面向数据流离线分析的无共享并行查询中间件研究*

王庆荣[1,2]

(1. 兰州交通大学 电子与信息工程学院, 兰州 730070; 2. 兰州理工大学 电气工程与信息工程学院, 兰州 730050)

摘 要: 通过分析网络监控离线分析处理的负载特征,给出了一种面向数据流离线分析处理的并行多策略查询中间件,并利用多策略及 DBMS 实现了局部结果的汇总,对需后处理查询的系统扩展性、不需后处理查询的系统扩展性分别进行了评价分析。评价分析结果表明:提出的基于面向数据流离线分析处理的并行多策略查询中间件,不但做到了中间件的轻量级,实现了查询内部的并行化;同时,能利用多策略及 DBMS 实现子节点间并行查询,能提高查询的响应时间,从而在多节点时保持较好的扩展比,避免了系统过载,提高了资源利用率。

关键词: 数据流; 无共享; 离线分析; 并行查询

中图分类号: TP31.13 **文献标志码**: A **文章编号**: 1001-3695(2012)07-2531-03

doi:10.3969/j.issn.1001-3695.2012.07.034

Research on data stream off-line analysis for shared-nothing parallel query middleware

WANG Qing-rong[1,2]

(1. *School of Electronic & Information Engineering, Lanzhou Jiaotong University, Lanzhou* 730070, *China*; 2. *School of Electrical & Information Engineering, Lanzhou University of Technology, Lanzhou* 730050, *China*)

Abstract: The paper analyzed the load characteristics of off-line analysis process based on network monitoring management. And it presented multi-policy data stream off-line analysis for parallel query middleware, summarizing the local results using multi-policy and DBMS. Finally, system scalabilities were valued and analyzed respectively which one needed post-processing query and another not needed. The results show that multi-policy data stream off-line analysis for parallel query middleware realizes not only the lightweight of middleware but also the parallelization of inner query. Meanwhile, query response time can be improved using multi-policy and parallel query among DBMS and sub-nodes. So in the multi-nodes condition, scalability ratio can be maintained. And system overload can be avoided. At the same time, resource efficient can be improved.

Key words: data stream; shared-nothing; off-line analysis; parallel query

随着传感器网络、网络管理等需求的推动,短小数据流成为一种日益重要的数据密集型应用,单个数据流查询可能需要处理上百 GB 甚至 TB 级的数据,查询响应时间和系统扩展能力已成为挑战。随着数据库规模日趋庞大,其查询日趋复杂,对数据库系统性能的要求也越来越高。另一方面,随着并行计算机系统迅速的发展,在并行计算机系统上实现数据库管理系统能极大地提高数据库管理系统的性能[1]。近年来,国内外学者就并行查询开展了大量的研究,文献[2]借鉴分布式数据库查询处理中基于半连接的优化思想,提出了基于半连接的并行查询处理算法;文献[3]探讨了基于流机制的并行查询执行技术;文献[4]采用虚拟分区实现查询内并行,但由于采用全复制,影响了大规模数据的扩展性,不能够处理海量数据的并行处理。文献[5]利用动态规划算法,提出了数据流上并行查询优化技术。本文通过分析网络监控离线分析处理的负载特征,提出了一种并行的多策略查询数据库中间件。

1 数据流离线查询负载特征分析

随着计算机和信息技术的快速发展,导致数据库规模日益庞大,海量数据流数据的查询,具有费时长、资源消耗大等特点,可以汇总成以下几类查询:

a)不需进行二次处理的数据,可直接将结果返回给用户,例如:

```
SELECT * FROM eventlog WHERE TS >= t1 AND TS <= t2;
```

b)需要转义后进行处理,并且进行二次处理的,例如:

```
SELECT AVG(env_num) FROM eventlog WHERE TS >= t1 AND TS <= t2;
```

需要将上述语句转换为两条语句。第一条语句在所有的子数据库节点上执行:

```
SELECT INSERT INTO temp AS SELECT SUM(env_num) AS sum_env_num, COUNT(env_num) AS count_env_num FROM enentlog WHERE TS >= t1 AND TS <= t2;
```

第二条语句汇总子数据库节点结果进行处理:

```
SELECT SUM(sum_env_num)/SUM(count_env_num) FROM temp;
```

c)子查询不需要转义而需进行二次处理的,例如:

```
SELECT * FROM eventlog WHERE TS >= t1 AND TS <= t2 ORDER BY eventcount DESC;
```

需要将上述语句转换为下列两条语句。第一条语句在所

收稿日期: 2011-11-09; **修回日期**: 2011-12-16 **基金项目**: 教育部人文社科规划资助项目(11YJAZH132,11YJCZH170);甘肃省自然基金资助项目(1107RJZA166)

作者简介:王庆荣(1977-),女,宁夏固原人,副教授,博士研究生,主要研究方向为城市交通控制及智能交通系统(zhudd003@163.com).

有的子数据库节点上执行：

INSERT INTO temp2 SELECT * FROM eventlog WHERE TS >= t1 AND TS <= t2 ORDER BY eventcount DESC；

第二条语句汇总结果进行处理：

SELECT * FROM temp2 ORDER BY eventcount DESC；

d)完全不需要转义的查询

SELECT * FROM eventlog WHERE rownum < 1000001。

2 并行多策略查询系统结构设计

2.1 并行查询系统结构

数据库链路技术可将远程数据源映射到本地，采用 union 将多个数据源的执行结果汇总后返回给用户，查询以单线程/单进程方式执行，即使有多个远程数据源参与到一个查询当中，在任一个时刻只能有一个远程数据源。这种环境中的串行化意味着失去一个通过重叠处理不同远程数据源来提高性能的机会，返回第一条记录的结果取决于所有执行的数据源的查询时间总和。如将子查询并行，则查询时间只受限于最慢的执行查询节点，能很好地保证系统的扩展性。而对于一些不依赖最慢执行结果的查询，则可以将最快返回的查询返回给用户，这样使用户获得的第一条记录时间最短。图 1 中获取第一条记录的时间是 5 min，能在第 5 min 获取第一条记录只是一种特殊情况，要保证所有的 SQL 能获取最好的查询性能，则能够在运行过程根据查询的语句选用不同的并行查询策略[6]。

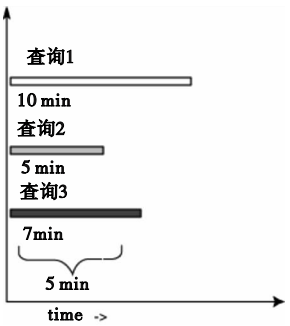

图1 并行查询

并行查询系统由数据库节点和查询节点组成(图 2)。数据库节点存储数据并执行局部查询，查询中间件运行在单独的查询服务器上，分解并派发查询到数据库节点，将得到的中间结果进行后处理。数据库节点与查询中间件通过高速网络和查询服务器连接。

假定系统中有 N 个数据库节点，对应的数据集合是 $D_i(1 \leqslant i \leqslant N)$，$D_i$ 之间没有数据重叠。客户端连接到查询服务器后，提交查询 Q；查询服务器中的并行管理调用 SQL 解析器得到子查询集合 $\{S_i \mid 1 \leqslant i \leqslant N\}$ 和后处理语句 P。然后在各数据库执行相应子查询得到结果集合 $M = \cup S_i(D_i)(1 \leqslant i \leqslant N)$，查询服务器根据 SQL 语句类型采用不同的策略返回结果给用户[7,8]。

2.2 多策略查询

从上述查询语句的分类可以看出，查询类型不同处理语句 P 的类型也不相同，有的需要在查询服务器上执行，有的则完全不需要执行。P 的执行过程如图 3、4 所示。

由图 3 可见，子数据库节点执行相应子查询得到结果集合 $M = \cup S_i(D_i)(1 \leqslant i \leqslant N)$ 在查询服务上汇总后，使用后处理语句 P 进行二次处理后将结果返给用户。用户得到的第一条记录时间，系统受限于整个系统中最慢的子查询节点。并且当系统子节点数据库规模扩大时，子查询结果并发地返给查询服务器时，可能受限于查询服务器的处理能力。

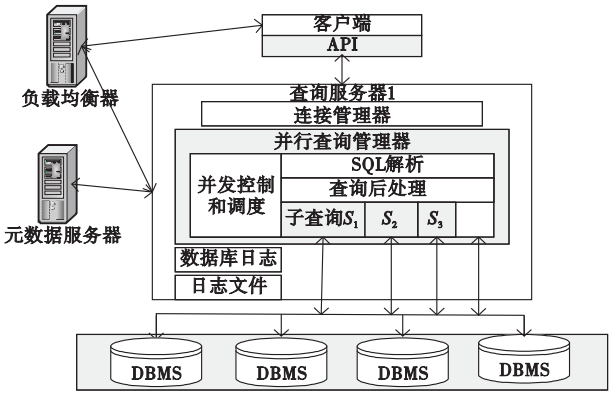

图2 并行查询中间件系统结构

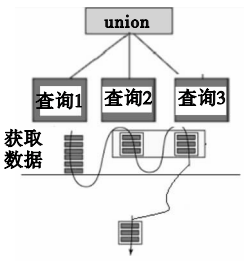

图3 查询子数据库

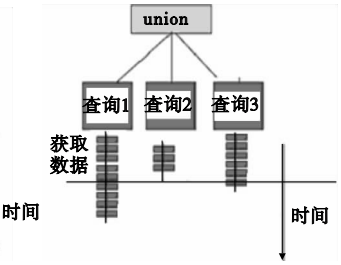

图4 查询服务器

由图 4 可见，查询服务器首先将最先返回的子查询结果 $(\min(S_i(D_i)))$，$1 \leqslant i \leqslant N)$ 发给用户；依次将其他返回结果返回给用户。且用户得到第一条记录的时间不受限于整个系统执行最慢的子查询节点。

如将属于图 4 类型的查询使用图 3 进行后处理查询的方式来处理，系统的性能受限于最慢的数据库处理节点，且数据库子查询的数据需要查询服务器再次执行后处理后才能返回给用户，造成严重的网络和 I/O 性能浪费。

多策略查询需要一个 SQL 分析模块区分查询的类型：a)能够区分查询类型；b)能够正确地转义查询成为子查询；c)能够查询类型产生正确的后处理语句。需要能够进行图 3 中后处理的后处理汇总模块[9,10]，需要能够处理图 4 中多个结果集的多结果处理模块。

3 系统扩展性评价分析

扩展性是衡量并行数据库的重要指标。通常用扩展比来表示当系统规模扩大时，大数据量分析查询的性能变化[8]

$$su = \frac{small\_problem\_elapsed\_time\_on\_small\_system}{large\_problem\_elapsed\_time\_on\_large\_system} \quad (1)$$

扩展比的理想值为 1，称做线性扩展。

3.1 需后处理查询的系统扩展性

子节点查询执行流程包括得到第一条记录、获取整个结果集和写数据到查询服务器三个过程。因系统采用高速网络，子节点和查询服务器间的传输时间忽略不计，则用户获取第一条记录的执行时间为

$$T = T_{oh} + T_{res} + T_{ins} + T_{pt} \quad (2)$$

其中：T_{oh} 指开始到产生第一个结果的时间；T_{res} 是获取整个结果集所用时间；T_{ins} 是子节点写结果集到查询服务器时间。

当系统扩展到 n 个数据库节点时，总的中间结果数据量是

单机的 n 倍[9]。假定数据库写入速度的极限值为 ρ 且达到 ρ 后不再下降。设 r_s 为单机产生的中间结果,则总写入时间不小于 $n \times r_s/\rho$。同时假设后处理的时间同数据量成正比。总的执行时间为:$T_{oh} + \max\{T'_{ins}\} + n \times T_{pt}$,$n \times rs/\rho \leqslant T'_{ins} \leqslant N \times T_{ins}$,含 n 个数据库节点的系统扩展比为

$$\frac{T_{oh} + \max(T_{res}, T_{ins}) + T_{pt}}{T_{oh} + \max(T_{res}, T'_{ins}) + nT_{pt}}, \text{其中} \frac{rs}{\rho} \leqslant T'_{ins} \leqslant nT_{ins} \quad (3)$$

显然,各阶段的执行时间主要依赖于查询类型和输入关系的记录量。定义查询的选择比为该查询结果集记录数和输入的所有关系记录数的比值。根据系统涉及的原始查询(S)、子查询(S_1)和后处理查询(S_2)的选择比,将查询分成如下几类:

a)S 小,S_1 大。这样需要后处理的记录数较多,最终结果集较小。

如果产生中间结果的速度远比写入速度慢,扩展比为:$\frac{T_{oh} + T_{res} + T_{pt}}{T_{oh} + T_{res} + nT_{pt}}$。当产生中间结果的聚合速度超过 ρ 时,数据库写入是瓶颈,扩展比为:$\frac{T_{oh} + T_{res} + T_{pt}}{T_{oh} + n \times r_s/\rho + nT_{pt}}$,如第 4 类语句。

b)S 小,S_1 选择比也小。后处理数据量小,T_{ins} 和 T_{pt} 可忽略不计,$\frac{T_{oh} + T_{res} + T_{pt}}{T_{oh} + T_{res} + nT_{pt}} \approx \frac{T_1}{T_1} = 1$,如第 1 类语句。

c)S 大。子查询产生大量结果,最终结果也很多。中间结果数量巨大,写数据库成为瓶颈。此时扩展比为:$\frac{T_{oh} + T_{ns} + T_{pt}}{T_{oh} + nT_{ins} + nT_{pt}}$,如第 3 类语句。

3.2　不需后处理查询的系统扩展性

子节点查询执行流程,包括得到第一条记录和获取整个结果集两个过程,因为系统采用高速网络,子节点和查询服务器间的传输时间忽略不计。则用户获取第一条记录执行时间为:$T = T_{oh}$,由于不需二次处理,可直接将结果按照子查询的查询速度返回给用户:$T = \min(T_{oh})$。当各子节点返回时间相同时,系统的扩展比为:$T_{oh}/T_{oh} = 1$。当结果集较大时,使用图 4 的策略能够限制改善系统的性能,减少系统的响应时间。

4　结束语

本文通过分析网络监控离线分析处理的负载特征,给出了一种面向数据流离线分析处理的并行多策略查询中间件,一方面实现查询内部的并行化,另一方面对于不同类型的查询采用不同的查询策略。采用 DBMS 作为后处理工具,做到了中间件的轻量级。采用面向资源的调度,调度使用不同资源的查询同时执行,提高资源利用率,避免系统过载。同时对查询根据不同的类型采用不同的执行策略提高执行效率,提高了系统的扩展性。

参考文献:

[1] DEWITT D J, GRAY J. Parallel database systems: the future of high performance database processing [J]. Communication of ACM, 1998, 35(6): 417-434.

[2] 王建国,吴建平,陈修环. 一种综合数据流和控制流的协议测试集自动生成法[J]. 清华大学学报,2000,40(3):117-125.

[3] 王意洁, 王勇军. 基于半连接的并行查询处理算法的研究[J]. 软件学报,2006,12(2):219-224.

[4] 陈红,文继荣,王珊. 基于流机制的并行查询执行技术[J]. 计算机科学,2006,27(4):9-13.

[5] JERMAINE C, DOBRA A. The sort merge-shrink join [J]. ACM Trans Database Systems, 2008, 31(4): 1382-1416.

[6] 王勇,焦丽梅. 面向网络数据管理的并行查询处理[J]. 计算机工程与应用,2009,43(30):5-10.

[7] 周虹. 数据流上并行查询优化技术[J]. 佳木斯大学学报,2009,27(1):42-46.

[8] 陈刚,顾进广,刘玲睿. 空间数据流的无阻塞快速连接算法[J]. 华中科技大学学报,2010,38(12):43-47.

[9] 李国徽,陈媛,陈基雄. 一种并发高效的连续时空查询处理算法[J]. 华中科技大学学报,2009,33(12):90-93.

[10] 唐桂芬,杨伟锋,黄双临. 一种高效的累进式空间连接查询处理算法[J]. 华中科技大学学报,2010,37(2):318-324.