传统事务模型的并行性*)

The Parallelism of Traditional Transaction Model

张志强[1] 李建中[2] 周立柱[1]

(清华大学计算机科学与技术系 北京 100084)[1]

(哈尔滨工业大学计算机科学与工程系 哈尔滨 150001)[2]

Abstract Transaction is a very important concept in DBMS, which has several features such as consistency, atomicity, durability and isolation. In this paper, we first analyze the parallelism of traditional transaction model. Next we point out that we can investigate more parallelism with a high parallel processing manner underlying multi-processors parallel structures. We will then compare the influence of two different software architectures on database system parallelism.

Keywords Transaction, Parallelism, Transaction model, System software architecture

1 前言

在数据库管理系统中事务是一个非常重要的概念,它具有一致性(Consistency)、原子性(Atomicity)、持久性(Durability)和独立性(Isolation)等特点。为了在保证事务的这几个特性的同时又能使系统具有很高的吞吐量,人们提出了很多的并发控制技术,其中最基本的技术为锁技术[1,2]和时间印技术[3]。

锁技术是建立在可串行化理论的基础上,通过对数据的互斥存取来保证可串行性,即当一个事务存取一个数据项时不允许其它事务修改这个数据项。每个事务在存取一个数据项之前必须获得这个数据项上的锁;而时间印协议不需要锁的概念,对于系统中的每个事务,我们都为其分配一个时间印,不同的事务具有不同的时间印。由于时间印协议使得冲突操作按照时间印的顺序被处理,因此时间印协议保证了冲突可串行性。通过采用不同策略人们又提出了一些其它的技术来提高系统的并发性。如,验证技术[4]、多种并发控制粒度[5]、多版本技术[6]等。在分布式数据库和多数据库中对于事务处理人们也作了很多的工作[8~10]。上面的工作都是建立在传统事务模型的基础上,为了满足应用的需求和提高系统的性能,人们又提出了新的多级事务模型[11~13],但是关于并行数据库的事务处理几乎还没有这方面的工作。本文对传统事务模型的并行性作了分析,并从系统软件结构的角度探讨了进一步开发传统事务模型的并行性。

本文首先分析了传统事务模型的并行性,给出了事务间和事务内的并行性概念,并对系统所能够支持的并行性进行了分类,最后讨论了系统结构对系统所能提供的并行性的影响。

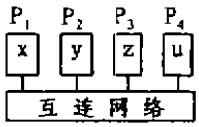
2 传统事务模型的并行性

在这里我们将传统事务模型的并行性分为事务内的并行性和事务间的并行性两类,下面我们分别对其进行讨论。

2.1 事务内的并行性

事务内的并行性包括操作的并行性与操作之间的并行性两种。前者主要是指具体数据库操作的并行执行,如,数据被分配在多个处理结点上,当执行一个查找操作时,多个处理结点可以同时在各自的局部数据库中查找相应的记录,从而大大提高了响应时间。当前对并行数据库操作的并行算法研究主要集中于对数据库操作的并行性开发上。操作间的并行主要是指各个数据库操作之间的并行执行。下面我们通过一个例子来看一下操作间的并行性。

例 1 数据库中的数据项:x、y、z 和 u 分别存储在 P_1、P_2、P_3 和 P_4 四个处理结点上,如图 1 所示。



事务 T
t_1: R(x) /*R(x)*/
t_2: y=y+x /*W(y)*/
t_3: R(z) /*R(z)*/
t_4: u=u-2*z /*W(u)*/

图 1

在事务 T 中一共有四个数据库操作,我们可以看

*)本课题得到国家自然科学基金(69373024),国家 863 高科技计划基金(863-511-9511-002)的资助。张志强 讲师,博士研究生,主要的研究方向有并行数据库、Web 数据库、数据挖掘等。

到操作R(x)只与W(y)有关联，而与R(z)和W(u)没有关联，即R(x)可以与R(z)、W(u)并行执行，W(y)与R(z)、W(u)可以并行执行，而且并不影响整个事务执行的正确性。通过上面的例子我们也看到，多处理机系统为事务操作间的并行性开发提供了可能。目前对于开发这种事务操作间的并行性的研究还很少，这里涉及到事务操作间的相关性判定问题。

2.2 事务间的并行性

实际应用中，系统里往往同时存在多个用户应用程序，而每个应用程序里又包含多个事务，每个事务里又含有多个SQL语句。这里我们将事务间的并行性分为两类，第一类是指不同应用程序的事务之间的并行性；第二类为同一个应用程序内不同事务间的并行性。前面提到的并发控制技术主要针对事务间的并行性，实际应用中只是考虑了第一类事务间的并行性，而没有考虑第二类事务间的并行性。本文对处于同一应用程序内不同事务的并行性进行了讨论，在第4节指出利用进程的概念我们可以获得第二类事务间的并行性。

3 系统的并行性

这里提到系统的并行性是指一个系统所能支持的最大事务并行性。系统的并行性不仅与系统的硬件体系结构有关而且与系统所采用的软件结构也有密切的关系。例如，在单处理结点情况下，系统只能是顺序执行用户应用程序，即使系统采用分时的方法也只能提供不同用户应用程序之间的并发执行。在多处理结点的并行结构下，如例1所示，除了能获得事务间的并行性之外，我们还能够获得事务内的并行性。在单处理结点的情况下SQL语句都是顺序执行的，而在拥有多处理结点的并行结构的情况下一般都采用并行数据操作算法来实现SQL语句中的操作(操作并行性)，所以在下面的分类中不考虑SQL语句本身的并行性问题；另外，对于非嵌入的交互式SQL系统可以把它当作一个简单的事务来处理，所以我们在这里不考虑这种情况。因此，我们可以得到系统的并行性的8种组合，详见表1。

表1

	应用程序之间执行顺序	程序内各事务之间执行顺序	事务内SQL语句之间执行顺序
1	顺序执行	顺序执行	顺序执行
2	顺序执行	顺序执行	并行执行
3	顺序执行	并行执行	顺序执行
4	顺序执行	并行执行	并行执行
5	并行执行	顺序执行	顺序执行
6	并行执行	顺序执行	并行执行
7	并行执行	并行执行	顺序执行
8	并行执行	并行执行	并行执行

从表1中我们可以看到，第8种组合是最佳的组合，因为它能够提供最大的并行性。下面我们将看到通过对系统软件结构的调整，可以使系统的并行性得到提高。

4 传统事务模型的执行方式

现在来看一下传统事务模型的执行方式，我们分单处理结点与多处理结点两种情况进行讨论，并且在多处理结点的情况下，提出了一种高并行性的执行方式。为了方便起见，讨论是在例2的基础上进行的，并且假设以一个SQL语句的执行时间为一个系统分时的时间单位。

例2 下面是两个用户的应用程序，分别含有两个事务。

```
应用程序1
{
 Transaction A
 {
  SQL 1
  SQL 2
 }//End of Transaction A
 Transaction B
 {
  SQL 3
  SQL 4
 }//End of Transaction B
}//End of Application 1

应用程序2
{
 Transaction C
 {
  SQL 5
  SQL 6
 }//End of Transaction C
 Transaction D
 {
  SQL 7
  SQL 8
 }//End of Transaction D
}//End of Application 2
```

4.1 单处理结点情况

在单用户的情况下，应用程序1与应用程序2之间只能顺序执行，谁先提交给系统，系统就先执行谁。另一个只能等到前一个执行完毕后才能开始执行。假设应用程序1先于应用程序2提交给系统，则一种可能的执行方式如图2所示。我们可以看到，应用程序2的事务C需要等待4个时间单位，即在应用程序1的事务A和事务B执行完后才能开始执行。

在采用分时技术来模拟多用户的情况下，我们可以获得不同应用程序的事务之间的并发执行，一种可能的执行顺序如图3所示。从图3中我们可以看到，应用程序2的事务C不需要等到应用程序1中的事务A和事务B都执行完后再开始执行，而是在事务A执行完SQL1后就可以开始执行。同时我们也可以看到系统执行这两个应用程序的总体时间并没有缩短，只是通过延长一些事务的响应时间来获得较短的等待时间，提高系统的吞吐量。

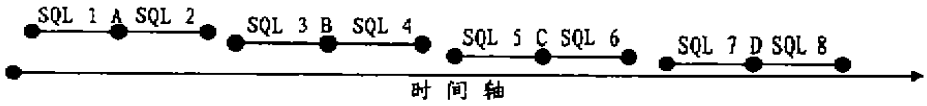


图 2　单用户情况下的执行过程

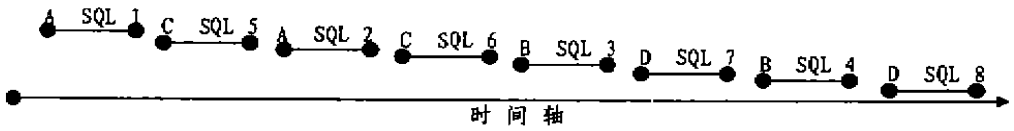


图 3　分时情况下的执行过程

4.2　多处理结点并行结构情况

在多处理结点的并行结构下，可以利用多个处理结点来获得更高的系统并行性，改善系统的响应时间。目前所采用的执行方式都只考虑了第一类事务间的并行性，而没有考虑第二类事务间的并行性，这时例 2 的一种可能的执行过程如图 4 所示。从图 4 中我们可以看到，应用程序 2 的事务 C 不必等到应用程序 1 中的事务 A 的 SQL1 执行完就可以开始执行，因为这两个事务可能在不同的处理结点上并行执行。同时还可以采用并行操作算法来实现对 SQL 语句的并行执行，在图 4 中我们用比上面图 3 中短一些的线段来表示 SQL 语句执行时间。系统总体的处理时间比单处理结点情况大大缩短了。我们还可以看到，事务 B 只能在事务 A 执行完毕后才能开始执行，同时只有在事务 C 执行完毕后事务 D 才能开始执行。

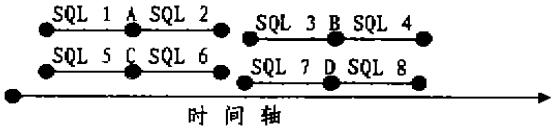


图 4　当前采用的执行方式

为了获得更高的系统并行性，进一步改善系统的吞吐率和系统的响应时间，针对上面提到的问题，我们做如下改进：在系统中把事务作为一个基本的独立执行单元，而不仅仅是作为一个基本的管理单元和逻辑运行单元，例如，用一个进程来实现一个事务。这样，不仅能够获得第一类事务间的并行性，而且还能获得第二类事务间的并行性，使系统能够支持第 7 种或第 8 种并行性。图 5 描述了例 2 的一种可能的执行的过程，在这里我们假设事务内的 SQL 语句之间采用顺序执行方式。从图 5 中我们看到，应用程序 1 中的事务 A 和事务 B 实现了并行执行，应用程序 2 中的事务 D 也不必等到事务 C 执行完毕后再开始执行。系统总体的响应时间又进一步缩短了。这种执行方式的具体实现细节见文[15,16]。

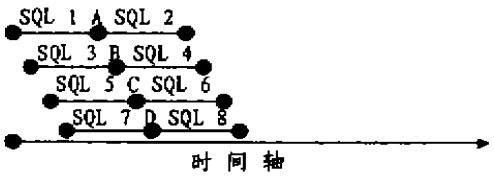


图 5　高并行性执行方式

5　系统软件结构对系统并行性的影响

这里系统结构采用如图 6 所示的逻辑上带一前端机的 SN 型并行结构，其中前端机是系统与用户的交互界面，事务均在前端机送交系统，并且在前端机进行编译、词法、语法和完整性约束等检查，同时对后端机进行统一管理；后端机主要用来处理由前端机发来的各种对数据库的操作，整个数据库的数据按照某种划分原则均匀分布在后端机上。由于每个后端机均为一个独立的处理结点，因此它们有很大的自治性。互联网络主要用来实现前端机与后端机，后端机与后端机之间的通信，这里事务的抽象模型仍采用传统的事务读写模型。下面我们主要讨论不同的系统软件结构对系统并行性的影响。

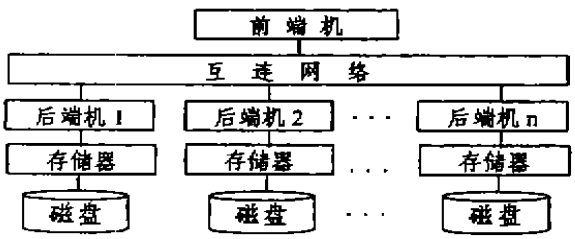


图 6　带一前端机的 SN 型并行结构

前端机与后端机软件结构采用什么实现方式，对系统所能提供的并行性有直接的影响，这里对前端机和后端机分别进行讨论，为了简单起见我们以两种可

能的实现方式为例来讨论(当然还有其它的实现方式)。以上面提到的例2为例,并假设应用程序1先于应用程序2提交给前端机。

前端机实现方式

1. 所有的功能集成为一个进程,如将查询优化、关系模式维护等功能都放在一个进程中。
2. 同时存在多个进程,如,查询优化进程、关系模式维护进程和数据维护进程等。每个进程完成相应的功能。

后端机实现方式

1. 所有功能集成为一个进程,如将数据维护、关系模式维护、投影和连接等功能由一个进程来完成。
2. 与前端机的2类似,系统一启动就存在多个进程,如,数据操作算法调度进程、数据维护进程、关系模式维护进程、投影子进程和连接进程等。

因此,整个系统的实现方式共有4种组合,记作(i,j),其中前端机采用实现方式i,后端机采用实现方式j。下面我们分别来讨论:

(1,1):前端机只能是顺序调度事务,应用程序1和应用程序2之间、程序内各事务之间、事务内的SQL语句之间都只能顺序执行。多个后端处理结点的环境虽然能够提供不同事务之间的并行执行,但由于前端机结构的限制,系统最多能够实现数据库操作自身的并行执行。因此在这种软件结构下系统只能提供表1中的第1种并行性。

(1,2):与前一种情况一样,由于前端机结构的限制,系统只能提供表1中的第1种并行性。但是此时,后端机采用第二种实现方式,在每个后端机上提供了多个事务并发执行的可能性。

(2,1):前端机可以实现多个事务的并发执行,但在每个后端机上只能顺序执行事务。在不同的后端机上可以实现不同事务之间的并行执行,同时还可获得数据库操作的并行执行。因此,系统能够支持表1中的第5种并行性,并且有可能达到第7种并行性。

(2,2):前端机可以实现多个事务的并发执行,每个后端机也能实现多个事务的并发执行。与前一种情况相似,在不同的后端机上可以实现不同事务之间的并行执行,同时还可获得数据库操作的并行执行。系统能够支持表1中的第5、7种并行性。系统的吞吐率比上一种情况要大。

我们看到在这几种组合中,(2,1)和(2,2)组合能够使系统提供更高的并行性,通过对事务实现方式和系统软件结构的调整,我们还可以获得更好的系统并行性。前端机与后端机采用不同的软件结构,都对系统的吞吐率有所影响,其中前端机的软件结构对系统所能提供的并行性起着关键性的作用。

结论 本文对传统事务模型的并行性进行了分析,对当前的执行方式进行了改进,提出了一种高并行性的执行方式。相应的结果已经应用于我国黑龙江大学研制的HPDB并行关系数据库系统,具体的实现细节以及涉及到的问题在这里就不再赘述,有兴趣的读者可以参看文[15,16]。

参考文献

1 Eswaran K P, et al. The Notions of Consistency and Predicate Locks in a Database System. CACM, 1976, 19(11)

2 Silberschatz A, Kedem Z. Consistency in Hierachical Database Systems. Journal of the ACM, 1980, 27(1)

3 Bernstein P A, Goodman N. Timestamp-based Algorithms for Concurrency Control in Distributed Database Systems. In: Proc. of VLDB, 1980

4 Kung H T, Robinson J T. Optimistic Concurrency Control. ACM Trans. on Database Systems, 1981, 6(2)

5 Ries D R, Stonbraker M. Effects of Locking Granularity in a Database Management System. ACM Trans. on Database Systems, 1977, 2(3)

6 Berstein P A, Goodman N, Lai M Y. Analyzing Concurrency Control when User and System Operations Differ. IEEE Trans. on Software Engineering, 1983, SE-9(3)

7 Papadimitriou C H. The Theory of Database Concurrency Control. Computer Science Press, Rockville, MD 1986

8 Al-Houmaily Y, Chrysanthis P. Two-Phase Commit in Gigabit-Networked Distributed Databases. In: Proc. of 8th Intl. Conf. on Parallel and Distributed Computing Systems. Sep. 1995

9 Mohan C, Dievendorff D. Recent Work on Distributed Commit Protocls Recoverable Messaging and Queuing. Data Engineering Bulletin, 17(1)

10 Georgakopolous D, et al. On Serializability of multidatabase transactions through forced local conflicts. In: Proc. of the Seventh Intl. Conf. on Data Engineering, Kobe, Japan, 1991

11 WeiKum G, Hasse C. Multi-Level Transaction Management for Complex Objects: Implementation, Performance, Parallelism. The VLDB Journal, 1993, 2(4)

12 WeiKum G. Principles and Realization Strategies of Multilevel Transaction Management. ACM Trans. on Data Systems, 1991, 16(1): 132～180

13 Schaad W, Schek H J, WeiKum G. Implementation and Performance of Multi-Level Transaction Management in a Multidatabase Enviroment. In: Proc. of the 5th Intl. Workshop on Research Issues on Data Engineering: Distributed Object Manamane, Taipei, Taiwan, 1995

14 李建中,孙文隽. 并行关系数据库管理系统引论. 科学出版社, 1998

15 张志强,李建中,等. HPDB系统事务处理的设计与实现. 黑龙江大学信息技术研究所研究报告, 1998

16 张志强,李建中,周立柱. HPDB系统事务处理的设计与实现 南京大学学报(专刊), Vol. 36, p52～59, 第八届全国青年计算机学术会议, 2000. 10, 南京