文章编号:2095-476X(2015)3/4-0090-04

嵌入式混合型实时事务并发控制研究

陈亚峰[1], 王艳军[2], 李蔚[2]

(1.河南省轻工业学校 计算机系, 河南 郑州 450000;
2.郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450001)

摘要:针对嵌入式混合型实时事务并发控制新特点,引入替代和补偿,提出一种混合型并发控制算法,即无冲突混合并发控制MCC-CCCP算法.该算法将冲突分为类内和类间两种,利用ET-CCCP和CCCP协议分别解决硬实时事务类内和软实时事务类内的冲突;设计的LC-CCCP算法通过比较选出冲突数量最少的替代参与到并发控制中,同时采取一定的补偿措施降低冲突数量,以解决不同类间冲突.模拟实验表明,该混合型算法降低了事务重启率和错失率,具有一定的优越性.

关键词:嵌入式实时数据库;实时事务;混合型并发控制
中图分类号:TP311.13 **文献标志码**:A **DOI**:10.3969/j.issn.2095-476X.2015.3/4.020

Research on embedded hybrid real-time transaction concurrency control

CHEN Ya-feng[1], WANG Yan-jun[2], LI Wei[2]

(1. *Deptartment of Computer, School of Light Industry in He'nan Province, Zhengzhou* 450000, *China*;
2. *College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450001, *China*)

Abstract: Based on new characters of embedded hybrid real-time transaction concurrency control, a hybrid concurrency control algorithm, namely mixed concurrency control-CCCP (MCC-CCCP), was proposed by adding substitution and compensation. Conflict was divided into the intra-class and inter-class in the algorithm. ET-CCCP protocol solved the intra-class conflict of hard real-time transaction, while CCCP protocol solved the intra-class conflict of soft real-time transaction. The LC-CCCP algorithm was designed to resolve the inter-class conflicts by comparing to select the least number of alternative involved in the concurrency control and taking some compensation measures to reduce the number of conflicts. Simulation experiments showed that the hybrid algorithm decreased the rate of restart and transaction miss, which had certain advantages.

Key words: embedded real-time database; real-time transaction; hybrid concurrency control

0 引言

在现实世界,实时事务通常是混合并存的[1].混合型实时事务包含真实时事务和非实时事务两部分.硬实时事务、软实时事务和固实时事务称为真实时事务[2].

嵌入式混合型实时事务混合并存各类截止期不同的实时事务[3],对不同类型的实时事务的处理需要采取不同的策略.单一类型的实时事务并发控制策略一般对事务到达方式、系统性能评价及事务对数据的访问方式有着特殊要求,所以无法满足嵌入式混合型实时数据库系统的要求.目前,混合型

收稿日期:2014-11-22

作者简介:陈亚峰(1982—),男,河南省项城市人,河南省轻工业学校讲师,主要研究方向为嵌入式数据库.

实时事务的处理已经引起国内外研究者的重视,主要从类内和类间两种类型的冲突出发,采取不同的策略进行控制. K. Y. Lam 等[1]提出一个二层并发控制模型,其中主并发控制主要解决类间冲突,从并发控制着重处理类内冲突;V. C. S. Lee 等[4]提出动态调整时间戳 OCC-DA 算法以避免不必要的重启;L. Sha 等[5]定义了一个嵌入式混合型实时数据库系统的框架,对不同类型事务采取不同的并发控制策略,对类间事务冲突采取忽略的措施;K. Y. Lam 等[6]在经典 PCP 策略的基础上,提出了适应混合事务的 RCP 策略,对硬实时事务和软实时事务分别采取不同并发控制算法,而当出现类间冲突时优先考虑硬实时事务;王强等[7]提出了一种新的并发控制协议 MCC-DATI. 以上并发控制协议在特定环境下具有一定的优越性,其中无冲突并发控制协议 CCCP[8]在实时事务处理中最具代表性,但是通用性不够强,系统资源未得到有效利用.

鉴于此,本文拟提出一种新的嵌入式混合型并发控制算法,即无冲突混合并发控制 MCC-CCCP 算法,以提高系统的可靠性和事务的成功率,更适用于嵌入式混合型实时数据库系统.

1　MCC-CCCP 算法思路

本算法考虑硬实时事务、软实时事务混合并存的嵌入式混合型实时事务,将冲突分为类内和类间两种. 硬实时事务内部之间采取 ET-CCCP 协议[9],加大了相容性检测范围,选取合适的替代予以执行,以保证事务得以顺利提交,减少事务重启的次数,使该类实时事务尽可能地满足截止期的要求. 软实时事务内部之间采取传统的 CCCP 协议,引入替代,为就绪事务的执行提供较多时间,以提高系统的吞吐率. 对类间冲突设计了最少冲突的 CCCP 协议 LC-CCCP(least conflict-CCCP),针对不可避免的冲突,通过比较选出冲突数量最少的替代参与到并发控制中,以达到冲突数量降到最低的目的. 在消解相互之间的冲突时,采取一定的补救手段.

2　LC-CCCP 算法设计与实现

LC-CCCP 算法充分考虑并借鉴了 CCCP 协议引入事务替代的特性和基于推测的并发控制协议能提供稳定性能的优点. 首先分析处在接纳队列事务的所有替代与活跃事务是否相互冲突:如果其中存在某个替代与活跃事务集中的所有事务均不相互冲突,那么该替代被选出并参与并发控制;相反,如果不存在这样的替代,则分别确定所有替代的冲突集,经过比较选出冲突集中冲突数量最少的替代参与并发控制. 同时对硬实时事务引入一定的补偿手段并采取一定的补救措施.

2.1　相关说明

1)活跃事务集. 活跃事务集用 $Active(T)$ 表示,并且 $Active(T) = Run(T) + Ready(T)$. 其中,$Run(T)$ 为正在执行的事务集,$Reday(T)$ 为处在就绪状态的事务集,$Accepted(T)$ 为处在接纳状态的事务集.

2)活跃替代集. 活跃替代集用 $Active(FX)$ 表示,并且 $Active(FX) = Run(FX) + Ready(FX)$. 其中,$Run(FX)$ 为处在执行状态的替代集,$Ready(FX)$ 为处在就绪状态的替代集.

3)对于某活跃事务 T_i,至少有一个替代是活跃替代,即

$$T_i \in Active(T) \Rightarrow FX_i \in Active(FX)$$

4)若替代 FX_i 与另一个替代 FX_j 存在冲突操作,就称这两个替代互为冲突操作,并记作 $CT(FX_i, FX_j) = 1$;相反,则称 FX_i 与替代 FX_j 互为非冲突替代,并记作 $CT(FX_i, FX_j) = 0$.

5)如果事务 T_i 和另一个事务 T_j 存在某方面冲突操作,就称两个事务互为冲突事务,并记作 $CT(T_i, T_j) = 1$;反之,则称 T_i 和 T_j 互为非冲突事务,并记作 $CT(T_i, T_j) = 0$.

6)如果处在接纳状态队列中的事务 T_i 与某一活跃事务 T_j 互为冲突事务,则对于该接纳事务 T_i 的一切替代都与该活跃事务 T_j 处在活跃状态的替代互为冲突替代,即

$$CT(T_i, T_j) = 1, T_i \in Accepted(T), T_j \in Active(T) \Rightarrow \forall FX_i \in T_i, FX_j \in T_j, CT(FX_i, FX_j) = 1$$

7)如果处于接纳状态队列中的事务 T_i 和某一活跃事务 T_j 互为非冲突事务,那么对于该接纳事务 T_i,总会存在某一替代与该活跃事务 T_j 处在活跃状态的替代互为非冲突替代,即

$$CT(T_i, T_j) = 0, T_i \in Accepted(T), T_j \in Active(T) \Rightarrow \exists FX_i \in T_i, FX_j \in T_j, CT(FX_i, FX_j) = 0$$

8)若 FX_i 为处在接纳状态队列中 T_i 的某一替代,则与其冲突的活跃事务组成的全集记为 $CT(FX_i)$,并记 $Num(CT(FX_i))$ 为 $CT(FX_i)$ 中活跃事务成员的数量.

2.2　具体实现

算法实现如下:

For($i=1$;$T_i \in AcceptedQueue(T)$
&&$i \leqslant Num(AcceptedQueue(T))$;$i++$)

//对接纳队列中的所有事务,如果其中存在某个替代与活跃事务集中的所有事务均不相互冲突,那么该替代进入就绪队列. 其中 $AcceptedQueue(T)$ 表示的是接纳队列.

{For($y=1$&&$a=0$;$FX_y \in T_i$&&
$y \leqslant Num(T_i)$&&$a=0$;$y++$)
{For($j=1$&&$b=0$;$T_j \in Active(T)$&&
$j \leqslant Num(Active(T))$&&$b=0$;$j++$)
{If($CT(FX_y,T_j)=1$)
Then $b=1$;
Else $b=0$;
}
If($b=0$)
Then $a=1$,$t=y$;
}
If($a=1$)
Then join FX_t into $Ready(T)$;
Else{For($y=1$;$FX_y \in T_i$&&$y=Num(T_i)$;$y++$)
{$Num(CT(FX_y))=0$;
For($j=1$;$T_j \in Active(T)$&&$j=Num(Active(T))$;$j++$)
If($CT(FX_y,T_j)=1$)
Then
{$Num(CT(FX_y))++$;
Join T_j into $CT(FX_y)$;
}
}
$Min=Num(CT(FX_1))$;
$k=1$;
For($i=1$;$FX_i \in T_i$&&$i=Num(T_i)$;$i++$)
{If($Num(CT(FX_i))<Min$)
Then{$Min=Num(CT(FX_i))$;
$k=i$;
}
}
Select FX_k into $Ready(T)$;
}
}

2.3 补救措施

如果被夭折的活跃替代是硬实时事务的替代,为防止过载带来的灾难性后果,该算法引入了补偿,即替代在执行中已涵盖了需要的补偿,当硬实时事务的替代夭折时,调度相应的补偿任务予以执行. 该算法采取了立即补偿的方式,即当替代失败后立即调度补偿任务,当该替代的影响被消除后再执行其他替代.

3 实验结果与分析

模拟实验采用 SimPack 软件包编写,具体参数见表 1.

表 1 实验参数表

实验参数	参数值
数据库的大小	100
事务访问数据对象中的被访问到的平均页面数	5
修改被访问页的可能性	50%
页面命中率	80%
实时事务包含功能替代集的平均数	3
事务空余时间与处理时间的平均比例	1∶2
软实时事务∶硬实时事务	19∶1

图 1 为 MCC-CCCP 与 CCCP 事务重启率的比较. 由图 1 可以看出,传统 CCCP 协议的事务重启率要比 MCC-CCCP 算法的事务重启率高,而且随着事务到达数的增加,二者的差别越来越大. 其原因是:当硬实时事务与硬实时事务发生冲突时,MCC-CCCP 加大了相容性检测范围,选取合适的替代予以执行,保证了事务可以顺利提交,减少了事务重启的次数. 当软实时事务与硬实时事务之间发生冲突时,从冲突集中选取最少冲突的替代参与并发控制;此外为使事务能顺利提交,引入了补偿措施. 而传统的 CCCP 机制,一旦新接纳事务 T 中所有的替代与就绪队列中的某些事务存在资源冲突时,那么该新接纳事务 T 就不能进入就绪队列,如果时间允许将会被重启,否则夭折.

图 2 为 MCC-CCCP 与 CCCP 事务错失率的比较.

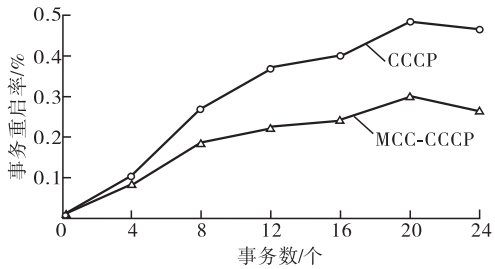

图 1 MCC-CCCP 与 CCCP 事务重启率的比较

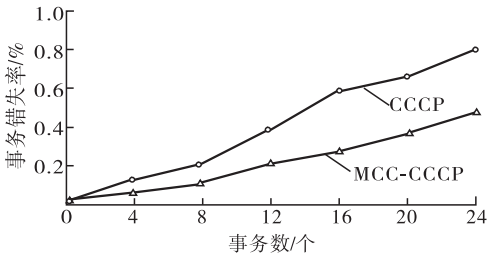

图 2　MCC-CCCP 与 CCCP 事务错失率的比较

由图 2 可以看出,MCC-CCCP 算法的事务错失率比传统 CCCP 协议的事务错失率低. 原因是:MCC-CCCP 加大了相容性检测范围,选取合适的替代予以执行,使硬实时事务尽可能地满足截止期的要求;当不同类间发生冲突时,从冲突集中选取最少冲突的替代参与并发控制,除此之外,采取补偿等措施也可使得这些实时事务能较好地满足定时限制. 而 CCCP 协议一旦检测到接纳事务队列中与就绪事务队列中存在事务冲突,那么接纳事务将不被执行并且等待,当能满足定时限制时,该事务被重新接纳;否则该事务只能被夭折. 显然混合型的并发控制算法的错失率低一些.

以上实验结果表明,从事务的重启率和事务的错失率来看,MCC-CCCP 的性能优于传统的 CCCP 协议,该算法能更好地适应嵌入式混合型实时数据库系统.

4　结论

本文提出了一种无冲突混合并发控制算法,将冲突分为类内和类间两种,LC-CCCP 协议引入了替代和补偿,通过比较选出冲突数量最少的替代参与到并发控制中,同时采取一定的补偿措施以降低冲突数量,解决不同类间冲突;硬实时事务内部之间的冲突采取 ET-CCCP 协议,软实时事务内部之间的冲突采取 CCCP 协议. 通过事务重启率和事务错失率两个指标进行模拟实验比较,结果表明该混合型并发控制算法的性能优于 CCCP 协议,更适用于嵌入式混合型实时事务.

参考文献:

[1] Lam K Y, Kuo T W, S H Lee T. Strategies for resolving inter-class data conflicts in mixed real-time database systems[J]. Journal of Systems and Software, 2002, 61(1):1.

[2] 李刚,魏涛,李蔚,等. 嵌入式数据库嵌套实时事务模型研究[J]. 郑州轻工业学院学报:自然科学版,2009,24(5):59.

[3] 梁平,张晓龙,陈和平,等. 基于实时日志的故障恢复策略[J]. 武汉大学学报:理学版,2014,60(2):160.

[4] Lee V C S, Lam K W, Hung S L. Concurrency control for mixed transactions in real-time databases[J]. IEEE Transactions on Computers, 2002, 51(7):821.

[5] Sha L, Rajkumar R, Son S H, et al. A real-time locking protocol[J]. IEEE Transactions on Computers, 1991, 40(7):793.

[6] Lam K Y, Kuo T W, Tsang W H, et al. The reduced ceiling protocol for concurrency control in real-time databases with mixed transactions[J]. Computer Journal, 2000, 43(1):65.

[7] 王强,王宏安,金宏,等. 一种面向混合实时事务调度的并发控制协议[J]. 计算机研究与发展,2005,42(1):18.

[8] 夏家莉. 嵌入式实时数据库系统中无冲突并发控制协议 CCCP[J]. 计算机研究与发展,2004,41(11):1936.

[9] 夏家莉,韩增波,陈辉. 基于功能替代模型的无冲突并发控制协议[J]. 计算机工程,2010,36(15):57.