

硕士论文初稿

论文名称：优质媒体报表的作业调度系统的设计与实现

姓 名：农佳武

学 号：GS132A153

专业方向：移动云计算

所属院系：软件学院

学院指导老师：林军

企业指导老师：郝文育

企业导师单位：安客诚全球信息服务（南通）有限公司

论文起止时间： 2015.03-2015.12

**2015年9月**

## **目 录**

目 录 I

1. 课题背景介绍 I

2．系统关键技术及服务介绍 VI

3．作业调度系统核心功能分析 XXII

4．系统改进、性能优化总结 XXIV

5．创新点及缺陷 XXV

6．致谢 XXVI

7．参考文献 XXVI

## **课题背景介绍**

1.1.课题背景与意义

进入二十一世纪之后，进入二十一世纪之后，我们已经进入海量数据的时代[1]，每天百度搜索引擎请求次数达到50亿次，每分钟Youtube上传的影片时长为100小时，而Twitter的3亿用户每天发送超过5亿条推文，Google每天需要处理超过30PB的数据[2]。面对如此庞大的数据量，怎样从这些数据中得出有价值的信息？对数据进行深入挖掘的需求凸显。金融公司或银行通过数据挖掘从信用卡交易记录中发现欺骗模式，电信服务商(ISP)通过数据挖掘发现客户变化模式，科研工作者通过数据挖掘发现望远镜观测到的星体。而随着Web2.0和移动应用的蓬勃发展，传统数据库中存储的数据所占比重逐渐减小，存储在NoSQL中的数据越来越多，如微博、即时消息、语音、视频和页面点击等[3-5]。人类互联网的大数据时代，我们面临着一个严重的问题就是信息过载，在互联网时代有许多探索解决信息过载的方法，信息分类网站和搜索引擎就已经在解决信息过载问题上取得了成功。通过信息分类来解决信息过载的网站有Yahoo!和Sina，而Google和百度则是通过搜索来解决信息过载的。在解决信息过载的核心技术中，Map/Reduce就是一种应运而生的面向海量信息处理的云计算编程模型。但是，一个Map/Reduce作业是功能单一的程序，其能完成的功能十分有限。为了完成一个复杂的任务，Map/Reduce作业之间就需要相互协作。目前已有许多面向Map/Reduce的工作流引擎开源软件产品，例如Hadoop生态圈的Hamake、Oozie、Azkaban和Cascading[6]。

互联网的应用与发展不仅促进了各个新兴产业的产生与发展，影响了每个人的生活，同时也为各行业提供了机遇与挑战。对于广告商的营销方式，如投放广告、搞促销活动等方式已经远远不能满足他们需要[7]。常规营销方式的制定往往需要通过较长的周期及过高的成本收集客户数据然后人工分析，再制定相应的营销策略与方式，甚至在缺少数据的情况下盲目进行营销策略的制定，所以很难达到企业预期的效果。而通过数据库营销，企业可以方便地收集和积累客户信息，构建庞大的顾客信息库，然后通过云计算技术对海量数据快速准确地筛选和分析，从而有效地进行客户数据挖掘与客户关系维护[8]。

1.2.课题研究内容

互联网的应用与发展不仅促进了各个新兴产业的产生与发展，影响了每个人的生活，同时也为各类行业提供了机遇与挑战。对于企业来说，常规的营销方式，如投放广告、搞促销活动等方式已经远远不能满足他们需要。常规营销方式的制定往往需要通过较长的周期及过高的成本收集客户数据然后人工分析，再制定相应的营销策略与方式，甚至在缺少数据的情况下盲目进行营销策略的制定，所以很难达到企业预期的效果。而通过数据库营销，企业可以方便地收集和积累客户信息，构建庞大的顾客信息库，然后通过云计算技术对海量数据快速准确地筛选和分析，从而有效地进行客户数据挖掘与关系维护。

以下是优质媒体报表服务的全局架构图:

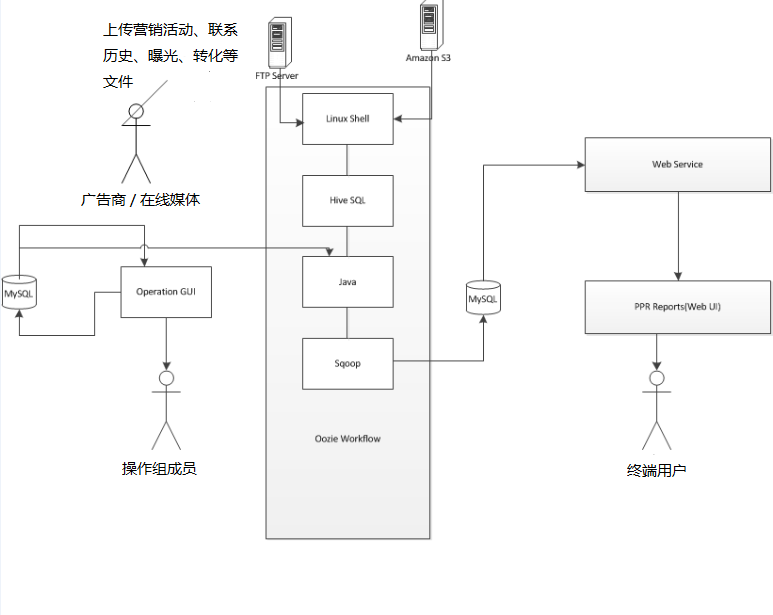


图1 优质媒体报表服务全局架构

PPR的用户主要分为三类：数据文件供应者，操作组成员，以及终端用户。首先用JAVA去读取MySQL数据库，获取上次读取的文件列表、媒体的映射关系，数据文件的装载方式（增量或全量）到Hive临时表，接着由数据文件供应者向FTP服务器或亚马逊S3服务器上传营销活动相关的数据文件，广告商主要提供Campaign，Campaign Cell及Conversion文件，用于提供营销活动的编号信息、参与活动的人群，人群细分信息、以及活动的转化情况等。优质媒体主要提供Exposure文件，用于提供广告曝光次数，广告被点击次数，被保存次数等。实现将广告商和优质媒体数据文件中的人员一一匹配，主要是通过Acxiom提供的Contact History文件。接着由Oozie统一调度，Oozie默认每天定时用Linux Shell到FTP和S3服务器上取出文件。由于优质媒体较多，某些媒体提供的文件需要提取密码后解压，均是在此步进行处理。在判断取得的文件列表与上次取得的有变化后，触发下一步Hive SQL执行。Hive的运算，最终只会更新本次数据文件导致的受影响的数据行，也就是有变化的营销活动相关数据，并不会将原先计算的结果更新。图２展示了Hive的数据预处理过程：

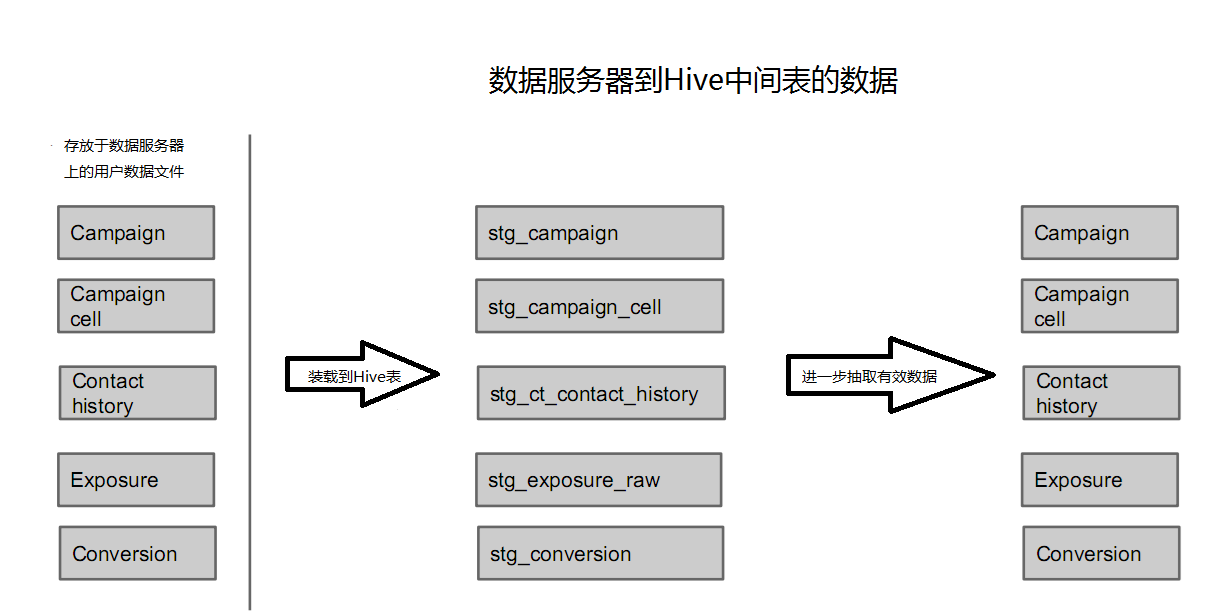


图2 Hive预处理部分的数据流向

Hive的预处理其实相当简单，根据取到的文件，建立Hive表(Load过程)，接着进行表内容的提取(Ingest)，Hive预处理主要是对脏数据做进一步过滤，并对重复数据进行去重。Hive的核心步骤是以Acxiom数据科学家提供的算法对预处理后的数据进行聚合(Aggregate)，随后Hive临时表通过Sqoop同步至MySQL数据库，Oozie部分的工作就算完成了。而在此间的每一步骤执行前，Oozie都会通过JAVA去读取Operation GUI上由操作组成员配置好的参数，才去执行该步骤。而由于一个营销活动与一个广告商是一对一的关系，因而Hive表的第一分区便是基于广告商级别的，第二分区是基于优质媒体的，第三分区基于营销活动，再往下细分则需要根据实际的运算需要。如此一来，有助于利用Hive分区对数据聚合阶段的某些运算进行过滤，缩短工作时间。对于优质媒体报表服务的Oozie核心部分，可以用下图加以概括。



图3　优质媒体报表系统Oozie部分的顶级工作流概貌

可以看到，在每一次Oozie作业失败或是完成后，均会向指定的Email发送邮件，通知管理员工作流的进展情况。在Oozie部分完成后，Web UI部分要负责的是对运算结果的友好呈现，生成多维度的分析报表，这部分均通过调用后台的Web Service完成。

综上所述，本文提出了基于Oozie的营销活动报表分析服务——优质媒体报表（Premium Publisher Reporting）的架构，实现海量数据的处理与存储，并将其加以分析，吸收过往营销活动的经验，集中力量将企业的优势资源应用到新一轮的营销活动中。通过事实的检验，报表分析在企业的每一次营销活动中都起到了至关重要的作用，企业能够利用报表的分析结果对新一轮营销活动的计划或决策进行修正和优化，合理分配、利用企业的资源，科学地设计与实施新一轮的营销活动，不断地朝最大化企业收益的方向前进。文章主要研究内容如下：

(1)总结云计算的相关知识及发展历程，分析 Hadoop 实现的开源云平台，对现有的大数据分析工具与工作流引擎进行归纳和总结。

(2)针对目前海量数据挖掘产品不支持以任务流的方式进行数据挖掘的不足，本文设计基于Oozie工作流引擎的海量数据挖掘工作流模型，研究分析优质媒体报表分析系统其在海量数据挖掘过程中的作业调度。

(3)本文将优质媒体报表的调度过程分为数据导入、Hive 处理、出错检测与回滚、报表生成与展示四个阶段，抽象并模块化这些阶段，然后对这几个阶段的作业调度进行深入分析与探讨。

(4)设计并实现优质媒体报表中扩展Oozie模块功能的内部子系统Operation GUI，实现Hive数据处理工作中产生数据倾斜或系统瓶颈部分的修正，最后，通过实际的客户数据文件对系统进行测试与调优。

1.3.系统总体方案

通过以上分析，实现本系统的总体方案如下：

(1)开发一个基于Oozie的工作流系统，其核心机制是通过Hive脚本对客户（广告商和在线媒体）提供的数据文件进行预处理以及运算分析，最终生成的几张Hive临时表将通过Sqoop同步至MySQL数据库，作为后期Web Service运算的基准表。

(2)开发一个加以辅助的可视化子系统，其用户应为操作组的人员（如此设计主要是为了避免由广告商或媒体商直接操作时，由于对业务的不熟悉而造成不必要的错误），为报表服务提供一个灵活的配置环境以及错误监测机制，扩展作业系统的功能。

(3)优化优质媒体报表服务的系统瓶颈，拆分系统流程，主要是提供运算过程中的数据错误检测分析，并支持Oozie在指定步骤下的重运行。

## **2．系统关键技术及服务介绍**

**2**.1. Hadoop

Hadoop是一个由Apache基金会所开发的分布式系统基础架构。

用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

Hadoop实现了一个分布式文件系统（Hadoop Distributed File System），简称HDFS。HDFS有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS放宽了（relax）POSIX的要求，可以以流的形式访问（streaming access）文件系统中的数据。

Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，则MapReduce为海量的数据提供了计算。

Hadoop 由许多元素构成。其最底部是 Hadoop Distributed File System（HDFS），它存储 Hadoop 集群中所有存储节点上的文件。HDFS（对于本文）的上一层是MapReduce 引擎，该引擎由 JobTrackers 和 TaskTrackers 组成。通过对Hadoop分布式计算平台最核心的分布式文件系统HDFS、MapReduce处理过程，以及数据仓库工具Hive和分布式数据库Hbase的介绍，基本涵盖了Hadoop分布式平台的所有技术核心。

**HDFS**

对外部客户机而言，HDFS就像一个传统的分级文件系统。可以创建、删除、移动或重命名文件，等等。但是 HDFS 的架构是基于一组特定的节点构建的，这是由它自身的特点决定的。这些节点包括 NameNode（仅一个），它在 HDFS 内部提供元数据服务；DataNode，它为 HDFS 提供存储块。由于仅存在一个 NameNode，因此这是 HDFS 的一个缺点（单点失败）。

存储在 HDFS 中的文件被分成块，然后将这些块复制到多个计算机中（DataNode）。这与传统的 RAID 架构大不相同。块的大小（通常为64MB）和复制的块数量在创建文件时由客户机决定。NameNode 可以控制所有文件操作。HDFS 内部的所有通信都基于标准的 TCP/IP 协议。

**NameNode**

NameNode 是一个通常在 HDFS 实例中的单独机器上运行的软件。它负责管理文件系统名称空间和控制外部客户机的访问。NameNode决定是否将文件映射到DataNode 上的复制块上。对于最常见的3个复制块，第一个复制块存储在同一机架的不同节点上，最后一个复制块存储在不同机架的某个节点上。注意，这里需要您了解集群架构。

实际的 I/O事务并没有经过 NameNode，只有表示 DataNode 和块的文件映射的元数据经过 NameNode。当外部客户机发送请求要求创建文件时，NameNode 会以块标识和该块的第一个副本DataNode IP地址作为响应。这个NameNode 还会通知其他将要接收该块的副本的DataNode。

NameNode在一个称为FsImage 的文件中存储所有关于文件系统名称空间的信息。这个文件和一个包含所有事务的记录文件（这里是 EditLog）将存储在NameNode 的本地文件系统上。FsImage和EditLog 文件也需要复制副本，以防文件损坏或 NameNode 系统丢失。

NameNode本身不可避免地具有SPOF（Single Point Of Failure）单点失效的风险，主备模式并不能解决这个问题，通过Hadoop Non-stop namenode才能实现100% uptime可用时间。

**DataNode**

DataNode 也是一个通常在 HDFS实例中的单独机器上运行的软件。Hadoop 集群包含一个 NameNode 和大量 DataNode。DataNode 通常以机架的形式组织，机架通过一个交换机将所有系统连接起来。Hadoop 的一个假设是：机架内部节点之间的传输速度快于机架间节点的传输速度。

DataNode 响应来自 HDFS 客户机的读写请求。它们还响应来自 NameNode 的创建、删除和复制块的命令。NameNode 依赖来自每个 DataNode 的定期心跳（heartbeat）消息。每条消息都包含一个块报告，NameNode 可以根据这个报告验证块映射和其他文件系统元数据。如果 DataNode 不能发送心跳消息，NameNode 将采取修复措施，重新复制在该节点上丢失的块。

**文件操作**

可见，HDFS 并不是一个万能的文件系统。它的主要目的是支持以流的形式访问写入的大型文件。

如果客户机想将文件写到 HDFS 上，首先需要将该文件缓存到本地的临时存储。如果缓存的数据大于所需的 HDFS 块大小，创建文件的请求将发送给 NameNode。NameNode 将以 DataNode 标识和目标块响应客户机。

同时也通知将要保存文件块副本的 DataNode。当客户机开始将临时文件发送给第一个 DataNode 时，将立即通过管道方式将块内容转发给副本 DataNode。客户机也负责创建保存在相同 HDFS名称空间中的校验和（checksum）文件。

在最后的文件块发送之后，NameNode 将文件创建提交到它的持久化元数据存储（在 EditLog 和 FsImage 文件）。

**Linux 集群**

Hadoop 框架可在单一的 Linux 平台上使用（开发和调试时），官方提供MiniCluster作为单元测试使用，不过使用存放在机架上的商业服务器才能发挥它的力量。这些机架组成一个 Hadoop 集群。它通过集群拓扑知识决定如何在整个集群中分配作业和文件。Hadoop 假定节点可能失败，因此采用本机方法处理单个计算机甚至所有机架的失败。

实际开发环境下，Hadoop的搭建在个人测试时使用得更多的是伪分布式环境（即在单节点集群上利用多个线程模拟完全分布式的环境），后来经多次测试总结发现，所用的用户数据稍微大些时，单节点集群略显吃力，数据处理时间过长，导致测试和开发的时间均被阻塞，不足以保证项目的开发进度，后改为完全分布式的Hadoop环境实现。该实现中以Cloudera进行Hadoop环境的安装，其中将一台内网主机作为Master，另外两台内网主机作为Slave，并且在内网环境下配置了一台FTP服务器，在开发环境中尽可能地模拟出生产环境下的程序运行状态。

上述提及的Cloudera 由来自 Facebook 、谷歌和雅虎的前工程师杰夫·哈默巴切 (Jeff Hammerbacher) 、克里斯托弗·比塞格利亚 (Christophe Bisciglia) 、埃姆·阿瓦达拉 (Amr Awadallah) 以及现任 CEO 、甲骨文前高管迈克·奥尔森 (Mike Olson) 在 2008 年创建。在Cloudera系列产品中，我们采用Cloudera Manager实现了Hadoop环境下的管理、监控、诊断和集成功能，Cloudera CDH（Cloudera Distribution Hadoop）对Hadoop进行了相应的封装，简化了Hadoop的安装。Cloudera HUE则是CDH专门的一套web管理器，它包括HUE UI，HUE Server，HUE DB。HUE提供所有的CDH组件的shell界面的接口。利用HUE提供的可视化环境，我可以们很方便地编写Map/Reduce任务，查看修改HDFS的文件，管理Hive的元数据，运行Sqoop数据同步，编写Oozie工作流等大量工作。使用这一系列工具极大程度地简化了我们配置Hadoop环境、进行产品开发以及新特性的功能测试过程。

目前我们配置了两套Hadoop环境，一套用于本地开发测试，另一套用于生产环境的应用部署。利用HUE提供的强大UI界面，我们可以通过分析日志，来了解我们所编写的作业调度系统其运行状况。并且，HUE提供了Hive的执行环境，我们在书写程序的运算逻辑时，可以不用等到文件下载完成便先在HUE环境上做最基本的脚本测试，如此一来，所书写的Hive质量也能在多次调试后得到一定程度的保证。

2.2 Oozie工作流引擎

Oozie是一个工作流引擎服务器,用于运行Hadoop Map/Reduce和Pig 任务工作流。同时Oozie还是一个JAVA Web程序,运行在JAVA Servlet容器中,如Tomcat。 O ozie工作流中拥有多个Action,如Hadoop Map/Reuce Job, Hadoop Pig Job等,所有的Action以有向无环图(DAG Direct Acyclic Graph)的模式部署运行。所以在Action的运行步骤上是有方向的,只能上一个Action运行完成后才能运行下一个Action。

Oozie工作流通过HPDL(一种通过XML自定义处理的语言,类似JBOSS JBPM的JPDL)来构造。

Oozie工作流中的Action在运程系统运行如(Hadoop,Pig服务器上)。一旦Action完成,远程服务器将回调Oozie的接口并通知 Action已经完成,这时Oozie又会以同样的方式执行工作流中的下一个Action,直到工作流中所有Action都完成(完成包括失败)

Oozie工作流中包含可控制的工作流节点(control flow node)和Action节点(action node)。Control flow node其实可以理解为Oozie的语法，比如可以定义开始(start)，结束(end)，失败(fail)节点。开始节点就表示从该节点开始运行。同时也提供一种机制去控制工作流的执行过程,如选择(decision),并行(fork)，join节点。简单地举一个Oozie工作流运行的样例：

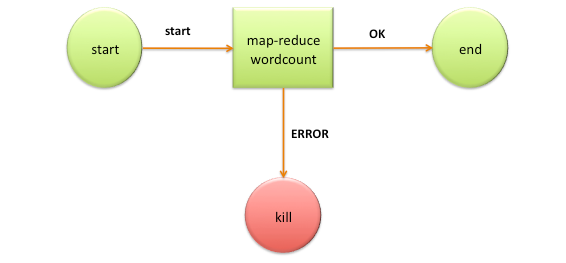


图4 工作流图—WordCount程序的工作流样例

Oozie工作流提供各种类型的Action用于支持不同的需要,如Hadoop

Map/Reduce，Hadoop File System,Pig,SSH,HTTP,Email,JAVA,以及Oozie子流程。Oozie也支持自定义扩展以上各种类型的Action 。

Oozie工作流允许自定义参数,如${inputDir}。



图5 书写Oozie的配置文件

配置打包后通过Oozie Client提交给Hadoop就直接可以运行了，在HUE环境上，我们甚至可以直接将配置文件上传到HDFS处即可启动相应的Oozie 作业。

2.3. Hive 数据仓库

Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供完整的SQL查询功能，可以将SQL语句转换为Map/Reduce任务进行运行。其优点是学习成本低，可以通过类SQL语句快速实现简单的Map/Reduce统计，不必开发专门的Map/Reduce应用，十分适合数据仓库的统计分析。

Hive是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 HQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 Map/Reduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

　　使用Hive的命令行接口，和操作关系数据库很相似，但是Hive和关系数据库还是有很大的不同，从宏观的角度比较Hive和关系数据库的区别，具体如下：

（1）Hive和关系数据库存储文件的系统不同，Hive使用的是Hadoop的HDFS（Hadoop的分布式文件系统），关系数据库则是服务器本地的文件系统；

（2）Hive使用的计算模型是Map/Recude，而关系数据库则是自己设计的计算模型；

（3）关系数据库都是为实时查询的业务进行设计的，而Hive则是为海量数据做数据挖掘设计的，实时性很差；实时性的区别导致Hive的应用场景和关系数据库有很大的不同；

（4）Hive很容易扩展自己的存储能力和计算能力，这个是继承Hadoop的，而关系数据库在这个方面要比数据库差很多。

Hive的架构图如下所示：

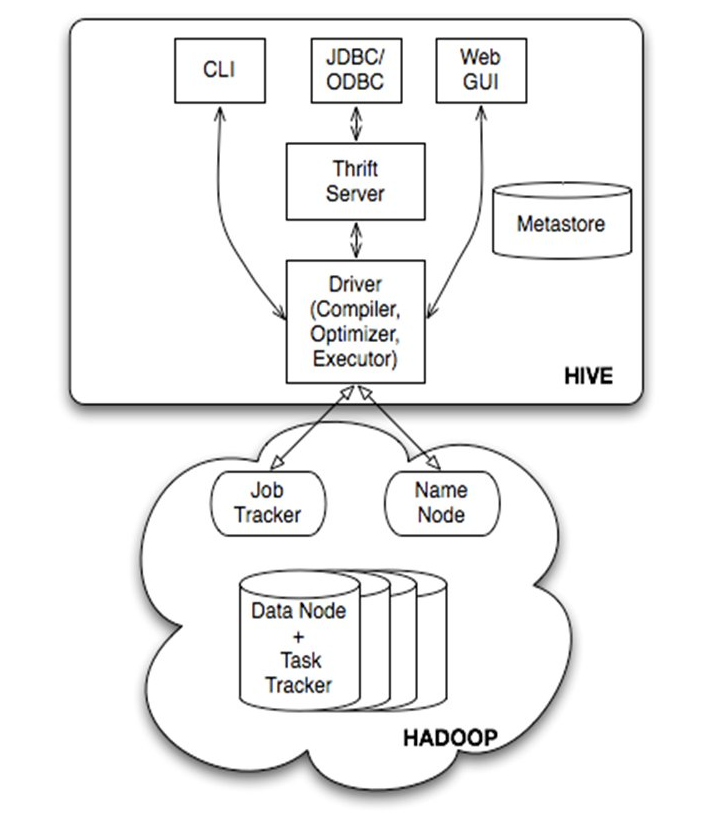


图6 Hive的整体架构图

由上图不难看出，Hadoop 和Map/Recude是Hive架构的根基。

Hive架构包括如下组件：CLI（Command Line Interface）、JDBC/ODBC、Thrift Server、WEB GUI、Metastore和Driver(Complier、Optimizer和Executor)，这些组件可以分为两大类：服务端组件和客户端组件。首先讲讲服务端组件：

**Driver组件**：该组件包括Complier、Optimizer和Executor，它的作用是将我们写的HiveQL（类SQL）语句进行解析、编译优化，生成执行计划，然后调用底层的Map/Recude计算框架。

**Metastore组件**：元数据服务组件，这个组件存储Hive的元数据，Hive的元数据存储在关系数据库里，Hive支持的关系数据库有derby、MySQL。元数据对于Hive十分重要，因此Hive支持把Metastore服务独立出来，安装到远程的服务器集群里，从而解耦Hive服务和Metastore服务，保证Hive运行的健壮性。

**Thrift服务**：Thrift是Facebook开发的一个软件框架，它用来进行可扩展且跨语言的服务的开发，Hive集成了该服务，能让不同的编程语言调用Hive的接口。

客户端组件：

**CLI**：Command Line Interface，命令行接口。

**Thrift客户端**：上面的架构图里没有写上Thrift客户端，但是Hive架构的许多客户端接口是建立在Thrift客户端之上，包括JDBC和ODBC接口。

**WEBGUI**：Hive客户端提供了一种通过网页的方式访问Hive所提供的服务。这个接口对应Hive的HWI组件（Hive Web Interface），使用前要启动HWI服务。

　　着重讲讲Metastore组件,具体如下：

Hive的Metastore组件是Hive元数据集中存放地。Metastore组件包括两个部分：Metastore服务和后台数据的存储。后台数据存储的介质就是关系数据库，例如Hive默认的嵌入式磁盘数据库derby，还有MySQL数据库。Metastore服务是建立在后台数据存储介质之上，并且可以和Hive服务进行交互的服务组件，默认情况下，Metastore服务和Hive服务是安装在一起的，运行在同一个进程当中。我也可以把Metastore服务从Hive服务里剥离出来，Metastore独立安装在一个集群里，Hive远程调用Metastore服务，这样我们可以把元数据这一层放到防火墙之后，客户端访问Hive服务，就可以连接到元数据这一层，从而提供了更好的管理性和安全保障。使用远程的Metastore服务，可以让Metastore服务和Hive服务运行在不同的进程里，这样也保证了Hive的稳定性，提升了Hive服务的效率。Hive的执行流程如下图所示：

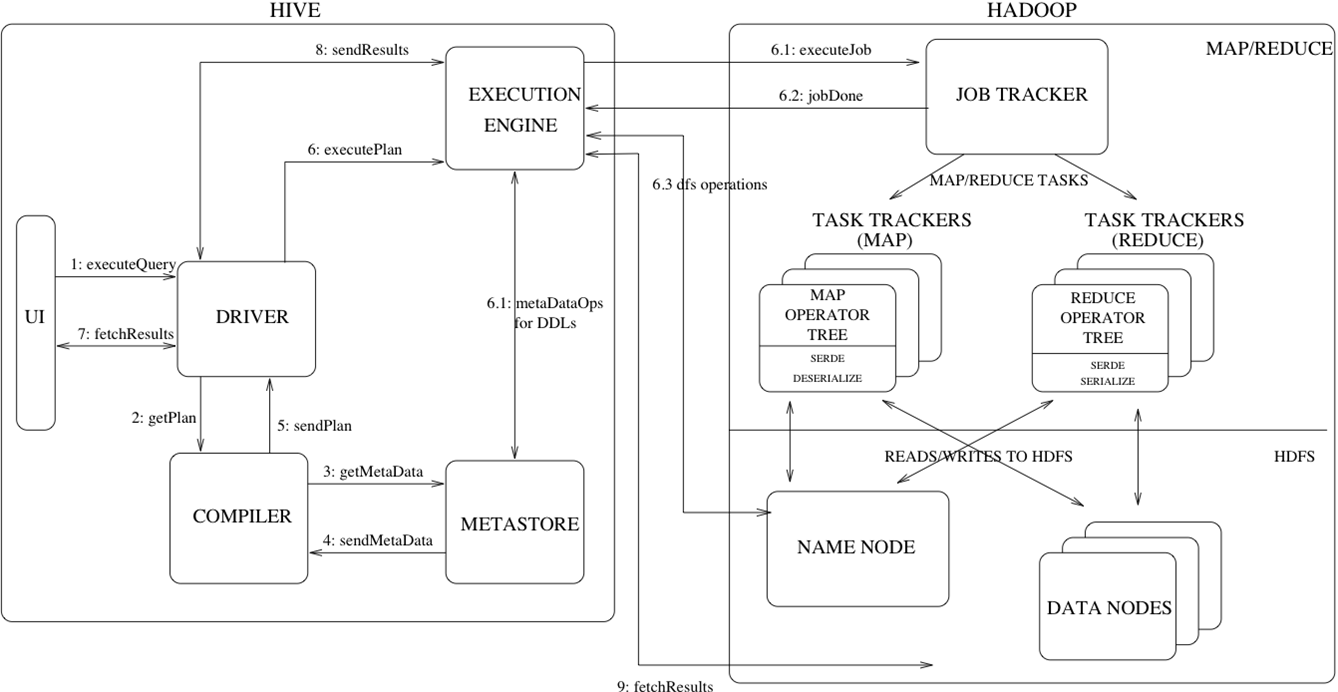


图7 Hive执行流程图

经过上文的了解，现深入分析下Hive与关系数据库的区别，具体如下：

（1）关系数据库里，表的加载模式是在数据加载时候强制确定的（表的加载模式是指数据库存储数据的文件格式），如果加载数据时候发现加载的数据不符合模式，关系数据库则会拒绝加载数据，这个就叫“写时模式”，写时模式会在数据加载时候对数据模式进行检查校验的操作。Hive在加载数据时候和关系数据库不同，Hive在加载数据时候不会对数据进行检查，也不会更改被加载的数据文件，而检查数据格式的操作是在查询操作时候执行，这种模式叫“读时模式”。在实际应用中，写时模式在加载数据时候会对列进行索引，对数据进行压缩，因此加载数据的速度很慢，但是当数据加载好了，我们去查询数据的时候，速度很快。但是当我们的数据是非结构化，存储模式也是未知时候，关系数据操作这种场景就麻烦多了，这时候Hive就会发挥它的优势。

（2）关系数据库一个重要的特点是可以对某一行或某些行的数据进行更新、删除操作，Hive不支持对某个具体行的操作，Hive对数据的操作只支持覆盖原数据和追加数据。Hive也不支持事务和索引。更新、事务和索引都是关系数据库的特征，这些Hive都不支持，也不打算支持，原因是Hive的设计是海量数据进行处理，全数据的扫描时常态，针对某些具体数据进行操作的效率是很差的，对于更新操作，Hive是通过查询将原表的数据进行转化最后存储在新表里，这和传统数据库的更新操作有很大不同。

（3）Hive也可以在Hadoop做实时查询上做一份自己的贡献，那就是和HBase集成，HBase可以进行快速查询，但是HBase不支持类SQL的语句，那么此时Hive可给HBase提供SQL语法解析的外壳，可以用类SQL语句操作HBase数据库。

2.4.Sqoop

Sqoop是一款开源的工具，主要用于在Hadoop(Hive)与传统的数据库(MySQL、PostgreS…)间进行数据的传递，可以将一个关系型数据库（例如：MySQL ,Oracle ,Postgres等）中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。

Sqoop项目开始于2009年，最早是作为Hadoop的一个第三方模块存在，后来为了让使用者能够快速部署，也为了让开发人员能够更快速的迭代开发，Sqoop独立成为一个Apache项目。Sqoop是一个用来将Hadoop和关系型数据库中的数据相互转移的工具，可以将一个关系型数据库（例如：MySQL,Oracle,Postgres等）中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。

对于某些NoSQL数据库它也提供了连接器。Sqoop，类似于其他ETL工具，使用元数据模型来判断数据类型并在数据从数据源转移到Hadoop时确保类型安全的数据处理。Sqoop专为大数据批量传输设计，能够分割数据集并创建Hadoop任务来处理每个区块。[21]

2.5. 亚马逊S3服务器

尽管云计算这一术语并不新鲜（Amazon 在 2007 年就开始提供它的云服务），但从 2008 年起它才开始真正成为流行词，这期间，Google 和 Amazon 的云服务逐渐获得了公众的关注。 Google 的 App Engine 使用户能够在 Google 基础设施上构建和托管 Web 应用程序。连同 S3，Amazon Web Services 还包括 Elastic Cloud Compute (EC2) 计算 Web 服务，该服务可以在 Amazon 的基础设施上托管应用程序。其他公司也开始蠢蠢欲动，准备与 Amazon 和 Google 一决高下，其中就包括 Microsoft的Azure，甚至 Sun Microsystems（其云计算还没有正式推向市场）也想分一杯羹。

以 Amazon 的 S3 产品为例。顾名思义，这是一个公开的服务，使 Web 开发人员能够存储数字资产（如图片、视频、音乐和文档等），以便在应用程序中使用。使用 S3 时，它就像一个位于 Internet 的机器，有一个包含数字资产的硬盘驱动。实际上，它涉及到许多机器（位于各个地理位置），其中包含数字资产（或者数字资产的某些部分）。 Amazon 还处理所有复杂的服务请求，可以存储数据并检索数据。您只需要付少量的费用（大约每月 15 美分 /GB）就可以在 Amazon 的服务器上存储数据，1 美元即可通过 Amazon 服务器传输数据。

Amazon 的 S3 服务没有重复开发，它公开了 RESTful API，使您能够使用任何支持 HTTP 通信的语言访问S3。JetS3t 项目是一个开源JAVA库，可以抽象出使用 S3 的 RESTful API 的细节，将 API 公开为常见的 JAVA 方法和类。

2.6. Bootstrap 前端框架

由于PPR内部的GUI系统所面向的用户主要为操作组成员及日常的开发和测试人员，系统基本上不会面临高并发问题，对于实时性的要求也不高（Hive设计的初衷本就是针对海量数据的离线计算），因此我们选用了jQuery + Bootstrap + SpringMVC + MyBatis + MySQL的架构，着眼于系统核心功能的实现，缩短开发周期。

Bootstrap来自Twitter，是目前非常流行的前端框架。它是基于HTML、CSS、Javascript的，简洁灵活，并且内置许多美观常用的Web组件，使得我们进行Web开发时更加快捷。Bootstrap提供了优雅的HTML和CSS规范，它是由动态CSS语言Less写成。Bootstrap一经推出后颇受欢迎，一直是GitHub上的热门开源项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。国内一些移动开发者较为熟悉的框架，如WeX5前端开源框架等，也是基于Bootstrap源码进行性能优化而来。Bootstrap是一组用于网站和网络应用程序的工具。Bootstrap是自由软件，包括HTML、层叠样式表（CSS）及JavaScript的框架，提供字体排印、窗体、按钮、导航及其他各种组件，并提供了Javascript扩展，旨在使动态网页和Web应用的开发更加容易。Bootstrap是前端框架。所谓“前端”，指的是展现给最终用户的界面。与之对应的“后端”是在服务器上面运行的代码。在2015年6月时，Bootstrap是GitHub上面被标记为“Starred”次数最多的项目。Starred次数超过81,000，而分支次数超过了32,000次。

Bootstrap原名Twitter Blueprint，由Twitter的Mark Otto和Jacob Thornton编写，本意是制作一套可以保持一致性的工具和框架。在Bootstrap之前，开发界面需要使用不同的代码库，这样很容易导致不一致的问题，从而增加了维护的负担。Twitter开发者Mark Otto说：“我和几个开发者一起设计创建一个新的内部使用的工具，然后我们发现有机会可以做更多的事儿。从那之后，我们发现我们设计的工具比别人设计的更强壮。几个月之后，我们做出了Bootstrap的原型，在公司内分享文档、设计和资源。”

经过一个小组几个月之后的努力，Twitter的许多开发者把它当作Hack Week（在Twitter开发者中流行的类似于黑客松的一星期）的一部分，开始参与开发。大家把Twitter Blueprint改名为Bootstrap，并且在2011年8月19日将其作为开源项目发布。此后项目继续由Mark Otto、Jacob Thornton和一个核心开发小组维护，此外还有众多来自社区的贡献者。

在2012年1月31日，Bootstrap 2发布。这一版增加了十二列网格布局和响应式组件，并且对许多组件进行了修改。Bootstrap 3于2013年8月19日发布，开始将移动设备优先作为方针，并且开始使用扁平化设计。

Bootstrap采用模块化设计，并且用LESS样式表语言来实现各种组件和工具。一个名为bootstrap。less的文件包括了这些组件和工具，开发者可以修改这个文件，自行决定项目需要哪些组件。通过一个基本配置文件可以进行有限的定制，此外也可以进行更加深入的定制。

LESS语言支持变量、函数、运算符、组合选择器和一个叫做Mixin（混入）的功能。从Bootstrap 2.0开始，Bootstrap文档包括一个叫做“自定义”的特别选项，开发者可以根据自己的实际需要来选择包含的组件和效果，然后生成和下载已经编译好的包。网格系统和响应式设计以1170像素宽为基准。此外开发者也可以自定义基准。这两种情况下，Bootstrap都能提供四种变体：手机竖屏、手机横屏和平板电脑、PC低分辨率、高分辨率，每个变体都会自动调整网格宽度。

**CSS**

Bootstrap对一系列HTML组件的基本样式进行了定义，并且为文本、表格和表单元素设计了一套独特的、现代化的样式。

**可重用组件**

除了基本HTML元素，Bootstrap还包括了其他常用的界面元素，例如带有高级功能的按钮（例如按钮组合、带有下拉菜单选项的按钮、导航栏、水平和垂直标签组、导航、分页等等）、标签、高级排版、缩略图、警告信息、进度条等。

这些组件都使用CSS的类实现。在页面中需要将其对应到特定的HTML元素上面。

**Javascript组件**

通过jQuery，Bootstrap加入了一些Javascript组件。它们提供了例如对话框、工具提示、轮播等功能。此外还增强了一些用户界面元素的功能，例如输入框的自动完成。Bootstrap 2.0支持以下Javascript插件：Modal（模态对话框）、Dropdown（下拉菜单）、Scrollspy（滚动监听）、Tab（标签页）、Tooltip（工具提示）、Popover（浮动提示）、Alert（警告）、Button（按钮）、Collapse（折叠）、Carousel（轮播）、Typeahead（输入提示）、Affix（附加导航）。

图8 Boostrap实现的基本模板样例

2.7. SpringMVC

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的SpringMVC框架或集成其他MVC开发框架，如Struts1，Struts2等。

**MVC框架是什么**

模型-视图-控制器（MVC）是一个众所周知的以设计界面应用程序为基础的设计模式。它主要通过分离模型、视图及控制器在应用程序中的角色将业务逻辑从界面中解耦。通常，模型负责封装应用程序数据在视图层展示。视图仅仅只是展示这些数据，不包含任何业务逻辑。控制器负责接收来自用户的请求，并调用后台服务（manager或者dao）来处理业务逻辑。处理后，后台业务层可能会返回了一些数据在视图层展示。控制器收集这些数据及准备模型在视图层展示。MVC模式的核心思想是将业务逻辑从界面中分离出来，允许它们单独改变而不会相互影响。

在Spring MVC应用程序中，模型通常由POJO对象组成，它在业务层中被处理，在持久层中被持久化。视图通常是用JSP标准标签库（JSTL）编写的JSP模板。控制器部分是由dispatcher servlet负责，在本教程中我们将会了解更多它的相关细节。

一些开发人员认为业务层和DAO层类是MVC模型组件的一部分。我对此持有不同的意见。我不认为业务层及DAO层类为MVC框架的一部分。通常一个WEB应用是3层架构，即数据-业务-表示。MVC实际上是表示层的一部分。

**Dispatcher Servlet(Spring控制器)**

在最简单的Spring MVC应用程序中，控制器是唯一的你需要在Java web部署描述文件（即web。xml文件）中配置的Servlet。Spring MVC控制器 ——通常称作Dispatcher Servlet，实现了前端控制器设计模式。并且每个web请求必须通过它以便它能够管理整个请求的生命周期。

当一个web请求发送到Spring MVC应用程序，dispatcher servlet首先接收请求。然后它组织那些在Spring web应用程序上下文配置的（例如实际请求处理控制器和视图解析器）或者使用注解配置的组件，所有的这些都需要处理该请求。

在Spring3.0中定义一个控制器类，这个类必须标有@Controller注解。当有@Controller注解的控制器收到一个请求时，它会寻找一个合适的handler方法去处理这个请求。这就需要控制器通过一个或多个handler映射去把每个请求映射到handler方法。为了这样做，一个控制器类的方法需要被@RequestMapping注解装饰，使它们成为handler方法。

handler方法处理完请求后，它把控制权委托给视图名与handler方法返回值相同的视图。为了提供一个灵活的方法，一个handler方法的返回值并不代表一个视图的实现而是一个逻辑视图，即没有任何文件扩展名。你可以将这些逻辑视图映射到正确的实现，并将这些实现写入到上下文文件，这样你就可以轻松的更改视图层代码甚至不用修改请求handler类的代码。

为一个逻辑名称匹配正确的文件是视图解析器的责任。一旦控制器类已将一个视图名称解析到一个视图实现。它会根据视图实现的设计来渲染对应对象。

2.8. MyBatis

MyBatis 是支持普通 SQL查询，存储过程和高级映射的优秀持久层框架。MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

每个MyBatis应用程序主要都是使用SqlSessionFactory实例的，一个SqlSessionFactory实例可以通过SqlSessionFactoryBuilder获得。SqlSessionFactoryBuilder可以从一个xml配置文件或者一个预定义的配置类的实例获得。

用xml文件构建SqlSessionFactory实例是非常简单的事情。推荐在这个配置中使用类路径资源（classpath resource)，但你可以使用任何Reader实例，包括用文件路径或file://开头的url创建的实例。MyBatis有一个实用类——Resources，它有很多方法，可以方便地从类路径及其它位置加载资源。

(1)加载配置：配置来源于两个地方，一处是配置文件，一处是Java代码的注解，将SQL的配置信息加载成为一个MappedStatement对象（包括了传入参数映射配置、执行的SQL语句、结果映射配置），存储在内存中。

(2)SQL解析：当API接口层接收到调用请求时，会接收到传入SQL的ID和传入对象（可以是Map、JavaBean或者基本数据类型），Mybatis会根据SQL的ID找到对应的MappedStatement，然后根据传入参数对象对MappedStatement进行解析，解析后可以得到最终要执行的SQL语句和参数。

（3）SQL执行：将最终得到的SQL和参数拿到数据库进行执行，得到操作数据库的结果。

(4)结果映射：将操作数据库的结果按照映射的配置进行转换，可以转换成HashMap、JavaBean或者基本数据类型，并将最终结果返回。

MyBatis 最强大的特性之一就是它的动态语句功能。如果您以前有使用JDBC或者类似框架的经历，您就会明白把SQL语句条件连接在一起是多么的痛苦，要确保不能忘记空格或者不要在columns列后面省略一个逗号等。动态语句能够完全解决掉这些痛苦。

　　尽管与动态SQL一起工作不是在开一个party，但是MyBatis确实能通过在任何映射SQL语句中使用强大的动态SQL来改进这些状况。动态SQL元素对于任何使用过JSTL或者类似于XML之类的文本处理器的人来说，都是非常熟悉的。在上一版本中，需要了解和学习非常多的元素，但在MyBatis 3 中有了许多的改进，现在只剩下差不多二分之一的元素。MyBatis使用了基于强大的OGNL表达式来消除了大部分元素。

## **3．作业调度系统核心功能分析**

**3**.1. Oozie部分核心功能分析

本文将优质媒体报表的调度过程分为数据导入、Hive 处理、出错检测与回滚、报表生成与展示四个阶段。现具体分析上述每一阶段具体的运作流程：

第一阶段，数据导入。广告商和在线媒体会根据要求，不定期地将其在网络中所做的营销活动用户数据以约定的格式上传至S3服务器或FTP服务器，可以是简单的CSV文件，也可以是压缩后的gzip文件。作业调度系统的第一步初始化历史信息，连接上文件服务器尝试拉取文件，如果发现有压缩包则进行解压，并删除原文件。在这一步骤中极有可能因网络不良而失去连接，针对这一情况，我们已经在上文提及，在GUI系统中对FTP的重连次数进行配置，若多次连接失败，管理员和开发人员将会收到错误邮件。在解压完毕后，会根据数据文件名生成一个记录了文件列表的文件，这一文件将用于与前一天的列表文件进行对比，目的是当发现了文件有变动时，进行Oozie的下一步骤。

当Shell脚本检测出文件变动后（一般是分析出已经具备了转化Conversion文件，可以开始对特定营销活动进行运算），将通知Oozie进行下一步操作。此时，Oozie会将文件列表写入ＭySQL（这一信息用于记录工作流的运行情况，还能为下一步Hive临时表的生成提供对比依据）。

第二阶段，将进行Hive处理。Hive脚本首先会检测是否已经存在相应的广告商和在线媒体，如果没有则在HDFS内新建一个文件夹，以广告商、在线媒体以及营销活动ID作为分区信息，目的是在进行表间运算时，利用分区信息加快表间连接或是条件筛选的速度。Hive SQL会把文件映射成表，将不同类型的文件映射成不同的数据表，这就完成了数据装载(Load)和抽取(Ingest)的过程。这两个阶段在开始和结束时，都会与数据库同步信息，将装载前、装载后、抽取前、抽取后处理的表行数记录到数据库（此后的每一阶段都会同步这些处理信息），而这些统计信息会在OperationGUI中进行整理和展示，目的是为运行状况提供监控数据。工作流会继续往下进行数据的聚合（Aggregation）运算，在广告商+媒体+营销活动的级别生成一份基准表，同时也会根据OperationGUI内的配置信息，生成维度更细的基准表（到营销活动不同单元划分的级别），所生成的基准表都是在为最终的报表UI提供服务。最后会以Sqoop的方式，将基准表增量式地更新到远程MySQL服务器上，至此，Hive运算的环节就结束了。

第三阶段，出错检测与回滚。在前两个阶段的作业调度过程中有提到，工作流会在每一步骤开始与结束处向数据库同步当前的数据处理信息，这一统计信息我们可以在OperationGUI中查看并检查。如果在统计表中发现异常数据，比如在数据抽取过程开始时，数据量共3000条；而在某一Hive SQL进行数据抽取结束后，抽取的数据量仅为100条，我们基本上可以断定这一步骤相关的运算出现了错误。一方面可以在HUE中查看日志，复查问题产生的原因。另一方面可以在OperationGUI中定位可能产生问题的Hive脚本。如果需要进行回滚操作，我们则需要向服务器上传一个符合本次运算特征的文件（通常是Conversion文件），通知Oozie工作流触发此步骤的再次运行。

第四阶段，Web UI对用户进行报表展示。由于Hive已经处理并生成了几种类型的基准表，对于用户感兴趣的指标，我们已经可以通过一般的SQL语句处理得到报表数据，除个别指标需要进行离线计算外（如某个营销活动的组合测量值指标），其余的基本可以通过在线运算实现。而需要进行离线计算的报表，我们也会预先生成六个常用指标的基准表，当不符合用户需求时再做额外的运算。如此，报表运算的压力已经从各个环节分解出去。

3.2. OperationGUI子系统核心功能分析

GUI系统最核心的功能便是辅助Oozie工作流，为其提供最大程度的灵活性，这一特点主要表现在以下几点：

（1）用户具备了对自己期望的文件装载方式进行配置的途径，即便是Hive不支持存储过程，在数据库与Oozie的交互过程中，作业调度系统能根据所配置的参数选择相应的Hive SQL执行；

（2）工作流中定义的每一步骤开始和结束的运行情况，都会同步至数据库中，而这些待分析的数据都在OperationGUI中得到展示，是提供系统分析的关键环节；

（3）不同的用户可以为自己定义测量值的组合，也可以为自己关注的测量值定义别名，当在Web UI里进行报表展示时，系统将根据用户类型动态展示报表指标，在GUI这里控制有利于放宽用户文件的格式限制（毕竟广告商和在线媒体的数据文件在他们长期运营中已经形成自身独特的格式），整个系统已经相当灵活。

## **4．系统改进、性能优化总结** 一个长期稳定运行的系统是不存在的，稳定只是相对的概念。即使是系统未开发完成，在我们发现诸多意料不到的问题同时发生时，一定要抽出时间回头审视系统架构，探讨是否有必要对系统进行改造，而不是一味地拖延开发进度。本系统在开发初期发现了基本Oozie对业务支持的局限性，立即从以书写Oozie的xml配置文件控制工作流的方式改为纯Java调度作业系统的方式，一方面使作业调度的工作仅由一位开发人员进行控制，其余相关的Shell脚本，Hive SQL，错误监控分析等开发工作得以分发出去，并行进行。另一方面在此基础上稍加改进，通过Java调度时将中间结果同步数据库的方式，增加了工作流运行过程中的运行情况监控数据记录，使得日志更加立体而全面，也使得作业调度的单步重运行成为现实，极大程度地扩展了工作流的功能。

结合本系统持续几个月的运行状况，对于Hive部分的性能优化，我们有以下几点经验总结：

（1）好的业务模型设计，会计数据处理事半功倍。

（2）解决数据倾斜问题，可以有效提升系统稳定性，优化时把握整体，单个作业最优不如整体最优。

（3）设置合理的map reduce的task数，能有效提升性能。(比如，10w+数量级的计算，用100多个reduce是相当浪费的，1个足够)。

（4）自己动手写Hive SQL解决数据倾斜问题是个不错的选择。set hive.groupby.skewindata=true;这是通用的在参数层面的算法优化，但算法优化总是漠视业务，习惯性提供通用的解决方法。数据仓库开发人员往往更了解业务，更了解数据，所以通过业务逻辑解决倾斜的方法往往更精确，更有效。

（5）对count(distinct)不加处理的方法，尤其数据大的时候极容易产生倾斜问题，不能抱侥幸心理。

（6）对小文件进行合并，是行至有效的提高调度效率的方法，假如我们的作业设置合理的文件数，整体调度效率也会产生积极的影响。

## **5．创新点及缺陷**

基于以上分析及实际产品的运行情况总结，与主流的工作流调度系统相比，本系统所具备的创新点主要有以下几点：

（1）GUI子系统弥补了Oozie工作流引擎无可视化界面支持的短板，提供的配置环境让系统的调度更为灵活。

（2）整个系统统一使用Java程序调度并提供各个阶段的日志信息，有效地弥补了Oozie作业失败时不利于分析维护的缺点。

（3）该系统提供了Oozie在指定步骤的单步重运行的功能，为不必要再次计算的环节节省了大部分的时间。

该系统仍然存在以下缺陷：

（1）虽然作业系统已经可以支持单步重运行，但仍需要向S3服务器上传一份可以触发特定步骤的文件，如何使这一操作能更加简单地实现，依然是我们努力改进的方向。

（2）随着用户的逐步增加，报表业务将越来越复杂，Hive脚本的书写、运算过程分析难度加大，使得有时候对于异常的Hive脚本我们难以处理，系统维护和优化的难度不断加大，如何增加系统的鲁棒性和可扩展性是我们努力的方向。

## **6．致谢**

## **7．参考文献**

[1] Kang E L, Braverman A J. Learning from Massive Distributed Data Sets[C]//AGU Fall Meeting Abstracts. 2013, 1: 02.

[2] Shah G, Shet K C. Design Architecture-Based on Web Server and Application Cluster in Cloud Environment[J]. arXiv preprint arXiv:1403.5392, 2014.

[3] Pokorny J. NoSQL databases: a step to database scalability in web environment[J]. International Journal of Web Information Systems, 2013, 9(1): 69-82.

[4] Cattell R. Scalable SQL and NoSQL data stores[J]. ACM SIGMOD Record, 2011, 39(4): 12-27.

[5] 姚林,张永库. NoSQL的分布式存储与扩展解决方法[J]. Computer Engineering, 2012, 38(6).

[6] 李乔,郑啸.云计算研究现状综述[J].计算机科学, 2011, 38(4): 32-37.

[7] 徐晓峰,大规模用户在线行为数据分析. 上海交通大学. 2013-01-11

[8] 章伟星,基于Hadoop的海量广告日志分析系统的设计与实现. 哈尔滨工业大学. 2013-06-01

[9] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//ACM SIGOPS Operating Systems Review. ACM, 2003, 37(5): 29-43.

[10] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J].

Communications of the ACM, 2008, 51(1): 107-113.

[11] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[J].

ACM Transactions on Computer Systems (TOCS), 2008, 26(2): 4.

[12] Wang L, Von Laszewski G, Younge A, et al. Cloud computing: a perspective study[J]. New Generation Comput ing, 2010, 28(2): 137-146.

[13] Calder B, Wang J, Ogus A, et al. Windows Azure Storage: a highly available cloud storage service with strong consistency[C]//Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011: 143-157.

[14] Amazon Web Services[EB/OL]. http://aws.amazon.com.

[15] IBM：积极推进“大数据”时代革新[J]. 中国电子报，2011，(22)：116.

[16] 阿里云：开放数据处理服务 ODPS [EB/OL]. www.aliyun.com/product/odps.

[17] Yan J, Yang Y, Raikundalia G K. A Data Storage Mechanism for Peer -to-Peer Based Decentralised Workflow Systems[C]//SEKE. 2003: 354-358.

[18] Cascading. Open Source Implementation of Cascading[EB/OL]. <http://www.cascading.org/>.

[19] Zhang C, De Sterck H. Cloudwf: A computational workflow system for clouds based on Hadoop [M]//Cloud Computing。 Springer Berlin Heidelberg, 2009: 393-404.

[20] 朱明. 数据挖掘[M]. 中国科学技术大学出版社, 2008.

[21] Apache Sqoop：云端大数据分析的关键一环 http://www.cstor.cn/textdetail\_8101.html