云计算时代的大数据查询分析优化

无法建索引

陈杨明  
计算机科学与技术/硕士1年级  
 哈尔滨工业大学（深圳）  
 中国-深圳  
 510717841@qq.com

徐鹏  
计算机科学与技术/硕士1年级  
哈尔滨工业大学（深圳）  
中国-深圳 [2633129202@qq.com](mailto:2633129202@qq.com)

施鉴洋  
计算机科学与技术/博士1年级  
哈尔滨工业大学（深圳）  
中国-深圳  
19b951026@stu.hit.edu.cn

王勋广  
计算机科学与技术/硕士1年级  
哈尔滨工业大学（深圳）  
中国-深圳  
 2872841099@qq.com

团队简介

团队的成员均是哈尔滨工业大学（深圳）计算机科学与技术学院的研一在读学生，其中队员三人均来自ices实验室。指导老师张晓峰教授，也是来自ices实验室的一员，负责教授“高级数据库原理”课程。

组长陈杨明是本次比赛的核心成员，为整个系统设计提供了核心思路，其研究方向为自动问答系统。曾在2017年广东省大学生计算机作品赛中，所在团队项目《基于kinect的康复训练及评估系统》获得广东赛区二等奖，还参与过多个大学生创新创业项目，有较多的项目开发经验。

组员徐鹏研究方向为AutoML在云端的应用，曾在在第十届英特尔软件创新创业大赛中，所在团队项目《Caremore儿童云安全预警系统》获得一等奖，并在英特尔DCG部门担任Cloud DeepLearning AI Intern，研究AutoML领域CNN量化模型的学习与搜索。

组员施鉴洋研究方向为多媒体数据挖掘及视链学习。曾在2017年全国大学生数学建模竞赛中获得国家二等奖，在其他计算机专业比赛中也均有斩获，具有较多的工程开发经验。

组员王勋广研究方向为图像处理和生成对抗网络。曾在2017年第四届SDN大赛获三等奖，同年获得第29届FIRA SimuroSot组世界冠军。

摘要

在数据爆炸增长的大数据时代，如何利用蓬勃发展的新硬件对查询分析系统的执行进行加速，逐渐引起工业界和学术界的重视。本赛题基于一个模拟商业决策查询的数据集（包含customer表，orders表，lineitem表），对特定的SQL语句的执行阶段进行优化，而查询效率作为赛题的核心评价指标，是我们所着重关注的焦点。针对问题的特性，本文提出了一中基于GPU的高并发查询系统的实现方法。系统通过并行任务分配，数据压缩，排序和构建索引等操作，实现了对大规模数据情况下的查询操作优化。实验表明，系统在赛题所给数据集上具有较好的性能。

关键词

GPU，数据压缩，并行，排序，索引

1 核心架构

面对庞大的数据量，对整个数据集进行简单直接的处理往往无法获得理想的效果。在这样的情况下，减小数据量过大造成的一系列影响便成了关键所在。

由此，团队设计了一套基于分治和并行思想的系统。系统的核心架构主要由指令模块，压缩模块，搜索模块，排序模块，数据模块和中间件组成，分别实现文件分块，索引建立和数据查询等核心功能。

系统的核心架构如图1所示。

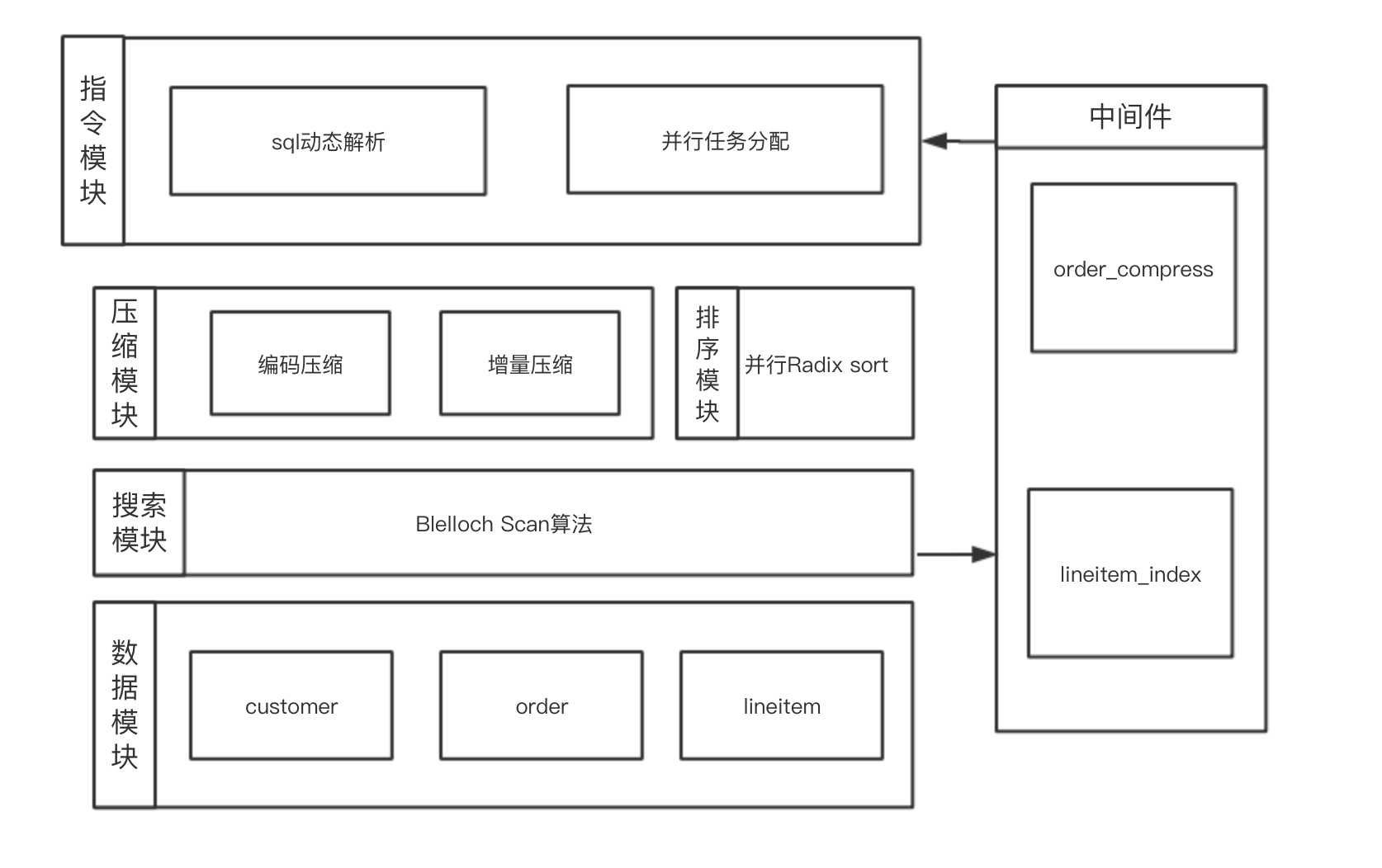
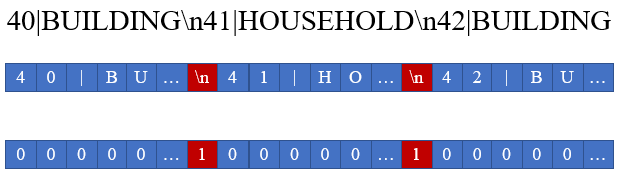
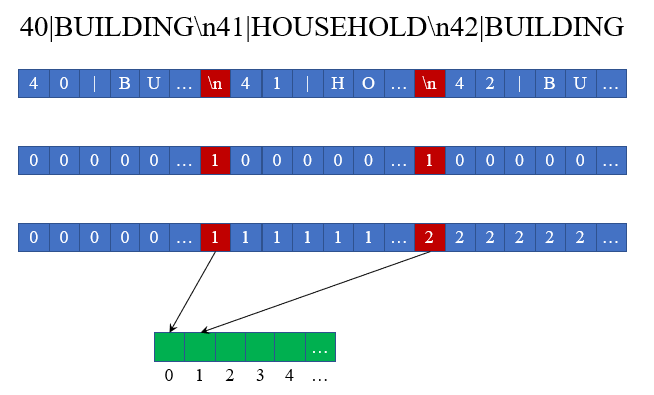


图1：系统核心架构

作为实现数据快速查询的基础，系统需要对数据进行压缩，排序和建立索引操作。由于本赛题数据量巨大，对数据分块便成为了首先需要完成的重要任务。在处理之前，系统对文件进行分块，保证各块文件进入显存单独处理。之后，利用分块数据建立索引，并进行数据查询。

2 系统实现

基于系统的核心架构设计，团队针对每个功能模块进行了具体设计。

2.1 文件分块

对于文件的快速分块，我们的系统采用类似插值查找的方式对order表和lineitem表进行快速划分，计算出把数据表等分为 份所分别对应的数据量。

具体步骤为，以文件平均长度为步长，分别计算出不同块所对应的分割位置，利用公式（1）计算得到目标数据位置，通过快速定位到文件的分割位置：

(1)

其中，表示目标的起始位置，表示当前的终止位置，表示每行记录在内存中的长度。

2.2 建立索引

对不同表之间建立索引的策略，是决定最终查询性能的关键所在。

对于customer表，整个表将被直接读入0号GPU的显存进行处理。

处理过程，一个与customer表字符串对应的辅助数组将被建立，其中与customer表对应为换行符的位置将会置为1，其他位置为0。效果如图2所示。

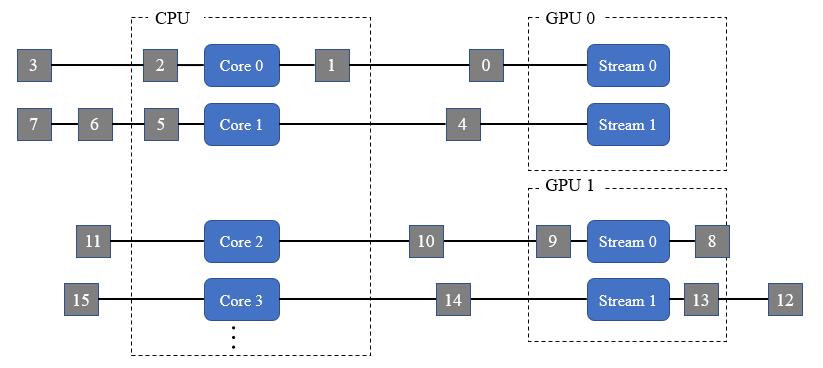
图2：辅助数组结构

采用Blelloch Scan算法[1]求出辅助数组前n项和，计算出每个字符前面换行符的数量。然后，系统根据换行符建立所需的索引。通过customer表中换行符的地址，计算得到的每行数据的起始地址，存储为customer\_address表。具体过程，如图3所示。

图3：customer表索引

这样，每个线程可以根据customer\_address表找到它要处理的起始位置，从而高效地利用GPU的并行能力处理各个数据。

同时，在处理customer\_address表的过程中，我们采用了编码压缩的方式进行了数据压缩。将customer表中的c\_mkgsegment字段压缩成数值，再将对应的数值根据c\_custkey保存在hash数组中，然后将整个hash数组存储为cid\_department表。

对于order表，为了充分利用机器的两个GPU，减少其重叠功能的等待时间，我们的系统采用了流的处理方式[2]。

每个GPU被分配了2个流，从而可以由4个流各自处理order表的4个部分。为了加速CPU对GPU的响应，系统为CPU分配了4个线程，让每个CPU线程分别负责GPU的一个流。具体流程如图4所示。

图4：多线程处理模式

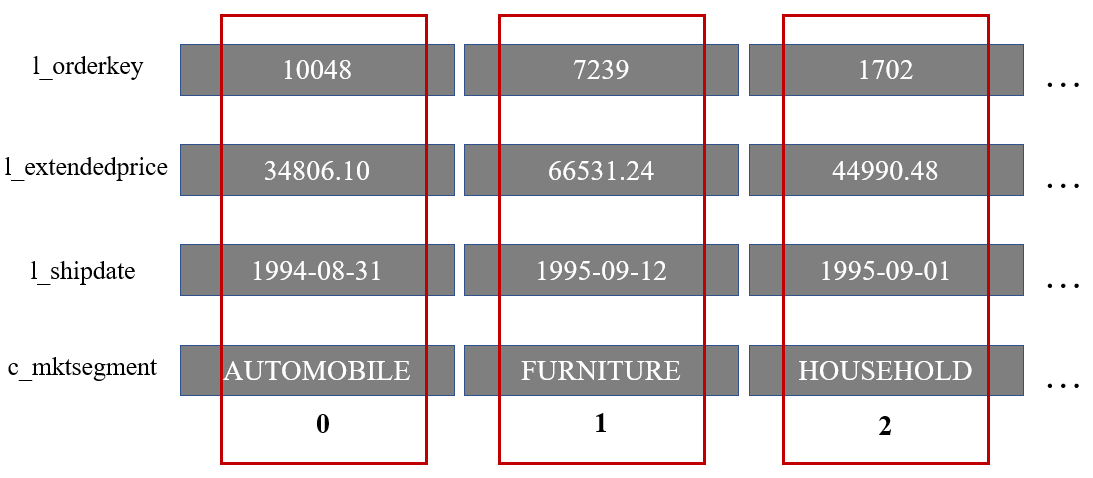
划分出的每一个部分采用与处理customer表类似的Blelloch Scan算法[1]求出order表中每行数据每个属性的起始地址，每个线程单独处理每行数据的每个属性。

之后，系统按照以下步骤生成compress\_order表：

1. 将order表中的o\_custkey字段根据cid\_department表替换成department\_type；
2. 将order表中的c\_orderdate字段通过增量压缩的方式形成0-3000范围内的数值；
3. 将department\_type与日期合并，根据order表中o\_orderkey的值来存放order的地址；
4. 在完成以上的操作后，即可把获得的compress\_order表存入磁盘；

对于lineitem表，系统为GPU分配4个流，分16轮处理让每个流处理lineitem表的16个部分。采用Blelloch Scan算法求出lineitem表中每行数据每个属性的起始地址，让每个线程单独处理每行数据的每个属性。

之后，系统对lineitem表和customer表同时进行操作，在建立索引的过程中，lineitem表和customer表数据被转换成了如图5所示新的数据结构。

图4：多线程处理模式

具体步骤分为如下几步：

1. 将lineitem表中l\_orderkey字段根据compress\_order表替换成department\_type;
2. 将lineitem表中的l\_shipdate字段，根据增量压缩的方式形成0-3000范围内的shipdays；
3. 考虑对内存的高效利用和查询条件的快速判别，将department\_type和shipday合并为一个字段，其中，department\_type在高位，shipdays在低位，记为type\_shipdays；
4. 使用并行Radix sort算法[3]，把位置指针按type\_shipdays排序，将其中的l\_orderkey和l\_extendedprice按位置指针取出来，可获得按部门划分并在部门内按日期排序的数据lineitem\_index表(其中由于lineitem表是分成64部分，lineitem\_index表也是分成64部分的)；
5. 在完成以上的操作后，即可把获得的lineitem\_index表存入磁盘；

总的来说，我们的方案将同一条记录的不同属性通过分离为不同数组，再在各数组之间使用数组下标对齐每一条记录的各个属性建立联系。在保证了在条件查找中各个属性之间访问的独立的同时，也可以充分利用GPU特性从而快速获得所需的正确查询结果。

2.3 数据查询

系统先把compress\_order表分成两部分加载进两个GPU的显存，对于lineitem\_index的64部分，再筛选出每部分部门和l\_shipdate符合的部分，并传输入显存。

按compress\_order表对剩下的数据作进一步筛选，把筛选出的数据先按l\_orderkey排序，接着使用按l\_orderkey归约的方式并行地将符合条件的数据结合起来。将结果数据按价格排序，取出top K个值。每个流再对自身topK个值进行排序。取出每个流各自对应的topK值，最后再对数据进行整合，得到最终结果。

3 系统性能

系统在所提供的数据集（customer表1G,order表21G,lineitem表86G）上经过实验得到结果如下表1所示

表格 1 性能测试结果

|  |  |
| --- | --- |
| 操作 | 时间（s） |
| 处理customer表 | 0.7 |
| 处理 order表 | 16 |
| 处理lineitem表 | 66 |
| 写索引时间 | 40 |
| 单个命令查询一次时间 | 3.3 |
| 单个命令查询3次时间 | 6 |

实验结果表明，系统在一定的时间完成数据处理以后，在所给数据集上表现出了较好的查询性能。

4 系统展望

随着数据量的不断增加，系统的查询性能会对计算设备所具有的内存大小产生更高的依赖，此时便无法保证可以将所有的数据都放入到内存里。

考虑到这个问题，在我们所设计的数据结构中，尽量的保证了在不丢失数据之间的关系的同时，又可以独立地处理各部分乃至各属性的数据。这样的特点，为在内存不足时实现冷热数据分离，建立动态热数据表提供了更好的优化空间，实现更细粒度的冷数据与新数据在内存中的交换，为计算资源有限的设备获得良好的查询性能提供了可能。

致谢

感谢赛事主办方为我们提供了这样一个珍贵的机会，使得我们可以和其他优秀的选手们同台竞技，相互学习，一起成长。

感谢哈尔滨工业大学对我们的培养，也让我们能够有实力有勇气站上这个舞台。感谢张晓峰老师和ices实验室在竞赛过程中给我们提供的指导和帮助。

衷心感谢各位队友的不懈努力，在面对各门考试和课程作业的压力下，仍然坚持到了最后，完成了整个比赛。即使作为数据库比赛方面初学者，我们在很多地方稍显生疏。但无畏结果，奋斗拼搏的经历会成为我们终生难忘的记忆。

最后感谢新硬件的发展，感谢计算机体系结构的黄金十年，让我们可以用新的硬件技术来解决重要的问题。我辈青年，当志存高远，在比赛中磨练打扎实基础，以求在将来为我国计算机领域做出自己的贡献。星星之火，亦可燎原。

参考

[1] Ananth Grama; Vipin Kumar; Anshul Gupta (2003). Introduction to Parallel Computing. Addison-Wesley. pp. 85, 86. ISBN 978-0-201-64865-2.

[2] John Wiley, Sons, Inc. Professional CUDA C Programming. 10475 Crosspoint Boulevard Indianapolis, IN 46256. www.wiley.com

[3] V. J. Duvanenko, "Parallel In-Place Radix Sort Simplified", Dr. Dobb's Journal, January 2011

会议名称：ACM伍德斯托克会议

会议简称：WOODSTOCK'18

会议地点：美国德克萨斯州埃尔帕索

ISBN：978-1-4503-0000-0 / 18/06

年份：2018

时间：6月

版权年份：2018年

版权声明：维护权利

DOI：10.1145 / 1234567890

RRH：F。Surname等。

价格：$ 15.00