一种高效的合作实时事务并行检验点算法*)

李国徽　王洪亚　刘云生
(华中科技大学计算机学院　武汉430074)

摘　要　许多数据和活动上都有很强时间性的应用在地理上同时具有分布性,这种应用需求使得分布式实时数据库的研究成为数据库研究领域的热点。在实时事务执行时,事务故障或数据竞争会导致事务重启,为了减少因重启而损失的工作量,可以采用检验点技术以利于事务时间正确性的满足。在一些分布式实时数据库应用中,不同结点的事务通过消息交换形成合作关系,当某一事务记检验点时,为保证合作事务间的全局一致性,相关事务也要相应地记检验点。传统的协同检验点方法没有考虑应用的定时约束,不能很好地支持分布式实时事务处理。本文提出了一种高效的并行协同检验点方法,该算法既具有最小协同检验点特性又使全局检验点过程延时最小。实验表明该算法减少了全局检验点阻塞时间,有利于分布式实时事务截止期的满足。

关键词　分布式合作实时事务,并行检验点方法,检验点依赖,全局检验点

An Efficient Concurrent Checkpointing Algorithm for Distributed Cooperative Real-Time Transaction Processing

LI Guo-Hui　WANG Hong-Ya　LIU Yun-Sheng
(School of Computer Sci. & Tech., Huazhong Univ. of Sci. & Tech. Wuhan 430074)

Abstract　Distributed real-time database systems have gained more and more research interests in the last decade. When transactions have to restart due to transaction failures and data contention, taking checkpoints periodically can reduce the amount of lost work and help real-time transactions to meet their deadlines. In some distributed real-time database applications, transactions in different sites can communicate with each other by message exchange. To maintain the consistency among transactions, when a transaction initiates a checkpoint, the elated transactions are forced to take their own corresponding checkpoints. Traditional coordinated checkpoint methods don't take the timing constraints of applications into consideration and are unsuitable for supporting distributed real-time transaction processing. This paper proposes a concurrent coordinated checkpointing method, which minimizes the time latency involved in a global checkpoint and force a minimal number of transactions to take their checkpoints. Extensive experiments show that the proposed algorithm results in the reduced time latency and an obvious reduction in the transaction deadline-missing rate as well.

Keywords　Distributed cooperative real-time transaction, Concurrent checkpointing method, Checkpoint dependency, Global checkpoint

1　引言

数据库与实时计算相结合的实时数据库已经得到越来越多的研究和应用,随着实时数据库应用的不断深入和计算机网络技术的发展,许多地理上分布的应用同时在数据和活动上又有很强的时间性,这种应用需求使得分布式实时数据库成为数据库研究领域的热点。

在实时事务的执行过程中,除了要保证传统的ACID特性,其时间正确性(定时约束的满足)的保证也非常重要[6]。在实时数据库中,事务故障和数据竞争都会导致重启,对于常见的长寿或开端实时事务,重启可能导致损失大量已做的计算而影响事务截止期的满足。检验点技术的采用可以减少事务被迫重启时工作量的损失[1],一个检验点是事务某个时刻状态的一个快照(Snapshot),它保存在存储器上,供事务故障或数据冲突时事务回滚到最近检验点而无须完全重启。

在一些分布式实时数据库应用中(如股票交易系统,通信系统和战地指挥系统等),不同结点的实时事务通过通信形成合作关系。我们称有合作关系的每个事务所记的检验点为局部检验点,所有局部检验点构成一个全局检验点。全局检验点必须是一致以保证当某一事务回滚时,与该事务有关的其它事务都回滚到一个全局一致状态[5]。

分布式计算环境中常见的检验点方法有异步检验点和协同检验点两种[2~4,6]。在异步(asynchronous)检验点方法中,各事务独立地记检验点,发生故障时恢复部件从一系列局部检验点中构造一个全局一致状态。事务记检验点时没有消息交换,但每个事务都要保留多个检验点,这对许多存储空间有限的实时应用是不合适的(如嵌入式实时应用)。恢复部件要花费时间来裁剪掉那些冗余的检验点以节省存储空间,恢复时最坏情况下可能会出现多米诺效应(所有事务回滚到最初的状态)[5],恢复时大量的消息交换也会导致恢复时间的增加,这些缺点使异步检验点方法不适合于分布式合作实时事务处理。

在协同(coordinated)检验点方法中,当一个事务记检验点时,为了保持合作事务全局状态的一致性,相关的事务都要记检验点,所以记检查点的花费相对异步方法大,但每个事务只需存储最新的检查点,减少了存储空间的要求。当某一事务回滚时,相关的事务都只回滚到最新检验点,减少了恢复时间且避免了多米诺效应,因此它适合于支持分布式合作实时事务处理。

文[2]中提出了一种非阻塞的协同检验点方法,在记检验

*)本文工作受青年国家自然科学基金项目资助(编号:60203017)。李国徽　博士,教授,主要研究方向包括移动、实时及主动数据理论及实现技术和现代数据工程。王洪亚　博士研究生,讲师,主要研究方向包括移动计算、实时及主协数据库理论及实现。刘云生　教授,博士生导师,主要研究方向包括移动、实时及主动数据库理论及实现技术和现代数据工程。

点过程中相关进程(事务)可继续执行,但文[4]发现该方法会导致不一致状态的出现。文[4]给出了最小协同检验点方法的定义,该类方法保证记检验点的进程(事务)个数最少,这样使存储空间的需求最小,但它没有过多考虑检验点过程时延问题。文[7]中提出了一种具有 O(n)消息复杂度的协同检验点方法,但它要求系统中每个进程(事务)都要记检验点,增加了那些并不需要记检验点事务的时间和空间代价。文[3]提出一个顺序检验点算法(SCA),检验点过程中顺序通知要记检验点的(进程)事务,这使得检验点过程延时较大,会严重影响具有定时约束的实时事务截止期的满足。本文提出了一种基于图论的协同检验点方法,该方法既使参与全局检验点过程的事务最少,又使检验点过程延时最小。实验表明该算法减少了检验点阻塞时间,有利于实时事务截止期的满足,提高了系统性能。

2 事务模型和检验点依赖

2.1 分布式合作实时事务的经历模型

实时事务是具有显式定时约束的事务,在一些分布实时数据库应用中,不同结点的实时事务通过消息通信结成合作事务集。在分布式合作实时事务集的执行过程中,除了对象事件、时间事件及事务事件外[8],由于事务间的消息通信有:

定义1 实时事务 t 的一个消息发送或接收原语的执行称为一个消息事件。一个消息事件总是与一事务相联,我们用 $send(m)$ 或 $receive(m)$ 表示事务发送或接受消息 m 的事件,用 ME_t 表示与事务 t 相关联的消息事件的集合。

定义2 与一实时事务相联的所有事件的事件经历称为该事务的经历,记为 H_t,形式化表示为:

$$H_t::=\langle E_t,<_t\rangle,E_t=OE_t\cup TE_t\cup CE_t\cup ME_t$$

其中 $<_t$ 为 E_t 中各事件在发生时间上的偏序关系,OE_t 为对象事件集,TE_t 为事务事件集,CE_t 为时间事件集,ME_t 为消息事件集。

定义3 设 T 为一个分布式合作实时事务集,与 T 中各事务相联的所有事件的经历称为 T 的合作执行经历,记为 H_T,形式化表示为:

$$H_T::=\langle E_T,<_T\rangle,\ E_T=OE_T\cup TE_T\cup CE_T\cup ME_T$$

$$OE_T=\bigcup_{t\in T}OEt;$$

$$TE_T=\bigcup_{t\in T}TEt;CE_T=\bigcup_{t\in T}CEt;ME_T=\bigcup_{t\in T}MEt$$

其中 $<_T$ 为 T 中各事务相联的事件在发生时间上的偏序。

当 T 中事务合作完成一个任务时,不但要保证其传统事件经历的正确性,还应保证消息事件经历的正确性。

定义4 T 的消息事件经历是正确的当且仅当在任意时刻 τ,系统中不存在孤儿消息,即对 T 中任意两个事务 $t_i,t_j(i\neq j)$有:

$$!\exists\ m,\ send(m)\notin P(H_T,t_i)\ \wedge\ receive(m)\in P(H_T,t_j)$$

其中 m 为到时刻 τ 为止 T 中事务间传递的任一消息,$P(H_T,t)$为合作事务集 T 的事务合作执行经历在 t 上的投影[8]。

在下边的讨论中,我们用 $C_{i,m}$表示事务 t_i 的第 m 个局部检验点。事务 t_i 在发送消 m 后记检验点 $C_{i,k}$表示为 $send(m)\in C_{i,k}$,反之为 $send(m)\notin C_{i,k}$;类似地,事务 t_i 在接收消息 m 后记检验点 $C_{i,k}$表示为 $receive(m)\in C_{i,k}$,反之为 $receive(m)\notin C_{i,k}$。当 T 记全局检验点时,必须保证全局检验点所反映的消息事件经历是正确的,称这样的全局检验点为一致全局检验点。

定义5 T 的一个全局检验点是一致的当且仅当对 T 中任意两个事务 $t_i,t_j(i\neq j)$满足:

$$!\exists\ m,\ send(m)\notin C_{i,m}\wedge receive(m)\in C_{j,n}$$

上式表明全局检验点是一致的当且仅当 T 中不存在这样的消息,该消息的接收事件包含在目标事务的局部检验点中而发送(该消息的)事件不包含在源事务局部检验点中[5]。

2.2 检验点依赖和最小检验点算法

为叙述方便,本文下面部分将实时事务简称为事务,分布式合作实时事务集简称为事务集。

定义6 设事务 t_i、t_j 是事务集 T 中两个事务,它们最近的检验点分别为 $C_{i,m}$和 $C_{j,n}$,t_i 直接检验点依赖于 t_j(记作 t_i DCD t_j)当且仅当:

$$\exists\ m,\ receive(m)\notin C_{i,m}\wedge send(m)\notin C_{j,n}$$

定义7 事务 t_i 传递检验点依赖于 t_j(记作 t_i TCD t_j)当且仅当:

$$\exists\ t_k,\ (t_i\ \mathrm{DCD}\ t_k)\wedge((t_k\ \mathrm{DCD}\ t_j)\vee((t_k\ \mathrm{TCD}\ t_j))$$

在下边的讨论中,除非特别说明,我们不区分直接检验点依赖和传递检验点依赖,把它们统称为检验点依赖。当事务 t 在某一时刻发起检验点过程时,为了保证全局检验点的一致性,所有检验点依赖于 t 的事务都要记相应的检验点。如果一个检验点方法保证只有检验点依赖于 t 的事务才记检验点,则称该方法为最小协同检验点方法[4]。

3 一种高效的并行检验点方法

完整的协同检验点过程包括应记检验点事务识别、检验点内容提取和检验点存储三个步骤[4],其中应记检验点事务的识别方法对阻塞时间有很大的影响。

3.1 消息交换

假定合作事务集中有 N 个事务,每个事务有唯一的标识 i ($1\leqslant i\leqslant N$),在本文算法中,为了跟踪事务之间的检验点依赖关系,每个事务有一个集合 TS_RMF$_i$ 与之相联,它记录了 t_i 最近一个检验点之后给它发送了至少一条消息的事务标识,当 t_i 记检验点时,TS_RMF$_i$ 中的所有事务也要记检验点。为了在检验点过程中更容易地标识出更多的需要记检验点的事务,在事务正常的消息通信过程中要包括以下附加的步骤:

(1) 当事务 t_i 发送一条消息给 t_j 时将 TS_RMF$_i$ 和 t_i 的事务标识 i 随同消息一起发送给事务 t_j

(2) 当事务 t_j 收到了一条 t_i 发送来的消息后执行下面操作:

```
STEP 1  For each identifier k in TS_RMFi
            IF k ∉ TS_RMFj
                TS_RMFj = TS_RMFj ∪ {k}
STEP 2  For the transaction ti (Note that the identity number of ti is
        i)
            IF i ∉ TS_RMFj
                TS_RMFj = TS_RMFj ∪ {i}
```

3.2 检验点算法

本文算法基于两段提交协议,当事务 t_i 发起检验点过程后,相关事务都被要求做一个临时检验点,所有相关事务都记了临时检验点并给 t_i 以肯定答复后,t_i 发送广播消息请求相关事务将临时检验点变为永久检验点,上述步骤的实现细节对系统的性能有很大的影响,在本文算法中,t_i 发起检验点过程后具体的细节如下:

(1) 合作事务集 T 中所有事务一旦收到检验点请求,就暂停计算过程,不能再发送消息给别的事务(与所有的阻塞(blocking)协同检验点方法相同)。

(2) t_i 给 TS_RMF$_i$ 中每个事务 t_j 发送一个检验点请求消息,格式如下:

〈initiator，weight〉

其中 initiator 是检验点发起事务，即事务 t_i 的标识 i；weight 是赋给每个需记检验点的事务的权值，用它来确定检验点过程是否完成。例如，如果 TS_RMF$_i$ 中有 N 个事务标识，则 t_i 发送给这些事务的消息格式为〈i，$1/N$〉。同时 t_i 将自己的 TS_RMF$_i$ 置于消息中一起发送给 TS_RMF$_i$ 中每个事务 t_j，以实现检验点请求消息数量的最小化。

(3) 当一个事务 t_j 收到了检验点请求〈i，w〉后，它首先检查 TS_RMF$_j$，看有没有事务标识 k 在 TS_RMF$_{rec}$（TS_RMF$_{rec}$为事务 t_j 收到的检验点请求消息中的事务标识集合）中没出现过，如果有 p 个这样的事务，则将 t_j 收到的权值 w 平均分给 t_j 和这 p 个事务，即 t_j 的权值被改为 $w/(1+p)$，最后将检验点发起事务标识 i，$w/(1+p)$组成的二元组发送给这 p 个事务。注意消息中还附带有 TS_RMF$_j$∪TS_RMF$_i$ 以确保每个事务至多只收到一条检验点请求。

(4) 对于每个事务 t_k，它至多会收到一个检验点请求，t_k 根据自己的情况给检验点发起者以肯定或否定（表示自身能否记一个检验点）响应消息，其中应包含检验点请求中的权值以使发起该全局检验点的事务 t_i 决定检验点过程何时结束。

(5) 当 t_i 收到了所有应答（所有权值之和为1）后，根据应答结果广播一个消息给所有的相关事务，通知它们或将临时检验点转化为永久检验点（当所有的事务都回答能记检验点），或丢弃临时检验点（至少有一个事务回答不能记检验点）。

定理1 上述检验点过程的事务标识算法是正确的。

证明：为了证明定理，只需证明上述事务标识过程所标识出的事务集是 CPCLOSER$_r$(t_i)的超集，即 CPCLOSER$_r$(t_i)中的事务都能通过检验点过程标识出来。这可以通过对 CPCLOSER$_r$(t_i)中事务数目 n 做归纳来证明。当 $n=1$即 CPCLOSER$_r$(t_i)中只有一个事务 t_k 时，根据 CPCLOSER$_r$(t_i)的定义和上述消息交换过程，在 TS_RMF$_i$ 中必有事务 t_k 的标识 k，因此当 t_i 发起检验点过程时，它必然给 t_k 发送一条检验点请求。假设当 $n=k$ 时，该检验点过程能标识出 CPCLOSER$_r$(t_i)中的所有事务。当 $n=k+1$时，也就是说 CPCLOSER$_r$(t_i)中有 $k+1$个事务，用 $t_1,t_2,\cdots,t_k,t_{k+1}$表示。不失一般性，我们假设 t_1到 t_k 已经被标识出，根据 CPCLOSER$_r$(t_i)的定义，在$\{t_1,\cdots,t_k\}\cup\{t_i\}$中必有一个事务 t_m，使得 t_{k+1} DCD t_m，即 t_{k+1}与 t_m 至少有一次消息交换，在 t_m 的 TS_RMF$_m$ 中有标识 $k+1$。当 t_m 接收到一个检验点请求〈i，w〉和 TS_RMF$_{rec}$时，它检查 $k+1$是否在 TS_RMF$_{rec}$中，如 $k+1\in$TS_RMF$_{rec}$表示 t_{k+1}已收到检验点消息，否则 t_m 给其发送一条检验点消息，从而 t_{k+1}可以被标识出。综上所述定理得证。

由于本文的检验点方法即是一致的又只要求检验点依赖于发起者的事务记检验点，因此有：

推论1 本文的检验点方法是最小协同检验点方法。

4 实验性能评估

4.1 系统模型和性能指标

模拟实验是在本实验室开发的一个分布式实时数据库系统原型 ARTs-Ⅰ的基础上进行的。全局数据库(GDB)为2400个数据库页的集合，这些页平均分配到10个结点形成局部数据库(LDB)。对数据的访问和并发控制都发生在页级，各结点上事务都以独立的泊松率到达并随机的组成合作事务集，同一事务只参与一个事务集。我们称事务集中事务个数为事务集容量，事务集容量从3到10之间变化。事务接纳后按照公式 $D_T=A_T+SF*R_T$ 确定事务截止期，其中 D_T 为事务截止期；A_T 为事务到达系统的时间；SF(slack factor)为松弛因子，由它控制事务截止期的松紧；R_T 为执行估算时间。按照截止期早者优先(EDF)给事务分配优先权，且其优先权在整个生命期保持不变，根据事务的优先权进行事务调度。并发控制采用2PL-HP(2PLHigh-Priority)协议。

事务集中检验点过程的发起者随机选择，两个连续检验点过程间隔呈 $\theta_{CK}=20s$ 的指数分布。每个事务向事务集中其它事务发送消息的时间间隔呈 $\theta_{MT}=4s$ 的指数分布，发送对象随机选择。事务故障的概率为0.5%。由于检验点的目的是减少事务故障和被迫重启时计算量的损失，因此将检验点直接记录在随机存储器中以减少时间代价，检验点内容提取采用增量检验点方法[6]。

实验的性能指标包括平均检验点识别延迟和事务错过率。衡量协同检验点方法的主要性能指标有检验点的数目和检验点过程时间代价，由于文[2]中方法会导致不一致状态，文[7]中方法不是最小协同检验点方法，而 SCA 和本文方法都是最小协同检验点方法，所以我们对 SCA 和本文方法的检验点过程时间代价进行了比较(两者的检验点数相同且都是最小的)。完整的检验点过程包括从发起者发出检验点请求到所有应记检验点的事务都记完检验点为止的这段时间，在标识完所有应记检验点的事务后，SCA 和本文方法执行过程相同，因此实验中我们只比较了应记检验点事务识别的时间代价。定义事务集中从发起者发起检验点请求到最后一个应做检验点的事务被确认为一次检验点过程识别延迟。某种容量事务集的平均检验点识别延迟定义为该容量所有事务集的检验点过程识别延迟的平均值。

事务错过率(Miss Rate，MR)指系统中超截止期事务的比率，是衡量实时数据库系统性能的重要指标，我们在不同荷载下比较了不采用检验点技术、采用 SCA 和采用本文方法的事务错过率。定义当0<MR<20时系统处于轻荷载下，当20<MR<100时系统处于重荷载下。

4.2 实验数据分析

在保持事务集组成模式、消息发送模式和检验点发起模式等不变的前提下，使实验系统分别不采用检验点技术、采用 SCA 和采用本文方法运行一小时。在每个结点事务到达率为4时不同容量事务集的平均检验点识别延迟如图1所示，图中 Y 轴以所有事务对之间的最大消息延迟为基准单位。

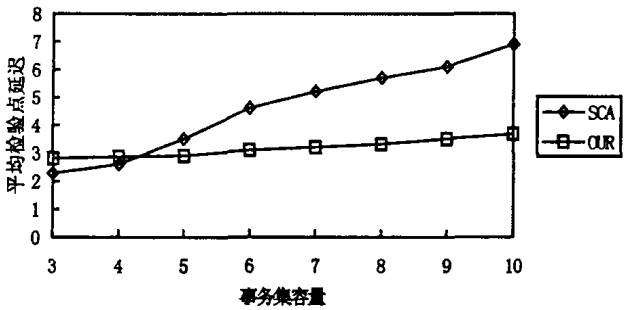

图1 平均检验点识别延迟比较

由图1可以看出 SCA 的平均检验点识别延迟随着事务集容量增加而迅速增加，这是由于顺序通知的消息延迟所造成的。由于本文方法并行执行检验点过程，因此平均检验点识别延迟要优于 SCA。

(下转第75页)

(2)作为备份过程的一个集成部分,检查数据库块是否残缺不全,这可以减少对整个数据库频繁使用 Analyze Table Validate Structure 语句的要求;

(3)支持增量物理备份,这个与导出/导入的增量截然不同;

(4)支持多线程备份;

(5)提供一个集成的分类系统,该分类系统在鉴别所需的备份磁带时能够把混乱减到最小。

结束语 Oracle 本身提供了强大的安全机制,但同时也存在着一些漏洞,普遍存在于 Oracle 数据库中的用户问题:

(1)权限过大:对 Oracle 数据库编程和浏览的一般用户常常具有 DBA (数据库管理员权限),能对数据库系统做任何修改或删除。

(2)安全性差:很多 Oracle 用户缺省存储位置都在系统表空间,这样不仅影响系统的正常工作,而且不同用户的数据信息互相影响、透明,保密性差。随着数据的不断加入,有可能使整个数据库系统崩溃。

(3)密码有规律:在 Oracle 调试初期形成的用户名和密码一致的不良习惯保留到现在;系统用户 SYS 和 SYSTEM 的密码也众所皆知。

因此真正保障数据库应用系统的安全性,仅仅保证 DBMS 的数据库安全保障体系还不能彻底解决系统的安全问题,保障数据库应用系统的安全性还需要从安全策略上加以保障,需要从体系结构上保障,需要增强物理安全、网络安全以及系统安全性,DBA 需要经常检查安全策略与措施,堵塞漏洞,防范于未然。

参 考 文 献

1 Oracle9i Security Overview [M]. Oracle Corporation, 2002. 60～90
2 Oracle9i 基础与提高[M]. 飞思科技产品研发中心编著. 北京:电子工业出版社,2003. 370～450
3 Dillon S. Oracle 编程入门经典[M]. 清华大学出版社,2002. 458～498
4 Introduction to Oracle Identity Management [M]. An Oracle White Paper, Dec. 2003. 4～6
5 Gorman T. Unraveling the Sweater Oracle Database Security [M]. 2002. 4～8
6 蒋继洪. 计算机系统、数据库系统和通信网络的安全与保密[M]. 成都:电子科技大学出版社,1995
7 Nanda A. 现实问题的细粒度审计. http://www.oracle.com/technology/global/cn/pub/articles/nanda_fga_pt2.html 2004-1-9
8 陈吉平. Oracle 9i RMAN 参考使用手册[J]. 2002. 10～12
9 朱鲁华,陈荣良. 数据库加密系统的设计与实现. 计算机工程,2002(8)

(上接第71页)

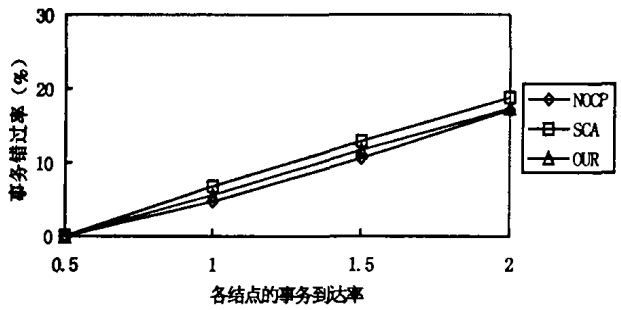

图2 轻载荷下事务错过率比较

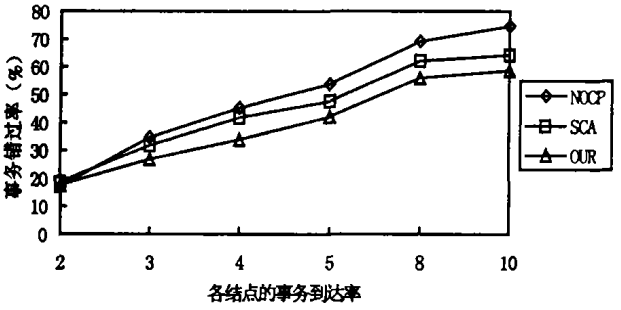

图3 重载荷下事务错过率比较

轻、重载荷下不同事务到达率时的 MR 如图2、3所示,从图中可以看出在系统处于轻载荷时,采用检验点技术的事务错过率比未采用时略高,并且采用 SCA 比采用本文方法的事务错过率要高。这是由于轻载荷下系统中事务数较少,发生数据竞争的可能性比较小,事务故障是导致事务重启的主要原因,这使得总事务重启率相对较小。采用检验点技术多花费了记检验点的时间,因此事务错过率相对未采用时略高。SCA 由于检验点延迟比本文方法法要大,因此有更高的事务错过率。注意在轻载荷下采用本文方法对系统性能并没有太大的影响。

当系统处于重载荷时,采用检验点技术的事务错过率比未采用时要低很多,检验点技术的优势逐渐显示出来。这是因为在重载荷下系统中事务数量不断增多,数据竞争的概率增加造成总事务重启率变大(注意合作事务集中任一事务重启都会导致事务集中其它相关事务的重启),采用检验点技术则不用完全重启事务而减少了计算量的损失,因此有利于事务定时约束的满足,提高了系统性能。本文方法的时间代价要优于 SCA,因此有更低的事务重启率。

结论 随着实时数据库应用的深入和计算机网络应用的普及,分布式实时数据库成为新的研究热点。在一类分布式合作实时数据库应用中,不同结点的事务通过消息通信结成合作关系。为了减少因事务故障和数据冲突造成的事务重启而损失的工作量,可以采用检验点来定期记录事务执行的快照,本文的主要工作有包括提出了一种新的分布式合作实时事务经历模型;给出了事务间由于消息通信而形成的检验点依赖关系的形式化定义;提出并实现了一种高效的并行最小协同检验点方法,该方法能降低一个全局检验点的时间延迟,从而更有利于实时事务定时限制的满足;设计并实现了一个实验系统,对文中方法进行了性能评价。

参 考 文 献

1 Grey J, Reuter A. Transaction Processing: Concepts and Techniques. San Mateo, Calif. Morgan Kaufmann, 1993
2 Prakash R, Singhal M. Low-cost checkpointing and failure recovery in mobile computing systems. IEEE Tran. Parallel and Distributed Systems, Oct. 1996. 1035～1048
3 Koo R, Toueg S. Checkpointing and roll-back recovery for distributed systems. IEEE Tran. on Software Engineering, Jan. 1987. 23～31
4 Cao Guohong, Singhal M. On coordinated checkpointing in distributed systems. IEEE Tran. on Parallel and Distributed Systems, Dec. 1998. 1213～1225
5 Randell B. System structure for software tolerance. IEEE Transactions on Software Engineering, 1975, SE-1(2): 220～232
6 Deng Y, Park E K. Checkpointing and Rollback-Recovery Algorithms in Distributed Systems. Journal of Systems and Software, Apr. 1994. 59～71
7 汪东升,邵明珑. 具有 O(n)消息复杂度的协调检查点设置算法. 软件学报,2003,14(1):43～48
8 刘云生著. 现代数据库技术. 北京:国防工业出版社,2001