实时主动数据库并发控制协议

雷向东, 赵跃龙, 袁晓莉

(中南大学 信息科学与工程学院, 湖南 长沙, 410083)

摘要: 提出了实时主动数据库系统实时多版本两阶段封锁并发控制协议(RTMV2PL)。该协议将多版本并发控制的优点和两阶段封锁并发控制机制优点结合起来。多版本两阶段封锁机制消除了只读事务和更新事务的冲突, 只读事物从不重启动。对实时主动数据库系统的事务优先级重新定义。通过模拟仿真与传统的 HP2PL 和 OCC-TI-WAIT-50 协议进行比较。研究结果表明, 并发控制协议不但能有效地降低事务的重启动率和延误截止时间率, 而且提高只读事物的响应时间。当事务触发率高, 导致系统负载高时, 它的性能仍比其他协议的性能好。

关键词: 实时数据库系统; 主动数据库系统; 并发控制; 多版本两阶段封锁协议
中图分类号: TP311.13 文献标识码: A 文章编号: 1672-7207(2005)05-0852-06

A concurrency control protocol in real-time active database systems

LEI Xiang-dong, ZHAO Yue-long, YUAN Xiao-li

(School of Information Sciences and Engineering, Central South University, Changsha 410083, China)

Abstract: Real-time multiversion two-phase locking (RTMV2PL) concurrency control protocol was presented for real-time active database systems(RTADBSs). The protocol presented combines the advantages of multiversion concurrency control mechanism with the advantages of two phase locking. Multiversion two-phase locking mechanism eliminates the conflicts between read-only and update transactions. Read-only transactions are never restarted. The priority of transactions is redefined for RTDABSs. The protocol presented is compared with concurrency control HP2PL and OCC-TI-WAIT-50 protocol by simulation experiments. The simulation results show that the new protocol can not only effectively reduce the miss rate of transactions and the restart rate of transactions, but also significantly improve the response time of read-only transactions. When transaction triggering probability is higher, which results in increasing system load, RTMV2PL still performs better than other protocols.

Key words: real-time database systems; active database systems; concurrency control; multiversion two phase locking protocol

实用主动数据库系统(RTADBSs)在电话通信网络管理、自动航空交通管制、自动金融交易、过程控制等领域有广阔的应用前景。近几年来, 人们对主动数据库(ADBSs)已进行大量的研究, 但对实时主动数据库研究报道较少[1-3]。在 RTADBSs 中, 事务不仅有时间约束如截止时间, 而且可能触发其他事务, 形成事务触发链。传统实时并发控制协议不适用于 RTADBSs, 主要有 2 个缺点: 首先, 传统并发控制协议解决冲突策略忽视了事务链的作用。当一个事务不得不重启动时, 被这个事务触发的

收稿日期: 2004-11-11
基金项目: 国家"863"计划资助项目(511-910-092)
作者简介: 雷向东(1964-), 男, 湖南常宁人, 博士, 从事数据库理论研究
论文联系人: 雷向东, 男, 博士; 电话: 0731-8837504(H); E-mail: leixiangdong@mail.csu.edu.cn

事务也必须重启动。实时事务处理并发控制机制必须减少处理浪费[4]。另一个缺点是事务的优先级由最早截止时间优先(EDF)方式决定。在 ADBSs 中事务的优先级必须考虑到事务触发链的影响。目前，大多数 RTADBSs 研究主要集中在乐观并发控制机制和扩展锁机制。众所周知，在乐观并发控制机制中大量的事务重启动，强烈地影响了并发控制机制性能。在锁机制中，事务可能花很长时间等待数据锁，并且优先级高的事务可能被优先级低的事务阻塞。在此，作者提出实时多版本两阶段封锁并发控制协议。该协议将多版本并发控制控制的优点和两阶段封锁优点结合起来。多版本两阶段封锁并发控制控制有一个很好的特性：读请求从不失败，而且不必等待。在典型数据库系统中，读操作比写操作频繁，这个优点对于实践来说至关重要。

1　实时主动事务执行特性

在 ADBSs 中，主动规则定义何时执行动作以及执行什么动作。主动规则的事件—条件—动作(ECA)模型使用广泛[5]。这种模型中规则通常具有如下形式：on 事件 if 条件 then 动作。当事件发生时，一个或多个规则可能触发。一旦规则被触发，规则的条件就被检查。若条件满足，规则动作将被执行。规则中动作的执行可能导致新的事件发生，这些事件又可能触发其他规则，而这些规则接着又可能触发其他规则。由此所得触发动作形成一条链。如果几条规则被同一事件触发，那么这些规则必须按某种顺序执行。事件—执行绑定定义了什么时候执行规则。因此，在触发事物和被触发事物之间存在提交依赖和流产依赖。如果事务 T_j 依赖于 T_i，称 T_i 是 T_j 的监护事务。根据以上讨论，实时主动事务执行有下列特性：

a. 在触发事物和被触发事物之间存在提交依赖和流产依赖；

b. 被触发事务串行在触发事务之后；

c. 如果一个事务重启动，触发事务必须重启动；

d. 被触发事务的截止时间比所有它的监护事务截止时间更迟。

2　新协议

2.1　多版本两阶段封锁并发控制机制

多版本两阶段封锁并发控制机制试图将多版本并发控制控制的优点和两阶段封锁并发控制机制优点结合起来，该协议区分只读事务和更新事务。只读事务在执行读操作时遵从多版本时间排序协议，更新事务执行强两阶段封锁协议，即持有全部锁直到事务结束，事务可以按其提交的顺序串行化。多版本技术消除了只读事物和更新事物之间的冲突，因为只读事物总是读提交后的版本，不与更新事物竞争资源。因此，只读事物从不重启动，事物重启动数量显著减少。

每个事务 T_i 与 1 个时间戳关联，记为 $T_S(T_i)$，该时间戳在事务开始前赋予。对每个数据项 Q，有一个版本序列< Q_1，Q_2，…，Q_m > 与之关联。每个版本 O_k 包含 3 个数据字段：

a. content 是 Q_k 的版本值；

b. $W_{TS}(Q_k)$ 是创建 Q_k 版本的事务时间戳；

c. $R_{TS}(Q_k)$ 是所有成功读取 Q_k 版本的事务的最大时间戳。只读事务在执行读操作时遵从多版本时间排序协议，因此，只读事务 T_i 发出读数据项 Q 时，返回值是时间戳小于 $T_S(T_i)$ 最大版本 Q_k 的内容。显然，一个事务读取在它之前的最近版本。

当更新事务读取一个数据项时，它在获得该数据项的共享锁后读取该数据项最新版本值。当更新事务想写一个数据项时，它首先要获得该数据项上的排他锁，然后为此数据项创建一个新版，写操作在新版本上进行。新版本的时间戳最初置为 ∞，它大于任何可能的时间戳，在创建此版本的事务 T_i 完成之前，阻止其他只读事务对此版本的读操作。

2.2　优先级认定

因为实时主动事物具有实时和主动特性，最早截止时间优先(EDF)的事务优先级策略不适应于实时主动事务。为设计优先级敏感的协议，事务的优先级需要重新考虑。图 1 所示为 5 个并发事务 T_1，T_2，T_3，T_4 和 T_5 执行轮廓。在时间 t_2，T_1 触发事务 T_2。在时间 t_4，T_1 触发事务 T_4。在时间 t_5，T_3 触发事务 T_5。在 OCC 机制由于 T_3 的截止时间早，T_3 的优先级高于 T_1。如果 T_1 和 T_3 有不可调和的冲突，决定重启动 T_1，T_2 和 T_4 也必须重启动。重启动决定忽略了 T_2 和 T_4 的截止时间。T_2 和 T_4 实际上没有机会完成，因而错过截止时间。在这种情况下，如果 T_3 重启动，T_5 重新触发，T_5 比 T_2 和 T_4 有比较好机会完成，浪费的处理时间相对比较低。因此，在 RTADBSs 设计优先级敏感的协议，事务的优先级需要重新考虑。

从上面的例子中可以发现，传统的并发控制忽略了事务链的影响。主动数据库系统中实时并发控

制协议中，事务优先级不能使用 EDF 策略。显然，随着依赖监护事务的事务数量增加，重启动监护事务的代价越大。因此，随依赖事务(记为 $D_{ep}(T_i)$)的增加，监护事务 T_i 的优先级应提高。事务 T_i 的优先级记为 $P(T_i)$。

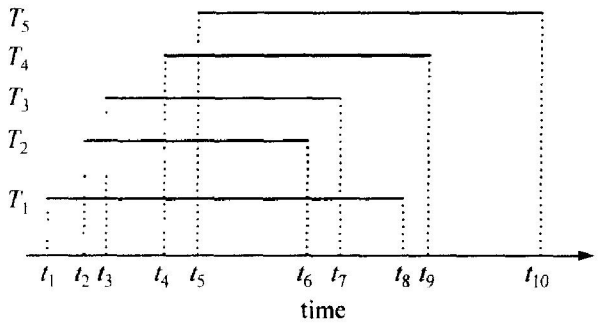

图 1 5 个并发事务 T_1，T_2，T_3，T_4 和 T_5 执行轮廓

Fig. 1 Execution profiles of five concurrent transactions

2.3 减少错过截止时间事务数量

错过截止时间的事物数量强烈地影响了实时并发控制机制的性能。为了提高协议性能，协议必须减少错过截止时间事务数量。在 RTADBSs 中，当事务不得不重启动，被触发的事务也必须重启动。解决冲突时，必须考虑触发链的影响。在图 1 中，如果事务 T_3 和 T_1 有不可调和的冲突，并且 T_1 被重启动，T_2 和 T_4 也必须重启动。T_2 和 T_4 实际上没有机会完成。如果 T_3 被重启动，将有较好的机会来完成。从上面例子可看到一个更有效的解决冲突的方法。假设更新事务 T_i 在数据项 Q 上申请排他锁，事务 T_j 持有排他锁，事务 T_i 优先级高于 T_j，并且依赖 Tj 的事务重动启动不会错过截止时间，并发控制协议重启动 T_j，事务 T_i 获得在数据项 Q 上排他锁，其他情况 T_i 等待。重启动规则描述如下:

```
if( ∀Tk(Tk∈Dep(Tj) && ((Et(Tk)+ Ct)< Dl(Tk))) && P(Ti)> P(Tj))
{
  restart Tj;
  Ti obtains exclusive lock on data item Q;
}
else
{
  Ti wait;
}
```

其中，$E_t(T_k)$ 为事务 T_k 的估计运行时间，$D_l(T_k)$ 为事务 T_k 的截止时间。

在 RTADBSs，更新事务执行时间包括 CPU 操作时间，等待数据项可用时间和 I/O 操作时间[6]。访问一个数据项的时间包括由于 CPU 竞争等待 CPU 时间(Wait CPU Time)，CPU 服务时间 T_{CPU}，等待数据项可用时间 W_{lock}，由于磁盘竞争等待磁盘时间 W_{disk} 和磁盘服务时间 T_{disk}。因此，访问 $N_{data}(T)$ 个数据的事务估计执行时间为:

$$E_t(T) = N_{data}(T) \cdot (W_{CPU} + T_{CPU} + W_{lock} + R_{ph} \cdot B_t + (1 - R_{ph}) \cdot (W_{disk} + T_{disk}))。$$

在多版本两阶段封锁并发控制中机制只读事物总是读提交后的版本，不与更新事物竞争资源。因此，访问 $N_{data}(T)$ 个数据的只读事务估计执行时间为:

$$E_t(T) = N_{data}(T) \cdot (W_{CPU} + T_{CPU} + R_{ph} \cdot B_t + (1 - R_{ph}) \cdot (W_{disk} + T_{disk}))$$

其中，R_{ph} 为存储系统页命中率。协议持续跟踪这些数据，得到其最近平均值。

3 协议算法

根据上述讨论，作者提出了实时多版本乐观并发控制协议(RTMVOCC)。协议主要由 3 个函数组成: read-only-transaction(T_i)，update-transaction(T_i)和 scheduler (T_i, Q)。函数 read-only-transaction(T_i)在只读事务 T_i 读数据项时运行，函数 update-transaction(T_i)在更新事务访问数据项时运行，当事务 T_i 对数据项 Q 加锁发生冲突时函数 scheduler(T_i, Q)进行仲裁。3 函数描述如下:

```
void read-only-transaction(Ti)
{
  while(! End of Ti && Ct< = Dl(Ti))
  {
    if (op= read(Q))
    {
    select Qk that Ts(Qk) is the largest timestamp less than Ts(Ti);
    read(Qk);
    }
  }
  if(End of Ti && Ct< = Dl(Ti))
  {
    if(any guardian transaction of Ti is not
```

```
committed)
            exit;
          else
            commit Ti;
          }
        }

        void update-transaction(Ti)
        {
          while(! End of Ti && Ct< = Dl(Ti))
          {
          if(op= read(Q))
          {
            request shared-lock(Q);
            if(Ti obtains shared-lock(Q))
            {
              select the latest version Qk;
              read(Qk);
            }
            else
              Ti wait;
          }
      if(op= write(Q))
      {
          request exclusive-lock(Q)
          if(Ti obtains exclusive-lock(Q))
          {
            create a new version Qk of Q;
            Ts(Qk) ← ∞;
          }
          else
            Ti waits;
        }
      }
      if(End of Ti && Ct< = Dl(Ti))
      {
          if(any guardian transaction of Ti is not
committed)
            exit;
          else
          {
            set the timestamp on every version crea-
ted to Ts(Ti);
            release all locks held by Ti;
            commit Ti;
          }
        }
      }
      void scheduler(Ti, Q)
      {
          look for Tj that holds lock on Q;
          flag= FALSE;
        if(P(Ti)> P(Tj))
        {
          flag= TRUE;
          for each Tk ∈Dep(Tj) do
          {
          If((Et(Tk)+ Ct)> = Dl(Tk))
            flag= FALSE;
          }
        }
        if(flag= = TURE)
        {
          Ti obtains lock on Q;
          restart Tj;
        }
        else
        {
          Ti waits;
        }
      }
```

4　性能评价

通过仿真实验评价新协议的性能。RTADBSs 由共享内存多处理机组成，数据平衡分布在各个磁盘上，数据库系统在虚拟页式管理上实现。当事务试图访问页时，系统使用 R_{ph} 概率决定这页是否在主存中。每个事务 T_i 的 截止时间由下面公式给出:

$$D_l(T)= P_{et}(T)\times S_f+ A_t(T)\ 。$$

这里 $P_{et}(T)$ 是预计执行时间，它是事务长度的函数，S_f 是松弛因子，$A_t(T)$ 是事务 T 到达时间。测试的主要参数是在不同事务到达率时事务延误截止时间率(miss rate) 和事务重启动率(restart rate)。模拟程序用 C + + 编程实现，基于 SIMPACK 工具[7]。仿真模拟实验主要参数: CPU，4 个; 磁盘，6 个; CPU 时间为 10 ms/页; 磁盘时间为 20 ms/页; 页命中率为 0.8; 数据库页数为 1 000 页; 数据项为 10/页; 事务触发率(P_{tri}) 为 0.05; 松弛因子(S_f) 为 2~ 5。

为了比较并发控制协议的性能，选用了 2 个协

议作为基准协议: 一个是 HP2PL[8, 9], 另一个是 OCC-TI-WAIT-50[10, 11]。HP2PL 是广泛使用的基于锁的并发控制协议。在 HP2PL 协议中, 若申请锁事务优先级高于持锁的事务优先级, 则申请锁事务获得锁; 否则, 申请锁事务等待。OCC-TI-WAIT-50 是著名的乐观并发控制协议。OCC-TI-WAIT-50 协议使用了"50% 原则", 即当冲突集中半数以上事务比进入有效性检查事务优先级高时, 该事务等待; 否则, 令冲突集事务重启动。

图 2 和图 3 所示为系统负载对协议性能的影响。可见, 提出的并发控制协议的性能要好于 HP2PL 和 OCC-TI-WAIT-50 协议的性能。当事务到达率低时(正常系统负载), RTMV2PL 协议延误截止时间率和重启动率比 HP2PL 协议和 OCC-TI-WAIT-50 低得多。例如, 当事务到达率为每秒 12 个事务时, RTMV2PL 的延误截止时间率只有 15.6%, OCC-TI-WAIT-50 的延误截止时间率高于 32%, HP2PL 的延误截止时间率高于 48%。多版本机制消除了只读事务和更新事务的冲突, 只读事物从不重启动, 事务重启动数显著减少。因此, 事物延误截止时间率大大降低, 协议性能提高。

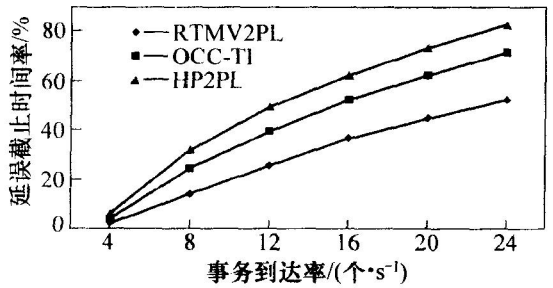

图 2 延误截止时间率与事务到达率的关系曲线

Fig. 2 Miss rate of transaction vs. arrival rate of transaction (baseline)

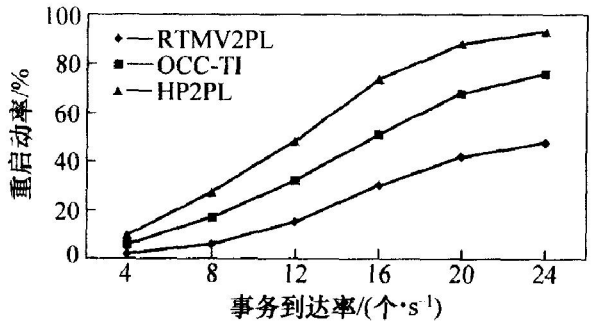

图 3 重启动率与事务到达率的关系曲线

Fig. 3 Restart rate of transaction vs. arrival rate of transaction (baseline)

图 4 和图 5 所示为高事物触发率对协议性能的影响。高事物触发率导致出现高系统负载。RTMV2PL 有相对较低的曲线, 说明 RTMV2PL 比其他协议有较低的事务重启动。当事务到达率超过系统超负载点时(大约每秒 20 个事务), 事务的重启动率急剧下降。这是因为系统已不能调度和管理所有进入的事物, 由于资源竞争, 事物在 CPU 和磁盘队列上等待, 实际上花在操作上的时间很少。在事务触发率高时, RTMV2PL 的性能仍然比其他协议好。

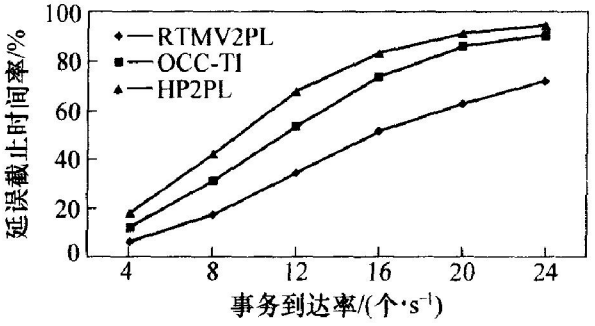

图 4 延误截止时间率与事务到达率(高事务触发率)的关系

Fig. 4 Miss rate vs. arrival rate with high transaction triggering probability

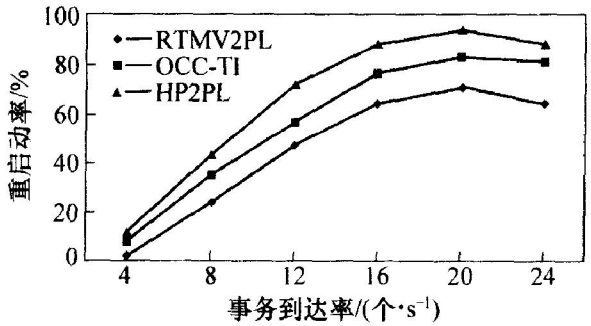

图 5 重启动率与事务到达率(高事务触发率)的关系

Fig. 5 Restart rate vs. arrival rate with high transaction triggering probability

5 结 论

a. 研究了实时主动事务执行模型, 由于事务具有主动特性, 传统实时并发控制协议不适应于 RTADBSs。

b. 提出了实时多版两阶段封锁并发控制协议(RTMV2PL)。多版本两阶段封锁机制消除了只读

事务和更新事务的冲突，只读事物从不重启动，事务重启动数显著减少。

c. 通过仿真实验比较了 RTMV2PL，HP2PL 和 OCC-TI-WAIT-50 共 3 种协议的性能。研究结果表明，当事务到达率低时(正常系统负载)，RTMV2PL 协议延误截止时间率和重启动率比 HP2PL 协议和 OCC-TI-WAIT-50 低得多。在高事物触发率时 RTMV2PL 有相对较低的曲线，说明 RTMV2PL 比其他协议有较低的事务重启动。RTMV2PL 性能仍然比其他协议好。

参考文献:

[1] Carey M, Jauhari R. On transaction boundaries in active database: A Performance perspective [J]. IEEE Trans Knowledge and Data Engineering 1991, 3(3): 320－336.

[2] Botzer D, Etzio O. Self-tuning of the relationships among rule's components in active databases systems [J]. IEEE Trans Knowledge and Data Engineering, 2004, 16(3): 375－379.

[3] Datta A, Sang H S. A study of concurrency control in real-time active database systems [J]. IEEE Trans Knowledge and Engineering, 2002, 14(3): 465－484.

[4] Datta A, Son S H. Limitations of priority cognizance in conflict resolution for firm real-time database systems [J]. IEEE Trans Computers, 2000, 49(5): 483－501.

[5] Silberschatz A, Korth H F. Database System Concepts [M]. New York: McGraw-Hill, 1997.

[6] LEI Xiang-dong, YUNAN Xiao-li. Validation concurrency control protocol in real-time parallel database systems [J]. Journal of Central South University of Technology(English Edition), 2002, 9(3): 197－201.

[7] Ozsoyoglu G, Snodgrass R. Temporal and real-time databases: A Survey [J]. IEEE Trans Knowledge and Data Engineering, 1995, 7(4): 513－532.

[8] Kao B, Lam K－Y. Maintaining temporal consistency of discrete objects in soft real-time database system [J]. IEEE Trans Computers, 2003, 52(3): 373－388.

[9] Fishwick P A. SIMPACK: Getting started with simulation programming in C and C++ [R]. Gainesville: Computer and Information Sciences, Univ of Florida, 1992.

[10] Abbott R, Garcia-Molina H. Scheduling real-time transactions: a performance evaluation [J]. ACM Trans Database System, 1992, 17(3): 513－560.

[11] Ulusory O. Analysis of concurrency control protocols for real-time database systems [J]. Information Sciences, 1998, 11(1): 19－47.