基于合格标记的数据操作并行执行算法的研究

王意洁,王勇军,胡守仁
(国防科技大学计算机学院并行与分布处理国家重点实验室,湖南长沙 410073)

摘 要: 本文针对面向对象数据库及其查询的特点,提出了一种基于合格标记的数据操作并行执行算法,理论分析和模拟结果都验证了它的实用性和有效性.

关键词: 面向对象数据库; 数据操作; 并行
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2000) 08-0028-04

Research of the Mark-based Data Operation Parallel Executing Algorithm

WANG Yi-jie, WANG Yong-jun, HU Shou-ren
(National Key Laboratory For Parallel & Distributed Processing, Institute of Computer, NUDT, Changsha 410073, China)

Abstract: According to the features of the object-oriented database and its query, the mark-based data operation parallel executing algorithm is proposed, the theoretical analysis and simulation results show that it is an efficient practical algorithm.

Key words: object-oriented database; data operation; parallel

1 引言

目前,国际上已经有几个研究机构涉及了并行面向对象数据库(POODB)这个研究领域,但还没有研制出比较成熟的原型系统.从现在的研究情况[1~8]来看,虽然研究者们的思路各不相同,但他们的研究存在一些共同的问题,基本上是借鉴并行关系数据库(PRDB)的思想和技术,并根据面向对象数据库(OODB)的特点进行适当的修改和扩充.通过深入分析,我们发现,OODB 与关系数据库(RDB)之间存在着本质区别,因此,对 PRDB 的思想和技术加以"改造利用"的方法是不能充分利用 OODB 及其查询的本质特点的.

针对 OODB 及其查询的本质特点,结合查询处理的分阶段执行策略[9]和基于对象类的混合式数据放置策略[9,10],本文提出了一种基于合格标记的数据操作并行执行算法,并给出了性能评价的结果.

分阶段执行策略[9]的主要思想是:第一步,通过在对象类之间处理和传播对象标识 *Oid*,以 *Oid* 的形式确定"合格"的对象,也就是查询的前期结果;第二步,将系统定义的或用户定义的函数作用于第一步得到的前期结果,从而产生最终结果.

数据放置策略的目标应该是:各处理机之间负载均衡,各处理机之间的通信开销达到最小,避免不必要的磁盘 I/O 请求.基于对象类的混合式数据放置策略[9,10]包括混合式数据划分策略和基于对象类的数据分配策略.混合式数据划分策略兼顾了水平划分策略和垂直划分策略二者的优点,它的具体思想是:首先,对数据库中的对象进行垂直划分,产生两种数据子集:一种是由二元组(*Oid*,属性值)构成的属性值集;另一种是由二元组(*Oid*, *Oid*)构成的关系集,它们分别存贮对象的属性值和对象与其它对象之间的关系;然后,对两种数据子集基于 *Oid* 进行水平划分,得到混合式数据子集.基于对象类的数据分配策略[9,10]对混合式数据划分策略的结果进行合理分配.

2 基于合格标记的数据操作并行执行算法

为了开发操作内的并行性,POODB 的研究者提出了基于传统数据放置策略的并行数据操作算法[3~7],为了便于区分,我们称其为传统的并行数据操作算法.虽然传统的并行数据操作算法比较有效地开发了操作内的并行性,但是,对 OODB 本质特点的考虑不够充分,而且磁盘 I/O 量和网络传输量都较大.针对传统并行数据操作算法的不足,提出了基于合格标记的数据操作并行执行算法,它与传统并行数据操作算法的主要区别是:利用"合格标记"来记录数据操作的结果,而不是利用数据拷贝来记录操作的结果.

对于选择操作来说,"合格"意味着对象满足选择条件,那么,如何有效地进行"合格标记"呢?在一个由 N 个处理机结点通过高速互联网连接的无共享结构中,每个结点的磁盘中都有一张用于记录对象的"合格"标记的表格——标记表,存储于该结点的磁盘上的每一个对象都与标记表中的一项相对应,标记表中的每一项由对象的 *Oid* 和相应的标记构成.这个标记表由若干个子表构成,每个子表对应于存储在该结点的磁盘上的一个混合式数据子集.

收稿日期:1999-05-11;修回日期:2000-01-15
基金项目:国家自然科学基金(No. 69903011);部委级预研项目(15.4.1)

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

从查询优化的角度考虑,在查询处理时,优先进行选择操作和投影操作,所以我们对连接操作的讨论也是在选择操作和投影操作的基础上进行的. 对于连接操作来说,"合格"不仅意味着对象满足选择条件,而且对象之间存在一定的关系. 所以,我们利用由两个对象的 *Oid* 构成的二元组 *(Oid, Oid)* 作为连接操作的"合格标记". 我们称这样的二元组为关联模式,所有的关联模式构成关联模式集,它也象标记表一样分布于各个处理机结点上.

查询的基本数据操作有很多,其中,选择(selection)、投影(projection)和连接(join)是三个比较重要的操作,因此,我们主要研究这三个操作的并行执行. 为了便于比较,我们同时给出了传统的数据操作并行执行算法和基于合格标记的数据操作并行执行算法的描述.

目前,基于无共享结构所进行的有关 POODB 的研究都较充分地考虑了 CPU 处理、磁盘 I/O 和互联网络通信三者的重叠,并进一步考虑了网络发送与网络接收的重叠. 为了有效实现这种重叠,数据操作并行执行算法中在每个结点上分配一个接收缓冲区,为每个磁盘分配一个读盘缓冲区和一个写盘缓冲区,为每个远程结点分配一个发送缓冲区.

给定类 *C* 和类 *D*,每个类都有 *m* 个性质,一个无共享结构由 *N* 个处理机结点通过高速互联网连接. 若采用传统的数据放置策略,那么 *C* 和 *D* 在所有结点的磁盘上均匀划分, *Ci* 和 *Di* 分别表示存储于结点 *i* 的磁盘上的 *C* 和 *D* 的对象集合, $i = 1, 2, \cdots, N$;若采用基于对象类的混合式数据放置策略,那么 *C* 和 *D* 也在所有结点的磁盘上均匀划分, *Ci* 和 *Di* 分别表示存储于结点 *i* 的磁盘上的 *C* 和 *D* 的混合式数据子集, $i = 2, \cdots, N$,那么, $Ci = \{Cij \mid j = 1, 2, \cdots, m\}$, $Di = \{Dij \mid j = 1, 2, \cdots, m\}$,其中 *Cij* 和 *Dij* 分别表示 *C* 和 *D* 的对应于性质 *j* 的数据子集(属性值集或关系集)中存储于结点 *i* 的磁盘上的部分, $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, m$. 不论采用哪种数据放置策略,操作的执行结果可以分散于各结点中,不必收集于某一个结点.

2.1 选择

我们首先给出选择操作的问题描述:给定类 *C*,从 *C* 中选取满足条件 Cond 的所有对象,其中条件 Cond 涉及 *C* 的一个或多个性质. 这里,设 *NC* 表示条件 Cond 所涉及的 *C* 的性质的数目.

传统的选择操作并行执行算法的具体思想是:每个结点 *i* 从本地磁盘中读取 *Ci*,每次一页. 依次处理每一个读入的对象:判断它是否满足条件 Cond,若满足,则将它放入结果的写盘缓冲区,等待后续处理. 当写盘缓冲区被填满时,将它写回磁盘.

基于合格标记的选择操作并行执行算法的具体思想是:

Step-1　每个结点 *i* 从本地磁盘中读取与 *Ci* 相对应的标记子表.

Step-2　每个结点 *i* 从本地磁盘中读取 *Cij* 各一页, $j \in \{j_1, j_2, \cdots, j_{NC}\}$,共 *NC* 页, $\{j_1, j_2, \cdots, j_{NC}\}$ 表示条件 Cond 所涉及的性质集合. 这样,从磁盘读入的不是一个个完整的对象,而是每个对象中与 Cond 有关的 *NC* 个性质,这 *NC* 个性质只拥有一个对象的部分信息,我们称它们为部分对象. 依次处理每一个读入的部分对象:判断它是否满足条件 Cond,若满足,则在标记子表中作"合格"标记. 根据需要,不断地从磁盘读入 *Cij* 的后续页,直到所有部分对象均被处理完为止.

Step-3　将与 *Ci* 相对应的标记子表写入本地磁盘.

2.2 投影

我们首先给出投影操作的问题描述:给定类 $C, Ai\ (i = 1, 2, \cdots, m)$ 表示 *C* 的 *m* 个性质,设 *o* 是属于 *C* 的一个对象,它的 *m* 个性质值为 $ai\ (i = 1, 2, \cdots, m)$, $\{Aj1, Aj2, \cdots, Ajp\} \subseteq \{A1, A2, \cdots, Am\}$,那么 *o* 在 $Aj1, Aj2, \cdots, Ajp$ 上的投影为 $\{aj1, aj2, \cdots, ajp\}$, *C* 在 $Aj1, Aj2, \cdots, Ajp$ 上的投影为 *C* 中每个对象在 $Aj1, Aj2, \cdots, Ajp$ 上的投影构成的集合.

传统的投影操作并行执行算法的具体思想描述如下:每个结点 *i* 从本地磁盘中读取 *Ci*,每次一页. 依次处理每一个读入的对象:将每个对象的投影放入结果的写盘缓冲区,等待后续处理. 当写盘缓冲区被填满时,便将它写回磁盘.

在基于对象类的混合式数据放置策略中,每个类是按照该类在每个性质上的投影构成的集合的方式存储的,也就是说,在数据放置中已经蕴含了投影操作. 所以,可以认为基于合格标记的投影操作并行执行算法的响应时间趋于零.

2.3 连接

我们首先给出连接操作的问题描述:给定类 *C* 和类 *D*,找出所有满足连接条件的对象二元组 *(COi, DOj)*. 其中, *COi* 和 *DOj* 分别表示类 *C* 和类 *D* 中的对象. OODB 与 RDB 的许多本质区别导致了 OODB 中的连接与 RDB 中的连接也不尽相同. 文献[11]中对比了 OODB 和 RDB 中的连接,并根据连接所涉及的两个类之间的关系将 OODB 中的连接分为两类:隐式连接和显式连接. 隐式连接中的两个类之间存在关系(relationship). 显式连接中的两个类之间不存在关系.

现有的 OODB 并行连接算法[3~7]大多是基于散列的算法,我们提出的基于合格标记的连接操作并行执行算法也是以散列为基础的. 下面,我们分别讨论隐式连接和显式连接的并行执行.

(1) 隐式连接

传统的隐式连接并行执行算法的具体思想是:

Step-1　建表(building)阶段. 具体而言,主要是指每个结点将本结点磁盘上的 *C* 类对象的有关信息(包括选择操作的结果,连接性质值,检索性质值等)传送到与其相关联的 *D* 类对象所在的结点上,并根据接收到的信息为以后的连接建必要的表格(如以连接性质为关键码建 Hash 表).

Step-2　探询(probing)阶段. 具体而言,主要是指根据所建表格匹配符合连接条件的 *C* 类对象和 *D* 类对象,并将结果存入磁盘.

传统的隐式连接并行执行算法的执行过程中,结点之间不仅传送 *Oid* 而且传送投影后的大量描述信息,同时大量的描述信息不仅要从磁盘读出而且最后又要存入磁盘. 通过分析,我们发现算法的网络传输量大,磁盘 I/O 量大,磁盘占用空间大,这些都直接影响了连接操作的执行效率. 基于合格标记的隐式连接并行执行算法通过关联模式(即:由两个对象的 *Oid* 构成的二元组,如: *(c2, d5)*)来记录连接结果,这就避免

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

了大量描述信息从磁盘读出和写入以及在网络上传送. 在网络上只需传送 *Oid* 构成的关联模式,同时也减少了磁盘占用空间,由于连接结果是通过关联模式记录的,而相关联的两类对象往往位于不同的结点上,所以既需要 *C* 类对象向 *D* 类对象所在结点发送 *Oid* 信息并在 *D* 类对象所在结点上确定关联模式,又需要 *D* 类对象向 *C* 类对象所在结点发送 *Oid* 信息并在 *C* 类对象所在结点上确定关联模式. 选择操作的结果是一组 *Oid*,可将其视作选择结果表. 下面,给出基于合格标记的隐式连接并行执行算法的描述:

Step-1 建表阶段. 具体而言,每个结点从磁盘读出类 *C* 选择结果表,以 *Oid* 为关键码建选择结果 Hash 表;每个结点从磁盘读出类 *D* 的选择结果表,以 *Oid* 为关键码建选择结果 Hash 表.

Step-2 探询建表阶段. 具体而言,每个结点从磁盘读出类 *C* 的关于连接性质的部分数据子集,通过探询类 *C* 的选择结果 Hash 表,判断类 *C* 部分数据子集中的二元组的合格性,并将合格的二元组传送到指定的结点上;同时,根据接收到的信息,为类 *D* 建立必要的表格(如以连接性质为关键码建连接 Hash 表).

Step-3 探询阶段. 具体而言,每个结点从磁盘读出类 *D* 的关于连接性质的部分数据子集,通过探询类 *D* 的选择结果 Hash 表,判断类 *D* 部分数据子集中的二元组的合格性,若二元组合格,则继续探询连接 Hash 表,对符合连接条件的 *C* 类对象和 *D* 类对象进行连接,形成关联模式集,并将类 *D* 的关联模式集写入磁盘,同时发送到 *C* 类对象所在结点;每个结点将接收到的关联模式写入磁盘.

(2) 显式连接

传统的显式连接并行执行算法的具体描述如下:

Step-1 建表(building)阶段. 具体而言,主要是指每个结点将本结点磁盘上的 *C* 类对象的有关信息(包括选择操作结果,连接性质值,检索性质值等)传送到所有其它 $N-1$ 个结点上,并根据本结点上的 *C* 类对象的有关信息和接收到的信息为以后的连接建立必要的表格(如以连接性质为关键码建 Hash 表).

Step-2 探询(probing)阶段. 具体而言,主要是指根据所建表格匹配符合连接条件的 *C* 类对象和 *D* 类对象,并将结果存入磁盘.

基于合格标记的显式连接并行执行算法与基于合格标记的隐式连接并行执行算法很相似,它的具体描述如下:

Step-1 建表阶段. 每个结点从磁盘读出类 *C* 的选择结果表,并以 *Oid* 为关键码建选择结果 Hash 表;每个结点从磁盘读出类 *D* 的选择结果表,并以 *Oid* 为关键码建选择结果 Hash 表.

Step-2 探询建表阶段. 每个结点从磁盘读出类 *C* 的关于连接性质的部分数据子集,通过探询类 *C* 的选择结果 Hash 表,判断类 *C* 部分数据子集中的二元组的合格性,并将合格的二元组传送到其它所有 $N-1$ 个结点上;同时,根据接收到的信息,为类 *D* 建立必要的表格(如以连接性质为关键码建连接 Hash 表).

Step-3 探询阶段. 每个结点从磁盘读出类 *D* 的关于连接性质的部分数据子集,通过探询类 *D* 的选择结果 Hash 表,判断类 *D* 部分数据子集中二元组的合格性,若合格,则继续探询连接 Hash 表,对符合连接条件的 *C* 类对象和 *D* 类对象进行连接,形成关联模式集,并将类 *D* 的关联模式集写入磁盘,同时发送到 *C* 类对象所在结点;每个结点将接收到的关联模式写入磁盘.

3 性能评价

由于实验环境等客观条件的限制,我们采用理论分析与模拟实验相结合的方法来对比分析基于合格标记的数据操作并行执行算法与传统的数据操作并行执行算法的性能.

我们编制模拟测试程序计算传统的数据操作并行执行算法和基于合格标记的数据操作并行执行算法的响应时间. 模拟的工作环境是:多台 SGI Challenge 服务器(MIPS R4400 芯片,128MIPS)通过网络互联,网络启动时间为 0.05ms,传输率为 75Mb/s.

通过改变重要参数的取值,观察两种算法响应时间比值 ratio(传统并行执行算法的响应时间/基于合格标记的并行执行算法的响应时间)的变化. 由模拟测试结果可以看出:

- 在一般情况下,基于合格标记的数据操作并行执行算法明显优于传统的数据操作并行执行算法;
- ratio 与对象的大性质数目成正比,趋于线性关系;
- ratio 与选择率成正比;
- ratio 与大性质的大小成正比,趋于线性关系.

4 结论

数据操作的执行效率直接影响到数据库系统的性能,数据操作的并行执行对于并行数据库系统而言是至关重要的,数据放置与数据操作的并行执行是息息相关的. 基于合格标记的数据操作并行执行算法利用"合格标记"来记录数据操作的结果,而不是利用数据拷贝来记录操作的结果,从而有效地降低了磁盘 I/O 量和网络传输量. 它比较依赖于最初的数据放置,而且不涉及结果数据的放置,这不仅减少了对后续的数据操作的影响,而且减少了数据偏斜产生的可能性.

从模拟测试结果可以看出,在一般情况下,基于合格标记的数据操作并行执行算法优于传统的数据操作并行执行算法;当数据操作涉及的对象的大性质数目越多,大性质占用的存储空间越多,基于合格标记的数据操作并行执行算法就越优于传统的数据操作并行执行算法(反映在算法的响应时间比值 ratio 越大). 这充分说明了基于合格标记的数据操作并行执行算法抓住了 OODB 及其查询的本质特点,具有一定的实用性和有效性.

参考文献:

[1] Michael J. Carey, David J. DeWitt, Michael J. Franklin, and et al. Shoring up persistent applications [A]. In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data [C], 1994, 124 - 137.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

[2] David J. DeWitt, Jeffrey F. Naughton, John C. Shafer, and Shivakumar Venkataraman. Parsets for parallelizing OODBMS traversals: Implementation and performance [A]. In Proceedings of the Third International Conference on Parallel and Distributed Information Systems [C], 1994, 111 - 120.

[3] Russell F. Haddleton. An implementation of a parallel object oriented database system [R]. Technical Report, University of Virginia, 1995.

[4] Russell F. Haddleton. Parallel set operations in complex object oriented queries [D]. PhD thesis, University of Virginia, 1998.

[5] N. Bassiliades, I. Vlahavas. Hierarchical query execution in a parallel object-oriented database system [J]. Parallel Computing, 1996, 22(3): 1017 - 1048.

[6] Michael Gesman. Mapping a Parallel Complex-Object DBMS to Operating System Processes [A]. In Proceedings of the Second International Euro-Par Conference [C]. Euro-Par '96, 1996, 852 - 861.

[7] Daniel F. Lieuwen, Daoid J. Dewitt, and Manish Mehta. Parallel Pointer-based Join Techniques for Object-Oriented Databases [J]. In Proc. of the Second Int'l Conf. on Parallel and Distributed Information Systems, January 1993: 172 - 181.

[8] Leung, C. H. c and Taniar, D. Parallel Query Processing in Object-Oriented Databases Systems [A]. Proceedings of the Sixth Austrakasian Database Conference (ADC '95) [C]. Adelaide, Australia, January 1995: 119 - 131.

[9] 王意洁. 面向对象数据库的并行查询处理与事务管理 [D]. 博士学位论文, 国防科技大学研究生院, 1998, 10.

[10] 王意洁, 王勇军, 胡守仁. 并行面向对象数据库中的基于对象类的混合式数据放置策略 [J]. 计算机学报, 1998, 21(增刊): 178 - 183.

[11] 何炎祥, 郑振楣, 石树刚. 面向对象数据库 [M]. 武汉大学出版社, 1995, 6.

作者简介:



王意洁　1971 年 9 月生, 1998 年获博士学位, 现为国防科技大学计算机学院并行与分布处理国家重点实验室助理研究员, 研究领域为数据库和并行分布处理技术.



王勇军　1971 年 1 月生, 1998 年获博士学位, 现为国防科技大学计算机学院网络与信息安全研究所助理研究员, 研究领域为虚拟现实、数据库和并行分布处理技术.

(上接第 34 页)

参考文献:

[1] Lefelhocz C, Lyles B, Shenker S, et al. Congestion control for best-effort service: Why we need a new paradigm [J/OL]. IEEE Network, 1996, January/February: 10 - 19.

[2] Almquist P. Type of service in the internet protocol suite [J/OL]. Internet RFC 1349, July 1992.

[3] Braden R. Integrated services in the internet architecture: An overview [J/OL]. Internet RFC1633, July 1994.

[4] Braden B. Resource ReServation Protocol (RSVP)-Version 1 functional specification [J/OL]. Internet RFC 2205, September 1997.

[5] Blake S, Black D, Carlson M, et al. An architecture for differentiated services [DB/OL]. Internet Draft < draft-ietf-diffserv-arch-02. txt >. October 1998.

[6] Black S, Bernet Y, Binder J, et al. A Framework for Differentiated Services [DB/OL]. Internet Draft < draft-ietf-diffserv-framework-01. txt >. October 1998.

[7] Jacobson V, Kerals M. Congestion avoidance and control [A]. In Proceedings of SIGCOMM '88 [C], Stanford, CA, Aug. 1998, ACM.

[8] Allman M, Paxson V, Stevens W, et al. TCP Congestion Control [DB/OL]. Internet Draft < draft-ietf-tcpimpl-cong-control-03. txt >. December, 1998.

[9] Demers A, Keshav S, Shenker S. Analysis and simulation of a fair queueing algorithm [A]. In Proc. ACM SIGCOMM '89 [C], 1989: 3 - 12.

[10] Floyd S, Jacobson V. Random Early Detection Gateways for Congestion Avoidance [J]. IEEE/ACM Transactions on Networking, 1998, 1(4): 397 - 413.

[11] Alt-Hellal O, Altman E, Elouadghiri D, et al. Performance of TCP/IP: the case of Controlled Sources [A]. International Conference for Computer Communications-ICCC '97 [C], Cannes, France, November 19-21, 1997

[12] Bonald T. Comparion of TCP Reno and TCP Vegas via Fluid Approximation [R]. In Workshop on the modeling of TCP, December 1998.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net