文章编号:1671-4598(2022)10-0209-07 DOI:10.16526/j.cnki.11—4762/tp.2022.10.032 中图分类号:TP311 文献标识码:A

一种面向分布式数据库的多并发 OLAP 型查询性能预测方法

刘骥超[1,2]，杜建华[1,2]，谢寒生[1,2]，刘霄燕[1,2]，谢名干[3]
(1. 海南省气象信息中心，海口 570203；2. 海南省南海气象防灾减灾重点实验室，海口 570203；
3. 海南省澄迈县气象局，海南 澄迈 571900)

摘要：数据库查询性能预测即预测查询的执行延迟时间，数据库查询延迟预测技术能够用于查询调度和查询进度显示等；当前的查询性能预测技术都具有一定的局限性，不能准确预测分布式数据库中多并发查询导致 CPU、I/O 资源争用情形下的查询延迟；针对这一问题，文章提出了一种在分布式数据库中预测并发 OLAP 型查询延迟的技术，该技术通过建立查询干扰度和查询敏感度模型来衡量资源（I/O，网络）竞争的激烈程度，并据此来预测不同执行环境下的查询性能；TPC-DS 的实验结果表明，分析型查询预测的误差率在 25%以下，说明该技术能够较准确地预测查询执行时间。

关键词：性能预测；查询延迟；并发查询；分布式数据库；并发 OLAP

A Concurrent OLAP Query Performance Prediction Approach for Distributed Database

LIU Jichao[1,2], DU Jianhua[1,2], XIE Hansheng[1,2], LIU Xiaoyan[1,2], XIE Zhaogan[3]
(1. Meteorological Information Center of Hainan Province, Haikou 570203, China;
2. Key Laboratory of South China Sea Meteorological Disaster Prevention and Mitigation of Hainan Province, Haikou 570203, China; 3. Meteorological Bureau of Chengmai County, Chengmai 571900, China)

Abstract: Database query performance prediction is predicting query latency time. Database query latency prediction technology can be used for query scheduling and query progress display. However, current query performance prediction techniques often have some limitations and be hard to accurately predict the query latency caused by the concurrency queries in the distributed database, because of the CPU and I/O resource contention. Aimed at this problem, a prediction concurrent OLAP query latency technique is proposed in distributed database. The proposed method measures the intensity of competition in the resources of I/O and network by setting up the concurrent query interference (CQI) and query sensitivity (QS) models, And accordingly predicts the query performance under the different execution environments. The experimental results of the TPC-DS show that the mean relative error of analytical query prediction is less than 25%, which shows that the technology can predict the query execution time more accurately.

Keywords: performance prediction; query latency; concurrent query; distributed database; concurrent OLAP

0 引言

数据库中并行执行查询能够带来许多优势。例如，它能够缩短多个查询的整体运行时间并且提高硬件的利用率，但是，对于并发查询中的某一个查询，相较于其单独执行，它的执行时间可能会延长或缩短。主要原因是多个查询之间相互影响[1]，有些查询能够促进该查询的执行，有些则由于与该查询存在资源竞争而延长查询执行。

并发查询性能预测对于查询调度控制[2-6]等具有很大的应用价值，例如，如果事先能够知道查询执行时间，那么就可以改变多个查询的顺序，继而达到用户 SLA 要求。准确的查询性能预测技术还能够用于查询进度显示[7]，以便了解当前查询的执行进度，然后 DBA 就可以做下一步的决策，等待该查询执行完毕或杀死该查询。查询性能预测还对查询优化器具有一定的指导作用，比如：查询优化器可以更好地创建并发查询感知的查询计划，以此来缩短查询的整体执行时间。

由于查询性能预测技术具有很大价值，所以，针对该方面有很多研究，这些研究主要面向两类查询，一是 OLTP 型查询[8]，OLTP 主要指在关系型数据库中一些对时间要求比较高的事务型查询，通常该类查询的执行时间较短；二是 OLAP 型查询[9]，OLAP 型查询主要运用于数据仓库，这种类型的查询面对的数据量比较大，执行时间比较长。本文主要面向 OLAP 型查询。当前已经有一些技术能够对分析型查询做性能预测[10]，但这些技术在实用性和

收稿日期:2022-05-16； **修回日期**:2022-07-04。
基金项目:海南省自然科学基金青年基金(420QN373);海南省气象局科研项目(hnqxSJ202104)。
作者简介:刘骥超(1988-),男,河南沁阳人,硕士,工程师,主要从事气象数据管理与应用方向的研究。
通讯作者:杜建华(1981-),男,湖北洪湖人,硕士,高级工程师,主要从事气象数据理论和应用方向的研究。
引用格式:刘骥超,杜建华,谢寒生,等.一种面向分布式数据库的多并发 OLAP 型查询性能预测方法[J].计算机测量与控制,2022,30(10):209-215,221.

扩展性方面存在一定的局限。

本文提出了一种能够量化多个查询之间的资源竞争并根据资源竞争情况预测并发查询的延迟时间的方法。在现实情况中，同一个查询在不同的并发查询组合下，执行时间等性能指标将产生差异，主要原因在于多个查询要竞争使用 I/O 和网络等资源。在 I/O 资源方面，分析型查询通常包含许多表连接操作，每个表的数据都需要从磁盘中读取，这会引发大量的 I/O 操作，而且若几个查询扫描不同的数据表，则会争用 I/O，导致 I/O 时间变长，继而使得查询变慢。对于网络资源，数据存放在分布式数据库集群的各个节点中，这些节点位于不同的物理机器上，在执行查询时，数据需要通过网络在节点间迁移，因此，网络带宽资源的争用也是影响性能的重要元素。模型能够根据不同的查询组合情况，量化资源（I/O、网络）的使用和争用情况，从而预测查询延迟时间。

本文提出了两个模型，分别是：查询干扰度（CQI，concurrent query interference）和查询敏感度（QS，query sensitivity）。查询干扰度是指多个并发查询对主查询（要预测的查询）的干扰程度，干扰度越大说明资源竞争越激烈，主查询的执行延迟也越大。查询敏感度是指主查询在不同的查询资源竞争情况下所表现出来的敏感程度即查询性能，不同的查询组合会引起不同程度的资源竞争。本文利用 CQI 和 QS 模型预测分布式数据库中并发 OLAP 型查询延迟。

1 相关工作

当前已有一些针对查询性能预测的研究，并且有较好的效果。另外，查询进度显示通常也需要预测执行时间。所以，接下来，本文将介绍关于这两面的研究并比较与本文技术的区别。

1.1 查询进度指示器

查询进度指示器（QPI，query progress indicators）用于指示一个查询的进度，这样可以直观地了解查询已经花费了多少时间和还需要多少时间能够执行完毕。当前，针对 QPI 已经有了一些研究，Surajit 等人[11]对查询完成的比例建模，他们没有考虑并发查询的情况，只是针对单个查。G. Luo 等人[12-13]把磁盘 I/O、运行时间、消息字节数等作为度量，运用机器学习预测长查询的进度，但是他们在预测查询的延迟时间时，该查询必须是在运行中，而本文的方法在查询执行前就能够得知查询需要多少时间，且主要考虑并发查询的情况，相对机器学习方法来说，更加简单。

1.2 查询性能预测

文献[14－16]都是运用机器学习的方法来预测查询性能，但都没有处理并发查询的情形。对于多并发查询，文献[9，17－19]首先对性能预测技术进行了研究，他们的共同点都是针对分析型查询，而且考虑查询之间的交互作用并建立回归模型，即能够预测不同并发度下的查询性能。Mumtaz 等人[20-21]扩展了上述的研究，他们考虑了不同组合中查询的相互影响，根据实验数据建立模型，从而得到了较高的准确率。Muhammad 等人[7]对运行中的查询进行预测，而 Barzan 等人[8]主要对 OLAP 型的查询进行预测。Chetan 等人[22]主要研究在数据仓库中，查询延迟如何随着工作负载的改变而改变。Jennie 等人[23]对并发查询的资源竞争建模，使得预测的误差较低。文献[24]预测了在查询在大数据量下的单独执行时间，它把 SQL 查询映射为一系列基础操作符并计算操作符消耗时间之和来计算总执行时间。文献[25]主要针对 OLTP 型并发查询，运行机器学习方法以系统状态的 36 个指标为基础进行性能预测。

上述研究既有对 OLAP 型查询的研究，也有对 OLTP 型查询的研究，在 OLAP 型并发查询的情形下，都是基于单机数据库，并没有考虑分布式数据库的环境，即查询分布在多个机器上的情形，这种查询有网络开销。本文提出的查询干扰度模型考虑了网络资源开销，能更准确地预测分布式数据库并发查询的性能。

2 分析型任务

2.1 分布式数据库

本文使用分布式数据库 Greenplum 存储数据并查询。分布式数据库和单机数据库最大的区别在于分布式数据库是一个集群，集群中有多个节点，节点分为主节点和从节点，主节点用来管理从节点并生成查询计划，从节点存放数据并按照查询计划查询数据。当客户端提交一个 SQL 查询到 Greenplum 后，Greenplum 中的主节点首先解析查询语句并生成一个代价最小的查询计划，然后，发送给各从节点，从节点根据查询计划执行查询，并将得到的结果返回给主节点，最后，主节点得到从各从节点发送来的结果，汇总结果并返回给客户端。

分布式数据库能够存储海量的数据，且能够通过增加节点来扩展存储量，但由于数据分布在多个节点上，执行表连接查询时，通常需要迁移数据，所以，相较于单机数据库，影响查询性能的因素不仅包含从节点的 CPU，内存和 I/O 性能，还包括节点间网络的性能。

2.2 分析型任务特征

分布式数据库执行查询时，影响查询性能有 4 个重要的因素，分别为：CPU、内存、I/O 和网络，在这 4 个因素当中，CPU 和内存相较于 I/O 和网络，影响较小，因为通过实验观察到，在多个查询并行执行的时候，集群中各个节都有足够的 CPU 资源。另外，对于每个数据库实例，配置充足的内存。当有多个查询执行时，如果分配给该实例的内存耗尽，操作系统会将内存中的部分数据移到磁盘中，也就是发生页置换，引起 I/O 操作。分布式数据库最耗时的操作是扫描表，即发生 I/O，其次是节点间数据传输。所以，本文主要研究磁盘 I/O 和节点间网络因素对查询性能的影响。

在本文的研究中，主要针对分析型查询，其特点是包含大量的 join 操作，每个 join 需要通过 I/O 和网络读写数据和传输数据，其查询时间一般较长。多个查询争用 I/O 和网络资源，这必然会对查询产生影响。

本文用到的查询是由 TPC－DS 中的 10 个模板生成[26]。选取的查询模板分别：17，20、25、26、32、33、61、62、

65、71，它们都为 IO 敏感型查询，即花在 I/O 上的时间较多或占整个查询时间的比例较大，有些查询的 I/O 时间占整个执行时间的 90%以上。

2.3　取样

本文定义 MPL（multi-programming level）为同时运行的查询个数，当 $MPL=3$ 时，表示当前有 3 个查询同时执行，且这 3 个查询构成一个查询组合。由于一共有 10 个查询，当 MPL 为 2 时，可以两两结合组成查询组合，但当 MPL 等于大于 3 时，手工设计查询组合会变得越来越复杂，所以，选用 LHS（latin hypercube sampling）技术取样，下面以一个例子说明如何使用该方法构建查询组合。以 MPL 等于 3 为例，在 python 中，运行 $X=$lhs（3，10）生成抽样矩阵 $\boldsymbol{X}$：

$$\boldsymbol{X}=\begin{bmatrix}0.48598874 & 0.53752996 & 0.14463418\\ 0.63232065 & 0.25919125 & 0.56607826\\ 0.56970233 & 0.32660218 & 0.25538848\\ 0.86806865 & 0.60768378 & 0.92189821\\ 0.24429758 & 0.46828485 & 0.60020496\\ 0.74722981 & 0.86315018 & 0.30160745\\ 0.96099145 & 0.79773071 & 0.46703065\\ 0.37610543 & 0.07210812 & 0.77056908\\ 0.05637045 & 0.12928659 & 0.02091329\\ 0.10821039 & 0.97343076 & 0.88174788\end{bmatrix}$$

矩阵 $\boldsymbol{X}$ 中的元素都位于 0～1 之间，然后把这些元素化为整数。变换的过程是将矩阵 $\boldsymbol{X}$ 中的每个元素扩大 10 倍，然后再向上取整得到整数矩阵 $\boldsymbol{Y}$：

$$\boldsymbol{Y}=\begin{bmatrix}5. & 6. & 2.\\ 7. & 3. & 6.\\ 6. & 4. & 3.\\ 9. & 7. & 10.\\ 3. & 5. & 7.\\ 8. & 9. & 4.\\ 10. & 8. & 5.\\ 4. & 1. & 8.\\ 1. & 2. & 1.\\ 2. & 10. & 9.\end{bmatrix}$$

$\boldsymbol{Y}$ 包含 10 种整数，并且 $\boldsymbol{Y}$ 中的每行每列的数字都不重复，接下来把这 10 个数字映射成具体的查询 ID，映射的关系如下：

表 1　查询映射表

序号	1	2	3	4	5	6	7	8	9	10
query	17	25	26	32	33	61	62	65	71	20

根据映射表最终得到在 MPL 为 3 下的查询组合为：

$$\boldsymbol{Z}=\begin{bmatrix}33. & 61. & 25.\\ 62. & 26. & 61.\\ 61. & 32. & 26.\\ 71. & 62. & 20.\\ 26. & 33. & 62.\\ 65. & 71. & 32.\\ 20. & 65. & 33.\\ 32. & 17. & 65.\\ 17. & 25. & 17.\\ 25. & 20. & 71.\end{bmatrix}$$

在矩阵 $\boldsymbol{Z}$ 中，每一行代表一个查询组合，每一行的第一个查询代表该查询组合的主查询，其余都是并发查询。在每个查询组合中，每个查询构成一个查询流，查询流就是让每个查询运行 n 次，这样就得到 n 个样本数据，这里只取后 $n-1$ 次的数据。

2.4　预测评价

为了测量预测模型质量，本文使用平均相对误差 MRE（mean relative error）来衡量，该度量的定义如下：

$$MRE=\frac{1}{n}\sum_{i=1}^{n}\frac{|observed_i-predicted_i|}{observed_i}\tag{1}$$

$observed_i$ 为实际运行的时间，$predicted_i$ 为预测的时间，MRE 越低，说明预测模型越准确。

3　预测模型

该部分讲述如何对主查询和并发查询的 I/O 和网络资源争用进行建模。本文提出了查询干扰度和查询敏感度两个模型，查询干扰度用来描述主查询当前执行环境的优劣，也即描述资源的争用情况，而查询敏感度是描述主查询在不同并发查询执行时，它的性能如何变化。

3.1　查询干扰度

I/O 和网络是影响数据库查询性能最重要的两个因素，在一定程度上能够决定查询总延迟。假定一个查询组合为 m，它包括主查询 q 和与主查询并行执行的查询 $C=\{c_1,c_2,\cdots,c_n\}$，并发查询的个数为 n。首先，得到每个并发查询 c_i 单独运行时需要的 I/O 和网络资源，此时，没有发生资源竞争。然后，估计每个并发查询与主查询争用资源对主查询的影响。最后，评估由于并发查询之间争用资源对主查询的影响。定义变量如表 2 所示。

表 2　主要符号含义

符号	含义
P_q	表示查询 q 中 I/O 时间占总运行时间的百分比
S_t	单独扫描表 t 所需要的时间
ρ_{c_i}	主查询与并发查询共享的 I/O 时间
φ_{c_i}	并向查询之间共享的 I/O 时间
$\tau_{\min c_i}$	并发查询 c_i 单独执行的时间
$\tau_{\max c_i}$	并发查询 c_i 在最差环境下执行时间
$\tau_{q,m}$	查询 q 在查询组合中的执行延迟
t_q	查询 q 的迁移数据量
σ_{c_i}	并发查询 c_i 对主查询的网络干扰
γ_{c_i}	并发查询 c_i 对主查询的干扰度
$\gamma_{q,m}$	查询 q 在查询组合中的 CQI 值
$c_{q,m}$	查询 q 的 PRP 值

3.1.1 Baseline I/O

Baseline I/O 指的是查询的基准 I/O，即当一个查询独立执行时，它的 I/O 时间占用执行总时间的百分比，所占百分比越大，则表示此查询需要越多的 I/O 资源。用表示一个并发查询中 I/O 所占的百分比。

3.1.2 Positive I/O

当主查询与并发查询共同执行时，如果一个并发查询与主查询扫描不同的表，并发查询会对主查询产生“干扰”，这是因为不同查询会争用 I/O。当并发查询与主查询扫描相同的表时，这种“干扰”会大大减小，甚至会对主查询产生“促进”作用，因为在数据库中，当频繁扫描一个表时，会把这个表的数据存入到共享缓存中，此后再请求这个表的数据，会直接从共享缓存中取，从而避免了重复性的 I/O 操作。下面通过模型来量化并发查询与主查询的相互作用。

假设表 t 是主查询 q 与并发查询 c_i 共同扫描的表。定义如下值：

$$h_{tc_i}=\begin{cases}1 & \text{如果表 } t \text{ 是共同扫描的表}\\ 0 & \text{如果表 } t \text{ 不是共同扫描的表}\end{cases} \tag{2}$$

可以看到 h_{tc_i} 取值只有 0 和 1，下面计算共享的 I/O 时间。

$$\rho_{c_i}=\sum_{i=1}^{n} h_{tc_i} * S_t \tag{3}$$

上述公式中，n 表示主查询和并发查询需要扫描表的总个数，S_t 表示扫描表 t 花费的时间。本文用 select * from [table] 形式的查询语句获取扫描表 [table] 花费的总时间，即该查询语句执行时间中的扫描表的时间。在 Greenplum 中，表的数据分布到各个节点中，查询在各个节点中执行，所以，多个查询如果包含共同的表，则可“省去”重复在磁盘上扫描表的时间。公式（3）计算由于共享 I/O 的而节省的时间。

3.1.3 Concurrent I/O

除了考虑主查询和并发查询的共享 I/O 之外，还需要度量并发查询之间的 I/O 影响。即主查询与两个并发查询 a 和 b 共同执行时，a 和 b 由于并发执行所节省的 I/O 时间。首先定义表 t 为并发查询和其他非主查询共同扫描的表：

$$l_{t,c_i}=\begin{cases}1 & \text{查询 } c_i \text{ 扫描表 } t\text{，主查询不扫描}\\ 0 & \text{其他情况}\end{cases} \tag{4}$$

定义 d_t 为扫描表 t 的并发查询的个数，这里 d_t 必须大于 1。另外，由于只考虑并发查询之间的表扫描情况，所以这里的表 t 不能出现在主查询当中。计算由于并发查询共享 I/O 而节省的时间为：

$$\varphi_{c_i}\sum_{i=1}^{n} l_{t,c_i} * (1-\frac{1}{d_t}) S_t \tag{5}$$

上面公式中的 n 同样是主查询和并发查询需要扫描表的总数。

3.1.4 网络争用

本文面向的是分布式数据库，数据分布在集群中的各个节点中，SQL 查询中的表连接操作一定会发生数据传输，即把在一个节点上的数据迁移到另一个节点上。Greenplum 中数据迁移的方式有两种：广播和重分布。广播就是把一个节点上的数据传输给其他所有节点，从而每个节点都有一个表的完整数据。重分布就是把表的数据根据关联键计算哈希值，然后再重新分布到各个节点上。假设一个表的记录数为 N，那么重分布的数据量为 N，广播的数据量为 N^* 节点数，通过上述的方式就可以计算一个连接操作的数据迁移量。主查询的数据迁移总量为 t_q，并发查询的迁移数据量为，t_{c_i} 定义并发查询对主查询的网络干扰为：

$$\sigma_{c_i}=\frac{t_{c_i}}{t_q} \tag{6}$$

从上式可以看出，并发查询 c_i 的数据迁移量越大，对主查询的干扰就越大；反之，则越小。这是由于系统的网络带宽是一定的，当网络中有其他查询传输数据时，必然会影响主查询的数据传输。

3.1.5 查询干扰度模型

得到上述各个变量以后，就可以定义并发查询对主查询的影响。

$$\gamma_{c_i}=\frac{\tau_{\min c_i} * P_{c_i}-\rho_{c_i}-\varphi_{c_i}}{\tau_{\min c_i}}+\sigma_{c_i} \tag{7}$$

公式（7）可以理解为并发查询 c_i 与主查询直接竞争 I/O 和网络的激烈程度，公式（7）的前半部分是主查询减去了并发查询与主查询共享 I/O 的时间，在网络争用确定的情况下，当 γ_{c_i} 越大，则表示并发查询与主查询共享 I/O 的时间越短，竞争 I/O 的时间越长，在这种情况下，会延长主查询的查询延迟。当 γ_{c_i} 越小，则表示并发查询与主查询的资源竞争越小，对主查询的延迟影响越小。

在一个查询组合中，定义主查询的 CQI 值为 $\gamma_{q,m}$，计算公式如下：

$$\gamma_{q,m}=\frac{1}{n}\sum_{i=1}^{n}\gamma_{c_i} \tag{8}$$

上述公式即取各并发查询的 γ_{c_i} 平均值。

3.2 查询敏感度

查询敏感度是指主查询对不同执行环境的敏感程度，这里不同的执行环境就是不同的 MPL 以及在相同 MPL 下的不同查询组合。敏感程度指主查询的性能变化，不同的执行环境会引起不同的资源竞争，进而导致主查询的性能变化。下面详细介绍该模型。

3.2.1 查询性能区间

查询性能区间 PR（performance range）指的是一个查询延迟时间范围，这个区间中的值表示查询在不同环境中的执行时间，区间的最大值为 $\tau_{\max c_i}$，表示查询在最恶劣的资源环境下的执行时间。本文通过不断读取大文件并在不同节点间交换传输这些文件模拟最差的环境。最小值为 $\tau_{\min c_i}$，表示在当前环境中，只有这个查询在执行时的延迟时间。上述两个值表示在极端执行环境中的执行查询，在其余环境下的查询执行时间都位于该查询性能区间中，主查

询的 PRP（performance range point）值定义如下：

$$c_{q,m}=\frac{\tau_{q,m}-\tau_{\min q}}{\tau_{\max q}-\tau_{\min q}} \tag{9}$$

当知道了 $c_{q,m}$的值以后，将该值代入公式（9）反推可得到 $\tau_{q,m}$，即主查询的查询延迟。接下来，介绍如何预测 $c_{q,m}$的值。

3.2.2　查询敏感度模型

在 3.1 节介绍了 CQI 模型，该模型可以用于预测查询延迟（后面通过实验具体验证）。这意味着，给定一个查询组合 m 和一个主查询 q，可以利用公式（8）来计算 CQI 值，然后使用线性回归模型预测查询的性能。为了进一步说明查询性能和 CQI 之间的线性关系，本文引入查询敏感度 QS（query sensitivity）。

假定 CQI 和 PRP 存在线性的关系，定义如下公式：

$$c_{q,m}=\mu_q * \gamma_{q,m}+b_q \tag{10}$$

上述公式中，μ_q 为斜率，b_q 为截距，$c_{q,m}$与 $\gamma_{q,m}$是一种线性关系。

3.2.3　预测流程

在 Greenplum 中，利用 QS 模型预测查询 q 的流程如图 1 所示。

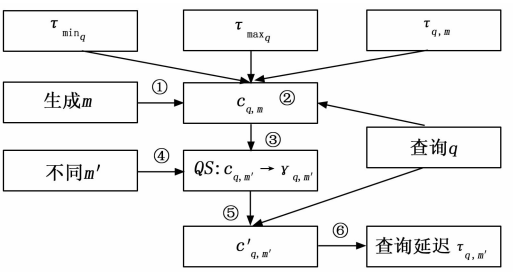

图 1　利用 QS 预测查询延迟

图 1 展示了预测查询延迟的具体流程。第一步，利用 LHS 生成查询组合 m，其中包括主查询 q；第二步，分别得到关于 q 的 $\tau_{\min q}$、$\tau_{\max q}$和 $\tau_{q,m}$，$\tau_{\min q}$和 $\tau_{\max q}$是能够预先得到的，而 $\tau_{q,m}$能从实验数据中得到。之后，代入公式（9）得到 $c_{q,m}$。通过上述方式，可以从实验产生的测试集中得到很多的（$c_{q,m}$，$\gamma_{q,m}$）值对，第三步，利用得到的（$c_{q,m}$，$\gamma_{q,m}$）值对使用基于最小二乘法线性的回归方法训练 QS 模型（公式（10）），得到查询 q 的 QS 模型，第四步中，当 q 处于另一个查询组合中时，要预测此时 q 的查询延迟，先计算此时 q 的 CQI 值 $\gamma_{q,m}$，接着第五步便可通过第三步中产生的 QS 模型得到 $c'_{q,m'}$，第六步把 $c'_{q,m'}$再次代入公式（9）即可得到在 m'查询组合中 q 的查询延迟 $\tau_{q,m}$。

4　实验评估

4.1　实验环境

实验的环境为 Greenplum 分布式集群，Greenplum 版本为 5.0.0－alpha＋79a3598。集群中共有 4 个节点，一个主节点和 3 个从节点，从节点主要用来存放数据并执行查询，主节点则负责分配查询和汇总结果。主节点的硬件配置为 32 GB 的内存，CPU 为 4 核 Intel（R）Xeon（R）CPU E5-2630 v2 @ 2.60 GHz，从节点的内存 16 GB，CPU 的核数和型号与主节点相同，在每个从节点中有 4 个数据库实例，每个数据库实例相当于一个完整 PostgreSQL 的数据库，用于处理一部分的数据。主节点和从节点的操作系统均为 CentOS 7.4，linux 内核版本为 3.10。

表和数据通过 TPC-DS 生成，TPC-DS 是一个决策支持的 benchmark。实验所用数据量大小为 50 G，选取 TPC-DS 中的 10 个模板生成 10 个查询用于训练和测试模型，这 10 个查询主要是 I/O 敏感型查询，执行时间较长，有利于提高预测模型的精度。

4.2　实验过程

在第 3 节详细讲述了如何利用查询干扰度和查询敏感度模型预测一个查询的延迟时间，下面通过实验验证上述的模型方法。具体实验过程说明如下：

在实验环境中，基于 Greenplum 数据库集群的默认配置，执行 TPC-DS 基准测试。并通过设置 Greenplum 的并行度参数，控制同时执行的查询数分别指定的 MPL（例如：1－5）。与此同时，通过性能监控工具 Telegraf 采集服务器的各项资源开销数据，并结合 Greenplum 返回的查询执行时间等性能数据，构建下述提到的查询干扰度模型和查询敏感度模型，并据此进行分析型查询的性能预测。

4.2.1　查询干扰度模型

3.1 节介绍了主查询 q 在查询组合 m 中的查询干扰度，接下来，通过实验验证通过该模型预测查询延迟是否有效。

1）查询干扰度模型分量：

查询干扰度评估了并发查询对主查询的干扰程度，干扰程度越大，对主查询的延迟也就越大。在查询干扰度模型中，下面分别对干扰度模型中的 4 个分量进行验证。

首先是基准 I/O（baseline I/O），当模型中只有 baseline I/O 时，计算一个并发查询的干扰度会变成如下形式：

$$\gamma_{c_j}=(\tau_{\min c_j} * P_{c_j})/\tau_{\min c_j} \tag{11}$$

公式只计算了并发查询与主查询竞争的 I/O 带宽，相当于并发查询与主查询是完全竞争的关系。

其次，在基准 I/O 的基础上，加入并发查询与主查询的正向交互作用因素，即 positive I/O，得到如下形式：

$$\gamma_{c_j}=(\tau_{\min c_j} * P_{c_j}-\rho_{c_j})/\tau_{\min c_j} \tag{12}$$

式（12）从争用的 I/O 时间中减去了由于共享表扫描所节省的时间，该公式主要考虑了并发查询对主查询的影响。

然后，再进一步考虑并发查询之间的交互，即 concurrent I/O，所得模型如下：

$$\gamma_{c_j}=(\tau_{\min c_j} * P_{c_j}-\rho_{c_j}-\varphi_{c_j})/\tau_{\min c_j} \tag{13}$$

式（13）即在式（12）的基础上加入了在一个查询组合中，并发查询之间的共享 I/O 时间。并发查询之间的作用会间接影响主查询的 I/O 操作。

最后，再加入网络争用的因素，得到 CQI 模型：

$$\gamma_{c_i} = \frac{(\tau_{\min c_i} * P_{c_i} - \rho_{c_i} - \varphi_{c_i})}{\tau_{\min c_i}} + \sigma_{c_i} \quad (14)$$

2）查询干扰度模型验证：

首先评估 CQI 的各个分量对误差率的影响，然后利用 CQI 预测查询延迟。当 *MPL* 为 3 时，上述各个变量对查询延迟的预测误差如下：

从图 2 可以看到，只利用 baseline I/O（对应公式（11））预测查询延迟的时候，它的误差较大，当加入并发查询交互（对应公式（12））的因素后，对误差率有明显的降低。考虑 concurrent I/O（对应公式（13））和网络争用因素（对应公式（14）），对预测的准确率没有明显提高，说明 positive I/O 是影响预测模型准确率的主要因素，其他因素能够小幅度的提升准确率。综上，CQI 考虑了并发查询之间的主要影响因素，是一个较好的预测模型。接下来，验证 CQI 模型对各个查询的误差。

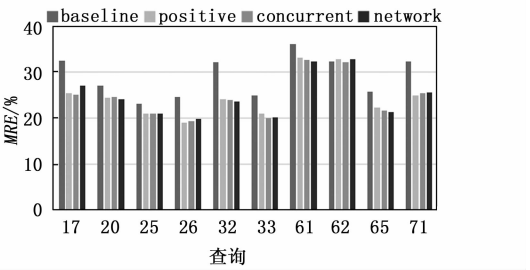

图 2 各个查询在不同度量下的误差

对于一个给定的查询组合，其中有一个主查询，其他查询为并发查询。实验产生的数据分为训练集和测试集。假设主查询的查询延迟和 CQI 存在线性关系，则通过训练集能够得到这个线性关系，然后，利用该线性模型对主查询进行预测，再与测试集数据比较并计算误差，从而得到图 3。

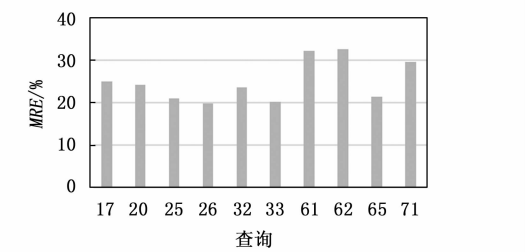

图 3 各个查询在 *MPL* 等于 3 时的误差

图 3 为 10 个查询在 *MPL* 为 3 时的预测情况，从图中可以看到，查询延迟预测的误差大部分在 25％以下。此处的误差是平均相对误差（*MRE*），其定义已经在公式（1）中给出。查询 61 和 62 的误差相对较高，这是由于这两个查询相对较为简单，查询时间较短，因而其它因素查询预测的干扰较大，导致误差相对较大。在实验中，查询 71 的误差也相对较高。该查询较为复杂，主要用于“对于指定的经理及其关联的所有 3 个销售渠道，找出其一个月内在早餐或晚餐时间段营收最高的产品”，其执行时间较长，因而误差相对率偏高。该查询的标准 SQL 形式如下：

```
select i_brand_id brand_id, i_brand brand,t_hour,t_minute,
    sum(ext_price) ext_price
    from item, (select ws_ext_sales_price as ext_price,
    ws_sold_date_sk as sold_date_sk,
    ws_item_sk as sold_item_sk,
    ws_sold_time_sk as time_sk
    from web_sales,date_dim
    where d_date_sk = ws_sold_date_sk
    and d_moy=12
    and d_year=2002
    union all
    select cs_ext_sales_price as ext_price,
    cs_sold_date_sk as sold_date_sk,
    cs_item_sk as sold_item_sk,
    cs_sold_time_sk as time_sk
    from catalog_sales,date_dim
    where d_date_sk = cs_sold_date_sk
    and d_moy=12
    and d_year=2002
    union all
    select ss_ext_sales_price as ext_price,
    ss_sold_date_sk as sold_date_sk,
    ss_item_sk as sold_item_sk,
    ss_sold_time_sk as time_sk
    from store_sales,date_dim
    where d_date_sk = ss_sold_date_sk
    and d_moy=12
    and d_year=2002
    ) tmp,time_dim
    where
    sold_item_sk = i_item_sk
    and i_manager_id=1
    and time_sk = t_time_sk
    and (t_meal_time = 'breakfast' or t_meal_time = 'dinner')
    group by i_brand, i_brand_id,t_hour,t_minute
    order by ext_price desc, i_brand_id
```

总体而言，除去类似于查询 61 和查询 62 这类因查询任务简单、执行时间短，易受干扰的查询，以及查询 71 这类非常复杂的查询，TPC-DS 的代表性查询在 *MPL* 为 2、4 和 5 时的资源预测情况与 *MPL* 为 3 时类似，大部分查询误差率仍在 25％以下。

4.2.2 查询敏感度模型

前面的实验验证了 CQI 可以较准确地预测查询延迟，而 CQI 是针对特定的查询组合，为此，本文提出了 QS 模型，它能够感知查询所处的不同环境（不同的查询组合），并做出预测。

对于一个特定的查询 q，找到包含这个查询的查询组合，然后以 q 为主查询构建 QS 模型，利用该模型预测执行时间，并与实际执行时间比较，得到结果如图 4 所示。

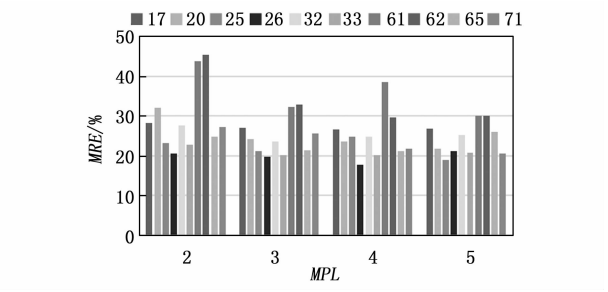

图 4　各个查询在不同 *MPL* 时的误差

从图 4 可以看出，不同的 *MPL*，除去查询 61 和 62，查询的误差都在 25%以下，有的甚至能够达到 20%以下。同样的，查询 61 和 62 的误差较高的原因在于它们的执行时间较短，从而造成误差较大。由此实验结果表明，QS 模型能够适应不同的查询执行环境（不同 *MPL* 下的不同查询组合），从而能够较准确地预测查询的执行延迟。

5　结束语

本文主要研究在分布式数据库中预测一个查询在多个其他查询并行执行的情况下其执行时间的技术，考虑了 I/O 和网络因素，这与其他基于机器学习的在单机数据库上的性能预测技术有明显不同。

本文主要提出了两个模型，分别为：查询干扰度和查询敏感度。查询干扰度（CQI）用于建立资源竞争的衡量模型，而查询敏感度（QS）用于衡量主查询对其他并发查询的感知度。查询敏感度是建立在查询干扰度的基础上，通过实验发现 CQI 与查询延迟存在线性关系，而 QS 则在 CQI 的基础上建立这种线性模型，从而利用 QS 预测查询延迟。

最后，实验结果表明模型的预测误差率大部分能够维持在 25%以下，能够较准确地预测查询的延迟时间。在此后的工作中，可以考虑如何使用更有效的指标来量化资源的竞争，以及如何利用查询执行计划辅助来建立预测模型。

参考文献：

[1] DUGGAN J, CETINTEMEL U, PAPAEMMANOUIL O, et al. Performance prediction for concurrent database workloads [C] //ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, DBLP, 2011: 337 - 348.

[2] MAGHAWRY E A, ISMAIL R M, BADR N L, et al. An enhanced queries scheduler for query processing over a cloud environment [C] //International Conference on Computer Engineering & Systems, IEEE, 2015: 409 - 414.

[3] 肖　垚. 面向数据市场的多租户查询负载优化处理技术 [D]. 上海：华东师范大学，2019.

[4] 张锦文. 查询交互量化与查询响应时间预测模型 [D]. 太原：太原理工大学，2018.

[5] AROSTEGI M, TORRE-BASTIDA A, NEKANE BILBAO M, et al. A heuristic approach to the multicriteria design of IaaS cloud infrastructures for Big Data applications [J]. Expert Systems, 2018, 35 (5): e12259.

[6] 游　琪. 多核环境下内存数据库并发调度技术优化研究 [J]. 计算机测量与控制，2017，25 (8)：234 - 236.

[7] SHEIKH M B, MINHAS U F, KHAN O Z, et al. A bayesian approach to online performance modeling for database appliances using gaussian models [C] // ACM International Conference on Autonomic Computing, ACM, 2011: 121 - 130.

[8] MOZAFARI B, CURINO C, JINDAL A, et al. Performance and resource modeling in highly-concurrent OLTP workloads [C] // ACM SIGMOD International Conference on Management of Data, ACM, 2013: 301 - 312.

[9] TONELLOTTO N, MACDONALD C. Using an inverted index synopsis for query latency and performance prediction [J]. ACM Transactions on Information System, 2020, 38 (3): 1 - 33.

[10] ZHOU X, SUN J, LI G, et al. Query performance prediction for concurrent queries using graph embedding [J]. Proceedings of the VLDB Endowment, 2020, 13 (9): 1416 - 1428.

[11] CHAUDHURI S, NARASAYYA V, RAMAMURTHY R. Estimating progress of execution for SQL queries [C] // SIGMOD. 2004: 803 - 814.

[12] LUO G, NAUGHTON J F, YU P S. Multi-query SQL progress indicators [C] //International Conference on Advances in Database Technology, Springer-Verlag, 2006: 921 - 941.

[13] LUO G, NAUGHTON J F, ELLMANN C J, et al. Toward a progress indicator for database queries [C] //ACM SIGMOD International Conference on Management of Data, Paris, France, DBLP, 2004: 791 - 802.

[14] AKDERE M, ÇETINTEMEL U, RIONDATO M, et al. Learning-based query performance modeling and prediction [C] //IEEE International Conference on Data Engineering: IEEE, 2012, 41 (4): 390 - 401.

[15] MARCUS R, PAPAEMMANOUIL O. Plan-structured deep neural network models for query performance prediction [J]. Proceedings of the VLDB Endowment, 2019, 12 (11): 1733 - 1746.

[16] WU W, YUN C, HACIGUMUS H, et al. Towards predicting query execution time for concurrent and dynamic database workloads [J]. Proceedings of the Vldb Endowment, 2013, 6 (10): 925 - 936.

[17] DUGGAN J, CETINTEMEL U, PAPAEMMANOUIL O, et al. Performance prediction for concurrent database workloads [C] // ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, DBLP, 2011: 337 - 348.

（下转第 221 页）