数据广播环境下移动实时事务的有效并发控制*)

李国徽 杨 兵 向 军 陈 辉
(华中科技大学计算机学院 武汉 430074)

摘 要 在无线数据广播环境下,从移动客户机到数据库服务器的上行带宽非常有限。传统的并发控制协议不适合这种不对称通讯环境。本文结合乐观并发控制协议和时标排序协议提出一种并发控制协议。只读事务满足弱一致性,包含所有更新事务和一个移动只读事务的可串行化图是非循环的。使用本文提出的并发控制协议,移动只读事务能够自主执行,移动更新事务能够较早地检测到数据冲突。模拟试验显示,我们提出的协议相对其它已经存在的并发控制协议来说,能够更好地满足事务截止期。

关键词 数据广播,移动实时事务,乐观并发控制,可串行化

Efficient Concurrency Control for Mobile Real-time Transactions in Data Broadcast Environments

LI Guo-Hui YANG Bing XIANG Jun CHEN Hui
(School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074)

Abstract In a wireless data broadcast environment, there is a limited upstream bandwidth from mobile clients to the fixed database server. Traditional concurrency protocols are unsuitable in such an asymmetric communication environment. In this paper, we introduce a concurrency control protocol which integrates optimistic concurrency control and timestamp ordering. A mobile read-only transaction is weak consistent and the serialization graph consisting of all update transactions and a mobile read-only transaction is acyclic. By using this protocol, mobile read-only transactions can execute autonomously and mobile update transactions can detect data conflicts earlier. Extensive simulation experiments are conducted and the result shows that compared with existing concurrency control protocols, more mobile real-time transactions can commit before their deadline expire.

Keywords Data broadcast, Mobile real-time transaction, Optimistic concurrency control, Serializability

1 引言

随着无线通信技术的飞速发展,越来越多的移动用户利用掌上电脑通过无线接口访问互联网。在传统的C/S系统中,数据是基于请求传送的,移动客户机向服务器发送明确的数据请求[1,2]。服务器接受到数据请求后,把请求数据返回移动客户机。这种数据传送方式叫 pull-based 数据请求方式。

在移动计算环境中,从服务器到移动客户机的下行带宽相对比较高,而从移动客户机到服务器的上行带宽非常有限,有时甚至没有上行带宽。为了充分利用相对宽裕的下行带宽,采用数据广播作为主要的数据传送方式,这种获取数据方式叫 push-based 传送方式。服务器周期性广播数据库数据,移动客户机监视广播通道并取回事务执行需要的数据,移动客户机执行的事务叫移动事务(MT)。大部分移动事务有时间限制,要求在其截止期前完成[3]。没有满足截止期的事务称之为"迟滞"事务。按照"迟滞"事务对系统的影响,移动实时事务(MRTT)分为两类:没有满足截止期,能够对系统造成致命影响的移动实时事务称为硬移动实时事务;对于没有满足截止期仅仅意味着没有继续执行必要的事务称为软移动实时事务。

由于有线分布式数据库系统中服务器和客户机存在大量通信,应用其上的传统并发控制协议显然不适合数据广播不对称通信环境,基于上行带宽的限制,大量移动实时事务的执行会被延迟,无法满足其截止期。与移动实时事务执行策略有关的另外一个重要问题是能耗,它在移动计算环境下极其匮乏[4,5]。如果使用传统的并发控制协议,当移动客户机和服务器存在大量数据传送时,移动客户机能耗巨大。并且,随着和服务器通信的移动客户机数量增加,服务器将不堪重负,成为整个系统瓶颈。

可串行化是事务执行的正确性标准[14]。它通过要求并发事务交叉执行和串行执行的结果一致来维护数据库一致性。目前,有两种典型的并发控制协议:基于锁的并发控制协议和乐观并发控制协议[9]。在无线通信环境下,如果移动客户机对每个数据对象要求生成请求锁,将增加移动客户机和数据库服务器额外的通信量。显然,基于锁的并发控制协议不适合数据广播环境[9,15]。

移动通信环境下采用可串行化作为事务并发执行的正确性标准代价相当大,在很多情况下没有必要采用这种保守的正确性标准[14]。移动实时数据库系统在保证数据库一致性基础上,可以不严格执行正确性标准来实现移动实时事务的并发执行。文[17]给出四种形式的查询执行一致性要求:严格一致性、强一致性、弱一致性和更新一致性,其严格程度依次减弱。这四种形式有一个隐含的必要条件:更新事务的单独执行必须是可串行化的。降低只读事务一致性要求能够提高性能。

*)本文工作受青年国家自然科学基金项目(编号:60203017)、留学回国人员基金及湖北省杰出青年基金的资助。李国徽 博士,教授,博士生导师;杨 兵 博士研究生。

在本文，针对移动实时事务的执行，结合乐观并发控制协议和时标排序协议提出一种新的并发控制协议：移动更新事务必须到数据库服务器进行最后验证，移动只读实时事务没有必要到数据库服务器进行最后验证而自主执行，并且采用弱一致性作为移动只读事务的一致性要求[17]。弱一致性有两个特点：(1)数据一致性；(2)本地串行化。数据一致性是指数据库服务器中更新事务是可串行化的，服务器数据库是一致的。本地串行化是指相对于更新事务来说移动只读事务是可串行的，但是可能没有考虑到其他只读事务。试验结果显示，通过降低移动只读事务的一致性要求，能够更好地完成并发控制，更多的事务可以满足其截止期。

本文是这样组织的：第 2 节是相关工作；第 3 节描述移动事务串行化执行的局限；第 4 节介绍数据广播环境下乐观并发控制协议；第 5 节提出一种新的移动实时事务并发控制协议。在第 6 节中给出一组试验，最后作出总结。

2 相关工作

目前，无线环境下的数据管理研究已经比较成熟[6~8]，但是绝大部分工作集中在数据索引、数据组织和数据失效报告，几乎没有移动计算事务处理方面的研究，尤其在数据广播环境下。文[8]针对高吞吐量数据库系统提出了一个典型的数据循环体系结构，整个数据库数据通过高带宽网络周期性广播。在读段，事务从广播通道中获取数据对象，并且假定事务在一个广播周期内完成读操作，维护了读集合的一致性。然而，这种体系结构存在很大的弊端：第一，在一般的多盘广播机制中，数据库数据对象不是以一个平坦模式，而是以不同的频率甚至不同的数据广播周期进行广播的[7]；第二，移动事务读集合没有在执行前预定义，对数据对象的访问顺序不要求和它在广播通道中出现顺序一致，如果移动事务请求的数据对象没有在当前广播周期中出现，它就不得不等待下一个广播周期该数据对象的出现。这样，一个移动事务的执行很可能跨越多个广播周期，读取到已经被数据库服务器更新、不一致的数据对象。

文[11~13]提出了几种数据广播方法来保证只读事务的正确性。多版本广播方法记录每个数据对象的不同版本，移动事务选择合适版本的数据对象来执行。在串行化图广播方法中，事务冲突图和数据对象同时广播，服务器和客户机共同维护冲突图进行冲突检查。这两种方法都采用可串行化作为正确性标准，但是广播通道包含大量额外信息(数据对象的多个版本或冲突图)，大大延长了广播周期。这将引起移动实时事务执行的时间延迟，导致更多移动实时事务不能满足截止期，同时它访问的数据也会超过它的有效期。

文[10]采用乐观并发控制协议来保证移动只读事务正确性。移动只读事务 T 被提交前，检查 T 是否和前一次数据广播结束后数据库服务器提交的更新事务发生冲突。一旦存在冲突，事务 T 夭折并重启动。移动更新事务必须到数据库服务器做最后的可串行化检查。但是文[10]中的乐观并发控制协议使许多移动事务由于冲突而夭折，甚至这些冲突并不违背可串行化正确性标准。文[16]把一个数据广播周期划分成多个子周期，在两个子周期之间的广播通道中包含前一个子周期广播结束后数据库服务器提交的更新事务产生的写集合。按照这种方式，许多没有必要取消的事务被避免取消，这些事务虽然发生冲突，但是并没有影响事务的并发执行。文[15]结合时标排序协议和乐观并发控制协议来加强移动事务执行的并发控制。尽管事务并发控制的执行是可串行化的，从下面的讨论可以看到，这个协议太保守，许多移动实时事务没有必要被取消，即夭折并重启动根本没有影响正确性标准的冲突移动实时事务是没有必要的，并且对 MRTT 截止期的满足造成相当大的负面影响。

由于在移动数据广播环境下维护可串行化正确性标准的代价相当大，文[9]采用更新一致性作为移动只读事务的正确性标准，这样，在无线环境下移动客户机能够在不连接服务器情况下读取最新的、一致的数据。这个策略中额外的控制信息是一个 $n\times n$ 矩阵，n 是数据库数据对象个数，移动事务依据这个矩阵来维护它的更新一致性。本文参照文[17]采用比更新一致性严格的弱一致性，这样控制信息显然少很多。

3 移动事务串行化执行的局限

文[15]提出移动事务的并发控制协议，数据库服务器采用传统的并发控制协议，如两段锁协议，来保证并发事务执行的正确性。服务器事务有一个时间戳，它按照事务提交顺序记录。被服务器更新的数据对象 d 也分配一个时间戳，记作 $WTS(d)$，记录所有更新事务对数据对象 d 进行写操作的最大时间。

移动事务时间戳间隔初始化为$[0,\infty)$。只读事务 T 能够在任何时候读取数据对象 d，T 的时间戳间隔下限 $LB(T)$ 取 $LB(T)$ 和 $WTS(d)$ 的最大值，即 $LB(T)=\max(LB(T), WTS(d))$。

一个数据广播周期开始时，包含前一次广播周期结束后被更新的数据对象 ID 和它们的时间戳到广播通道中。假设 CUT 是最近一次广播周期提交的更新事务集合，$CRS(T)$ 是移动事务 T 的前一次广播周期读集合。当接受到附加控制信息时，T 按照下面步骤处理：

$$UB(T)=\mathrm{mini}(TS(Ui),\ UB(T)),\ \forall Ui\in CUT \text{ AND } WS(Ui)\cap CRS(T)\neq\emptyset$$

任何时候 $LB(T)\geqslant UB(T)$，事务 T 将被取消并且重启。

上述算法能够保证数据库的一致性并且所有事务的并发执行是可串行化的(包括数据库服务器事务和移动只读事务)，同样这个算法太过保守以至许多串行执行不被通过，并且有些移动事务的夭折可避免。

如图 1，假定有三个事务：两个数据库服务器更新事务、一个移动只读事务。

U1: R(x) W(x)
U2: R(y) W(y)
T3: R(x) R(y)

图 1 三个事务的例子

现在，这些事务并发执行按照下面的方式：

$R3(x)\ R1(x)\ W1(x)\ R2(y)\ W2(y)\ R3(y)$

初始化 $WTS(x)=0$ 和 $WTS(y)=0$。当移动事务 $T3$ 读取数据对象 x 时，$T3$ 时间戳间隔初始化为$[0,\infty)$。当 $U1$ 对数据对象 x 进行写操作并且提交时，$T3$ 的时间戳上界调整为 $WTS(x)$，即 $UB(T3)=WTS(x)$。当 $T3$ 读取数据对象 y 时，$T3$ 时间戳间隔下界调整为 $WTS(y)$，即 $LB(T3)=WTS(y)$。很明显，当事务 $T3$ 读取数据对象 y，并且 $WTS(y)>WTS(x)$，则 $LB(T3)>UB(T3)$，事务 $T3$ 夭折并重启动。然而，由这三个事务组成的串行化图是无环的，并发执行是可串行化的。实际上移动只读事务在其它移动客户机的只读事

务读集合中没有信息，为了保证系统所有事务的执行是可串行化的，移动广播环境中绝大部分采用保守的可串行化并发控制协议，造成大量本来符合可串行化标准的事务夭折。这些都是移动实时数据库系统极力避免的。

4 数据广播环境下的乐观并发控制原理

在乐观并发控制协议中，向前验证协议能够解决移动客户机只读事务的“推迟”重启问题[15]，同时更新事务需要到数据库服务器做最后验证。而传统的并发控制协议，例如采用高优先级分布式两段锁协议，来维护全体数据库实时事务的串行化。移动事务中的验证是通过比较它的读集合和前一次广播循环结束后服务器更新事务提交的写集合来进行的。

数据广播周期包含数据库更新事务写集合中的数据对象ID作为控制信息，服务器同时广播数据对象和控制信息。参照控制信息，移动客户机能够检测到更新事务产生的数据冲突。

数据对象在进行读操作前，只读事务参照控制信息来决定是否继续执行。如果操作不能执行，移动事务就被夭折。移动客户机只读事务 T 决定是否进行读操作前，将进行下面的简单检查：

If $WS(Ui)\cap CRS(T)\neq\emptyset$ ，$\forall Ui\in CUT$ then restart T.

在这里，CUT 指的是最近一次广播周期中提交的更新事务集合，$CRS(T)$ 指的是事务 T 当前读集合。

这种策略有两个优点：第一，只读事务的“推迟”重启问题部分得到解决。第二，移动只读事务的执行不需要连接服务器。但是，如文[16]指出的那样，这个策略太过保守，许多移动事务夭折是没有必要的。为了解决这个问题，文[15]结合验证策略和时标排序协议，能够较好解决移动事务并发控制。然而，就像在第3节讨论的那样，这个算法还是太保守，许多串行化执行不允许。在第5节，我们将采用弱一致性作为移动只读事务正确性标准，提出一个并发控制协议来保证移动实时只读事务的弱一致性。针对更新事务采用新的时间戳分配策略、不同的一致性检测方法来完成更高层次的移动事务并发控制。

5 弱一致性协议

5.1 服务器端更新事务提交的时间戳分配

更新事务在最后提交时分配时间戳。在文[15]中，对于没有数据冲突的更新事务，也会按照它们提交的顺序分配不同时间戳。在我们的算法中，事务时间戳不是按照提交的顺序，而是按照可串行化执行的顺序进行分配，即服务器所有事务的并发执行是可串行化的，相关的串行化图 G 是非循环的。时间戳分配过程是基于串行化图的，在这个过程中，对每个在串行化图 G 中的事务 T，分配一个时间戳，记作 $TS(T)$。图2描述了时间戳分配算法。在时间戳分配算法执行前，只读事务从串行化图中删除。

```
TimeStamp_Assignment()
(1) Current_time=0;
(2) WHILE there are left transactions in GDO
{
    Select all the transactions in G which have no entering edges.
    For each selected transaction T, TS(T)=Current_time;
    Delete all these transactions (and the related going-out edges) from G;
    Current_time= Current_time+1;
}
```

图2 更新事务时间戳分配算法

$WS(U)$ 表示更新事务 U 的写集合。被更新数据对象 d 的时间戳表示为 $TS(d)$。$TS(d)=\langle LTTS,\ ID\_Tr\rangle$，这里的 ID_Tr 表示最近更新数据对象 d 的事务 ID，$LTTS$ 表示 $TS(D).ID\_Tr$ 的时间戳。

参照文[15]，当 U 被提交时，按照下面步骤执行：

(1)按照图2描述的算法给更新事务 U 分配一个时间戳。

(2)对于更新事务 U 写集合中的每个数据对象 d，更新 $TS(d)$。

(3)记录控制信息。

5.2 移动客户机只读事务读段

在这个协议中，读段的每个移动只读事务被分配一个时间戳间隔，用来维护移动事务的弱一致性。

按照前面的描述，每个更新事务有一个时间戳，几个更新事务可能有相同的时间戳。为了检测只读事务 T 是否满足弱一致性，只读事务关联一个时间戳间隔，用 $LB(T)$ 和 $UB(T)$ 分别表示只读事务 T 时间戳间隔的下限和上限。$LB(T)$ 和 $UB(T)$ 分别有两个属性，用 $\langle LTS,\ IDS\_Tr\rangle$ 表示，这里，LTS 表示更新事务时间戳，IDS_Tr 表示更新事务 ID。

在只读事务 T 开始执行时，它们时间戳初始化为：

$LB(T).LTS=0$；$LB(T).IDS\_Tr=\Phi$

$UB(T).LTS=\infty$；$UB(T).IDS\_Tr=\Phi$

可以使用时间戳间隔检测出非弱一致性。在读段，只要移动只读事务读取一个数据对象，它的时间戳就会被重新设定，从而体现由于自身和被提交更新事务引起的弱一致性要求。

任何时候只读事务 T 读取数据对象 d，按照下面的步骤执行：

```
Step 1. Read(d);
Step 2. If TS(d).LTTS>LB(T).LTS
        THEN
        {
        LB(T).LTS= TS(d).LTTS;
        LB(T).IDS_Tr= {TS(d).ID_Tr}
        }
        ELSE
         If TS(d).LTTS==LB(T).LTS
         THEN
          LB(T).IDS_Tr= LB(T).IDS_Tr ∪
                        {TS(d).ID_Tr}
Step 3. If LB(T).LTS>UB(T).LTS
        THEN restart T
        ELSE
        If   LB(T).LTS==UB(T).LTS   AND
        LB(T).IDS_Tr ∩UB(T).IDS_Tr≠Φ
             THEN restart T.
```

图3 只读事务 T 读取数据对象 d 执行步骤

在这个协议下，在事务执行结束前，非弱一致性被检测。

5.3 数据广播

数据广播中的控制信息能够帮助移动客户机检测移动只读事务的非弱一致性。控制信息包括前一个广播结束后被更新的数据对象 ID 和这些数据对象的时间戳信息。如果当前广播周期中含有控制信息的移动事务 T，并且它的读集合包含一些数据对象，$UB(T)$ 必须重新设定来表现自身和更新事务之间的串行化顺序，即当接受到控制信息，移动事务按照下面顺序执行。

对每个包含控制信息的数据对象 d，事务 T 进行读取时，执行步骤如下：

```
STEP 1: If TS(d).LTTS<LB(T).LTS
            THEN
           {
        UB(T).LTS= TS(d).LTTS;
        UB(T).IDS_Tr= {TS(d).ID_Tr}
           }
          ELSE
           If TS(d).LTTS==LB(T).LTS
           UB(T).IDS_Tr= UB(T).IDS_Tr ∪
                 {TS(d).ID_Tr}
STEP 2: If LB(T).LTS>UB(T).LTS
      THEN restart T
      ELSE
          If LB(T).LTS==UB(T).LTS AND
          LB(T).IDS_Tr ∩UB(T).IDS_Tr≠Φ
          THEN restart T.
```

图 4　包含控制信息的数据对象 d 及事务 T 读取数据对象 d 的执行步骤

5.4　正确性

在本小节，将给出我们提出协议的正确性证明。参照文[14]串行化图的定义，如果包含所有更新事务的串行化图是非循环的，并且包含所有更新事务和一个只读事务的任意串行化图也是非循环的，就说明事务的执行是弱一致性的。当服务器并发事务执行采用可串行化作为正确性标准，而且移动更新事务到服务器进行最后的验证时，很明显，包含所有更新事务的串行化图是非循环的。这样仅仅需要证明，任何包含所有更新事务和一个只读事务的串行化图是非循环的。

在我们提出的协议中，对于一个提交的只读移动事务 T，$LB(T).LTS \leqslant UB(T).LTS$，给 T 分配时间戳 $LB(T)$，为 $LB(T).LTS$ 和 $UB(T).LTS$ 的平均值，即 $TS(T)=(LB(T).LTS+UB(T).LTS)/2$。显然，$LB(T).LTS \leqslant TS(T) \leqslant UB(T)$

引理 1　在我们提出的协议产生历史事务执行中，假定 Q 和 U 分别是提交的只读事务和更新事务。如果在串行图 $SG(H)$ 中有一个有向边 $U \rightarrow Q$，那么 $TS(U) \leqslant TS(Q)$。

证明：如果在 $SG(H)$ 有一个边 $U \rightarrow Q$，那么这两个事务必定有一个或一个以上冲突读写操作（$Wu[x] \rightarrow RQ[x]$），这种可能只有在 Q 执行 $RQ[x]$ 前 U 完成它的写段才能出现。对于 $RQ[x]$，调整 $LB(Q).LTS$ 为 $TS(x).LTTS$，即 $TS(U)$。因此，$TS(U)=TS(x).LTTS \leqslant LB(Q).LTS \leqslant TS(Q)$。所以 $TS(U) \leqslant TS(Q)$

引理 2　如果在 $SG(H)$ 中有一个边 $Q \rightarrow U$，那么 $TS(Q) \leqslant TS(U)$。

证明：如果在 $SG(H)$ 有一个边 $Q \rightarrow U$，这两个事务一定有一个或者一个以上冲突读写操作（$RQ[x] \rightarrow Wu[x]$）。就意味着在 x 被 U 更新前 Q 读取了 x，并且广播到 Q 所在的移动客户机，即 $RQ[x]$ 不会被 $Wu[x]$ 影响。在我们提出的协议中，当移动客户机获得控制信息，$UB(Q).LTS$ 被调整到 $TS(x).LTTS$，即 $TS(Q) \leqslant UB(Q).LTS \leqslant TS(x).LTTS=TS(U)$，显然，$TS(Q) \leqslant TS(U)$。

观察 1　假定 $U1$ 和 $U2$ 是历史事务执行中被提交的两个更新事务，如果在 $SG(H)$ 中有边 $U1 \rightarrow U2$，那么就有 $TS(U1)<TS(U2)$。

由图 2 中更新事务时间戳分配算法可以直接得到。

定理 1　任何一个由本协议产生的历史事务的执行是满足弱一致性的。

证明：该定理即包含更新事务集合和一个只读事务的串行化图是非循环的。(1)假定只读事务在服务器端执行，采用可串行化并发控制协议，这个结论显然是成立。(2)假定只读事务在移动客户机执行，可以通过反证法来证明。

(2.1)假定循环串行化图仅仅包含更新事务 U 和移动只读事务 Q。于是，存在一个边 $Q \rightarrow U$ 和 $U \rightarrow Q$。按照引理 1 和引理 2，我们可以得到 $TS(Q)=TS(U)$ 和 $UB(Q).LTS=LB(Q).LTS=TS(U)$ 及包括 $LB(Q).IDS\_Tr$ 和 $UB(Q).IDS\_Tr$ 更新事务 ID。按照图 3 或图 4 中的步骤，移动只读事务 Q 将夭折并且重新启动。这和 Q 是被提交的事务并且出现在串行化图的假设相矛盾。

(2.2)假定串行化图包含 K（$k \geqslant 2$）个更新事务 $U1$，…，Uk 和一个移动只读事务 Q。在这个串行化图，至少在两个更新事务间有一条边，如 $U1 \rightarrow U2$。基于更新事务的时间戳分配算法，假定串行化图 $U1 \rightarrow U2 \rightarrow \cdots \rightarrow Q \rightarrow U1$。我们可以得到 $TS(U1)<TS(U2) \leqslant \cdots \leqslant TS(Q) \leqslant TS(U1)$，即 $TS(U1)<TS(U1)$，显然矛盾。

所以，对于一个移动只读事务 Q，包含 Q 和系统所有更新事务的串行化图是非循环的。按照我们协议得到的历史事务执行满足弱一致性。

下面通过两个例子来说明我们提出的协议能够完成更高级别的并发控制，并且移动只读事务满足弱一致性。

假定有 4 个事务，$T1$ 和 $T2$ 是更新事务，$T3$ 和 $T4$ 是两个移动只读事务。包含 4 个事务并发执行的历史记录描述如下：

$R4(y)W1(y)R3(y)R3(x)W2(x)R4(x)$

按照并发执行的历史记录获得的串行化图如下：

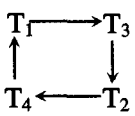

图 5　事务执行的串行化图

从图 5 可以看出，包含给定 4 个事务并发执行的串行化图是循环的，而并发执行的历史记录显示的却是不可串行化的。如果参照文[15]提出的协议，这些事务并发执行是不能接受的。然而，对移动只读事务 $T3$ 和 $T4$，它们满足弱一致性并且包含两个更新事务和 $T3$ 或 $T4$ 的串行化图是非循环的。数据库服务器是一致的，这样按照本文提出的协议，上面给出的并发执行历史记录就接受。

在事务 $T3$ 和 $T4$ 开始执行时，时间戳初始化如下：

$LB(T4).LTS=0$; $LB(T4).IDS\_Tr=\Phi$;

$UB(T4).LTS=\infty$; $UB(T4).IDS\_Tr=\Phi$

$LB(T3).LTS=0$; $LB(T3).IDS\_Tr=\Phi$;

$UB(T3).LTS=\infty$; $UB(T3).IDS\_Tr=\Phi$

在服务器端，由于 $T1$ 和 $T2$ 之间没有直接冲突，它们之间就没有直接路径。按照更新事务的时间戳分配算法，$T1$ 和 $T2$ 有相同的时间戳。假定 $TS(T1)=TS(T2)=\upsilon$。

事务 $T3$ 和 $T4$ 读取数据对象后，数据库服务器广播控制信息。$T3$ 和 $T4$ 时间戳调整如下：

$LB(T3).LTS=\upsilon$; $LB(T3).IDS\_Tr=\{T1\}$;

$UB(T3).LTS=\upsilon$; $UB(T4).IDS\_Tr=\{T2\}$

$LB(T4).LTS=\upsilon$; $LB(T4).IDS\_Tr=\{T2\}$;

$UB(T4).LTS=\upsilon$; $UB(T4).IDS\_Tr=\{T1\}$

对于事务 $T3$，尽管 $LB(T3).LTS=UB(T3)=\upsilon$，$LTS$，

$LB(T3).IDS\_Tr \cap UB(T3).IDS\_Tr=\Phi$，事务 $T3$ 能够提交，满足弱一致性正确性要求。事务 $T4$ 执行。

6 性能评价

模拟试验旨在研究我们提出的算法与 FBOCC[10]，EOCC[16]，TSOCC[15]在实时广播环境下的性能比较。参考文[10]，不考虑缓存对性能的影响，这样可能由于当前广播周期中请求数据不存在，移动事务不得不等待下一个广播周期中的请求数据。协议的主要性能指标是事务未满足截止期比率，即没有达到截止期事务的百分比。其它性能指标是事务重启率，即事务在提交前平均夭折并且重启的次数。一个事务重启后可能满足它的截止期，可以通过比较是否减少事务的重启次数来评比协议的优劣。事务响应时间，移动客户机提交事务所消耗的时间。事务重启可能不止一次。分别比较了文[10]中的传统乐观并发控制协议(FBOCC)、文[16]提出的协议(EOCC)、文[15]提出的协议(TSOCC)和我们提出的协议在移动只读事务(ROMT)和移动更新事务(UMT)方面的性能指标。

6.1 参数设置

参考文[10]，模拟试验包含一个服务器、一个客户机和一个用来传送数据对象和必需的控制信息的广播盘。在服务器端，采用传统向前验证乐观并发控制协议。移动客户机执行只读事务和更新事务。事务被提交时分配一个截止期。事务的截止期按照公式 submission_time+slack factor×predicted_execution_time 进行分配，这里的 predicted_execution_time 是一个事务长度和广播周期长度的函数。数据库数据对象按照正态分布的方式访问。表 1 列出了模拟试验的基本设置和参数。

表 1 基本设置和参数

移动客户端参数	值
每个事务操作数据个数	4
读操作概率 (UMT)	0.5
只读事务所占比率	70%
平均内部事务延迟	20 item-times (exponentially distributed)
减弱因素	2.0-8.0 (uniformly distributed)
服务器参数	值
每个事务操作数据个数	8
事务到达率	1 per 200 to 20 item-times
读操作概率	0.5
数据库中数据对象的数量	500
优先级策略	Earliest deadline first
保留空间大小	20

事务长度决定它进行读/写操作的次数。所有的数据都有相同大小，采用 item-time 作为时间单位，描述传送一个数据对象的时间，在广播周期开始，服务器用数据填充广播盘，数据库周期性广播所有数据对象及其相关控制信息。

6.2 试验结果

图 6 显示采用各种并发控制协议的移动事务重启率。从图 6 和图 8 可以看出，采用 OCC 的所有移动事务由于要到服务器进行最后验证，具有较高的事务重启率和事务响应时间。FBOCC 由于仅仅需要部分验证减少了事务重启率和事务响应时间。我们提出的协议具有较低的数据冲突，性能类似 FBOCC 和 EOCC。然而，随着服务器事务到达率的增加，EOCC 针对严重冲突和欺骗冲突采用不同验证方，性能显然优于 FBOCC，我们提出的协议采用弱一致性来作为移动实时事务的一致性要求，性能最好。图 7 比较了几个协议的未满足事务截止期的比率。

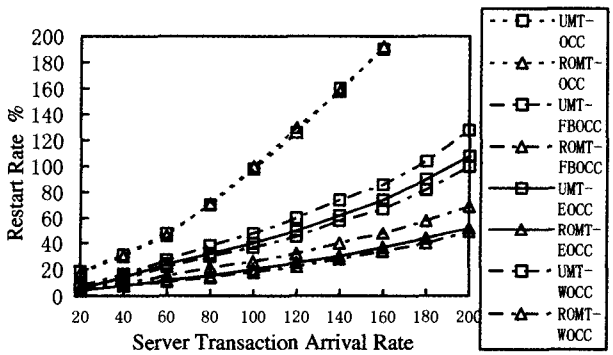

图 6 事务重启率

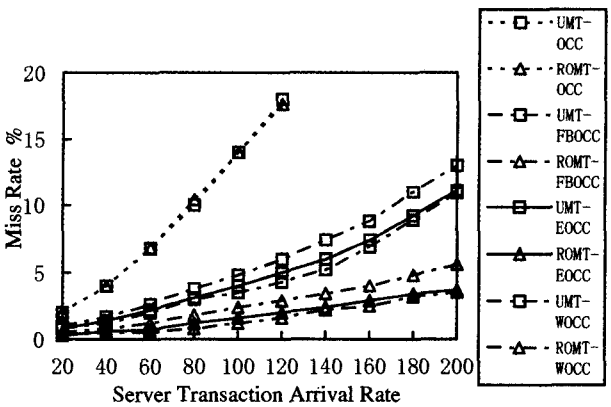

图 7 事务未满足截止期比率

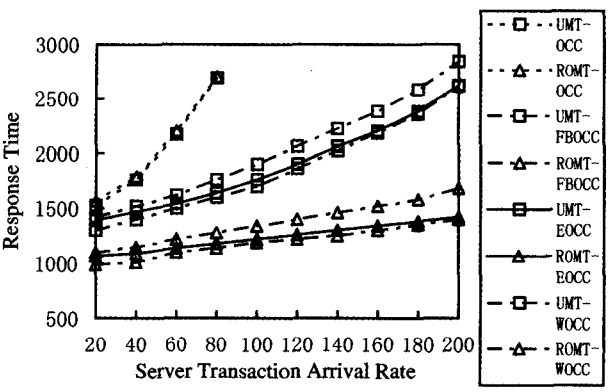

图 8 事务的响应时间

我们提出的协议需要移动事务满足弱一致性，能够不连接数据库服务器的情况下，自主执行。FBOCC 能够在移动客户机做部分验证，性能显然优于传统 OCC。*ROMT* 在服务器端保存验证信息，相对 UMT 能够更好满足事务截止期。ROMT 和 UMT 采用 EOCC 协议的性能比采用 FBOCC 协议和传统的 OCC 协议要好。主要原因在于验证方面，伪造冲突和严重冲突是有区别的。此外，只读事务的动态串行化顺序调整减少了 ROMT 的未满足事务截止期的比率。采用 OCC 协议的 ROMT 和 UMT 的性能比较相似，它们都需要到服务器端提交验证，随着服务器事务到达率的增加未满足事务截止期的比率也逐渐增加。尽管服务器负载不高，资源冲突影响不大，随着服务器事务数量增加，移动事务和服务器事务的数据冲突机也会加大，并且广播环境的带宽延迟比有线系统

大，重启事务不能满足其截止期的比率随之大大增加。

图8比较响应时间在不同协议之间的比较。可以看出，由于数据广播环境的缺点，OCC协议具有最长的平均响应时间，我们提出的协议由于较低的重启率和未满足事务截止期的比率，性能最好。

结论 在本文，我们结合乐观并发控制协议和时标排序协议提出了实时广播环境下的有效乐观并发控制协议，它支持不连接服务器的只读移动事务的自主执行，尽管移动更新事务不得不通过有限的上行带宽到数据库服务器做最后的验证，通过取消那些最终会夭折的事务，它还是节省了带宽的消耗。因此，通过把数据库服务器中更新事务执行写操作的数据对象实时包含在数据广播通道中，我们可以区分出严重冲突和伪造冲突，移动事务成功提交的可能大大增加。试验结果表明，移动只读事务满足弱一致性，更多事务能够满足截止期，我们能够获得更高层次的并发控制。

参考文献

1 Alonso R, Korth H. Database systems issues in nomadic computing. In: Proc. of the ACM SIGMOD Conference, Washington D C, June 1993. 388～392

2 Imielinski T, Badrinath B R. Mobile wireless computing: challenges in data management. Communications of the ACM, 1994, 37 (10): 18～28

3 Stabjivic J A, Son S H, Hansson J. Misconceptions about real-time databases. Computer, 1999, 32(6): 29～37

4 Pitoura E, Bhargava B. Building information systems for mobile environment. In: Proc. of the 3rd international conference on information and knowledge management, 1994. 371～378

5 Garcia-Molina H, Wiederhold G. Read-only transactions in a distributed database. ACM Transactions on Database Systems, 1982, 7(2): 209～234

6 Barbara D. Certification reports: supporting transactions in wireless systems. In: Proceedings of 17th International Conference on Distributed Computing System, USA, 1997. 466～473

7 Acharya S, Alonso R, Franklin M, et al. Broadcast disks: data management for asymmetric communication environments. In: Proceedings of the ACM SIGMOD Conference, 1995. 199～210

8 Herman G, Gopel G, Lee K C, et al. The datacycle architecture for very high throughput database systems. In: Proceedings of the ACM SIGMOD Coference, 1987. 97～103

9 Shanmugasundaram J, Nithrakashyap A, Sivasankaran R, et al. Efficient concurrency control for broadcast environments. In: ACM SIGMOD International Conference on Management of Data, 1999. 85～97

10 Lee V C S, Lam K W, Kuo T-W. Efficient validation of mobile transactions in wireless environments. Journal of Systems and Software, 2004(1): 183～193

11 Pitoura E, Chrysanthis P K. Scalable processing of read-only transactions in broadcast push. In: Proceedings of the 19th IEEE International Conference on Distributed Computing System, 1999. 432～439

12 Pitoura E. Supporting read-only transactions in wireless broadcasting. In: Proc. 9th Workshop on Database and Expert Systems Applications, Vienna, Austria, August 1998. 418～422

13 Pitoura E, Chrysanthis P K. Exploiting versions for handling updates in broadcast disks. In: Proc. 25th Int. Conf. on Very Large Data Bases, Edinburgh, UK, September 1999. 114～125

14 Bernstein P A, Hadzilacos V, Goodman N. Concurrency control and recovery in database systems. Addison-Wesley, 1987

15 Lee V C S, Lam K W, Son S H. Concurrency control using timestamp ordering in broadcast environments. The Computer Journal, 2002, 45(2): 410～422

16 Li Guohui, Yang Bing. Efficient Optimistic Concurrency Control for Mobile Real-Time Transactions in a Wireless Data Broadcast Environment. In: Proc. of 11th IEEE Int Conf. on Embedded and Real-time Computing Systems and Applications, August 2005. 443～446

17 Bober P M, Carey M J. Multiversion Query Locking. In: Proceedings of the VLDB Conference, Vancouver, Canada, August 1992. 497～510

(上接第35页)

的抗干扰性至少高出3.41dB。

结论 BOC调制方式应用逐渐广泛，因此对其抗干扰性的研究势在必行。本文对比于目前扩频通信中常见的BPSK调制方式，分析研究了BOC调制的功率谱和自相关特性及其存在的优势；在其潜在的干扰中，以阻塞式压制干扰为例，开展研究并进行仿真验证。通过仿真最终得出，基于本文所设计的信号同步环路，在同等条件下BOC(10,5)调制信号比BPSK(10.23)信号的抗干扰性至少高出3.41dB，为进一步的BOC调制信号抗干扰技术的研究提供了理论和仿真验证基础。

参考文献

1 邱致和. GPS M码信号的BOC调制[J]. 导航，2005，3(1)

2 谭显裕. GPS在导航战中的作用及其干扰对抗研究[J]. 现代防御技术，2001(6)：42～47

3 Betz J W. Effect of Jamming on GPS M Code Signal SNIR and Code Tracking Accuracy[J]. In: Proceedings ofION 2000 National Technical Meeting, Institute of Navigation, January 2000

4 Wolfert R, et al. Direct P(Y)-Code Acquisition under a Jamming Environment[J]. In: IEEE 1998 Position Location and Navigation Symposium, April 1998. 228～235

5 Davenport R G. FFT Processing of Direct Sequence Spreading Codes Using Modern DSP Microprocessors. In: Proceedings of the IEEE National Aerospace and Electronics Conference, NAECON 1991, 1: 98～105

6 邢兆栋，张其善，杨东凯. GALILEO接收机中BOC(1，1)信号的捕获[J]. 北京航空航天大学学报，2006，32(6)