基于统计的无阻塞连接算法

陈 刚 顾进广 李思川
(武汉科技大学计算机学院 武汉 430065)

摘 要 数据流上的关系查询处理技术是数据库研究领域的一大热点。优化无阻塞连接算法的关键在于提高内存连接阶段的效率。当内存空间满时,需要将内存数据刷新到外存相应分区,良好的刷新策略对于改进算法的性能至关重要。利用数据分布的特征,对关系连接的输出流,使用基于统计的方法,查找使用频率最低的元组,将使用频率较低的元组刷新到外存,以提高内存数据的效率。基于统计分析策略提高了刷新策略的准确性和效率及算法的适用范围。

关键词 数据流,无阻塞连接,内存刷新策略
中图法分类号 TP391 **文献标识码** A

Non-blocking Join Algorithm Based on Statistics

CHEN Gang GU Jin-guang LI Si-chuang
(School of Computer Science and Technology, Wuan University of Science and Technology, Wuhan 430065, China)

Abstract bstract Data stream query processing technology becomes a new and popular topic in database research area. The critical of improving non-blocking join algorithm is to improve the efficiency of memory join stage. If there are no more space for the coming tuple, some old tuples have to be flushed from memory to disk. A good refresh strategy is very helpful to increase join algorithm performance. The lowest frequently used tuples are searched from the result streams, then flush such tuples from memory to disk so that the tuples that are stayed in the memory would generate more results. Statistics join algorithm performance is increased obviously and it expands the adaptability of the data stream relation join algorithm.

Keywords Data stream, Non-blocking join, Memory flush strategy

1 引言

在Web环境下,连接的数据双方是远程数据,数据传输受网络环境的影响时缓时急,数据到达具有不可预测性[1]。传统的数据连接算法不能够在较短的时间内产生一定量的初始连接结果。为最小化用户与应用系统之间的交互时间,Ramon Lawrence[2]提出了一个通用的、可定制的Hash连接算法,能在连接的早期产生连接结果,通过向较小的关系集倾斜的刷新策略减少了I/O的次数,削减了整个执行时间。为了适应流水线式高速连接的需求,在普通的Hash连接算法基础之上产生了对称式Hash连接算法SHJ[3](Symmetric Hash Join),实现了无阻塞的连接。Tolga Urhan在SHJ对称式连接算法的基础上提出了一种分阶段的关系连接算法XJOIN[4],采用内外存切换的策略,将存储空间划分为很多小的分区,使得连接算法能够克服网络的不稳定因素,持续稳定地输出连接结果。在XJOIN算法的基础之上,MF. Mokbel等人在2004年提出了对称式刷新的HMJ算法[5],而2005年香港城市大学Tao Yufei提出了RPJ[6]算法,它利用一个评估函数实现了优化的阶段转换策略。

2 BSJ算法

本文给出一种新的连接算法BSJ(Based on Statistics Join),基于输出流上关键字的统计分析来设置刷新策略,提高了刷新的效率。

定义1 R_1 和 R_2 是两个有限的关系集合,它们有着共同的属性 A_{join},R_1 和 R_2 分别储存在远端的某数据源上,返回满足条件 $R_1.A_{join}=R_2.A_{join}$ 的 $R_1 \triangleright\triangleleft R_2$ 的结果,数据连接以如下方式进行:

(1) R_1 和 R_2 的数据通过网络以数据流的形式传输到本地数据流处理系统进行连接处理,数据在传输过程中可能遭遇网络延迟和阻塞。

(2) R_1 和 R_2 的连接结果以数据流的形式传输到目标应有系统,返回给用户。

不对关系 R_1 和 R_2 进行任何的预处理,所以元组的接受顺序是随机的。对于每个关系 $R_i(i=1,2)$,算法将已经到达的元组分别存储在内存空间 R_i^{mem} 和外存空间 R_i^{disk} 上。R_i^{mem} 和 R_i^{disk} 分别被划分为 n_{part} 个分区,划分标准是根据一个Hash函数 H 产生的$[1,n_{part}]$之间的一个整数来确定的。Hash函数的自变量是相互连接的两个关系的连接属性。接下来,用

到稿日期:2010-01-15 返修日期:2010-04-07 本文受国家自然科学基金(No. 60803160)资助。
陈 刚(1976—),男,博士,CCF会员,主要研究方向为数据流和实时数据库,E-mail:cg. hbwh@gmail. com;顾进广(1973—),男,教授,CCF会员,主要研究方向为网格技术。

$R_i^{mem}[j]$表示关系 R_i 的第 j 个内存分区。类似地，用 $R_i^{disk}[j]$ 表示关系 R_i 的第 j 个外存分区。

首先是 MM 阶段，与 HMJ 算法类似，将到达的元组与另外一个关系的对应内存中的数据进行连接，但是改进之处在于采用了一个优化的刷新策略来最大化连接结果。

其次是 reactive 阶段。该阶段将 DD 阶段和 MM 阶段混合执行，给出了一个评估函数用来根据当前的连接状况选择一个合适的阶段进行。该阶段可能包含有 2 n_{part} 个 DD 任务执行($R_1^{mem}[j] \triangleright \triangleleft R_2^{disk}[j], R_1^{disk}[j] \triangleright \triangleleft R_2^{mem}[j]$)以及 n_{part} 个 DD 任务执行($R_1^{disk}[j] \triangleright \triangleleft R_2^{disk}[j], 1 \leqslant j \leqslant n_{part}$)。每一次进入 Reactive 阶段都选择一个最高期望输出速率 Er/Et 的任务执行，Er 表示期望产生的元组个数，Et 表示任务执行的时间。该阶段沿用了 RPJ[5] 算法的评估思想，采用这种量化的粗略评估，虽然加大了算法的复杂度，但是能更准确地控制算法的运行。

在所有元组全部到达之后，要执行一个 clean-up 进程，在已经接收到的元组上执行一个全连接，返回前面遗漏的元组，以保证连接的完整性。

3 刷新策略

在 MM，MD，DD 3 个阶段的执行过程中，可以很明显地发现 MM 阶段的执行具有最高优先级。由于 MM 阶段执行速度最快，产生元组的速度也快，因此无阻塞连接算法中如何设计好 MM 阶段的执行算法，具有非常重要的意义。

当 R_1^{mem} 和 R_2^{mem} 的元组个数已经到达可分配的内存空间的极限时，需要采用将内存 R_{mem}^1 和 R_{mem}^2 中的部分数据搬移到外存上，以便后续元组可以被内存容纳。如何将内存中的部分数据搬移到外存，以腾出空间给随后到达的元组，并且最大化这一操作的效率，将是内存连接部分的主要任务。

刷新策略的设计目标就是尽量使保留在内存中的数据在下一次内存满之前，能与后续到达的元组产生最多的连接结果。

下面通过细化连接操作的过程，来分析影响 MM 阶段连接效率的因素。考查连接属性 A_{join} 域中的每个值。用 $n_i^{mem}(v)(i=1,2)$表示刷新之后内存 R_i^{mem} 中的值为 v 的元组个数，$p_i^{arr}(v)$表示下一个即将到达的元组属于关系 R_i，并且值为 v 的概率。$n_i^{arr}(v)$表示在下一次内存刷新之前 R_i 中将接收到的值为 v 的元组个数，n^{rslt} 表示在下一次内存刷新之前 MM 阶段产生的连接结果，那么有

$$n^{rslt} = \sum_{\forall v \in A^{join}} (n_1^{mem}(v) \cdot n_2^{arr}(v) + n_1^{arr}(v) \cdot n_2^{mem}(v) + n_1^{arr}(v) \cdot n_2^{arr}(v)) \quad (1)$$

上述等式是符合事实的，因为 MM 阶段的连接结果(t_1，t_2)只有以下几种产生方式：

(1)t_1 当前位于内存 R_1^{mem} 中，即 t_1 为没有被刷新出去的元组，而 t_2 将在后续到达。对应于式(1)中的 $n_1^{mem}(v) \cdot n_2^{arr}(v)$。

(2)t_1 后续到达，t_2 位于当前内存 R_2^{mem} 中，对应于式(1)中的 $n_1^{arr}(v) \cdot n_2^{mem}(v)$。

(3)t_1 和 t_2 均后续到达，不位于内存中，对应于 $n_1^{arr}(v) \cdot n_2^{arr}(v)$。

给出一个系统参数 n_{flush}，表示为了解决内存满而刷新到外存的元组个数。它也是从刷新之后开始到下一次内存满为止系统接收到的元组个数。如果用 $p_i^{arr}(v)$表示下一个元组属于关系 R_i 并且值为 v 的概率，那么 $n_i^{arr}(v)$可以表示为 $n_{flush} \cdot p_i^{arr}(v)$，这样式(1)可以写为

$$n^{rslt} = nflush \sum_{\forall v \in A^{join}} (n_1^{mem}(v) \cdot p_2^{arr}(v) + p_1^{arr}(v) \cdot p_2^{mem}(v) + n_{flush} p_1^{arr}(v) \cdot p_2^{arr}(v)) \quad (2)$$

在式(2)中，$p_i^{arr}(v)$由到达的数据的分布决定，不受算法的控制。从式(2)可以知道，为了最大化 n^{rslt}，刷新算法需要选择合适的 $n_1^{mem}(v)$和 $n_2^{mem}(v)$，使得式(3)的值达到最大。同时受到 $\sum_{\forall v \in A^{join}} (n_1^{mem}(v) + n_2^{mem}(v)) = M - n_{flush}$ 的约束，其中 M 是内存容量，也就是内存被刷新之前的元组的总量(内存是满的)，$M - n_{flush}$ 表示内存被刷新之后的元组的剩余总量。

$$y = \sum_{\forall v \in A^{join}} (n_1^{mem}(v) \cdot p_2^{arr}(v) + p_1^{arr}(v) \cdot n_2^{mem}(v)) \quad (3)$$

用一个具体的例子来说明这种情况。假设 A_{join} 的域包含两个值 1 和 2。R_1 的到达概率是 $p_1^{arr}(1)=35\%$，也就是说下一个元组属于 R_1 并且值为 1 的概率是 35%，$p_1^{arr}(2)=25\%$。相应地针对关系 R_2 有 $R_1^{mem}=10\%$，$p_2^{arr}(2)=30\%$。当内存满时，R_1^{mem} 和 n_{flush} 分别有 20 个元组，其中值为 1 和值为 2 的元组个数各有一半，目标是刷新出去 15 个元组，即 $n_{flush}=15$，那么采用该刷新策略，选择最小的 $p_2^{arr}(1)=10\%$，将其对应关系 R_1^{mem} 中值为 1 的元组移动 10 个到外存(因为 R_1^{mem} 中值位 1 的元组不足 n_{flush} 个)，这样 $n_{flush}=5$，次小的到达概率是 $p_1^{arr}(2)=25\%$，所以下一步从对应关系 R_2^{mem} 中移动 5 个值为 2 的元组到外存，完成刷新之后，$n_1^{mem}(1)=0$，$n_1^{mem}(2)=10$，$n_2^{mem}(1)=10$，$n_2^{mem}(2)=5$。

4 实验环境和结果分析

实验软件平台是 Cygwin，硬件是 2.4G 的 CPU，页面大小为 1024 个字节，每个记录的长度为 10 个字节。可用内存总量是 1000 个页面，大约可以容纳下 10 万条记录。为了模拟数据流的运行，创建了两种类型的数据流分布。第一种数据分布是协调一致的数据分布，它是指对于任何一个分区 j 有 $P(j|R_1)=P(j|R_2)$，即元组落入相同分区的概率相等。第二种是相反的数据分布，即 $P(j|R_2)=P(20-j|R_1)$。首先确定 $P(j|R_1)$，再来相应地调整 $P(j|R_2)$。例如，从元组的角度来看，假设两个关系流的公共属性为 A_{join}，两个关系流中 A_{join} 的取值都包含有值为 V 的元组，如果 R_1 和 R_2 中值为 V 的元组个数基本相等或者接近，那么可以认为 R_1 和 R_2 是分布协调一致的。如果两个 R_1 和 R_2 中取值为 V 的元组个数相差较远，那么认为 R_1 和 R_2 是分布不一致的。

从图 1 可以明显地看到 BSJ 算法优于 RPJ 算法，因为可靠网络传输情况下主要元组均由 MM 阶段产生，而刷新策略则是决定 MM 阶段效率的关键。所以此种情况下，BSJ 性能最好。

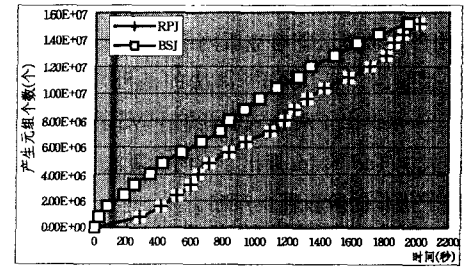

图 1 数据分布协调时的性能

(下转第 160 页)

guity resolution[D]. Pennsylvania:Pennsylvania,1998
[2] Nigam K,Laferty J,McCallum A. Using Maximum Entropy for Text Categorization[C]// Workshop on Machine Learning for Information Filtering. 1999:61-67
[3] 李荣陆,王建会,陈晓云,等. 使用最大熵模型进行中文文本分类[J]. 计算机研究与发展,2005,42(1):94-102
[4] 李素建. 汉语组块计算的若干研究[D]. 北京:中国科学院计算技术研究所,2002
[5] Berger A L,Pietra S A D,Pietra V J D. A Maximum Entropy Approach to Natural Language Processing[J]. Association for Computational Linguistics,1996,22(1):39-71
[6] Jelinek F, Mercer R L. Interpolated Estimation of Markov Source Parameters from Sparse Data[C]// Proceedings of the Workshop on Pattern Recognition. Practice,1980:381-398
[7] 于浚涛. 基于最大熵的汉语介词短语自动识别[D]. 大连:大连理工大学,2006
[8] Zhai C,Lafferty J. A Study of Smoothing Methods for Language Models Applied to Information Retrieval[J]. ACM Transactions on Information Systems,2004,22(2):179-214
[9] Gao J,Goodman J,Li M,et al. Toward a Unified Approach to Statistical Language Modeling for Chinese[J]. ACM Transactions on Asian Language Information Processing,2002,1(1):3-33
[10] 黄永文,何中市. 基于全局折扣的统计语言模型平滑技术[J]. 重庆大学学报:自然科学版,2005,28(8):51-55
[11] 黄建中,王肖雷. Katz 平滑算法在中文分词系统中的应用[J]. 计算机工程,2004,增刊(1):370-372
[12] 许家金. 兰开斯特汉语语料库介绍[EB/OL]. http://nlp.org/,2006
[13] Yang Xiaojun. Survey and Prospect of China's Corpus-based Researches[C]// The Corpus Linguistics Conference. Lancaster University(UK). 2003

(上接第 137 页)

[6] Dou D, LePendu P. Ontology-based Integration for Relational Databases[C]// Proceedings of the 2006 ACM Symposium on Applied Computing. Dijon,France,2006:461-466
[7] Wang Jin-peng, Lu Jian-jiang, Zhang Ya-fei, et al. Integrating heterogeneous data source using ontology[J]. Journal of Software,2009,4(8):843-850
[8] 吕艳辉,马宗民,王玉喜. 基于关系数据库的 OWL 本体构建方法研究[J]. 计算机科学,2009,36(7):153-156
[9] Laborda C P, Conrad S. Bringing Relational Data into the Semantic Web using SPARQL and Relational. OWL[C]// Proceedings of the 22nd International Conference on Data Engineering Workshops. Atlanta,USA,2006:55
[10] Erling O,Mikhailov I. Integrating Open Sources and Relational Data with SPARQL[C]// The 5th European Semantic Web Symposium and Conference. Tenerife,Spain,2008:838-842
[11] Weiske C, Auer S. Implementing SPARQL Support for Relational Databases and Possible Enhancements[C]// Proceedings of the 1st Conference on Social Semantic Web. Leipzig,Germany,2007:69-80
[12] Halevy A. Answering queries using views:A survey [J]. Very Large Data Bases Journal,2001,10(4),270-294
[13] Abiteboul S,Duschka O M. Complexity of answering queries using materialized views[C]// Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. Seattle,US,1998:254-263
[14] Beeri C,Levy A,Rousset M C. Rewriting queries using views in description logics[C]// Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS). Tuscon,Arizona,1997:99-108
[15] Baader F,Hollander B. Kris:Knowledge representation and inference system[J]. SIGART Bulletin,1991,2(3):8-14
[16] Levy A,Rajaraman A,Ordille J. Querying heterogeneous information sources using source descriptions[C]// Proceedings of the 22th International Conference on Very Large Data Bases. Bombay,India,1996:251-262
[17] Pottinger R,Halevy A. MiniCon:A Scalable Algorithm for Answering Queries Using Views[J]. The VLDB Journal,2001,10(2/3):182-198

(上接第 133 页)

图 2 显示的则是两个关系数据流分布不协调一致的情况,BSJ 算法能显示出其优越性。这是由于两个关系流分布不协调一致,也就是说在 R_1(或者 R_2)中可能存在一些关键字的大量取值,在 R_2 中与此相等的关键字含量过低,因而通过算法能够准确地找到关系流中出现概率较低的元组及其相应分区。

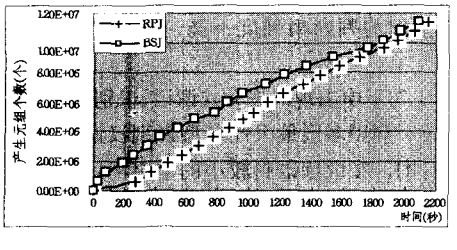

图 2 数据分布不协调时的性能

结束语 提出了一个新的内存刷新策略,再对数据流上的数据频率进行近似的统计分析,将分析结果应用于关系连接的输出流上,很好地反映了输入流中的数据分布情况,提高了刷新策略的准确性。

参 考 文 献

[1] Lawrence R. Early Hash Join:A configurable algorithm for the efficient and early production of join results[C]// VLDB. 2005:841-852
[2] Wilschut A N,Apers E M G. Pipelining in query execution[C]// Proc. of the International Conference on Databases,Parallel Architectures and their Applications. Miami,USA,March 1990
[3] Urhan T,Franklin M J. XJoin:Getting Fast Answers from Slow and Burst Networks[R]. CS-TR-3994. Computer Science Department,University of Maryland,1999
[4] Mokbel M F,Lu Ming,Aref W G. Hash-Merge Join:A Non-blocking Join Algorithm for Producing Fast and Early Join Results[C]// Proceeding of the 20th International Conference on Data Engineering. Washington:IEEE Computer Society,2004:251-263
[5] Tao Yufei,Yiu M L,Papadias D,et al. RPJ:producing fast join results on streams through rate-based optimization[C]// Proceedings of the 24th ACM SIGMOD International Conference on Management of Data. New York:ACM press,2005:371-382