作步骤	预计输出
测试实时查询	用户登录界面并输入 SQL	SQL 应该输出的结果
测试选择计算框架查询	用户登录界面输入 SQL 并选择另外的计算框架	SQL 应该输出的结果
测试带参数的查询	用户登录界面输入 SQL，SQL 中应含有${XX}之类的参数	SQL 应该输出的结果
测试 HDFS 文件上传	用户选择一个目录并点击上传	应当将文件上传到该目录
测试 HDFS 文件删除	用户选择一个文件并选择删除	应当将该文件进行删除

万方数据

表 6-4 ETL 优化测试用例表

Table 6-4 Test Case of Function ETL Optimising

用例名称

步骤 预计输出

节点	节点	节点	节点	节点	节点
A(1H)	-	-	-	-	-
B(2H)	C(0.8H)	D(1.7H)	-	-	-
分隔	分隔	分隔	E(2H)	F(1H)	
G(0.5H)	H(0.6H)	I(3H)	分隔	J(2H)	K(0.5H)
分隔	分隔	分隔	分隔	L(1H)	

运行代码

节点	节点	节点
A	-	-
C	D	-
B	分隔	F
分隔	E	K
G	分隔	J
H	分隔	L
I	-	-

其中输入和输出全部都是用二位数组表示，分别有 A、B、C、D、E、F、G、H、I、J、K、L 这 11 个节点，括号中的 H 表示该节点任务的用时。输入二维数组表示各个节点与其直接依赖的节点关系树，输出则是生成的优化后的任务树。

6.3 本章小结

本章在前面的讲述基础上，对整个系统进行了展示和测试，保证了平台的每个功能的可用性和可靠性，保证了界面的通俗性，简洁易用。

万方数据

7 结论

本项目最开始对项目背景做了简洁的介绍，并且简述了项目中所用到的一些关键技术，然后根据实际情况详细地做了需求分析，然后根据选择的技术讲述了平台的概要设计，然后在概要设计的基础上逐渐深入，最终展示出本项目的详细设计。本文表述的整个过程有很强的连贯性。

本平台主要有以下三个特点：

(1) 设计了完整维度、统计项的数据表，数据的存储和计算都有详细的设计规范；

(2) 将查询平台开放给运营人员，使操作者不需要很深的技巧就可以使用 SQL 查询原始数据，极大地满足了数据需求；

(3) 数据平台的数据流会每天经过算法优化，保证集群资源的利用率。

目前本平台已经开始使用，并且在使用的过程中不断测试和改进，实时查询功能提供了更灵活的界面去提取数据，同时使得 ETL 开发更快，能够满足绝大部分数据提取的需求，显著提升了工作效率，降低了数据人力成本。提升了运营部门的运营效率，最终带来的是公司的利润增长。

两个概况的页面基本能够展示又有的统计项和数据，并且维度全面，基本能够满足很多需求，并且自动化生产后，不再需要人工维护和查询，降低了成本，节省了人力。

今日概况模块实现了将每日的概况定时展示到前段，极大的方便了运营人员查看每日的游戏概况。方便了工作。

ETL 优化模块实现了定时检查数据流并计算最新的最优数据流并且更新，可以更好的利用集群，可以使得在任务流开启后，也能够额外承担部分查询任务，极大的优化了任务的开启过程。

万方数据

参考文献

[1] 洪焕坪,徐莹莹.中国网络游戏产业现状与发展趋势分析[J].商场现代化,2006(13):276.

[2] Qian Wang,Chao Yu,Yiming Zhang,Huiba Li,Ping Zhong. Delay - bounded skyline computing for large - scale real - time online data analytics[J]. Concurrency and Computation: Practice and Experience,2017,29(10).

[3] 犹锋.电网调度综合数据平台设计与实现[J].电子技术与软件工程,2017(17):169.

[4] 王文东,刘继梅.基于蚁群算法的云计算资源调度研究综述[J].电脑知识与技术,2017,13(23):161-163.

[5] 冉亮,党倩,杨泉伟,金铭,高林.海量历史准实时数据管理平台设计与实现[J].电力信息与通信技术,2017,15(07):20-24.

[6] 李险峰. 基于改进遗传算法的汽车装配生产线平衡问题研究[D].北京科技大学,2017.

[7] 吴金成,余红玲,伍冠桦,龚惠琴.交通一卡通大数据平台的构建研究[J].金卡工程,2017(05):63-66.

[8] K.W Chau,Ying Cao,M Anson,Jianping Zhang. Application of data warehouse and Decision Support System in construction management[J]. Automation in Construction,2003,12(2).

[9] Can Uzunkaya,Tolga Ensari,Yusuf Kavurucu. Hadoop Ecosystem and Its Analysis on Tweets[J]. Procedia - Social and Behavioral Sciences,2015,195.

[10] Harjeet Singh Jaggi,Sunny S. Kadam. Integration of Spark framework in Supply Chain Management[J]. Procedia Computer Science,2016,79.

[11] 王贝. 云计算环境下任务调度优化算法的研究[D].中国科学技术大学,2017.

[12] 段胜泽. 基于 Hadoop 的线缆生产的大数据服务平台的设计与实现[D].电子科技大学,2017.

[13] Katarzyna Rostek. Data Analytical Processing in Data Warehouses[J]. Foundations of Management,2010,2(1).

[14] 李晓璐. 基于模拟退火遗传算法的云计算任务调度的研究[D].华中师范大学,2016.

[15] 廖虹光. 支撑大数据的实时数据集成系统的研究与实现[D].电子科技大学,2016.

[16] 伊毅. 综合数据平台研究及在地市级供电公司的应用[D].华北电力大学(北京),2016.

[17] 朱朝阳,王继业,邓春宇.电力大数据平台研究与设计[J].电力信息与通信技术,2018,13(06):1-7.

[18] Max Klein,Rati Sharma,Chris H. Bohrer,Cameron M. Avelis,Elijah Roberts. Biospark: scalable analysis of large numerical datasets from biological simulations and experiments using Hadoop and Spark[J]. Bioinformatics,2017,33(2).

[19] 黄煜. 先进控制系统中数据监控平台的研究与开发[D].北京邮电大学,2015.

[20] 刘文娟. 并行遗传算法及其在网格任务调度中的应用研究[D].河北工程大学,2013.

[21] 杜冬艳. 电网调度综合数据平台管理研究[D].中国石油大学（华东）,2013.

[22] 仁庆道尔吉. 车间作业调度问题的多目标模型建立及其算法[D].西安电子科技大学,2013.

[23] B.K. Tripathy,Dishant Mittal. Hadoop based uncertain possibilistic kernelized c-means algorithms for image segmentation and a comparative analysis[J]. Applied Soft Computing,2016,46.

[24] Sasmita Panigrahi,Rakesh Ku. Lenka,Ananya Stitipragyan. A Hybrid Distributed Collaborative

万方数据

Filtering Recommender Engine Using Apache Spark[J]. Procedia Computer Science,2016,83.
[25] 郑玲,郑晓天.基于 WebSocket 的电力系统实时数据更新研究[J].计算机与现代化,2013(01):85-87.
[26] 付威. 面向实时数据仓库的达梦数据交换平台改进[D].华中科技大学,2012.
[27] 马永杰,云文霞.遗传算法研究进展[J].计算机应用研究,2012,29(04):1201-1206+1210.
[28] 赵海涛. 电网业务数据采集平台的设计与实现[D].电子科技大学,2012.
[29] Chen Xu. Big Data Analytic Frameworks for GIS (Amazon EC2, Hadoop, Spark)[M].Elsevier Inc.:2013-06-15.
[30] 汤可宗. 遗传算法与粒子群优化算法的改进及应用研究[D].南京理工大学,2011.
[31] 叶文宸. 基于 hive 的性能优化方法的研究与实践[D].南京大学,2011.
[32] 李艳生. 分布式并行计算智能调度策略的研究与实现[D].湖北师范学院,2011.
[33] Chongyang Xue,Feng Liu,Honghui Li,Jun Xiao,Zhen Liu. Research and Design of Performance Monitoring Tool for Hadoop Clusters[M].Springer India:2014-06-15.
[34] 师金钢. 基于 MapReduce 架构的实时数据仓库关键技术研究[D].东北大学,2011.
[35] 马永杰. 大型仓储系统的调度算法研究[D].兰州交通大学,2011.
[36] Shuangxi Chen,Chunming Wu,Yongmao Yu,Tibor Janda. Analysis of Plant Breeding on Hadoop and Spark[J]. Advances in Agriculture,2016,2016.
[37] 尹元韬,王焱.遗传算法改进策略研究进展[J].信息技术与信息化,2010(03):40-45.
[38] 樊唯钦.地区电力调度综合数据平台建设实践[J].电力系统自动化,2010,34(02):113-115.
[39] 王志国. 面向车间管理的离散生产系统精益运行驾驶舱研究[D].浙江大学,2009.
[40] Jyoti V. Gautam,Harshadkumar B. Prajapati,Vipul K. Dabhi,Sanjay Chaudhary. Empirical Study of Job Scheduling Algorithms in Hadoop MapReduce[J]. Cybernetics and Information Technologies,2017,17(1).
[41] 李运芝. 基于蚁群算法的网格任务调度研究[D].大连海事大学,2008.
[42] 孙玉涛. 网格计算环境中的动态任务调度算法研究[D].云南师范大学,2007.
[43] 朱玲. 智能控制系统中的实时数据采集与处理系统设计与实现[D].曲阜师范大学,2006.
[44] 尤海浪,钱锋,黄祥为,胡亮亮.基于大数据挖掘构建游戏平台个性化推荐系统的研究与实践[J].电信科学,2014,30(10):27-32.
[45] 张彦俊. 游戏运营中的数据挖掘[D].复旦大学,2011.
[46] Peter Braun,Alfredo Cuzzocrea,Timothy D. Keding,Carson K. Leung,Adam G.M. Padzor,Dell Sayson. Game Data Mining: Clustering and Visualization of Online Game Data in Cyber-Physical Worlds[J]. Procedia Computer Science, 2017, 112.
[47] 张书豪. 基于 IOS 的移动游戏数据统计分析平台的设计与实现[D].吉林大学,2015.
[48] 江鹤. 面向 CDN 日志业务的数据处理系统的设计与实现[D].中国科学院大学(中国科学院工程管理与信息技术学院),2017.

万方数据

作者简历及攻读硕士学位期间取得的研究成果

一、作者简历

姓名：许仕霖
性别：男
学号：16126229
出生日期：1994 年 5 月 27 日

教育经历：
2012 年 9 月-2016 年 6 月在南昌大学软件学院攻读学士学位
2016 年 9 月-2018 年 6 月在北京交通大学软件学院攻读硕士学位

参加实践工作：
2017 年 6 月-2017 年 10 月 百度多酷游戏公司技术产品部　大数据实习生
2017 年 12 月-2018 年 6 月 联想创投大数据业务部　　大数据实习生

万方数据

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：许仕豪 签字日期：2018 年 5 月 29 日

万方数据

学位论文数据集

表 1.1： 数据集页

关键词*	密级*	中图分类号	UDC	论文资助
移动游戏；数据平台；HDFS；ETL 优化	[redacted]公开			
学位授予单位名称*		学位授予单位代码*	学位类别*	学位级别*
北京交通大学		10004	工程	硕士
论文题名*		并列题名		论文语种*
基于 HDFS 的百度多酷移动游戏数据平台的设计与实现				中文
作者姓名*	许仕霖		学号*	16126229
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西直门外上园村 3 号	100044
工程领域*		研究方向*	学制*	学位授予年*
软件工程		软件工程	两年	2018
论文提交日期*	2018 年 6 月			
导师姓名*	卢苇		职称*	教授
评阅人	答辩委员会主席*		答辩委员会成员	
	何坚			
电子版论文提交格式 文本（√） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） 推荐格式：application/msword；application/pdf				
电子版论文出版（发布）者		电子版论文出版（发布）地		权限声明
论文总页数*	60			
共 33 项，其中带*为必填数据，为 21 项。				

万方数据