文章编号:1001 - 9081(2011)05 - 01404 - 05
doi:10.3724/SP.J.1087.2011.01404

用积分法处理分布式数据库中的多副本并发控制问题

刘　毅,林子雨
(乐山师范学院 智能信息处理及应用实验室,四川 乐山 614004)
(liuyi3310@yeah.net)

摘　要:在分布式系统中,采用的并发控制(CC)方法对事务处理系统的性能有着重要影响。介绍了主要的并发控制方法、加锁模型和两阶段锁(2PL)协议,提出了基于锁机制且遵守 2PL 协议的悲观控制方法——积分法。该方法既能减少网络中数据的传送量,又具有很好的并发性,可以很好地处理多副本并发控制问题。实验结果证明,该积分法可以比其他方法取得更好的性能。

关键词:积分;分布式数据库系统;并发控制
中图分类号: TP311.133.1　　**文献标志码**:A

Concurrency control in distributed database system with score-based method

LIU Yi, LIN Zi-yu
(*Laboratory of Intelligent Information Processing and Application, Leshan Normal University, Leshan Sichuan* 614004, *China*)

Abstract: The Concurrency Control (CC) scheme employed in distributed database system can profoundly affect the performance of transaction-processing system. There are many different kinds of solutions to the problem of CC in distributed database system, and they have both advantages and disadvantages. The major methods of CC, basic knowledge of lock-based model and the project of 2-Phase Locking (2PL) were introduced. A score-based method was proposed based on lock-mechanism and in accordance with 2PL protocol, which could reduce the amount of data transmission and had desirable performance of concurrency. It could be used to deal with the problem of CC. The experimental results show that the proposed method can achieve much better performance than other available ones.

Key words: score; Distributed Database Management System (DDBMS); Concurrency Control (CC)

0　引言

在分布式数据库中,为了提高系统的可用性、可靠性及存取效率,经常存放数据项的多个副本,当系统的某一或几个场地发生故障时,可以通过读取别的场地上的副本数据保证数据的正常处理。但是,副本的增加却带来新的问题,比如副本的选择及副本的更新等。一般来说,多副本环境下的数据更新代价很高,因为这里涉及到同步通讯开销。一种好的多副本并发控制方法既要有相对较小的同步通讯开销,又应具备很好的并发性,这是判断一个并发控制算法性能高低的重要指标。目前的许多方法,尽管它们已经被广泛地应用于商业应用中,但是都存在或多或少的缺陷,有的方法通讯开销小,却不具备很好的并发性;而对于那些并发性好的算法,其通讯开销还是比较可观。虽然对这个领域的研究可以追溯到 20 世纪 70 年代,但是,直到今天,仍可以在许多重要刊物上找到与之相关的研究文献,这足以说明对并发控制方法的研究还有很大的空间。本文提出了积分法,详细阐述了该算法的正确性,并以实验数据证明了积分法其相对于其他方法的良好性能。

1　相关工作

对多副本并发控制研究的历史比较久远,从 20 世纪 70 年代到现在,诞生了许多已经在实践中应用的有效方法[1-6]。根据同步发生在更新数据之前还是之后,这些方法被划分为两大类:悲观的控制方法和乐观的控制方法。对于前者,同步发生在访问数据之前,比较有代表性的算法包括文献[6-8];而对于后者,同步则发生在数据访问之后,以文献[5,9-11]等提出的算法为代表。悲观的控制方法又可以分为基于锁机制的方法和基于时间戳排序(Time-Stamp Ordering, TSO)的方法。许多学者对基于时间戳排序和悲观并发控制(Optimistic Concurrency Control, OCC)进行了大量的研究[12-13],但是前者更适用于分布式系统,在其他类型系统中性能不佳,而对于两阶段事务处理系统来说,OCC 则是一种比较可行的方法。OCC 方法是加锁方法的主要替代方法,对冲突较少的系统较为方便,特别是大多数事务是只读型的应用时更为合适,因为一个只读事务没有写入阶段及写入集,因此验证起来比较简单,而对于冲突较多的系统,使用乐观方法显然是不明智的,在高效事务处理系统中,基于锁机制的方法表现要优于乐观控制方法。

一种被广泛采用并被许多教科书重点介绍的基于锁机制的悲观控制方法就是“投票法”,投票法减少了在更新和读取操作中所涉及到的副本的数量,从而降低了网络通讯开销,减少了网络分割的负面影响,平衡了读写操作的开销,并使系统获得了良好的可用性和稳定性。有许多文献对基于投票法的系统的可用性和负载分配问题进行了详细的研究和讨论[14-15],这些研究与分析都基于这样几个假设:1)把渐近性

收稿日期:2010 - 08 - 11;**修回日期**:2010 - 10 - 18。　**基金项目**:四川省教育厅资助重点资助项目(07ZA035)。
作者简介:刘毅(1965 -),男,副教授,主要研究方向:数据库、软件工程。

行为作为整个系统的代表行为; 2)不管是远程还是本地,读写都具有相同的开销; 3)可扩展性不必是单调的。

投票法是一类方法的统称,其中包含几个有代表性的方法,如读-写全法[11]和多数法[9]。读-写全法的特点是,当事务对某一数据项加锁时,若加读锁,则只对其多副本中的任一个副本加锁;若加写锁,则要对该数据的所有副本加锁,即要向分布式数据库中所有拥有被锁对象的副本的场地发送报文申请锁,等待这些场地的回答,当均回答可以加锁时,才真正向这些场地发送数据项的新值,然后再进行解锁。显然这种方法在写锁情况下通信费用是很大的。当有一个场地拒绝加锁时,就不能获得被锁对象的写锁。为了解决通信费用较大的问题,Thomas 在文献[9]中提出了“多数法”,并详细分析了算法的正确性和可行性,且在文献[16]中被采用。该方法的基本思想是:只有在获得多于副本数一半以上的锁(包含读或写锁)以后,才可以获得对数据项的加锁,即加锁的副本数总大于未加锁的副本数。这样,在同一个时刻内两个事务就不可能对同一数据项及副本加排他锁,但可以加共享锁。

除投票法以外,还有其他一些方法也很具有代表性。在文献[10]中作者提出了主副本法,在该方法中,主副本法对某一数据项加锁时,不管其副本数是多少,只要对其中一个称之为主副本的加上锁,即得到该数据项的锁。因此,通信费用大大降低,但其并发的程度也降低了。此外,被人们所熟悉的方法还包括中心场地法[17]、主副本令牌法[17]、键值加锁法(Key Value Locking, KVL)[18]等。中心场地法中,由一专门的场地来管理加锁的请求,但容易形成瓶颈,且当中心场地发生故障时系统会瘫痪。主副本令牌法中,当某一场地拥有某一数据项的写令牌时,它就可以得到该数据项的读/写锁,当拥有读令牌时,它就可以得到该数据项的读锁。

在以上各种方法中,对查询操作多的系统使用读-写全法较好,对更新操作较多时可采用多数法;主副本法报文传送少但并发度低,没有充分利用多副本的优点。读-写全法出现死锁的可能性大。多数法死锁的可能比读-写全法少。基于以上的分析,本文提出了简单可行而又具有很好性能的积分法。

2　预备知识

本文所研究的积分法就是基于锁机制且遵守 2-阶段锁(2-Phase Locking, 2PL)协议的并发控制方法,所以在介绍积分法之前,先介绍 2PL 协议和锁机制的基本思想。

2.1　分布式两段锁协议(2PL 协议)

在集中式数据库系统中,2PL 协议是并发控制算法中的重要算法之一。其主要内容是,并发执行的多个事务中,事务对数据进行操作以前要进行加锁,且每个事务中的所有加锁操作在第一个解锁操作以前执行,即每个事务中的加锁操作和解锁操作分布在两个部分中。在分布式数据库中,如果全部的分布式事务都以 2PL 协议加锁,则系统中的各场地上的局部调度是可串行化的。2PL 协议正确性的证明过程可以参见文献[17]。

2.2　锁机制的基本思想

锁机制的基本思想是基于这样的原则:事务对任何数据的操作必须先申请数据项的锁,只有申请到锁以后,即加锁成功以后,才可以对数据项进行操作。操作完以后,要释放已经申请的锁。通过锁的共享及排斥的特性,实现事务的可串行化调度。在基本锁方法中,对于锁模型的锁不加以区分,这有可能降低事务的并发程度。因此,一般对锁的类型要加以区分,分为读锁和写锁两种。读锁是对数据项进行读操作时要加的锁。由于读操作是可共享操作,所以读锁也称为共享锁。写锁是对数据项进行写操作时要加的锁。写操作是不可共享的,因此,写锁也称为排他锁。写锁和排他锁之间的相容矩阵如下所示:

	读锁	写锁
读锁	共享	排他
写锁	排他	排他

3　积分法

3.1　积分法概述

大量的研究证明,消息的传送量和并发性是衡量一个加锁方法是否高效的重要指标,为了既可以减少网络中数据的传送量,又可以保证具有比较好的并发性,本文提出了“积分法”,它属于基于锁机制且遵守 2PL 协议的悲观控制方法。首先给出本文中出现的几个概念的定义。

定义 1　事务 T 从场地 s 获得积分,当下列条件成立:

1)事务 T 所在的场地 r 向场地 s 发送加锁请求;

2)场地 s 给场地 r 反馈信息;

3)场地 s 同意事务 T 的加锁请求;

定义 2　最小积分 $SCORE_{min}$ 是指事务 T 申请对某一数据项加锁时,若想获得对该数据项加锁的许可时所需的积分的最小值,即事务 T 至少获得该积分值以后才能对该数据项加锁。

定义 3　副本场地是指某数据项的副本所在的场地。

定义 4　总积分 $SCORE_{sum}$ 是指事务 T 从各个副本场地获得的积分的总和。

定义 5　事务 T 获得对某数据项的加锁权限当且仅当事务 T 获得的总积分 $SCORE_{sum}$ 达到(即不小于)最小积分 $SCORE_{min}$。

本方法中,加锁请求发起事务 T 所在的场地 r 可以向副本所在场地发送加锁请求,如果能够得到某场地的答复并且该场地表示“赞成”,事务 T 就可以从该场地获取积分;否则得不到积分。一旦事务 T 得到的总积分 $SCORE_{sum}$ 达到了加锁所要求的最低积分 $SCORE_{min}$,那么事务 T 就可以对该数据项加锁。如果已经没有可能得到所需的最小积分,则事务 T 进入等待状态。

3.2　初始化过程算法

采用积分法时,必须做一些准备工作。这些准备工作,由分布式数据库系统完成。准备工作包括:建立带有积分值的副本列表以及把该列表发送到分布式数据库(Distributed Data Base, DDB)中的各个场地上。初始化过程由“算法 1”完成。

算法 1　建立带有积分值的副本列表并发送到 DDB 中的各个场地上。

输入　n 个副本的信息。

输出　分布到各个场地上的带有积分值的副本列表; 最小积分值 $SCORE_{min}$。

1) 对 n 个副本按编号从小到大进行初始化,建立带有积分值的副本列表,设置第 1 个副本的积分值是 1,设置第 2 个副本的积分值为 2,以后的每个副本的积分值都是与它紧相邻的编号比它小的两个副本的积分值之和;
2) IF n 不能被 3 整除 THEN
3) 副本 n, $n-3$, $n-6$,…, k 的积分值之和作为 $SCORE_{min}$ 的值; //其中 k 为该序列中最后一个大于 0 的数
4) ELSE
5) 把副本 n, $n-3$, $n-6$, …, 3, 1 积分值之和作为 $SCORE_{min}$ 的值;
6) 各个场地保存带有积分值的副本列表

在上面的算法中, $SCORE_{min}$ 的确定方法是很关键的,因为事务 T 一旦达到所需的最小积分 $SCORE_{min}$,就可以对数据项进行加锁,因此 $SCORE_{min}$ 的值的设定必须能保证在同一时刻不能有两个或两个以上的事务取得对同一数据项加锁的权限。换句话说,也就是当有好几个事务请求对某一数据项加锁时,一旦它们之中的某个事务 T 获得的积分值达到了最小积分 $SCORE_{min}$,那么其他事务就不可能再有机会达到最小积分 $SCORE_{min}$。

定理 1 多个事务申请对某个数据项加锁时,在同一时刻,仅能有一个事务 T 获得对某数据项的加锁权限。

证明 我们假设有 m 个事务 $T_1, T_2, \cdots, T_m$ 同时申请对某一数据项加锁。

由定义 5 可知,事务 T 获得对某数据项的加锁权限当且仅当事务 T 获得的总积分 $SCORE_{sum}$ 达到(即不小于)最小积分 $SCORE_{min}$。

所以,要证明该定理,只需证明当某个事务 T_i 获得的积分值达到了最小积分 $SCORE_{min}$,那么其他事务就不可能再有机会达到最小积分 $SCORE_{min}$。

由算法 1 可知, $SCORE_{min}$ 的确定方法分成两种情况:副本数目 n 能被 3 整除时和副本数目 n 不能被 3 整除时。为了讨论和描述方便,我们假设 n 比较大。

情况一 当 n 不能被 3 整除时。$SCORE_{min}$ 的值是副本 n, $n-3$, $n-6$,…, k(k 为该序列中最后一个大于 0 的数)的积分值之和;当事物 T 获得 $SCORE_{min}$ 时,就得到了副本 n, $n-3$, $n-6$,…, k 的积分值,剩下的副本的积分被其他事务取得。事实上,我们不妨假设剩余的所有的积分都被事务 T_j 获得,即然后验证 T_j 的 $SCORE_{sum}$ 是否可以达到 $SCORE_{min}$。

当 $k=1$ 时,我们知道,事务 T_i 获得了副本 n, $n-3$, $n-6$,…, 4, 1 的积分,而事务 T_j 则获得了副本 $n-1$, $n-2$, $n-4$, $n-5$,…, 3, 2。

设 S_i 表示副本 i 的积分,则:

$$SCORE_{sum}(T_i) = S_n + S_{n-3} + \cdots + S_4 + S_1$$

$$SCORE_{sum}(T_j) = S_{n-1} + S_{n-2} + S_{n-4} + S_{n-5} + \cdots + S_3 + S_2$$

由算法 1 可知:

$$S_n = S_{n-1} + S_{n-2}$$

$$S_{n-3} = S_{n-4} + S_{n-5}$$

$$\vdots$$

$$S_4 = S_3 + S_2$$

因此有:

$$S_n + S_{n-3} + \cdots + S_4 = S_{n-1} + S_{n-2} + S_{n-4} + S_{n-5} + \cdots + S_3 + S_2$$

则又有:

$$S_n + S_{n-3} + \cdots + S_4 + S_1 > S_{n-1} + S_{n-2} + S_{n-4} + S_{n-5} + \cdots + S_3 + S_2$$

即 $SCORE_{min} > SCORE_{sum}(T_j)$,也即事务 T_j 获得的总积分不可能达到最小积分。

当 $k=2$ 时,同理可证。

情况二 当 n 能被 3 整除时。可仿情况一的证明过程进行证明。

综上所述,当某个事务 T_i 获得的积分值达到了最小积分 $SCORE_{min}$,那么其他事务就不可能再有机会达到最小积分 $SCORE_{min}$。

因此,定理可以得证。

为了更好地理解此算法过程,我们以具体实例进行说明。

情况一 当 n 能被 3 整除时。以 $n=12$ 为例子,如图 1 所示。按照算法 1,构成 $SCORE_{min}$ 的几个副本所在场地是:12,9,6,3,1。只要某申请加锁场地 r 获得了副本场地 12,9,6,3,1 的积分,那么其他申请加锁事务就算可以得到剩余所有副本场地的积分,也不会比 r 获得的总积分多。因此,把副本场地 12,9,6,3,1 的积分之和作为 $SCORE_{min}$ 的值,就可以保证一个申请加锁事务达到这个最小值时,其他申请加锁事务就不可能再有机会达到。

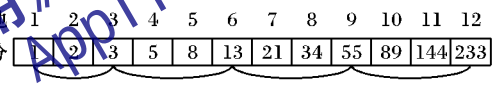

图 1 能被 3 整除时 $SCORE_{min}$ 的确定方法

情况二 当 n 不能被 3 整除时,以 $n=11$ 和 $n=10$ 为例。当 $n=11$ 时,如图 2 所示,我们只要把副本场地 11,8,5,2 的积分之和作为 $SCORE_{min}$,就可以保证满足条件。当 $n=10$ 时,如图 3 所示,我们只要把副本场地 10,7,4,1 的积分之和作为 $SCORE_{min}$,就可以保证满足条件。

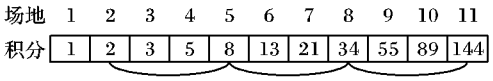

图 2 被 3 除余 2 时 $SCORE_{min}$ 的确定方法

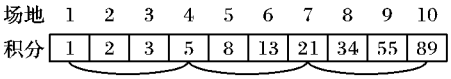

图 3 被 3 除余 1 时 $SCORE_{min}$ 的确定方法

3.3 数据项加锁算法

前面的准备工作完成以后,当某个事务 T 申请对某个数据项加锁时,就可以首先向那些构成最小积分的副本场地发送加锁请求,如果没有获得所需最小积分,就调用积分弥补过程,积分弥补过程失败,则进入等待状态。

算法 2

输入 n 个副本场地信息以及事务 T。

输出 加锁是否成功。

1) 把加锁请求发往的目的地的集合 Q 设置为空集;
2) IF n 不能被 3 整除 THEN
3) 把场地 n, $n-3$, $n-6$,…, k 加入到 Q 中;
// 其中 k 为该序列中最后一个大于 0 的数
4) ELSE
5) 把场地 n, $n-3$, $n-6$,…, 3, 1 加入到 Q 中;
6) 向集合 Q 中的各个场地发送加锁请求,并在本地存放的

副本列表中对这些场地做标记 0
　　//标记 0 表示该目的场地拒绝
7)　事务 T 所在场地 r 接收各目的地发送回来的反馈信息;
8)　IF 场地 i 的反馈信息是“赞成” THEN
9)　　在本地存放的副本列表中对场地 i 做标记 1;
　　//标记 1 表示该目的场地赞成
10)　IF 各目的场地都表示“赞成” THEN
11)　　对该数据项加锁;
12)　ELSE
13)　　计算那些表示“赞成”的场地的累计积分值 $SCORE_{sum}$;
14)　　把表示“拒绝”的场地加入集合 R;
15)　　complement();　　//调用积分弥补过程
16)　IF $SCORE_{sum}$ 能够弥补达到 $SCORE_{min}$ THEN
17)　　对数据项加锁;
18)　ELSE
19)　　事务 T 进入等待状态;

从算法 2 中可以看出,当多个事务同时申请对某个数据项加锁,并且每个都从其中一部分场地获得了积分,但都未能够达到最小积分时,它们就都进入等待状态,这种情况可以用通用的死锁处理机制来解决,释放其中一部分场地,使其他申请加锁事务能获得积分,从而最终获得对某数据项加锁的权利。产生死锁对性能的影响请参见文献[12,17,19],有关死锁探测的相关策略可以参见文献[20],文献[19]则详细论述了事务系统模型、标准静态和动态加锁方法、锁冲突和死锁、改善锁性能的方法、两阶段处理方法的性能以及乐观控制方法的分析。

3.4　积分弥补过程 complement

算法 3

输入　反馈信息表示“拒绝”的场地集合 R;

输出　是否弥补达到 $SCORE_{min}$。

1)　WHILE R 不为空 DO
2)　取出集合 R 中编号最大的场地 j,并从集合 R 中删除此元素;
3)　向编号比 j 小的并与 j 紧相邻的两个场地发送加锁请求;

如果能同时从这两个场地得到积分,则把二者的积分累加到 $SCORE_{sum}$ 中,这时如果积分达到 $SCORE_{min}$,就退出 complement 过程;否则,也就是没有同时得到二者的积分时,也退出 complement 过程。

4　性能分析

通信费用是衡量一个并发控制算法性能的重要标准。对于本文提出的积分法,从算法 2 中可以看出,如果第 10)步成立,则可以立即对该数据项加锁,这样传输的报文次数最少时就只有 $2n/3$,而多数法最少时需要 $3(n+1)/2$[17]。所以在最优情况下,积分法的表现优于多数法。同时,积分弥补过程 complement 是一个很快速的过程,并且能在有限的时间内给出弥补的结果,这比起多数法要反复发送报文向多个场地进行询问要高效得多。另外,相对于主副本法而言,积分法具有更好的并发性。

为了进一步证明积分法在性能方面相对于已有的其他算法的优势,进行了一系列实验。实验环境为:10 台计算机组成一个局域网,每台计算机的配置为 Window XP 操作系统、1.3 MHz 的 CPU、256 MB 内存和 SQL Server 2000 数据库服务器。使用 DELPHI 和 COM + 进行编程,采用 SQL Server 2000 提供的加锁语句对数据对象进行加锁,设置 5 个客户端同时对各个场地发起加读锁或加写锁请求,并且每个加锁操作持续 2 s 以保证出现加锁冲突,采用超时法处理死锁情况,对主副本法、多数法和积分法分别设计对照实验。在每种方法中,都会记录下从发送第一个加锁请求到最后加锁成功之间的总共加锁请求传递次数,最后取 5 个客户端的记录值的平均值作为最终评测结果。实验结果如表 1 ~ 4 所示。

表 1 反映的是主副本法、多数法和积分法在加读锁成功时的加锁请求传递次数,从中可以看出,主副本法的加锁请求传递次数不会随着副本数目的变化而变化,而在多数法和积分法中加锁请求传递次数则会随着副本数目的增加,但是就增加的幅度而言,积分法要明显小于多数法,而且副本数目越多,这种优势越明显。

表 1　加读锁成功时的加锁请求传递次数

副本数量	主副本法	多数法	积分法
3	1	2	1
4	1	3	2
5	1	3	2
6	1	4	3
7	1	4	3
8	1	5	3
9	1	5	4
10	1	6	4

表 2 反映的是主副本法、多数法和积分法在加写锁成功时的加锁请求传递次数,从中可以看出,主副本法的加锁请求传递次数随着副本数目的变化基本趋于平稳,而多数法和积分法则变化较大;对于传递次数的增加幅度而言,积分法要明显小于多数法,而且副本数目越多,这种优势越明显。

表 2　加写锁成功时的加锁请求传递次数

副本数量	主副本法	多数法	积分法
3	3.4	6.7	4.1
4	3.2	8.9	5.3
5	3.8	12.5	5.4
6	3.3	16.4	8.5
7	3.4	21.3	8.2
8	3.5	26.4	8.8
9	3.4	31.2	16.7
10	3.5	39.4	16.2

表 3 反映的是 10 个副本时单位时间内 5 个客户端处理读/写数据操作数量总和之比,我们用这个指标来反映 3 种方法的并发性,从中可以看出,在加写锁时,积分法具有明显的优势。

表 3　5 个客户端处理读/写数据操作数量总和之比

操作	主副本法:多数法:积分法
读	12:1:1.9
写	0.3:1:1.5

积分法在最佳情况下,确实要比多数法效率高,但是也存在一个问题,那就是由于副本场地的积分不同,则副本场地的重要性也有差别,那些积分小的副本被访问的几率比较小,相

反那些积分值很大的场地，会被经常访问，尤其是最后几个场地，一旦某个申请加锁事务连续失去最后几个副本场地的积分，那就意味着即使前面获得了再多的积分，也要进入等待状态。对于这个情况，本文认为可以采取与通信费用相结合的方法，来削减高积分场地的优势，这样，要研究的情况将很复杂，所以本文没有给予深入讨论。另外，当积分值最大的那几个场地如果出现故障，那么无论如何努力，申请加锁事务都不可能达到 $SCORE_{min}$，这样，就需要在死锁解除机制中增加这样一个功能，即分布式数据库管理系统检测出这样的场地故障以后，就需要通知各个处于等待状态中的申请加锁事务，让它们各自的 $SCORE_{min}$ 都减去发生故障的场地中的最大的积分值。此外，本文的研究并未考虑数据库系统的构成，即是同构数据库还是异构数据库，在实际应用中，要充分考虑可能的数据库系统构成，针对同构和异构数据库系统分别采用不同的细节处理过程，有关同构数据库系统的并发控制研究比较多，而对异构数据库系统的研究相对比较少。

图 4 显示了积分法(Score)和文献[5]最近提出的 Mobile 方法的性能比较。从图 4 中可以看出，在不同的副本数量的情形下，在成功提交的事务的数量方面，积分法的性能都要优于 SCORE。

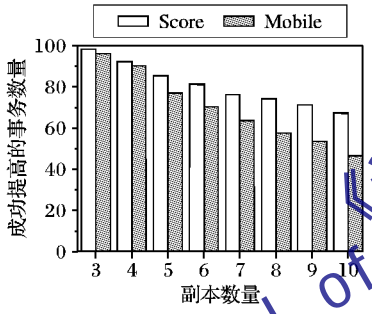

图 4 积分法和 Mobile 方法性能比较

5 结语

在分布式数据库系统中的多副本并发控制这个问题上，许多研究单位都进行了大量的研究，也提出了许多卓有成效的方法，本文所提的积分法只是针对多数法的通信费用和主副本法的并发程度方面的改进，但是其本身也存在一定的缺陷，在今后的研究工作中，将引入通信费用来进行综合考虑，从而对积分法进行优化。

参考文献：

[1] PLESHACHKOV P, CHARDIN P, KUZNETSOV S D. A dataguide-based concurrency control protocol for cooperation on XML data [C]// 9th East European Conference on Advances in Databases and Information Systems. Berlin: Springer, 2005: 268 – 282.

[2] ENOKIDO T, TAKIZAWA M. Role-based concurrency control for distributed systems [C]// 20th International Conference on Advanced Information Networking and Applications. Washington, DC: IEEE Computer Society, 2006: 407 – 412.

[3] IZADI K, ASADI F, HAGHJOO M S. XPLC: A novel protocol for concurrency control in XML databases [C]// International Conference on Computer Systems and Applications. Washington, DC: IEEE Computer Society, 2007: 450 – 453.

[4] ANSARI M, KOTSELIDIS C, JARVIS K, *et al*. Advanced concurrency control for transactional memory using transaction commit rate [C]// 14th International Euro-Par Conference on Parallel Processing. Berlin: Springer, 2008: 719 – 728.

[5] MOIZ S A, RAJAMANI L. An efficient strategy for achieving concurrency control in mobile environments [C]// 12th Asia-Pacific Network Operations and Management Symposium. Berlin: Springer, 2009: 519 – 522.

[6] KLEIN J. Concurrency and replica control for constraint-based database caching [C]// 14th East European Conference on Advances in Databases and Information Systems. Berlin: Springer, 2010: 305 – 319.

[7] BERNSTEIN P A, GOODMAN N. An algorithm for concurrency control and recovery in replicated distributed databases[J]. ACM Transactions on Database Systems, 1984, 9(4): 596 – 615.

[8] SINGHAL M, AGRAWALA A K. A Concurrency control algorithm and its performance for replicated database system [C]// Proceedings of the 6th International Conference on Distributed Computing Systems. Washington, DC: IEEE Computer Society, 1986: 140 – 147.

[9] THOMAS R H. A majority consensus approach to concurrency control for multiple copy databases [J]. ACM Transactions on Database Systems, 1979, 4(2): 180 – 209.

[10] ALSBERG P A, BELFORD G C, DAY J D, *et al*. Multicopy resiliency techniques [C]// Distributed Data Management. Washington, DC: IEEE Computer Society, 1978: 128 – 176.

[11] SINGHAL M. An optimistic concurrency control algorithm with conflict resolution in replicated database systems [C]// Proceedings of the 20th Hawaii International Conference on System Sciences. Washington, DC: IEEE Computer Society, 1987: 64 – 72.

[12] BERNSTEIN P A, GOODMAN N. Concurrency control in distributed database systems [J]. ACM Computing Surveys, 1981, 13(2): 185 – 221.

[13] LIM S, CHO H. Timestamp based concurrency control in broadcast disks environment [C]// 13th International Conference on AI, Simulation, and Planning in High Autonomy Systems. Berlin: Springer, 2004: 333 – 341.

[14] AMIR Y, WOOL A. Evaluating quorum systems over the Internet [C]// Proceedings of the IEEE International Conference on Fault-Tolerant Computing Systems. Washington, DC: IEEE Computer Society, 1996: 26 – 35.

[15] NAOR M, WOOL A. The load, capacity, and availability of quorum systems [J]. SIAM Journal Computer, 1998, 27(2): 423 – 447.

[16] HAMMER M M, SHIPMAN D W. Reliability mechanisms for SDD-1: A system for distributed databases[J]. ACM Transactions on Database System, 1980, 5(4): 431 – 466.

[17] 郑振楣, 于戈, 郭敏. 分布式数据库[M]. 北京: 科学出版社, 2000.

[18] Mohan C, ARIES K. A key-value Locking method for concurrency control of multi-action transactions operating on B-tree indexes [C]// Proceedings of the 16th International Conference on Very Large Data Bases. Brisbane: Morgan Kaufmann, 1990: 392 – 405.

[19] THOMASIAN A. Concurrency control [J]. ACM Computing Surveys, 1998, 30(1): 70 – 119.

[20] MAKKI K, PISSINOU N. Detection and resolution of deadlocks in distributed database systems [C]// Proceedings of the 1995 International Conference on Information and Knowledge Management. New York: ACM Press, 1995: 411 – 416.