分布式实时事务处理*)

Distributed Realtime Transactions Processing

刘云生　党德鹏

(华中科技大学计算机学院　武汉 430074)

Abstract　Many realtime applications are inheretly distributed. As developing in centralizied realtime database systems and mature of distributed database system, distributed realtime database systems have been a new hot point. This paper focuses on the distributed transactions processing, which is the key problem in DRTDBS, and gives some possible approaches, in order to provide some advice for future research.

Keywords　Realtime database, Distributed database, Distributed realtime database, Distributed realtime transaction processing

实时数据库系统利用数据库技术来有效地管理实时应用中的大量数据，以提高系统功能及灵活性[1,2]。事实上，诸如股票交易、导弹/飞行器控制、计算机集成制造系统、空中交通管制等许多实时应用本来都是分布式的，从而，随着实时数据库研究的深入及分布式数据库技术的成熟，分布式实时数据库便开始成为新的研究热点[3]，已经吸引了来自实时系统、数据库系统、并行/分布式系统等多个领域的专家学者。

分布式数据库系统是由若干相互自主的单节点数据库系统松散耦合成的一个总的数据库系统。其中数据以数据片段/复制的形式分布于各个节点上；事务具有访问共享数据的能力，即它可以访问存储于远程节点上的数据。有两种事务：仅访问和更新一个节点中数据的事务叫局部事务；需访问和更新多个节点中数据的事务称全局事务。分布式实时数据库研究的核心论题是事务处理，即事务的调度必须同时满足截止期、局部一致性及全局一致性。为此，执行事务的各节点需要交换调度信息，而信息交换引发的通讯延迟对于事务的响应时间来说是一个很大的开销，这对满足分布式实时事务的截止期很不利。可见，在分布式实时数据库中解决截止期与一致性这一对矛盾比在集中式实时数据库中困难得多，我们需要进一步研究分布式的实时并发控制协议、提交协议、事务调度及数据复制控制协议等。

1. 数据复制

如果关系 r 或数据片段 p 被复制，则在两个或两个以上的节点存有关系 r 或 p 的副本。极端情况下，我们采用全复制，此时系统中每个节点都存有 r 或 p 的一个副本。

1.1　实时数据复制对系统性能的影响

复制带来的优点在于：①更好的可用性。当包含 r 或 p 的节点之一发生故障时，r 或 p 可在另一节点上找到。因此，尽管一个节点上发生了故障，系统仍可以继续处理涉及 r 或 p 的查询。②更高的并行度。如果绝大多数对 r 或 p 的访问只是读取，那么几个节点就可以并行地处理涉及 r 或 p 的查询。r 或 p 的副本越多，在事务执行的节点上发现所需数据的可能性就越大。这样，数据复制减少了数据在节点间的移动。

另一方面，数据复制也带来了问题：增加了更新的开销。系统必须保证 r 或 p 的所有副本是一致的，否则就可能产生错误的计算。因此，只要 r 或 p 被更新了，更新就必须传播到包含副本的所有节点。研究表明，多副本更新导致的持有副本的各节点间通讯的开销对实时事务的时间约束来说是很可观的[3,4]。数据复制对面向更新的应用没有任何吸引力；然而，除非大多数事务是面向更新的或系统负载很高，存储数据的多副本(但不要太多)总可以得到更好的性能。这里需要深入研究的一个问题是依据应用类型数据访问的分布性数据可

*)本文研究得到国家自然科学基金、国防预研基金资助。刘云生　教授，博士生导师，主要研究领域为现代(实时、主动、分布式、时序、内存)数据库理论与技术及其集成实现、数据库与信息系统开发及实时应用，软件方法学及支撑环境。党德鹏　博士生，主要研究方向为实时数据库，主动数据库，分布式主动实时数据库。

用性要求来确定存多少个副本最佳[5]。

1.2 复制控制算法

关于复制控制算法，目前主要研究有以下五种：

• 集中加锁　所有对数据元素的加锁请求都要送到位于某个节点的集中式锁管理器。所有封锁和解锁请求都在此节点处理。该协议实现简单，但通讯开销也最大。

• 主副本更新　选择一个副本来作为主副本，所有数据项的更新操作都指向其主副本，然后，由主节点将更新向其它副本广播。它对那些读操作事务不要求完全一致或最新数据的应用系统，可更好地满足时间约束，但主节点易成为瓶颈。

• 基于令牌　允许各副本轮流担任主节点或存在多主节点，即依令牌持有者为主节点，这可提高更新操作的性能。

• 一致同意原则　协调节点向所有其它节点发送更新请求，只有协调节点收到所有副本所在节点回答的准备好消息，才开始执行更新，否则，更新被拒绝而不在任何接点上进行。它保证了所有数据项的副本都是正确的并且也是最新的，

• 大多数投票　对某一数据项的读写请求需发送到大多数副本上，只有收到超过了某个给定数目的可以读写的应答、读写操作方可进行，

我们需要大量实验研究，以评价比较这些算法的性能，并进一步研究其它更好的算法。

2. 并发控制

在一个分布式数据库系统中，每个节点上的调度器负责控制对存于该节点上的数据项的访问。局部和全局访问请求都依正在运行的并发控制协议进行排序，分布式实时数据库环境中需要运行实时并发控制协议的分布式版本。目前主要研究了分布式锁式并发控制协议[5]，其中主要研究了在分布式环境中锁冲突的解决：

• 分布式高优先级夭折协议(DPA)　其锁冲突解决算法如下：

```
LockConflict(tr,th)
  { if Priority(tr)>Priority(th)
      if local(th)
        local_restart(th)
      else
        global_restart(th)
      endif
    else
      block(tr)
    endif}
```

其中，t_h 表示持有锁的事务，t_r 表示请求锁的事务，local(t_r)表示 t_r 是局部事务，global(t_r)表示 t_r 是全局事务，以下类同。

• 分布式高优先级继承协议　其锁冲突解决算法如下：

```
LockConflict(tr,th)
{ if Priority(tr)>Priority(th){
   block(tr);
   Broadcast UpdatePriority(th,Priority(tr)to all sites)
     ;
   else
     block(th)
   end if }
```

由于低优先级事务可能涉及多个节点，需要将其优先级进行广播，算法如下：

```
UpdatePriority(th,Priority)
  { if Priority(th)<Priority
      the Priority of th's participants.=Priority
    end if ;
```

• 分布式混合式协议　其锁冲突解决算法如下：

```
LockConflict(tr,th)
 {if Priority(tr)>Priority(th){
   if ((Lih-Xik<=h)or committing)(th)){
    block(tr);
    Broadcast UpdatePriority(th,Priority(tr)to all sites
    ;
    else
    {if local(th) local_restart(th)
     else global_restart(th)
     end if ;
   end if
 end if }
```

其中的广播算法与上面的相同。

我们需要研究各种实时事务并发控制算法在引进分式式环境后引发的新问题并对各种算法的性能进行比较。

3. 提交处理

一个分布式实时事务需要分解为若干个子事务并发送到多个节点上协同运行。为了保证事务的原子性，需要原子提交协议以保证各参与节点上的子事务要么全部提交要么全部不提交，即保证执行事务的各节点在事务执行的最终结果上取得一致。传统分布式数据库中保证分布式事务原子提交特性的标准方法是2PC/3PC 协议[6]，但是因它们的不可预报性及过高的时间耗费而不适合于实时数据库环境，对于分布式实时数据库系统，目前提有两种方案：一是设计一种自适应的提交协议，它可在不同负载条件下，动态改变到不同的提交策略，比如必要时可采用放松了原子性的提交方法，二是使用补偿事务，各子事务自由提交，之后对不正确提交的事务用其补偿事务进行基于语义的 undo 操作，这是一种乐观的提交方法，

4. 子事务截止期分配

分布式实时数据库系统中的事务调度总是使用嵌套的事务模型，即一个事务由若干可能在不同节点运

(下转第25页)

工具、语音工具、用户管理、会话管理、共享文件等功能，Ashram 系统目前运行在由 Sun Ultra30、IBM RS/6000等工作站和 Pentium PC 等构成的异构分布式网络上（含各种多媒体设备），操作系统包括当前流行的 Windows 98/NT、AIX、Solaris 等。

Ashram 采用图1所示的半复制式体系结构，由3.1节所述的工具包开发其底层通信支持，采用会话管理工具包提供的 API 来管理和显示资源树。资源树中的共享文件功能采用共享文件工具包开发；另外，视频、音频的采集、播放、传输部采用了工具集的多媒体构件。

Ashram 以 Java 实现，使其适合于跨平台异构型网络应用。它主要有以下特点：①支持灵活耦合，允许用户对其与其他用户之间的交互粒度进行配置（如协同编辑器中用户进行的行、段为单位的协同粒度的配置）；②支持基于多媒体信息的组感知；③支持语音、视频及多媒体信息；④支持同步与异步工作的结合。

结论 本文讨论了多媒体支持的协同系统开发工具集的设计与实现。创新之处有：

1）根据用户所处工作位置相关性，采用了灵活的工作空间感知处理机制，特别是融入了实时视频信息，增强了协同工作组的协同感和真实感；

2）在多媒体多路传输中，对数据流和控制流采用清晰的层次结构，构件独立性和可扩展性好，并且将报文控制和 QoS 控制有机地结合起来，尽可能地满足实时性要求；

3）将协同系统中繁琐的协同方式的处理、网络信息处理、多媒体信息处理等过程以 API 类库形式提供给程序员，方便了协同系统开发，有效地缩短了开发周期。

参考文献

1 Hill R，et al. The Rendezvous Architecture and Language for Multi-User Applications. ACM Transactions on Computer-Human Interaction，1994，1(2)：81～125

2 NCSA Habanero，1996. NCSA Habanero Home Page. www.ncsa.uiuc.edu/SDG/Software/Habanero/

3 Dewan P A Tour of the Suite User Interface Software. In Scott Hudson，ed. Proc. of the 3rd ACM SIGGRAPH Conf. on User Interface Software and Technology，ACM，New York，1990. 57～65

4 Abdel-Wahab H，et al. An Internet Collaborative Environment for Sharing Java Applications. In Proc of the 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems（FTDCS'97），Tunis，Tunisia，1997. 112～117

5 Dourish P Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications. ACM Transactions on Computer-Human Interaction，1998，5(2)，109～155

6 Dewan P，Choudhary R. A High-Level and Flexible Framework for Implementing Multiuser User Interfaces ACM Transaction of Information Systems，10(4)，345～380

7 詹永照，茅兵，吴敏强，谢立 协作共享对象模型及其实现技术. 计算机学报

8 詹永照，窦万峰，李成锴，茅兵，谢立. 基于共享对象划分的工作空间感知处理模型. 计算机研究与发展，2000，37(3)：353～358

（上接第20页）

行的子事务组成。往往顶层事务的截止期不能反映子事务的紧急程度，而子事务却需要自己的截止期以便获得合适优先级去竞争资源，这就需要有效的算法来由总截止期导出各子事务的截止期[7]。考虑到各子事务间的执行依赖关系，该问题可分解为两个子问题：

·串行子事务截止期计算 设顶层事务 T 的到达时间记为 AT，截止期记为 DT，宽松时间记为 ST，执行时间记为 ET，$T=\{T_1,T_2,\cdots,T_n\}$，则各个子事务的截止期可用以下公式计算：

$$D_i=AT+\sum_{k=1}^{n-1}E_i+(i-1)\times ST/n$$

·并行子事务截止期计算 公式如下：

$$D_i=AT+\sum_{k=1}^{n-1}E_i+(i-1)\times ST/(k\times n)$$

将这两种方法结合，就可进行一般嵌套子事务的截止期分配。

这只是目前提出的一种粗略的分配方法，实际上，由于子事务间相互依赖的关系很复杂，各个子事务的紧急程度也可能差别很大，我们还需要进行大量的工作以研究更适合于各种不同类型应用特征的方法。

参考文献

1 Ramamritham K. Real Time Database Int. Journal of Distributed and Parallel Database，1992

2 刘云生. 实时数据库系统. 计算机科学，1994，21(3)

3 Kohler W H. A Survey of Techniques for Concurrency Control. ACM Trans. on Database System，1981，6(2)

4 Sha L，Rajkumar R，Lehoczky J. Concurrency Control for Distributed Realtime Databases. ACM SIGMOD Record，March 1988

5 Carey M，Livny M. Distributed concurrency control performance：A Study of Algorithms，Distribution，and Replication. In：Proc. of 4thIntl Conf. on Very Large Databases，August 1998

6 Silberschatz A，Korth H F，Sudarshan S. Database System Concepts U. S. A.：The McGraw Hill Company，1999

7 刘云生，李国徽. 实时数据库嵌套事务的并发控制. 小型微型计算机系统，1998，19