Journal of Computer Applications
计算机应用,2020,40(2):420 - 425
ISSN 1001-9081
CODEN JYIIDU
2020-02-10
http://www.joca.cn

文章编号:1001-9081(2020)02-0420-06
DOI:10.11772/j.issn.1001-9081.2019101762

并行查询下查询执行计划的选择

裴泽锋,牛保宁,张锦文,Amjad Muhammad
(太原理工大学 信息与计算机学院,太原 030024)
(*通信作者电子邮箱 niubaoning@tyut.edu.cn)

摘　要:查询是数据库系统的主要负载,其效率决定了数据库性能的好坏。一个查询存在多种执行计划,当前,查询优化器只能按照数据库系统的配置参数,静态地为查询选择一个较优的执行计划。并行查询间存在复杂多变的资源争用,很难通过配置参数准确反映,而且同一执行计划在不同情景下的效率并不一致。并行查询下执行计划的选择需考虑查询间的相互影响——查询交互。基于此,提出了一种在并行查询下度量查询受查询交互影响大小的标准QIs。针对并行查询下查询执行计划的选择,还提出了一种动态地为查询选择执行计划的方法TRating,该方法通过比较查询组合中按不同执行计划执行的查询受查询交互影响的大小,选择受查询交互影响较小的执行计划作为该查询的较优执行计划。实验结果表明,TRating方法为查询选择较优执行计划的准确率达61%,相比查询优化器提高了25%;而且在为查询选择次优执行计划时,其准确率也高达69%。

关键词:并行查询;查询交互;查询优化器;查询执行计划;较优执行计划
中图分类号:TP392　　**文献标志码:**A

Query execution plan selection under concurrent query

PEI Zefeng, NIU Baoning, ZHANG Jinwen, AMJAD Muhammad
(*College of Information and Computer*, *Taiyuan University of Technology*, *Taiyuan Shanxi* 030024, *China*)

Abstract: Query is the main workload of a database system, and its efficiency determines the performance of the database system. There are multiple execution plans for a query, and the existing query optimizers can only statically select a better execution plan for a query according to the configuration parameters of the database system. There are complex and variable resource contentions between concurrent queries, and such contentions are difficult to be reflected accurately through configuration parameters; besides, the efficiency of the same execution plan is not consistent in different scenarios. The selection of the execution plans for concurrent queries needs to consider the influence between queries — query interaction. Based on the above, a metric for measuring the influence of query interaction on the query under concurrent query called QIs (Query Interactions) was proposed. For the selection of query execution plan under concurrent query, a method called TRating (Time Rating) was proposed to dynamically select the execution plan for the query. In the method, the influence of query interaction on the queries executed with different plans in the query combination was measured, and the plan with small influence of query interaction was selected as the better execution plan for the query. Experimental results show that TRating can select a better execution plan for the query with an accuracy of 61%, which is 25% higher than that of the query optimizer; and the accuracy of the proposed method is as high as 69% when selecting suboptimal execution plan for the query.

Key words: concurrent query; query interaction; query optimizer; query execution plan; better execution plan

0　引言

查询是数据库系统的主要工作负载,查询效率的高低决定数据库运行性能的好坏。在数据库系统中,一个查询在真正执行前,查询优化器会生成多种查询执行计划(即查询在数据库中的执行方案)。尽管这些计划最终生成完全相同的结果集,但它们的执行效率却存在很大差异[1]。寻找较优的查询执行计划可以以较小的代价、在很大程度上缩短查询的响应时间,提高查询效率,进而提高数据库系统的性能。

目前,主流的数据库管理系统大都是通过查询优化器为查询选择较优执行计划,即查询的多种执行计划中,使查询响应时间较少的执行计划[1-3]。虽然不同数据库查询优化器实现有所差别,但它们都是依据执行代价——成本预算[4],为查询选择执行代价较小的执行计划作为该查询的较优执行计划。PostgreSQL作为目前成功的开源数据库之一,它的优化算法效率高于其他数据库[1],本文以PostgreSQL为对象,研究并行查询下查询执行计划的选择。查询执行前,客户端发出查询请求,解析器接收此请求并生成查询树,查询优化器中的

收稿日期:2019-09-18;**修回日期:**2019-10-03;**录用日期:**2019-10-24。　**基金项目:**国家自然科学基金资助项目(61572345)。

作者简介:裴泽锋(1995—),男,山西兴县人,硕士研究生,CCF会员,主要研究方向:数据库管理及优化;　牛保宁(1964—),男,山西榆社人,教授,博士,CCF会员,主要研究方向:大数据管理与分析、数据库系统的自主运算与性能管理;　张锦文(1988—),男,山西阳泉人,博士,CCF会员,主要研究方向:数据库系统的性能管理;　Amjad Muhammad(1978—),男,巴基斯坦人,博士,CCF会员,主要研究方向:数据库查询性能预测。

规划器接收查询树并生成可能的查询执行计划，优化器依据动态规划或基因算法等计算每个执行计划的执行代价，从中选出执行代价较小的一个执行计划作为较优执行计划，并据此制定一个完整的规划树传递给执行器去执行[1,5]。

在数据库管理系统中，查询优化器在为查询生成不同的执行路径时，会综合考虑数据表的扫描方式、表间连接方式及连接顺序，在选择较优执行路径时，会依据查询的复杂程度选用不同的算法。因此，当数据库中只有一个查询运行时，如果查询不太复杂，查询优化器可以做到避免选择最差执行计划，且在为此类查询选择较优查询计划上可保持一定的准确率。

随着数据量的急剧增长、用户需求的不断变化，查询语句越发复杂。由于查询优化器优化算法本身的局限性，它只能依据数据库配置参数静态地为查询选择一个较优的执行计划，如果查询语句过于复杂，在统计信息不准确的情况下，成本估算的误差会成倍放大[6]，因此会造成所选择的较优执行计划存在较大偏差。

特别地，在实际的数据库系统中，单一查询运行的情况比较罕见，大部分情况是不同查询在一起并行，因此，一个查询的运行会受到其他查询的影响，我们将其称为查询交互[7]。由于查询交互现象的存在，相比其单独运行，查询的响应时间会有显著变化，大部分查询的响应时间会变长。而且，查询并行数（*MPL*）越大，这种变化越发明显。

但是，目前的查询优化器并没有特别地考虑查询交互这一关键因素的存在，仅通过数据库参数配置很难准确反映查询交互。如果数据库配置参数固定，查询一旦确定，查询的较优执行计划随之确定。我们认为，在不同的查询组合（两个及两个以上查询并行运行时的查询集合）下，查询的较优执行计划可能也不同。假设当查询 Q 单独运行时，查询优化器为该查询选择的较优查询计划为 A。测试实验证明，多数情况下，当查询 Q 与其他查询并行执行时，较优的执行计划不是 A，如果按执行计划 A 执行查询 Q，会大幅度延长其执行时间，并严重影响其他查询的执行。因此，当多个不同查询并行时，目前的查询优化器并不能为查询选择当前查询组合下的较优执行计划。

为此，本文提出一种度量查询受查询交互影响程度大小的标准 QIs（Query Interactions），为选择查询执行计划时定量考虑查询交互因素打下基础。针对并行查询下较优执行计划的选择，本文提出一种为查询动态选择当前查询组合下较优执行计划的方法 TRating（Time Rating），该方法通过比较按照不同执行计划执行的查询在查询组合中受查询交互影响程度的大小，选择受查询交互影响较小的查询所对应的执行计划作为该查询的较优执行计划。

1　相关工作

目前，数据库管理系统通过查询优化器为查询选择较优执行计划。对于一个特定的查询，在查询执行前，查询优化器中的规划器负责为查询生成所有可能的执行计划，优化器依据成本预算从中选择执行代价最小的计划。针对复杂度不同的查询，查询优化器采用不同的算法进行处理。对于比较简单的查询，优化器采用动态规划算法，该算法搜索范围小且效率高；对于比较复杂的算法，则采用基因算法。基因算法是一种自适应的算法，可以在较短的时间内给出一个较优的解[1]。因此，对于数据库中单一运行的查询，查询优化器在为不太复杂的查询选择一个不坏的执行计划时可保持一定的准确率。但是，如果查询语句过于复杂，由于查询优化器成本估算算法的局限性，在统计信息估计不准确时，成本估算的误差会成倍放大，无法为查询选择较优的执行计划[7]。

特别地，在实际的数据库系统中，单一查询运行的情况比较少见，多数情况下是不同查询在一起并行，并行查询间存在查询交互，这种交互会导致某一查询的运行受到其他查询的影响[8]。由于查询优化器通过数据库配置参数静态地为查询选择查询执行计划，这种选择较优执行计划的方式尽管一定程度上反映了查询交互[9]，但是其并没有特别地考虑查询交互这一关键因素的存在，无法准确地反映这种查询交互现象。在并行查询情景下，一个不坏的执行计划在另一查询组合中可能会变得不好，甚至更差[9]。在这种情况下，如果仍用原有执行计划去执行该查询，会严重影响该查询和其他查询的性能。因此，通过查询优化器已无法为查询准确地选择当前查询组合下较优的执行计划[10-11]。

目前，有关考虑查询交互的并行查询下较优执行计划的选择还没有专门的研究，大部分研究都是通过度量查询交互进而预测并行查询下查询的开销及响应时间[12-13]，进而为查询调度提供一定的依据。目前对响应时间的预测方法主要集中于建立分析模型和统计模型。分析模型指把查询计划分段，估计每一段的工作量、系统的执行速率，然后求得该段的执行时间，最后把所有分段的执行时间加起来即可。统计模型指事先选取并运行一定量的样本，依据样本运行结果，给定输入和输出，根据训练集利用统计的方法训练模型，然后通过测试集来评判模型的性能，最后依据模型给出预测结果。利用统计模型预测并行查询的响应时间性能要优于分析模型。但是无论是分析模型还是统计模型，预测响应时间只能用于合理地调度一批查询的执行顺序。如果数据库中到达一批查询，按照先到先服务原则，先到查询需要首先被处理。在这种情况下，如果可以为该查询选择当前组合下较优的执行计划，这对提高查询效率，进而提高数据库运行性能尤为重要。

在多查询并行时，查询交互这一关键因素会影响查询的响应时间，进而影响查询效率。Duggan 等[14]利用 BAL（Buffer Access Latency）即查询平均页读取时间来度量查询交互。查询 Q 单独运行时的 BAL 值等于该查询读取页花费的总时间除以此查询读取页的总数。多查询并行时，一个查询的 BAL 值越大，则表示其受其他查询的影响程度越大，但这种度量只适用于中等大小查询。之后，张锦文等[15-16]提出 QueryRating 来度量两个查询并行时查询交互的大小，由于其直接采用查询响应时间的比值大小来衡量查询交互的大小，宏观直接且使用范围广；但 QueryRating 只适用于度量两个查询并行时，其中一个查询受另一个查询影响程度的大小。

基于以上查询交互度量使用范围的限制、查询优化器为查询选择当前查询组合下较优执行计划存在较大偏差的问题，本文提出了一种度量查询在多查询（$MPL > 2$）并行下受查询交互影响程度大小的标准 QIs，并基于查询交互提出了一种动态地为查询选择当前查询组合下较优执行计划的方法 TRating。当用户提交的查询到达时，依据先到先服务原则，为先到查询选择当前查询组合下的较优执行计划提供了依

据，具有一定的实际意义。

2 并行查询下较优执行计划的选择

2.1 QIs度量标准

Zhang等[15]提出的QueryRating的表达式如式(1)，它直接计算响应时间的比值，建模复杂度低且准确率高。

$$\gamma_{p/q} = t_{p/q} / t_p \tag{1}$$

其中：$\gamma_{p/q}$表示查询p和查询q并行时由于查询交互p受到q影响的大小，$t_{p/q}$表示查询p和查询q并行时查询p的响应时间，t_p表示查询p单独运行时的响应时间。

当$\gamma_{p/q} < 1$，则说明查询p在与q并行执行时的响应时间比其单独运行时的响应时间短，两者间存在合作关系，$\gamma_{p/q}$值越小，查询q对查询p响应时间的促进程度越大，两个查询间的交互程度就越小；当$\gamma_{p/q} = 1$，则说明查询q的加入对查询p的运行没有影响，查询p的响应时间没有发生变化；当$\gamma_{p/q} > 1$，则说明查询p在与q并行执行时的响应时间比其单独运行时的响应时间长，两者间存在资源竞争的关系，$\gamma_{p/q}$值越大，查询q对查询p响应时间的阻碍程度越大，两个查询间的交互程度就越大。

QueryRating可以度量两两查询并行时由于查询交互某一查询受到另一查询影响程度的大小。那么，如果多查询并行($MPL > 2$)时，由于查询交互，某一查询会受到其余任何一个查询的影响，如何度量多查询并行时特定查询所受查询交互影响程度的大小是为该查询选择当前查询组合下较优执行计划的基础。受Duggan等[14]提出的B2L&BAL模型预测查询性能的启发，本文通过考虑其他剩余每个查询对要研究查询的查询交互大小，提出了一种度量多查询并行时查询所受查询交互影响程度大小的标准QIs，表达式如下：

$$QIs_{q_i} = Dur_{q_i} * \sum_{j=1}^{n, i \neq j} \gamma_{q_i/q_j} \tag{2}$$

其中：

$$Dur_{q_i} = t_{q_i} \Big/ \frac{1}{n} \sum_{j=1}^{n} t_{q_j} \tag{3}$$

式(2)中：QIs_{q_i}表示多查询并行时，查询q_i受其他剩余查询查询交互影响的大小；t_{q_i}表示查询q_i单独运行时的响应时间；Dur_{q_i}(Duration)可以看作查询q_i所受其他查询查询交互影响的持续时长。

本文对式(2)、(3)作如下解释：多查询并行时，由于查询交互，对于某个查询，其余查询都会阻碍或促进该查询的执行。在考虑该查询受查询交互影响大小时，要分别考虑其余每个查询对该查询的影响即γ_{q_i/q_j}，因此，本文将不同的γ_{q_i/q_j}进行求和表示其余查询对该查询的影响大小。特别地，本文认为，该查询单独运行时间的长短会影响其受查询交互影响程度的持续时长，该查询单独运行时响应时间越长，则当其加入查询组合后，它受查询交互影响的持续时长也越长。因此，本文通过Dur_{q_i}表示查询q_i所受其他查询查询交互影响的持续时长。Dur_{q_i}越大，表示其响应时间所占比例越大，则当该查询加入查询组合时，它所受其他查询查询交互影响的持续时长也越大。

由式(2)、(3)可以看出，QIs_{q_i}值越大，表示多查询并行时，由于查询交互，查询q_i受其他查询的影响程度越大。

2.2 TRating方法的具体实现

在实际的数据库系统运行中，对于用户提交的查询，依据先到先服务原则，按照用户提交查询的先后顺序执行查询。那么，当某个查询到来时，为查询选择当前查询组合下较优的执行计划对提高查询效率十分必要。当按照特定执行计划执行的某查询在加入查询组合时，相比其他执行计划，按特定执行计划执行的某查询单独运行时的响应时间较短，且它所受查询交互影响程度越小，则其在查询组合中的响应时间越短。也就是说，按特定执行计划执行的某查询单独运行时的响应时间的长短和该查询所受查询交互影响程度的大小均会影响该查询在查询组合中最终的响应时间。响应时间越短的查询对应的执行计划即为该查询在当前查询组合中较优的执行计划。

为能准确地描述该方法，本文定义该方法中用到的符号见表1。

表1 相关符号定义

Tab. 1 Definition of related symbols

符号	释义
q_i	查询q_i，编号i
q_{i_j}	查询q_i的执行计划q_{i_j}，编号j
$D = \left\{ q_{i_j} / q_{n_m} \mid i \neq n \right\}$	按特定执行计划q_{i_j}执行的查询q_i和按特定执行计划q_{n_m}执行的查询q_n并行查询组合
$f_{q_{i_j}/q_{n_m}}$	q_{i_j/n_m}查询组合的编号f
$A = \left\{ q_i \mid i = 1, 2, \cdots, n \right\}$	数据库中的所有查询的集合，共n个查询
$B = \left\{ q_{i_j} \mid j = 1, 2, \cdots, m \right\}$	查询q_i的执行计划的集合，共m个
$C = \left\{ q_{i_j} / q_{n_m} \middle\vert q_i, q_j \in A, q_{i_j}, q_{n_m} \in B, i \neq n \right\}$	查询组合的集合
$\gamma_{q_{i_j}/q_{n_m}}$	按特定执行计划q_{i_j}执行的查询q_i和按特定执行计划q_{n_m}执行的查询q_n并行时查询q_i的QueryRating值
$k_{\gamma_{q_{i_j}/q_{n_m}}}$	$\gamma_{q_{i_j}/q_{n_m}}$的编号k

相比其他执行计划，如果按特定执行计划执行的某查询单独运行时的响应时间较长，即使该查询在查询组合中受其余查询的查询交互影响较小，它最终的响应时间也可能较长；同样地，如果按另一执行计划执行的该查询单独运行时的响应时间较短，即使该查询在查询组合中受其余查询的查询交互影响较大，它最终的响应时间也可能较短。考虑到这种现

象的存在，将 $Fac_{q_{w_k}}$ 值作为每个查询计划的基准值。建立 $Fac_{q_{w_k}}$ 的表达式如下：

$$Fac_{q_{w_k}} = t_{q_{w_k}} \Big/ \frac{1}{m}\sum_{k=1}^{m} t_{q_{w_k}} \tag{4}$$

其中，$Fac_{q_{w_k}}$ 表示按 q_{w_k} 执行计划执行的查询单独运行时的响应时间所占该查询所有执行计划单独运行时响应时间之和的比值，以此作为每个查询执行计划的基准值。

当 $Fac_{q_{w_k}}$ 值越小，如果按此执行计划执行的查询在加入查询组合时，其所受的查询交互影响也较小，则此执行计划即为该查询的较优执行计划。考虑到按特定执行计划执行的某查询单独运行时的响应时间的长短、该查询所受查询交互影响程度的大小均会影响该查询在查询组合中最终的响应时间，因此，本文提出 $TRating_{q_{w_k}}$（Time Rating）表示两者的乘积。$TRating_{q_{w_k}}$ 的表达式如下：

$$TRating_{q_{w_k}} = Fac_{q_{w_k}} * QIs_{q_{w_k}} \tag{5}$$

该方法具体流程如下：

1）构建两张索引表 IT_1、IT_2 和两张数据表 T_1、T_2，并将索引表 IT_1 和数据表 T_1 关联，索引表 IT_2 和数据表 T_2 关联。

索引表 IT_1 存储 A 中查询的查询编号 i、每个查询对应的所有执行计划编号 j，并将其分别设置为一级索引和二级索引。表结构见表2。

表2　索引表 IT_1

Tab. 2　Index table IT_1

查询编号	执行计划编号
i	j

索引表 IT_2 存储 C 中查询组合编号 f、每个查询组合中查询 q_i 和查询 q_n 所对应的 QueryRating 值 $\gamma_{q_{i_j}/q_{n_m}}$ 和 $\gamma_{q_{n_m}/q_{i_j}}$ 的编号 $k_{\gamma_{q_{i_j}/q_{n_m}}}$ 和 $k_{\gamma_{q_{n_m}/q_{i_j}}}$，并将查询组合编号设置为一级索引。表结构见表3。

表3　索引表 IT_2

Tab. 3　Index table IT_2

查询组合编号	$\gamma_{q_{i_j}/q_{n_m}}$ 编号	$\gamma_{q_{n_m}/q_{i_j}}$ 编号
f	k_1	k_2

数据表 T_1、T_2 分别对应索引表 IT_1、IT_2 中数据实际值，另数据表 T_1 需要存储每个 q_{i_j} 单独运行时的响应时间。

由索引表 IT_1、数据表 T_1 获取按特定执行计划执行的查询单独运行时的响应时间，由索引表 IT_2、数据表 T_2 获取该查询所受其他查询的 QueryRating。

2）给定 $MPL = w$，当某查询 q_w 到达数据库时，该查询与目前正在执行的 $(w-1)$ 个查询构成一个查询组合。

由索引 IT_1 表、数据表 T_1 得到该查询的所有执行计划 q_{w_k} 及其对应单独运行时的响应时间 $t_{q_{w_j}}$，计算出 $Fac_{q_{w_k}}$ 值。

3）比较所有执行计划对应的 $TRating_{q_{w_k}}$，较小的 $TRating_{q_{w_k}}$ 值所对应的执行计划即为该查询在当前查询组合下较优的查询执行计划。

3　实验结果与分析

3.1　实验环境与设计

本实验选用的测试基准为 TPC-H[17]，查询模板为 2，3，4，8，10，12，14，22。这些模板涵盖了分组、排序、聚集、连接、子查询等多表查询操作，其默认执行计划包含了各种基本表的扫描方式、表间连接方式及连接顺序。本实验分别给每个查询指定三个不同的执行计划，把按不同执行计划执行的同一查询看作不同查询。实验环境配置如表4。

表4　实验环境配置

Tab. 4　Experimental environment configuration

实验配置	参数说明
硬件环境	Server1： CPU：Inter Xeon CPU E3-1225 V2 @3. 2 GHz RAM：8 GB Server2： CPU：Intel Xeon CPU E5-2609 v4 @1. 70 GHz Mem：16 GB
操作系统	Windows 2008 Server CentOS Linux release 7. 4. 1708（Core）
数据库	PostgreSQL（缓冲池配置：2 000 MB）
编程语言	Java
测试基准	TPC-H（10 GB）

为查询选择当前查询组合下较优的执行计划，必须首先获取查询的多个执行计划。本文首先获取查询的多个执行计划，紧接着通过实验验证同一查询在不同查询组合下所选择的较优执行计划不同这一现象的大量存在。最后，给定 MPL，依据先到先服务原则，根据实际用户提交查询的先后，对比当前查询优化器，通过实验验证本文所提出的为查询选择当前查询组合下较优执行计划方法的准确性。

3.2　执行计划生成及其响应时间

由于执行计划的生成会综合考虑基本表的访问顺序、表间的连接方式和连接顺序，因此本文在生成查询执行计划时，对于单表查询着重考虑基本表的访问顺序，对于多表查询，着重考虑表间连接方式和连接顺序，以此生成代表性执行计划。

使用 explain 命令，PostgreSQL 查询优化器直接生成查询默认的执行计划。依据上述原则，通过关闭或打开执行计划开关，使得优化器允许或禁止使用一些扫描或连接方法，进而生成一定数量的执行计划，且其执行的查询响应时间少于或多于默认执行计划响应时间，且相差不宜过大。最后通过 pg_hint_plan 插件绑定查询计划，使查询按指定的执行计划去执行。下面通过图1、图2说明。

```
->Merge Semi Join (cost=0.86..186099017.03 rows=1 width=4)
Merge Cond:(partsupp.ps_partkey = part.p_partkey)
```

图1　默认执行计划

Fig. 1　Default execution plan

```
->Hash Semi Join (cost=66214.96..186067267.63 rows=1 width=4)
Hash Cond:(partsupp.ps_partkey = part.p_partkey)
```

图2　关闭 MergeJoin 后的执行计划

Fig. 2　Excution plan after closing MergeJoin

实验中，服务器只开启 PostgreSQL 数据库进程，分别单独运行每个查询、并行执行每两个查询组成的查询组合，利用 PostgreSQL 的 pg_stat_statement 模块获取查询的响应时间，通过多次运行求平均值的方法减少其他因素干扰，准确获取响应时间，从而求得响应时间的比值。

3.3 查询组合不同时的较优执行计划验证

本文分别对并行度*MPL*=2,4,6,8时的同一查询加入不同查询组合所选择的较优执行计划进行实验。实验结果表明,同一查询在不同查询组合下所选择的较优查询计划不同,而且这种现象普遍存在,平均约占71%。由于篇幅受限,图3仅对部分样例实验结果进行展示,并加以说明。

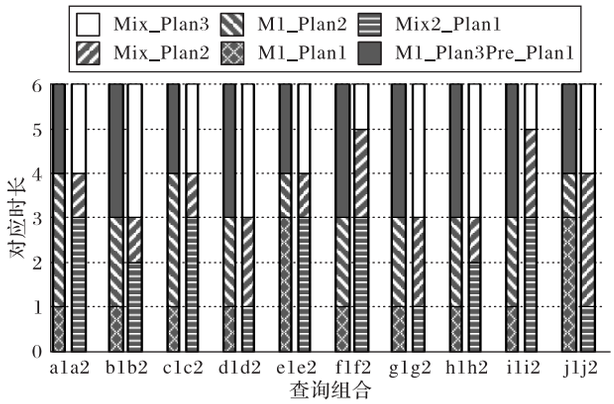

图3 关闭nestloop后*MPL* = 4时,同一查询加入不同的查询组合时的较优执行计划抽样

Fig. 3 Sampling of better execution plan of one query adding to different query combinations with nestloop closed and *MPL* = 4

图3中:每组由两个柱状图构成,它们分别表示同一查询加入不同的查询组合中构成新的查询组合,每个柱状图分成三段,每一段表示按不同的执行计划执行的此查询加入查询组合,自底向上分别表示执行计划1、2、3,执行计划1为PostgreSQL查询优化器给出的执行计划,每段长度分别对应按特定执行计划执行的此查询加入查询组合后的响应时间的长短。

由图3可知,多数情况下,每组的两个柱状图的三段长度均不相同,这说明当同一查询加入不同的查询组合时,所选择的较优执行计划并不相同。比如,当查询加入a1、a2查询组合时,对应与a1构成新的查询组合,该查询的较优执行计划为PostgreSQL查询优化器给出的执行计划即执行计划1;但对应与a2构成的新的查询组合,该查询的较优执行计划为执行计划2,不同于执行计划1。

本文对此种现象作如下解释:对于单一运行的查询,由于查询优化器搜索算法的限制,目前其仅能做到避免选择最差执行计划,当查询比较复杂时,为查询选择较优执行计划会存在较大偏差;对于并行运行的查询,优化器并没有特别地考虑查询交互这一主要因素,因为查询在不同查询组合下选择较优执行计划的偏差更大,因此,此种现象的大量存在十分正常。

3.4 TRating和查询优化器对比

模拟用户提交查询的实际场景,依据先到先服务原则,分别在*MPL*=2,4,6,8时计算TRating方法和查询优化器为查询选择较优执行计划的准确率,结果如表5所示。

由表5可知,在不同并行度下,与目前查询优化器相比,本文所提方法选择较优执行计划准确率平均提高了25个百分点。本文方法之所以准确率提高,是因为它在为查询选择当前查询组合下的较优执行计划时,特别地考虑了多查询并行时查询交互这一关键因素,并合理地度量了特定查询在查询组合中所受的查询交互影响的大小。尽管这样,随着并行度的增加、查询间交互的复杂化,预测准确率会稍有下降,属正常现象。

表5 TRating和查询优化器准确率对比

Tab. 5 Accuracy comparison between TRating and query optimizer

MPL	选择准确率		变化幅度
	优化器	TRating	
2	0.64	0.92	0.28
4	0.36	0.67	0.31
6	0.27	0.48	0.21
8	0.16	0.36	0.20

更可观的是,当前查询优化器是基于成本估算为查询选择较优执行计划,它需要计算每个执行计划所涉及各种操作的成本,计算量大。相对于查询优化器,本文TRating方法在为查询选择较优执行计划时,由于模型选择的合理性,所涉及的计算量较小。因此,实验发现,在相同的计算环境下,本文方法TRating耗时远小于当前查询优化器。除此之外,不难发现,在不同*MPL*值下,通过本文方法TRating选择较优执行计划的准确率均高于当前查询优化器,这是因为本文方法特别地考虑了并行查询下查询交互这一关键因素。因此,随着*MPL*即查询并行数的增加和查询交互的复杂化,虽然本文方法TRating准确率会有所下降,但仍优于当前查询优化器。

实验发现,在少数情况下,当本文方法不能为查询选择较优执行计划时,其所选择执行计划仍然为次优执行计划(即除去较优执行计划,使查询响应时间较少的执行计划),同样避免了最坏执行计划,也具有相当的实际意义。表6是*MPL* = 2,4,6,8时,本文TRating方法为查询选择次优执行计划的准确率。由表6可知,在不同并行度下,本文TRating方法为查询选择次优执行计划(包括较优执行计划)平均预测准确率高达69%,由此可知本文所提方法具有相当高的可靠性。

表6 TRating选择次优执行计划准确率

Tab. 6 TRating accuracy for suboptimal execution plan selection

MPL	TRating选择准确率	*MPL*	TRating选择准确率
2	0.97	6	0.61
4	0.73	8	0.46

4 结语

查询是数据库的主要负载,查询效率直接决定数据库系统的性能。多查询并行时,由于现有查询优化器只能依据参数配置静态地为查询选择一个不坏的执行计划,并没有特别地考虑查询交互这一关键因素,导致其在为查询选择当前查询组合下的较优执行计划时存在较大偏差。基于此,本文合理地度量了特定查询在多查询并行时受查询交互影响程度的大小,在此基础上,创新性地提出了动态地为查询选择当前查询组合下较优执行计划的方法TRating,并通过实验验证了其合理性。当用户提交的查询到达时,依据先到先服务原则,为先到查询选择当前查询组合下的较优执行计划提供了依据,具有一定的实际意义。随着查询并行度的增加、查询交互的复杂化,如何更准确地建立方法,或通过训练深度学习模型,保持其预测准确率不变是后续研究的重点。

参考文献(References)

[1] 孙振兴,向阳,刘增宝. PostgreSQL查询优化器分析研究[J]. 计算机技术与发展, 2011, 21(8): 141-144. (SUN Z X, XIANG Y, LIU Z B, Analysis and research on optimizer of PostgreSQL[J].

Computer Technology and Development, 2011, 21(8): 141-144.)

[2] 谷伟,陈莲君. 基于MySql的查询优化技术研究[J]. 微型电脑应用, 2013, 30(7): 48-50. (GU W, CHEN L J. Research on the query optimize technology of MySql[J]. Microcomputer Applications, 2013, 30(7): 48-50.)

[3] 刘维学. SQL Server 查询优化器原理与优化实例分析[J]. 计算机技术与发展, 2013, 23(11): 108-111. (LIU W X. Query optimization principle and optimized instance analysis of SQL Server[J]. Computer Technology and Development, 2013, 23(11): 108-111.)

[4] LI J, NEHME R V, NAUGHTON J. GSLPI: a cost-based query progress indicator[C]// Proceedings of the IEEE 28th International Conference on Data Engineering. Piscataway: IEEE, 2012: 678-689.

[5] 宋晓眉,叶晓俊,曾小青,等. PostgreSQL查询优化中的等价类研究与改进[J]. 计算机工程与应用, 2014, 50(14): 31-38. (SONG X M, YE X J, ZENG X Q, et al. Study and improvement on equivalence classes of PostgreSQL query optimization[J]. Computer Engineering and Applications, 2014, 50(14): 31-38.)

[6] CHAUDHURI S, KAUSHIK R, RAMAMURTHY R. When can we trust progress estimators for SQL queries?[C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2005: 575-586.

[7] MARKL V G, HAAS P J, ABOULNAGA A I, et al. System and method for updating database statistics according to query feedback: US7831592B2[P]. 2010-11-09.

[8] AHMAD M, ABOULNAGA A, BABU S, et al. Modeling and exploiting query interactions in database systems[C]// Proceedings of the 17th ACM Conference on Information and Knowledge Management. New York: ACM, 2008: 183-192.

[9] WEISSMAN C, MOELLENHOFF D, WONG S, et al. Query optimization in a multi-tenant database system: US7529728B2[P]. 2009-05-05.

[10] EWEN S E, KACHE H, MARKL V G, et al. Progressive refinement of a federated query plan during query execution: US7877381B2[P]. 2011-01-25.

[11] SIMON E, LLIRBAT F, FABRET F, et al. Query plan reformulation: US 8688683B2[P]. 2014-04-01.

[12] WU W, WU X, HACIGÜMÜŞ H, et al. Uncertainty aware query execution time prediction[J]. Proceedings of the VLDB Endowment, 2014, 7(14): 1857-1868.

[13] AHMAD M, ABOULNAGA A, BABU S, et al. Interaction-aware scheduling of report-generation workloads[J]. The VLDB Journal, 2011, 20(4): 589-615.

[14] DUGGAN J, CETINTEMEL U, PAPAEMMANOUIL O, et al. Performance prediction for concurrent database workloads[C]// Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. New York: ACM, 2011: 337-348.

[15] ZHANG J, NIU B. A clustering-based sampling method for building query response time models[J]. Computer Systems Science and Engineering, 2017, 32(4): 319-331.

[16] 张锦文,牛保宁,李爱萍. 查询交互响应时间预测模型的采样优化[J]. 小型微型计算机系统, 2015, 36(10): 2240-2244. (ZHANG J W, NIU B N, LI A P. An optimized sampling method for query interaction aware respond time modeling[J]. Journal of Chinese Computer Systems, 2015, 36(10): 2240-2244.)

[17] PostgreSQL Wiki. TP-C[EB/OL]. [2018-09-20]. https://wiki.postgresql.org/wiki/TPC-H.

This work is partially supported by the National Natural Science Foundation of China (61572345).

PEI Zefeng, born in 1995, M. S. candidate. His research interests include database management and optimization.

NIU Baoning, born in 1964, Ph. D., professor. His research interests include big data management and analysis, autonomous computing and performance management of database systems.

ZHANG Jinwen, born in 1988, Ph. D. His research interests include database system performance management.

AMJAD Muhammad, born in 1978, Ph. D. His research interests include database query performance prediction.