

**本 科 生 毕 业 设 计**

**开题报告**



**学生姓名: \_\_\_ 汤利康\_\_\_\_\_\_\_**

**学生学号: 3110102975**

**指导教师: 陈岭**

**年级与专业: 11级计算机科学与技术**

**所在学院: 计算机学院**

**一、题目： 基于直方图的大数据查询系统优化**

**二、指导教师对开题报告、外文翻译和中期报告的具体要求：**

**指导教师（签名）**

**年 月 日**

**毕业设计开题报告、外文翻译和中期报告考核**

**导师对开题报告、外文翻译和中期报告评语及成绩评定：**

|  |  |  |  |
| --- | --- | --- | --- |
| **成绩比例** | **开题报告**  **占（20%）** | **外文翻译**  **占（10%）** | **中期报告**  **占（10%）** |
| **分 值** |  |  |  |

**导师签名**

**年 月 日**

**答辩小组对开题报告、外文翻译和中期报告评语及成绩评定：**

|  |  |  |  |
| --- | --- | --- | --- |
| **成绩比例** | **开题报告**  **占（20%）** | **外文翻译**  **占（10%）** | **中期报告**  **占（10%）** |
| **分 值** |  |  |  |

**开题报告答辩小组负责人（签名）**

**年 月 日**

目录

[本科毕业设计开题报告 5](#_Toc288305490)

[1. 项目背景 5](#_Toc288305491)

[2. 目标和任务 5](#_Toc288305492)

[3. 可行性分析 5](#_Toc288305493)

[4. 初步技术方案和关键技术考虑 5](#_Toc288305494)

[5. 预期工作结果 5](#_Toc288305495)

[6. 进度计划 5](#_Toc288305496)

[本科毕业设计外文翻译 5](#_Toc288305497)

# 本科毕业设计开题报告

## 项目背景

* 1. **Impala大数据实时查询系统**

Impala是Cloudera公司主导开发的新型大数据查询系统，它提供SQL语义，支持对存储在HDFS和HBase的数据进行快速地、实时地查询。Impala使用与Hive相同的数据存储平台，元数据，SQL语法（Hive SQL），ODBC驱动程序和用户界面（Hue Beeswax）。Impala还提供了一个面向批量的实时查询平台。

在Cloudera的测试中，Impala的查询效率相比于Hive有数量级的提升。从技术角度分析，Impala系统性能提升如此之大主要有以下几点原因：

1. Impala不会把查询中间结果写入磁盘而避免了大量的I/O操作。
2. Impala系统通过自带的服务进程来进行作业调度，省掉了MapReduce作业启动的开销。
3. Impala抛弃MapReduce这个不太适合SQL查询的范式，而是使用MPP并行数据库思想，以便做更多的查询优化。
4. 使用了Data Locality的I/O调度机制，避免了网络数据传输带来的成本。

Impala开源项目自发布以来，以其优秀的性能，受到了学术界和工业界的广泛关注。目前，Cloudera已经发布了Impala2.0。本文中所介绍的Impala特点和将来所做的系统优化都针对于2.0。

* 1. **系统架构**

Impala是基于Hadoop的分布式的、高并发的SQL引擎。其主要架构如图1.1所示。

从图中可以得出，一个Impala系统主要由三个服务进程所构成。

1. Impald服务进程

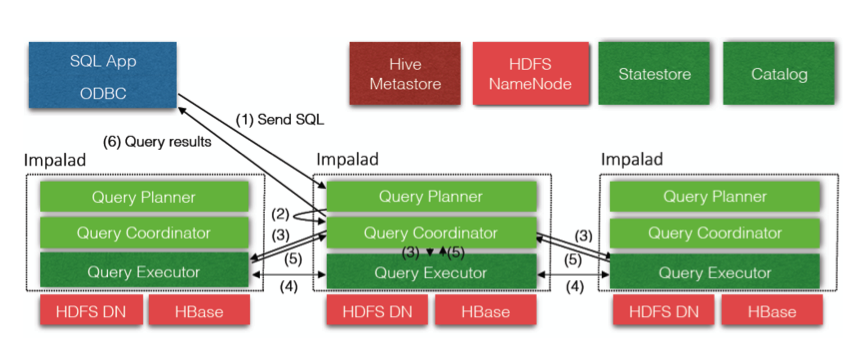
Impalad服务负责从Client接收查询请求，生成查询计划分发给其它Impalad，最后接收整合其它Impalad进程返回的结果，将结果返回给Client。它主要由查询计划器、查询协调器和查询执行器组成。其中，查询计划器负责将接收到的Client端查询请求转换为查询计划。这其中包括了查询计划树的生成，查询优化，查询分片等步骤。查询协调器根据查询计划以及集群状态分配查询任务，并最终收集各个节点返回的查询结果返回给Client。而查询执行器负责查询计划的执行。

1. Statestore进程

Statestore进程负责监控整个Impala系统。每个Impalad服务进程在启动时都会向Statestore注册，而当集群中的某一个节点发生故障时，Statestore将该故障信息通知给集群中所有的节点。这样，系统中的查询任务就不再会被分配给故障节点，提高了系统的鲁棒性。

1. Catalog服务

Catalog服务负责管理系统的元数据信息。Impala会缓存元数据信息。当某个Impalad服务进程处理DDL请求时，其它Impalad进程的元数据信息已经过时，这时就需要Catalog进程更新其它Impalad进程的元数据信息。

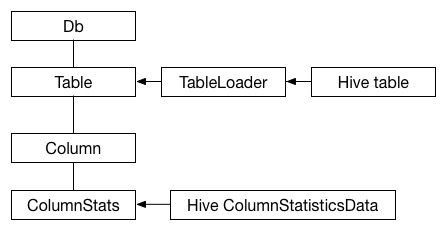
 图1.1

**1.3 Catalog服务原理与不足**

本文1.2节中已经提到Catalog的主要功能。因为接下来的优化主要是针对Impala系统中的Catalog模块。所以在以下篇幅中将介绍Catalog的原理以及目前存在的一些不足。

Catalog主要分为前端和后端两部分。

在每一个Impalad服务进程的前端部分，维护了如图1.2所示结构。

 图1.2

当某一个Impalad服务初始化时，会初始化一个Catalog对象，在其对象中包含了一个数据库缓存，接着载入Hive或者HBase中的数据库元数据信息。在初始化时，系统会为数据库中每张表创建一个入口，但是并不载入具体信息。当有某一个DDL请求涉及到相关Table时，Impalad会使用TableLoader类载入Hive中表的具体元数据信息。

而Catalog的后端部分只存在于一个节点上。它的主要功能是负责搜集各台主机上的Catalog更新信息。它主要维护了一个Catalog的版本号，任何节点上的Catalog改变都会引起其版本的改变。另外在后端Catalog上运行了一个GatherCatalogUpdatesThread线程来监控Catalog版本号，当发现后端版本号发生变化时，该线程会获取发生变化的元数据对象列表，将变化部分广播给整个系统上的每个节点。

通过以上的描述，可以得知目前Catalog只维护了Db、Table和Cloumn信息。但是对于一些查询优化方法来说，仅仅有这些信息还是不够的。如果能够在已有的Catalog的基础上增加一些统计信息，无疑会对查询优化提供更大的支持。因此，本文接下来的工作都将围绕该点展开。

**1.4代价模型与中间结果估计**

在制定查询计划时，查询优化是必不可少的，而查询优化的关键点之一是建立有效的代价模型。其中基于代价函数的代价模型用代价函数来表示查询处理的代价。

而为了提高代价模型的准确度，需要估计查询处理的中间结果的大小。直方图方法能够提供较精准的中间结果估计。而在直方图方法中，Maxdiff（V，A）方法误差率较小。

直方图在普通直方图的基础上限定，考虑每对相邻值之间的差。桶的边界是具有β-1最大差的对，其中β是用户指定的桶数。

基于以上，可以得出如果在Catalog服务中增加直方图的统计。将会大大有利于查询优化效率提升。

## 目标和任务

在现有Impala系统Catalog服务的基础上，增加直方图信息统计模块，以提高代价模型的准确度，为查询优化提供更大的空间。

直方图信息统计模块针对数据表中的列建立直方图，提供给查询优化器使用。为达到以上目标。我们将需要完成的任务分为三个阶段。

1. 熟悉Impala系统现有的Catalog服务的基础架构和工作流程。
2. 实现Maxdiff（A,V）直方图创建算法。使用JDBC连接数据库获得列数据，并创建直方图。
3. 将第二阶段中创建的直方图创建模块集成到Impala系统中，使Impala系统能够创建和维护直方图信息，提供给查询优化器使用。

## 可行性分析

* 1. **主要功能性能要求**

该项目添加的直方图信息统计模块需要能够较快地将请求需要的列直方图信息提供给查询优化器。让请求直方图数据引起的查询延迟竟可能小。因此，我们在创建直方图时不阻塞请求，而是判断请求是否会改变数据，如果改变了数据则在请求完成之后更新直方图信息。以此保证不造成用户请求延时。另外，我们认为在大数据环境中，查询操作的频数要比插入更新操作大很多，所以直方图的更新的频率不会很高，系统成本也不会很高。

* 1. **关键技术和风险**

1. 直方图计算

通过JDBC获得某一列数据，根据Maxdiff（A，V）直方图算法创建直方图。

1. 各个节点之间直方图信息共享

为了减少网络数据传输，我们将直方图的创建任务分配给该列数据所在节点。所以在每个节点上都可能存在部分直方图信息，当别的节点需要直方图信息时，需要向数据节点请求。我们将已经创建的直方图节点信息保存到Catalogd服务进程。节点通过Catalogd服务进程向直方图所在节点请求直方图信息。网络请求通过Thrift框架完成。

1. Thrift框架介绍

thrift是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml 这些编程语言间无缝结合的、高效的服务。在Impala工程中就使用了Thrift实现了后端和前端之间的通信。

## 初步技术方案和关键技术考虑

* 1. **直方图统计流程**

1. Catalog收到查询请求。
2. 调用直方图创建模块处理该请求，获得请求中涉及到的Column列表。对涉及到的Column列表建立直方图。根据Data-locality原则，将直方图的创建发送给该列所在数据节点执行。因为当数据量较大时，建立直方图的时间会比较慢。所以这个操作是在后台进行的。不会阻塞请求。该直方图会被缓存，在下一次涉及到相同Column的请求发生时会被利用到。所以针对于某一列的第一次请求事实上是无法使用直方图信息的。
3. 当某一个直方图建立完成后，会向Catalog进程报告已经建立某几个Column的直方图信息。当其它节点需要用到时，可以向该节点请求。在收到其它节点的直方图请求的同时，将会触发直方图统计模块，重新对直方图进行统计。
   1. **直方图创建方法**

当收到一个列的创建直方图请求后，通过JDBC连接HIVE查询该列数据，根据数据创建直方图。

* 1. **直方图创建算法**

本设计使用Maxdiff（A，V）直方图来估计中间结果。具体算法如下。

1. 从数据库请求经过排序的列数据。
2. 遍历列数据，将相同频数的列值合并为一个桶。当发现不相同频数的列值时，进入第3步。
3. 判断当前桶的平均基数是否与上一个相邻桶相同，如果是进入第4步，否则进入第5步。
4. 将当前桶与上一个相邻桶合并。进入第7步。
5. 判断当前直方图中的桶数是否达到最大桶数限制，如果是则将直方图中最小桶和其后桶合并。进入第6步。
6. 将当前桶加入直方图中。进入第7步。
7. 将当前桶设为上一个桶，返回第2步。

## 预期工作结果

1. 实现Maxdiff（A,V）直方图算法。能够根据数据库中获取到的列数据创建直方图。
2. 在Impala系统中增加直方图模块。能够较为快速地创建和维护直方图信息，提供给查询优化器使用。

## 进度计划

|  |  |  |
| --- | --- | --- |
| 时间范围 | 完成工作 | 备注 |
| 4月10日前 | 熟悉相关代码，完成总体设计。 |  |
| 4月20日前 | 完成直方图创建算法，能够通过JDBC查询某一列数据，创建直方图。 |  |
| 5月10日前 | 将直方图创建集成到Impala的Catalog中。能够根据查询请求创建和更新直方图。 | 直方图的创建由直方图列所在数据节点执行。 |
| 5月20日前 | 完成毕业设计报告 |  |
| 5月30日前 | 准备答辩 |  |