基于冲突率预测的自适应并发控制算法

范璧健 庄 毅
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 并发控制算法能够保证数据库事务集并发执行的正确性和一致性。为了提高并发事务的执行效率,提出了一种基于冲突率预测的自适应并发控制算法(ACC-PRC)。该算法将并发控制过程分为信息收集和策略选择两个阶段。信息收集阶段利用先验事务队列保证事务执行的可串行化,并且利用循环冲突队列收集系统的事务执行状态。策略选择阶段在循环冲突队列上运用改进的加权移动平均法预测下一阶段冲突率,并根据双向阈值决策下一阶段的并发策略。所提算法在事务到达率较高时能保持良好的事务执行效率,同时能够准确及时地感知冲突率的变化。对比实验表明ACC-PRC算法的综合性能优于HCC算法和ADCC算法。

关键词 并发控制,冲突率预测,策略选择
中图法分类号 TP311.133.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.11.054

Adaptive Concurrency Control Algorithm Based on Conflict-rate Prediction

FAN Bi-jian ZHUANG Yi
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Concurrency control algorithm can guarantee the correctness and consistency of the database transaction. In order to improve the efficiency of concurrent transactions, an adaptive concurrency control algorithm based on conflict-rate prediction(ACC-PRC) was proposed. The algorithm is divided into two stages: information collection and strategy selection. The information collection stage uses a priori transaction queue PTQ to guarantee the serializable execution of the transaction, and a cyclic conflict queue CQR is used to collect the transaction execution state of the system. The strategy selection stage uses the improved weighted moving average method to predict the next stage of conflict using the cyclic conflict queue, and then chooses appropriate concurrency strategies by bidirectional threshold. The algorithm maintains good transaction efficiency while the transaction arrival rate is relatively high. The results show that the integrate performance of ACC-PRC algorithm is better than that of the HCC algorithm and ADCC algorithm.

Keywords Concurrency control, Conflict-rate prediction, Strategy selection

1 引言

并发控制算法能够处理共享数据库中事务并发执行导致的一致性错误,同时保证数据库系统的一致性和完整性。而串行化是衡量事务并发执行正确性的重要标准。串行化标准通常用强两阶段锁S2PL算法,在该算法中锁一直保持到事务提交,这样锁可能被长期保持而影响性能,尤其是在长事务等待时。并发控制算法中最常用的两种方法是乐观并发控制和悲观并发控制。乐观并发控制认为事务重启的开销比队列等待的开销少,适用于冲突率较低的情况;悲观并发控制则认为所有事务都可能发生冲突并对事务操作对象加锁控制,适用于冲突率较高的情况。在实际事务并发过程中,冲突率不是固定不变的,因此乐观/悲观并发控制策略都具有一定的局限性。

为了解决上述问题,本文在对基于动态决策的并发控制算法(ADCC)研究的基础上提出了基于冲突率预测的自适应并发控制算法(Adaptive Concurrency Control algorithm based on Prediction of Conflict Rate, ACC-PRC),该算法将并发控制过程分为信息收集和策略选择两个阶段。信息收集阶段利用先验事务队列保证事务执行的可串行化,并且利用循环冲突队列收集系统的事务执行状态;策略选择阶段在循环冲突队列上运用改进的加权移动平均法预测下一阶段的冲突率,并根据双向阈值决策下一阶段的并发策略。本算法在事务到达率较高时保持良好的事务执行效率,同时能够准确且及时地感知冲突率的变化。

2 相关研究工作

乐观并发控制和悲观并发控制是两种主要的并发控制算法。在乐观并发控制算法中,当有操作提交时其就立即被接受,在一段时间之后再检查操作是否出错,如果运行正常则继

到稿日期:2015-10-18 返修日期:2015-12-03 本文受国家自然科学基金青年科学基金项目(61202351),国家博士后基金项目(一等)(2011M500124)资助。

范璧健(1991-),男,硕士生,主要研究方向为分布与并行计算,E-mail:fandy0409@sina.com;庄 毅(1956-),女,教授,主要研究方向为网络与分布式计算、信息安全、可信计算。

续执行，否则用中断/重启来撤销已经执行的操作。在悲观并发控制算法中，当有操作提交时必须首先决定操作应该被接收、拒绝或推迟。

悲观并发控制算法等传统的并发控制算法是基于封锁的协议。写锁和读锁是两种基本的锁类型。如果事务申请操作对象的锁与其他事务正在操作的锁类型匹配，则事务能够执行。如果锁类型不匹配，则事务必须等到其他事务释放对应的锁后才能继续执行。这些处理锁冲突的操作保证了数据库的可串行化。但是用锁来保证可串行化不可避免地会增加额外的开销，只有在锁冲突发生概率较高时，即在悲观并发控制算法中用锁才有意义。相反，在乐观并发控制算法中认为事务冲突发生概率较低，而发生冲突时，重启事务的消耗比事务等待的消耗要低。

针对上述情况，徐淑分析了常用实时并发控制算法的性能，提出了一种适用于无共享架构的并行实时数据库并发控制算法(PRMCC)[1]，该算法将第一个接收到事务请求的节点作为事务协调者，负责调用并发控制及决策，同时规定在数据最后一个操作结束后立即写回数据库，而不是写操作完成后立即重启。通过实验表明该算法在长短事务混合执行时有较高的执行率。Nystrom D. 提出了一种 CC 算法[2]，该算法将传统的悲观并发控制用作软事务的执行，而用版本来管理硬事务。Makni A. 提出了一种基于乐观并发控制的算法[3]，该算法能够较好地保证事务执行的一致性，但是算法性能没有得到保证。

Demsky B. 在传统并发控制算法的基础上，提出了一种新的并发控制方法 VIEWS[4]，VIEWS 能够简化复杂的锁算法，同时能够自动检测可能发生的数据冲突。Sheikhan M. 提出了一种基于神经网络的并发控制算法(NCC)[5]，尝试将 ART2 神经网络用于事务的优先级评价，对健康因子较高的事务优先处理。该算法在事务的 WS 和 RS 不确定的情况下也能较好地执行，但是对冲突率变化不敏感。文献[6]针对这种情况，提出了一种混合并发控制算法 HCC，该算法能够预测未来一段时间内的冲突率，从而选择不同的并发控制策略。HCC 算法虽然有效地降低了冲突中断个数，但是预测准确性还有待提高。CHEN Y. R. 在 HCC 算法的基础上提出了一种基于动态决策的并发控制算法(ADCC)[7]，该算法具有较强的适应性，对冲突率变化较为敏感，在数据库处于空闲或繁忙状态下都有较高的执行效率。

综上所述，现有的并发控制算法已有较高的运行效率，但是适应情况较为单一。ADCC 算法虽然有较强的适应性，但是预测效果没有得到有效的反馈，同时在并发控制策略切换时没有考虑到正在执行的事务状态，所以本文在 ADCC 的基础上进行进一步研究。

3 基于冲突率预测的自适应并发控制算法 ACC-PRC

针对上文中提到的现有并发控制算法的不足，本文提出了一种基于冲突率预测的自适应并发控制算法(ACC-PRC)。该算法用事务先验队列保证事务执行串行化，根据事务执行情况预测系统未来一段时间的冲突率，根据冲突率选择不同的控制策略。本文沿用 ADCC 算法中的一些定义并将其进行一定的修改，具体如下：

每个数据库对象都有新、旧两个版本，其中新版本是事务提交后正式生成的版本，旧版本是事务正在写入而尚未提交的版本。需要注意的是当某个对象首次写入时，可能只存在旧版本。

事务用数据结构 T 定义，其中包括全局唯一的事务标识符 Tid 和事务操作序列 Op。Op 操作序列中的每个操作定义为对对象的一次读/写操作，所有操作都是偏序关系。

每个事务都有对应的先验事务队列 PTQ。PTQ 用于保证事务调度的可串行化，其中存储了与事务 T 操作发生冲突的事务集合。

3.1 ACC-PRC 算法模型

图 1 为本文提出的算法模型图，当数据库收到新的事务请求时，为其分配全局唯一的数据结构 T，分析事务 T 并列出事务 T 的先验事务队列 PTQ，同时将其加入到待执行事务队列中。当事务 T 的先验事务队列为空时，事务 T 进入事务调度器中执行。当事务中所有写操作执行完后立即提交。事务调度器在执行事务的同时需要收集每个时间段内事务的执行情况，根据事务的执行情况决定下个时间段的并发控制策略。

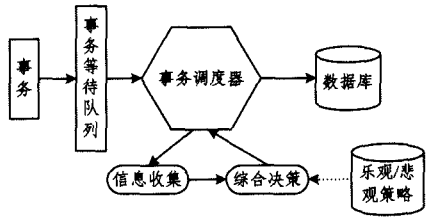

图 1 ACC-PRC 算法模型图

在每个时间段内 ACC-PRC 算法都分为两个阶段：信息收集阶段与综合决策阶段。信息收集阶段主要负责收集本时间段内的事务执行情况，其中包括事务冲突情况、事务读写状态、先验事务队列等信息。而综合决策阶段则是根据收集到的信息以及上个时间段决策反馈结果来预测冲突率，综合决策下一个时间段内需要采用乐观并发控制策略还是悲观并发控制策略。

3.2 信息收集阶段

ACC-PRC 算法的第一阶段为信息收集阶段。该阶段主要任务为收集本时间段内的事务执行情况。其中事务冲突情况用来预测下个阶段的冲突率，事务的读写状态和先验事务队列用来保证事务执行的可串行化。

3.2.1 冲突循环队列

当冲突率比较低时，乐观并发控制算法的运行效果比悲观并发控制算法好，否则，悲观并发控制算法更有优势。为了决策每个时间段内应该采用哪种并发控制算法，需要计算每个时间段内的冲突率。本文采用两个循环队列($CQ1$、$CQ2$)来记录最近 n 个时间段内事务的执行情况。循环队列 $CQ1$ 用来记录事务发生冲突的情况，$CQ1$ 中的元素 $NQ1$ 为一个时间段内的事务冲突数量。循环队列 $CQ2$ 用来记录数据库收到的事务请求数量，$CQ2$ 中的元素 $NQ2$ 为一个时间段内的事务请求数量。时间段 i 的冲突率 CR_i 可以用式(1)表示：

$$CR_i=\frac{NQ1_i}{NQ2_i} \tag{1}$$

图 2 给出了冲突循环队列的一个实例和最近 8 分钟内的冲突情况。

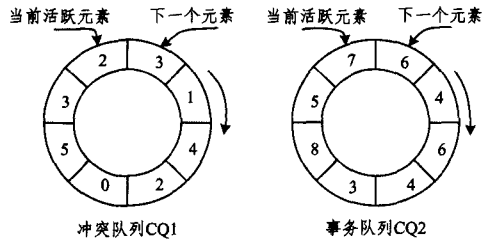

图 2　冲突循环队列实例

3.2.2　先验事务队列

本文采用先验事务队列来保证事务在执行时按照正确的读写顺序执行，即保证事务调度的可串行化。

定义 1　事务 T_i 的先验事务队列 PTQ_i 定义如下：

$$PTQ_i=\{T_0,\cdots,T_i,\cdots,T_j\}$$

其中，$\forall T_m\in PTQ_i$，T_m 是活跃事务，且 $T_m\prec T_i$。事务的偏序关系用二元关系$\prec$表示。

如果事务 T_m 与 T_i 的读写发生冲突且 $T_m\prec T_i$，则可用式(2)更新事务 T_i 的先验事务队列 PTQ_i。

$$PTQ_i=PTQ_i\cup T_m\cup PTQ_m \tag{2}$$

定义 2　系统的先验事务队列 PTQ 定义如下：

$$PTQ=\{PTQ_0,PTQ_1,\cdots,PTQ_m\}$$

其中，PTQ_i 表示当前系统中活跃事务 T_i 对应的先验事务队列。

使用先验事务队列检测事务执行的可串行化的算法思想及步骤如下：

Step1　当事务 T 的操作 Op 提交时更新该事务的先验事务队列；

Step2　如果该事务在其先验事务队列中则转 Step3，否则转 Step4；

Step3　事务不满足可串行化要求，将事务 T 加入到冲突事务集合 $CTOp$ 中，重启冲突事务集合中优先级最高的事务；

Step4　事务操作满足可串行化要求，执行事务。

ACC-PRC 的可串行化检测算法的伪代码如下所示。

```
//输入参数:事务操作 Op
//算法功能:检测事务操作 Op 是否符合可串行化
int Fun_Checkserial(Op){
  update(PTQT);
//将与操作 Op 相关的事务加入到事务 T 的先验事务队列
  if(T is in PTQT){//事务 T 在其先验事务队列中
    put T in CTOp;//将事务 T 加入到冲突事务集合
  restart Ti;//重启冲突事务集中优先级最高的事务 Ti
  return FALSE;
  }else{//事务 T 不在其先验事务队列中
  Execution Op;//执行事务操作 Op
  return TURE;
  }
}
```

3.3　综合决策阶段

ACC-PRC 算法的第二阶段为综合决策阶段，该阶段根据收集到的信息以及上个时间段的决策反馈结果来预测冲突率，综合决策下一个时间段内需要采用乐观并发控制策略或悲观并发控制策略。综合决策阶段的重点在于准确预测下个阶段的冲突率。本节将重点介绍冲突预测方法和乐观/悲观策略选择方法。

3.3.1　冲突率预测

ADCC 算法采用线性回归方程[7]来推导系统当前的冲突率，虽然考虑了预测结果的滞后性，但是线性回归方程只能适用于冲突率变化稳定的系统，而对非线性、随机性较大的冲突率变化的预测效果不佳。本文考虑采用改进的加权移动平均法来预测下一时间段的系统冲突率。

加权移动平均法以时间为标准，对离预测时刻越近的数据赋予越大的权值，这样可以弥补移动平均法对所有数据平等对待的不足，而且能够敏感地反应数据近期的趋势。该方法具有逻辑简单、预测精度高等特点，特别适用于短期趋势预测，基本的计算公式[8]为：

$$E_n=\frac{\sum_{i=1}^{n}a_iCR_{n-i}}{\sum_{i=1}^{n}a_i} \tag{3}$$

其中，E_n 表示第 n 次预测结果；CR_{n-i} 表示第 $n-i$ 个时间段检测到的冲突率；i 表示参考值个数；a_i 表示第 i 个参考值的权重。

加权移动平均法虽然对短期预测精确度要求较高，所预测的值更接近实际值，但是由于权值是预先设定的，预测值的准确性不能得到有效的反馈，这也限制了加权移动平均法精度的提高。本文考虑在现有公式的基础上引入反馈值的概念，将每次预测的结果反馈到下一次的预测中。反馈值的计算公式为：

$$F_n=\frac{\sum_{m=1}^{m}w_m\frac{E'_{n-m}}{CR_{n-m}}}{\sum_{m=1}^{m}w_m} \tag{4}$$

其中，F_n 表示第 n 次的反馈值；E'_{n-m} 表示第 $n-m$ 次的最终预测值；w_m 为权重值；F_n 大于 1 表示预测值偏大，F_n 小于 1 表示预测值偏小。

反馈值的概念是基于加权移动平均法的计算值提出的，最终的预测值等于加权移动平均值与反馈值的比值，预测值的计算公式为：

$$E_n'=\frac{E_n}{F_n} \tag{5}$$

最终预测值的具体计算步骤如下：

Step1　初始化数据，为各个观测值附加权重值 a_i；

Step2　根据式(3)计算加权移动平均值；

Step3　根据式(4)计算反馈值；

Step4　根据式(5)计算最终预测值；

选取合适的权重值对最终的预测结果有很大影响，考虑到冲突率预测有明显的时间特性，离预测点越近的观测值对结果的影响越大，本文采用衰减因子 k 来确定权值。权值的初始值 a_i 为 1。权值的计算公式为：

$$a_i=\frac{a_{i-1}}{k} \tag{6}$$

3.3.2　策略选择

ADCC 算法使用单个阈值 Eh 来决策系统策略[7]，当系统冲突率在阈值附近时需要频繁切换策略，不仅增加了系统额外的开销，而且不利于系统的稳定运行。本文采用双向阈值来决策系统策略，为悲观并发控制策略和乐观并发控制策略分配不同的阈值 $Eh1$ 和 $Eh2$，并要求 $Eh1>Eh2$，当预测冲突率大于 $Eh1$ 且当前系统为乐观并发策略时切换到悲观并发策略，当预测冲突率小于 $Eh2$ 且当前系统为悲观并发策略时切换到乐观并发策略，其他情况保持系统策略不变。

策略算法的流程如图 3 所示。

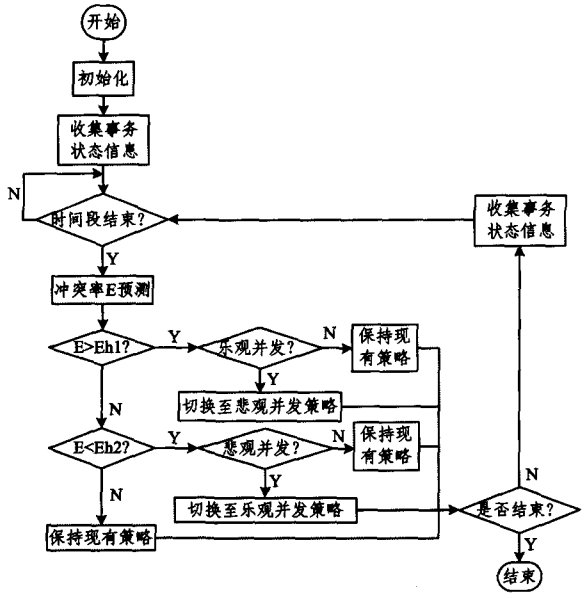

图 3　策略算法流程图

具体的策略选择算法如下：

Step1　初始化算法，将系统初始并发策略设置为乐观并发策略，循环冲突队列置 0。

Step2　当时间段结束时，收集本时间段内的事务执行状态，包括事务提交个数与冲突个数，将其加入循环冲突队列 *CQ*1，*CQ*2。

Step3　根据冲突率预测方法算出下一时间段的冲突率 E，若 $E>Eh1$，转 Step4；若 $E<Eh2$，转 Step5；否则转 Step6。

Step4　若当前系统为乐观并发策略，则将系统切换为悲观并发策略，执行完之后转 Step5。悲观并发策略中要求在事务执行之前判断事务操作是否有冲突，有冲突则执行优先级较高的事务，另一个事务进入等待状态，无冲突则正常执行。

Step5　若当前系统为悲观并发策略，则将系统切换为乐观并发策略，执行完之后转 Step5。乐观并发策略中在事务执行之前不判断是否有冲突，当有冲突发生时重启发生冲突的两个事务。

Step6　判断是否结束算法，如果不结束转 Step2，否则退出算法。

4　仿真实验与结果分析

为了测试 ACC-PRC 算法在并发事务中的性能，本文进行了仿真实验，实验环境为 3.20GHz 下的 Win7 系统，仿真实验参数如表 1 所列。

表 1　仿真实验参数表

仿真参数	值
数据规模(DatabaseSize)	500
事务到达率(ArrivalRate)	10－20
事务操作数(OperationNumber)	1－8
循环队列长度(CyclicQueueLength)	2－10
阈值 Threshold Eh1/Eh2	0－1

数据规模表示数据库中数据对象的数量。事务到达率表示每秒钟数据库收到的事务请求数。事务操作数表示每个事务中操作的数量。与本文提出的 ACC-PRC 算法对比的算法为前文提及的 HCC 算法和 ADCC 算法。算法性能的主要测试参数为预测偏差率、事务平均执行时间。其具体计算方法如下：

$$\text{预测偏差率(Deviation Rate)}=\frac{|\text{预测值}-\text{实际值}|}{\text{实际值}}$$

$$\text{事务平均执行时间(Average Execution Time)}=\frac{\text{事务的执行总时间}}{\text{进入系统的事务总数}}$$

本文使用 Matlab 编写所有的仿真程序，实验中根据帕累托法则，80%的事务请求集中在 20%的数据上操作。

表 2 给出了 ACC-PRC 算法在选用不同的循环队列长度以及衰减系数时的预测结果对比。实验中每次实验重复 3 次，预测偏差率取 3 次实验的平均值。由表可知 ACC-PRC 算法在循环队列长度 *CQL* 为 5，衰减因子 $k_1=1.5$，$k_2=1.75$ 时预测精度最高，因为冲突率预测有明显的时间性，统计周期数即循环队列长度过长，会使算法对冲突率变化的敏感度降低，预测精度降低。

表 2　不同参数下算法的预测偏差率表

CQL	k1	k2	DR	CQL	k1	k2	DR
2	1.5	1.5	4.73%	5	1.25	1.25	3.75%
3	1.5	1.5	4.02%	5	1.25	1.5	3.29%
4	1.5	1.5	3.59%	5	1.25	1.75	3.19%
5	1.5	1.5	3.24%	5	1.25	2	3.83%
6	1.5	1.5	3.75%	5	1.5	1.25	3.46%
7	1.5	1.5	4.38%	5	1.5	1.5	3.24%
8	1.5	1.5	5.12%	5	1.5	1.75	3.09%
9	1.5	1.5	6.23%	5	1.5	2	3.63%
5	2	1.25	4.21%	5	1.75	1.25	3.92%
5	2	1.5	3.86%	5	1.75	1.5	3.57%
5	2	1.75	3.63%	5	1.75	1.75	3.68%
5	2	2	4.04%	5	1.75	2	4.10%

图 4 为 3 种算法在预测精度方面的对比。图中横坐标为时间(单位：min)，实验中事务的到达率随着时间的增长而增长，但存在小幅的震荡。纵坐标为冲突率。从对比图中可以看出 3 种算法中 HCC 算法的精确度最高，ACC-PRC 算法相比于 HCC 算法精度稍低但是相差不大，ADCC 算法精确度最差。ADCC 算法采用线性回归的方式来预测冲突率，在冲突率剧烈变化时预测值偏差较大，具有一定的滞后性。HCC 算法采用 ART2 神经网络预测冲突率，神经网络通过多次迭代识别已存储的模式，实现预测，具有较高的精度，但神经网络多轮迭代牺牲了一定的时间效率。而 ACC-PRC 算法计算模型简单，在保证精度的情况下提升了效率。

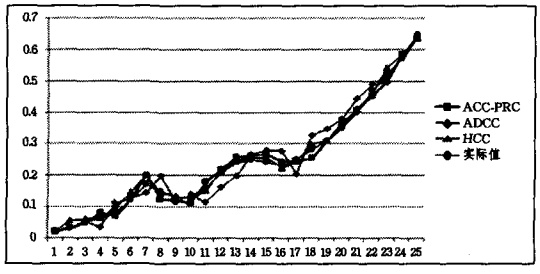

图 4　算法预测精度对比图

图 5 为 3 种算法中事务执行的平均时间对比图。实验时 ACC-PRC 算法的阈值 *Eh*1 和 *Eh*2 分别选取 0.3 和 0.5，ADCC 算法中阈值 *Eh* 选取 5。图中横坐标为时间(单位：min)，实验时事务的到达率随机变化，纵坐标为该时段内事务的平均执行时间(单位：ms)。从图中可以看出 ACC-PRC 算法中事务平均执行时间较短，事务执行效率较高。ADCC 算法和 HCC 算法的事务平均执行时间相差不大，在某些时刻 ADCC

(下转第 290 页)

Computer Science,2005,3399:277-289
[11] Zhang C,Naughton J,DeWitt D,et al. On Supporting Containment Queries in Relational Database Management Systems[C]// Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Santa Barbara, California, United States:ACM Press,2001:425-436
[12] Fu Li-zhen,Meng Xiao-feng. Reachability Indexing for Large-Scale Graphs:Studies and Forecasts[J]. Journal of Computer Research and Development,2015,52(1):116-129(in Chinese)
富丽贞,孟小峰. 大规模图数据可达性索引技术[J]. 计算机研究与发展,2015,52(1):116-129
[13] Chen Yang-jun,Chen Yi-bin. An Efficient Algorithm for Answering Graph Reachability Queries[C]// Proceedings of IEEE ICDE08. Piscataway,NJ:IEEE,2008:893-902
[14] Wang Hong-zhi,Li Jian-zhong,Luo Ji-zhou,et al. Hash-base Subgraph Query Processing Method for Graph-structured XML Documents[J]. Proceedings of the VLDB Endownment,2008,1(1):478-489
[15] Chen Li,Gupta A. Stack-based Algorithms for Pattern Matching on DAGs[C]// Proceedings of the 31st International Conference on Very Large Data Bases (VLDB). Trondheim,Norway:VLDB Endownment,2005:493-504
[16] Trißl S, Leser U. Fast and Practical Indexing and Querying of Very Large Graphs[C]// Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing, China:ACM Press,2007:845-856
[17] Jin Ruo-ming,Ruan Ning,Xiang Yang,et al. Path-tree:An Efficient Reachability Indexing Scheme for Large Directed Graphs [J]. ACM Trans on Database System,2011,36(1):73-84
[18] Cai Jing, Poon C. Path-hop:Efficiently Indexing Large Graphs for Reachability Queries[C]// Proc of ACM CIKM'10. New York:ACM,2010:119-128
[19] Kimelfeld B, Kosharovshy Y, Sagiv Y. Query efficiency in probabilistic XML models[C]// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver,Canada,2008:701-714

(上接第 283 页)

算法会表现出比 HCC 算法更高的时间效率。ACC-PRC 算法通过双向阈值的设置减少了事务冲突率变化导致的策略切换时间开销,保证了算法的平稳执行。

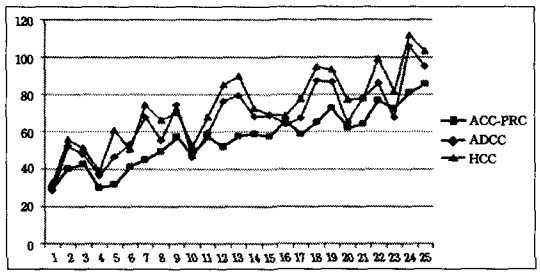

图 5 事务平均执行时间对比图

从图 4 和图 5 的对比实验中可以看出,ACC-PRC 算法具有较优的性能。HCC 算法采用 ART2 神经网络预测冲突率,虽然保证了较高的准确性,但是神经网络结构复杂,计算时间长,对于需要实时计算结果的并发情况来说,处理时间过长必然会导致事务执行效率的降低。ADCC 算法在冲突率变化剧烈时可能频繁地切换系统策略导致事务的执行效率低下。而 ACC-PRC 算法结构简单,并能够及时将预测结果反馈给系统,在保证计算精度的情况下,通过双向阈值避免了不必要的系统开销,提高了事务执行效率。

结束语 本文研究了现有的并发控制算法,并针对 HCC 算法和 ADCC 算法的不足,提出了一种新的并发控制算法:ACC-PRC 算法。该算法在 ADCC 算法的基础上将并发控制过程分为信息收集和策略选择两个阶段。信息收集阶段利用先验事务队列保证事务执行的可串行化,并且收集系统的事务执行状态。策略选择阶段在循环冲突队列上运用改进的加权移动平均法预测下一阶段冲突率,并根据双向阈值决策下一阶段的并发策略(乐观/悲观并发策略)。ACC-PRC 算法利用反馈值提高了预测冲突率的精度,并利用双向阈值避免了系统并发策略的频繁切换。对比实验表明,ACC-PRC 算法在预测精度、事务平均执行时间等方面优于 HCC 算法和 ADCC 算法。ACC-PRC 算法能够在事务到达率较高时保持良好的事务执行效率,同时能够准确及时地感知冲突率的变化。

参 考 文 献

[1] Xu Shu-ting,Sun Yong-qiang. Performance Study of Real-Time Multiversion Concurrency Control Protocols in Parallel Database Systems[J]. Chinese J. Computers, 2002, 25(2): 173-180(in Chinese)
徐淑顽,孙永强. 并行数据库实时多版本并发控制协议性能研究[J]. 计算机学报,2002,25(2):173-180
[2] Nystrom D,Nolin M,Teoanovio A,et al. Pessimistic concurrency control and versioning to support database pointers in real-time databases; proceedings of the Real-Time Systems[C]// 16th Euromicro Conference on Real-time Systems. 2004, 16: 261-270
[3] Makni A,Bouaziz R,Gargouri F. Formal Verification of an Optimistic Concurrency Control Algorithm using SPIN[C]// Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning. 2006:160-167
[4] Demsky B,Lam P. Views:Synthesizing Fine-Grained Concurrency Control [J]. Acm Transactions on Software Engineering & Methodology,2013,22(1):139-176
[5] Sheikhan M,Rohani M,Ahmadlueil S. A neural-based concurrency control algorithm for database systems [J]. Neural Computing & Applications,2013,22(1):161-174
[6] Sheikhan M,Ahmadlueil S. An intelligent hybrid optimistic/pessimistic concurrency control algorithm for centralized database systems using modified GSA-optimized ART neural model [J]. Neural Computing & Applications,2013,23(6):1815-1829
[7] Chen Y R,Zhuang Y. An Adaptive Decision Concurrency Control Algorithm [J]. Advanced Materials Research, 2014, 1046(1):512-515
[8] Liu Ya-qi,Xie Bo. Iron ore CIF price forecasting based on improved weighted moving average method[J]. Journal of Shanghai Maritime University,2015(2):55-59(in Chinese)
刘雅琪,谢波. 基于改进加权移动平均法的铁矿石到岸价格预测[J]. 上海海事大学学报,2015(2):55-59