一个两段加锁不死锁的充分条件

冯 涛[1] 李 俊[2] 王 涛[2]

(陕西师范大学杂志社 西安 710062)[1] (陕西师范大学计算机学院 西安 710062)[2]

摘 要 两段加锁是分布式系统中最广泛使用的并发控制算法。该算法除实现较为复杂外,其致命弱点是容易产生死锁。本文在分析两段加锁产生死锁原因的基础上,给出了一个不会产生死锁的两段加锁方法的充分条件和构造性定理以及实现的方法。

关键词 两段加锁,并发控制,死锁,回滚

The Sufficiency Condition of the Two-Phase Locking without Deadlock

FENG Tao[1] LI Jun[2] WANG Tao[2]

(Magazine Press of Shanxi Normal University, Xi'an 710062)[1] (College of Computer Science, Shanxi Normal University, Xi'an 710062)[2]

Abstract The Two-Phase locking is the most widespread concurrency control algorithm in the distributed system. This algorithm is not easily carried out, and it's fatal weakness is easily lead to the deadlock. This article give one sufficiency condition, theory of construction and realization method of the Two-Phase Locking without deadlock based on analysis of the reason that The Two-Phase Locking lead to deadlock.

Keywords Two-Phase locking (2P-L), Concurrency control, Deadlock, Roll-back

1 分布式系统中并发控制的定义与理论

自从 Eswaran 等。在 1976 年证明了用两段加锁方法,可以保证并发执行的事务的串行化后,两段加锁方法就成为分布式系中并发控制的主要方法。但两段加锁除实现较为复杂外,还会产生死锁,研究既不会产生死锁、又较为简单的算法,具有现实意义。

定义 1 设 E 表示并发执行事务 $T_1, T_2, \cdots, T_n$ 的执行过程。若某次并发执行的结果和某个串行事务序列 $T_{i1}, T_{i2}, \cdots, T_{in}$ 的执行结果完全相同,则称本次并发执行是可串行化的。

显然,事务的可串行化执行能保证事务的正确执行。

由于对同一文件可以并发读,但对同一文件设置了写锁后,则不允许再设置其它的文件加锁,所以对文件的加锁可分为排它锁(Lw)和共享锁(Lr)。

定理 1 利用两段加锁(Tow-Phase Locking)法对多个事务的并发执行是可串行化的。

(证明略)。

两段加锁虽可保证串行化,但可能引起死锁。如下例 T_1, T_2, T_3 分别为三个不同处理机上正在运行的事务,它们对同一结点的三个不同文件 x, y, z 中的数据进行处理:

T_1	T_2	T_3
BEGIN-TRANS	BEGIN-TRANS	BEGIN-TRANS
Read x	Read y	Read z
Write y	Write z	write x
END-TRANS	END-TRANS	END-TRANS

显然,当三个事务分别加完读锁后,再依执行写 y、写 z 和写 x,则会因对写 y、写 z、写 x 的写锁的申请形成了如图 1 所示的循环,而构成死锁。

以下,假设在该分布式系统中所有文件均无副本;且假定封锁的粒度为读写数据所在的文件;系统的加锁机制由每个结点的加锁管理器来完成。

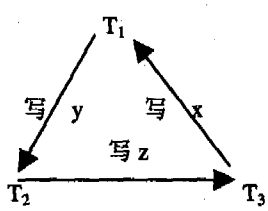

图 1

加锁管理器设有等待队列 Q,并决定事务 T_i 对本结点的文件的读或写的申请是否加锁或回滚(Roll-back)。

定义 2 设某结点有文件 $F_1, F_2, \cdots, F_m$,事务 $T_1, T_2, \cdots, T_n$ 在 t_0 时刻按逻辑时钟(timestamping 时间戳)访问该结点文件的序列为 $F_{i1}, F_{i2}, \cdots, F_{ij}$,且已按事务的读写要求加了相应的读锁(Lr)或写锁(Lw)$L(F_j)(1 \leqslant j \leqslant m)$,我们用符号"<"表示加锁的先后次序,称 $L(F_{i1}) < L(F_{i2}) < \cdots < L(F_{ij})$ 为该在时刻 t_0 时的结点锁序;类似地,把某个事务 T 对结点 P 的文件加锁的序称为这个结点的事务 T 锁序。明显地,事务 T 锁序是这个结点的结点锁序的一个子列。

显然,(1)对所有还未被访问文件 F_g,有 $L(F_{ij}) < L(F_g)$;(2)若本次对文件 F_g 的加锁申请为读锁(Lr),且有 $L(F_g) < \cdots < L(F_{ij})$,但 $L(F_g)$ 亦为读锁,则有 $L(F_{ij}) < L(F_g)$。

显然,"<"有传递性。

定理 2 设系统在 t_0 时刻,该结点的结点锁序为:$L(f_{k1}) < L(F_{k2}) < \cdots L(F_{kn})$,在 $t_1(t_1 > t_0)$ 时刻申请对文件 F_j 的加锁,若 $L(f_{ik}) < L(F_j)$,则加锁允许,并将该结点的锁序改为 $L(F_{i1}) < L(F_{i2}) < \cdots < L(F_{ik}) < L(F_j)$(当申请的加锁为排它锁时)或保持原有的锁序不变(当申请的加锁为共享锁时)作为结点在时刻 t_1 时的锁序,否则,将原来已对文件 F_j 加锁的子事务 T_k 回滚,从时刻 t_0 的结点锁序中删去相应的锁并将新的加锁请求加入到结点锁序和事务 T 锁序中,形成新的结

点锁序和事务 T 锁序。若所有的事务均按上述规则加锁，则这样的两段加锁机制是无死锁的。

证明：设系统在 t_0 时刻，该结点的结点锁顺序为：$L(f_{k1})<L(F_{k2})<\cdots L(F_{kk})$，事务 T_i 申请对文件 F_i 加锁，若有 $L(f_{ki})<L(F_j)$，则不会形成相互循环的等待状态，这由"<"的定义是显然的；若某次申请不满足 $L(f_{ki})<L(F_j)$，则说明本次申请的加锁和已有的加锁产生了冲突，从而形成了死锁的必要条件；但由于对冲突加锁的事务做了回滚操作，同时从结点锁序和事务 T_j 锁序中删除了相应的加锁，从而打破了 T_i 等待的条件，而事务 T_j 的再次加锁只能满足 $L(f_{ki})<L(F_j)$，这也就形成了一种按动态序加锁的方法，故不会形成死锁(证毕)。

2 无死锁的 2P-L 实现

2.1 结点加锁管理器的数据结构

(1)描述该结点锁序的双向队列 PQF：队列的每个结点为一结构类型，成员有文件名、写(Lw)或读锁 (Lr) 标记、实行加锁的事务名以及前向和后向指针，其头结点的两个指针分别指向队列的第一个元素和最后一个元素；

(2)描述每个事务锁序的队列 T_iQF：结点的结构除没有事务名外，其余相同。

结点加锁管理器采用分布式管理。

2.2 算法描述

当事务 T_i 要求对该结点的某个文件 F 加锁时

(1)取出 PQF 的最后一个元素的文件名 F_i、以及锁类型描述(Lw 或 Lr)；

(2)扫描队列 PQF，若 F 位于 F_i 之前且申请的加锁不属于共享锁，则转(5)否则转(3)；

(3)若 PQF 队列中没有 F，则将对 F 的加锁插入到 PQF 和 T_iQF 的队尾；

(4)若本次对 F 的加锁是共享锁(本次申请读文件 F，而 PQF 中对文件 F 的加锁也为读锁)，则将本次加锁插入到 T_iQF 的队尾，PQF 队列不变；

(5)把所有的对文件 F 加锁的子事务回滚(这些事务名从 PQF 可以查出)，并从 PQF 和相应的 T_iQF 队列中删除该事务对文件 F 的相应加锁(原有的其它锁序不变)；

(6)所有事务的提交，都应从 T_iQF 和 PQF 中删去相应的加锁。算法的正确性是显然的。

结束语 本文采用类似按序分配资源以预防死锁的方法，给出了两段加锁无死锁的充分条件，同时给出了加锁管理器的实现方法，由于结点和事务的锁序是按事务访问文件的顺序安排，且由于事务的回滚，原有的锁序也会发生改变，而不是事先规定的锁序，所以，它的效率和按序分配资源有本质的区别，该算法的实现也仅是几个队列的线性查找、删除、插入，算法复杂度为 $O(n)$，是完全可以接受的。算法的缺陷在于：(1)如何最大限度的减少事务回滚的次数；(2)若在同一结点或不同结点存在一个或多个副本时，如何优化该算法，还需进一步研究。

参 考 文 献

1 Al-Jumah N B, Hassanein H S, El-Sharkawi M. Implementation and modeling of two-phase concurrency control——a performance study [J]. Information and Software Technology, 2000,42:257～273

2 Kao A C B, Lam Kam-yiu. Comparing Two-Locking and Optimistic concurrency control Protocols in Multprocessor Real-Time Database. IWPDRTS(International Workshop on Parallel and Distributed Real-Time Systems)IEEE, 1997. 141～146

3 Tanenbaum A S. Modem Operating Systems(现代操作系统). 陈向群，等译. 北京机械工业出版社, 1999

(上接第 280 页)

化不影响分解的正确性。

如果在一个循环嵌套内存在对某个数组的多次访问，单环约束等式(5)往往会导致无法并行。而此类串行代码大量存在，为了使得算法更具有一般性可放松该约束等式。由于在计算划分和数据分布的过程中保留了依赖关系，因此对分解之后的代码仍可以运用依赖分析技术进一步的处理。这一步可以在代码自动产生时使用数据调整技术完成，通过在 SPMD 代码中增加显示的通信语句得到解决。

小结 本文主要阐述了在分布式存储环境中的一种基于线性代数的计算和数据自动分解算法，基于该算法实现的并行化编译器能够自动对串行代码进行无通信的分解。通过对计算划分和数组分布并行条件的深入分析，我们给出了完整的无通信分解约束等式，并对算法作了形式化的描述。在工程实现的基础上，为了增加算法的实用性，我们逐步放松了某些约束等式，并且对相应出现的分解结果进行了分析和实现，使得该算法的处理能力进一步的增强。

参 考 文 献

1 Knobe K, Lukas J D, Steele G L. Data optimization: Allocation of arrays to reduce communication on SIMD machies. Journal of Parallel and Distributed Computing, 1990,8:102～118

2 Lam M S, Wolf M E. Compilation techniques to achieve parallelism and locality. In: Proceedings of the DARPA Software Technology Conference, April 1992. 150～158

3 Huang C H, Sadayappan P. Communication-free hyperplane positioning of nested loops. In: Banerjee U, Gelernter D, Nicolau A, et al, eds. Languages and Compilers for Parallel Computing. Berlin, Germany, Springer-Verlag, 1992. 186～200

4 Kulkarni D, Kumar K G, Basu A, et al. Loop partitioning for distributed memory multiprocessors as unimodular transformations. In: Proceedings of the 1991 ACM International Conference on Supercomputing, June 1991. 206～215

5 Kumar K G, Kulkarni D, Basu A. Deriving good transformations for mapping nested loops on hierarchical parallel machines in polynomial time. In: Proceedings of the 1992 ACM International Conference on Supercomputing, July 1992. 82～91

6 Kennedy K, Kremer U. Automatic Data Layout for High Performance Fortra [A]. Proc Supercomputer [C]. San Diego, Calif, 1995

7 Wolf M E. Improving Locality and Parallelism in Nested Loops: [PhD Thesis]. Stanford University, August 1992

8 Anderson J M, Lam M S. Global optimizations for parallelism and locality on scalable parallel machines. In: Proceedings of the SIGPLAN '93 Conference on Programming Language Design and Implementation, Albuquerque, NM, June 1993. 112～135

9 Anderson J M. automatic computation and data decomposition for multiprocessors: [PhD Thesis]. Stanford University, March 1997

10 Guo Minyi. Efficient Techniques for Data Distribution and Redistribution in Parallelizing Compilers: [PhD Thesis]. Tsukuba University, 1998