面向网络的一种新分布式事务处理协议*

刘 煜** 蔡希尧
(西安电子科技大学软件工程研究所 西安 710071)

【摘要】 提出了一种新的适合于当前网络环境,基于客户机/服务器应用方式及远程过程调用(RPC)的分布事务处理模型及其实现协议。分析与实验表明它不仅满足网络中客户机/服务器应用的具体特点与要求,而且可以克服各种两段式提交协议中由于协调者故障造成的阻塞,同时避免了三段式提交等协议为解决阻塞而引起的开销过大,实现复杂等问题。

关 键 词 网络; 客户机/服务器; 分布事务处理; 远方过程调用
中图分类号 TP311.13

网络计算已成为计算机系统发展的必然趋势,基于网络的客户机/服务器体系结构是当前分布式系统广泛应用的逻辑互连方式,并已成为网络环境下分布式应用实际的标准开发模式。远程过程调用(RPC)是支持客户机/服务器模式工作的基本技术,它利用过程调用实现远程通信,具有与本地过程调用类似的语义。它可使网络中的计算机彼此访问对方机器上的资源与服务,实现网络内部的互操作性。

在网络系统中,事务处理具有越来越重要的意义。它不仅是分布式数据库技术的基础,而且逐渐应用到网络控制、管理等许多方面[1]。分布事务处理应具有如下特性:原子性、持久性、串行性、隔离性[2]。其中原子性最为重要,它保证事务或者全部完成,或者一个也不完成。为实现上述要求,Gray提出了两段提交协议(2PC),基本解决了分布事务处理问题,但该协议比较简单,效率不高,实际应用当中需要对其完善。随后出现了多种以2PC为基础的改进协议,性能较好的是Mohan等提出的假定异常终止(PA)及假定正常提交协议(PC)[3]。它们不仅能完成2PC的功能,而且在报文传送数目、报文延迟、恢复所需同步通信次数等方面有较大改进。但仍存在容易发生阻塞,无法适应永久节点故障等问题。随着网络、通信技术的发展,当前分布式系统向大规模、异构、客户机/服务器应用方式发展,对分布事务处理提出了新的任务与要求。人们相继研究出多种与此对应事务处理协议与模型,最有代表性的是K. Rothermel提出OC(Open Commit)协议[4]。它适用于开放式分布系统,可保证关键节点在节点故障情况下最终一致地结束事务,但它需要动态地调节协调者节点位置,并要改变事务执行树结构。这对一般客户机/服务器应用而言,算法复杂,开销大,不易实现。本文在前述各协议及我们研究实践的基础上,紧密结合网络环境下客户机/服务器分布事务处理的具体特点与要求,提出了一个简单、可靠的分布事务处理模型及其基于RPC的提交协议。该模型与协议解决了2PC与PA等协议无法解决的协调者永久故障引起的阻塞问题,又避免了OC与3PC(三段提交)协议等为解决这类问题而带来的算法复杂度太高、报文数量明显增加等不良后果。

① 1995年 10月 13日收稿,1996年 9月 25日修改定稿
* 国家教委博士点基金和国防预研基金资助项目
** 男 27岁 博士生

1 客户机/服务器模式下分布事务处理的特点和要求

设有如图1所示客户机/服务器网络商业应用环境,它由 $S_1, S_2, \cdots, S_n, C_1, C_2, \cdots, C_m$,共 $n+m$ 个网络节点及通信网络 N 组成。其中 $S_1, S_2, \cdots S_n$ 为服务器,$C_1, C_2, \cdots, C_m$ 为客户机。设某个用户将使用客户机 C_1 完成一次出国旅游的预定过程,首先需要订机票,设该过程要访问属于 A 公司的 S_1 服务器上的数据库;其次要订房间,设由 B 公司的 S_2 服务器上的数据库负责;最后需要预定旅游项目,设由 C 公司的 S_3 服务器上的数据库处理。由于 S_1、S_2、S_3 分属于三个不同的公司,无法将数据库集中到一处处理。并且,该客户要求通过客户机 C_1 或者一次完成全部预定任务,或者全部取消本次预定,因为只完成部分项目将无法使这次出国旅游顺利进行。这就是一个涉及到 S_1、S_2、S_3 及 C_1 四个网络节点的典型的分布事务处理问题。

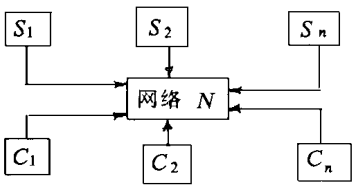

图1 客户机/服务器网络

当前,客户机/服务器应用环境下,网络可被认为由许多自治节点通过通信网络连接而成。网络规模可以很大,如除局域网外,还可能是城域网、广域网及无线网;网络中各节点的性质允许相差很大。服务器节点硬件配置较强,总线吞吐量大,工作速度高,并且一般具有完善的冗余、容错设施,如断电保护、多硬盘镜像、双热机备份等。其软件多采用稳定、可靠、具有保护和容错机制的高性能有操作系统,如 UNIX、Windows NT、Netware 等。并且当服务器节点出现硬件、软件故障时,还具有完善的故障恢复策略,可以较快地从故障状态回到正常状态。实用当中,如银行、保险、订票等商业系统的服务器普遍具备上述特点,能够不间断地提供可靠的服务。因此,我们称网络中服务器节点为可靠节点,并认为它们出现故障后,将在有限时间内恢复正常,不会永久处于故障状态。另一方面,客户机节点许多情况下只是一般的 PC机,除性能无法与服务器相比外,主要是其可靠性差,基本没有冗余、容错设施,操作系统以 DOS、Windows3.1等为主,容易出现故障,即使可恢复,也难于回到故障前状态。极端情况是用户因疏忽关机,可以认为该网络节点将一直无法从故障状态恢复。因此,称客户机节点为不可靠节点,允许它们永远处于故障状态。

在大多数客户机/服务器应用中,处于主动地位的是客户机上的应用程序,服务器只在接到请求后被动地提供服务。故在整个分布事务处理过程中,基本上是客户机上的应用程序控制着事务进行,即客户机节点一般是事务协调者的宿主节点。实际上,目前大多数客户机/服务器模式下的数据库都不能在多个节点间自动进行分布事务处理,而是以客户机上的应用程序作为协调者完成分布事务处理。

由两段提交等协议可知,协调者是事务处理的核心,它的任何故障将引起整个事务失败,并可能使所有参加者节点陷入阻塞状态。一旦出现阻塞,网络中大量资源被封锁,使其他事务执行效率降低或失败,并可能破坏数据的一致性与完整性。但是由前述分析可知,协调者所在的客户机节点正好属于网络中的不可靠节点,容易出现故障,这将造成客户机/服务器模式下分布事务处理的可靠性明显下降。

因此,在客记机/服务器分布事务当中,既要充分发挥客户机的处理能力,以客户机为核心,方便、灵活地控制分布事务的启动与执行,又需要考虑客户机与服务器节点在可靠性与性能方面的差异,尽量提高整个事务可靠性,减小出现阻塞的概率,确保在各种故障情况下,使各节点最终能够一致地结束事务。

2 分布事务处理模型及基于 RPC的实现协议

根据以上分析,建立如下分布事务处理模型: 分布事务的协调者是客户节点的应用程序,参加者是与分布事务有关的服务器节点。整个事务分布到多个服务器节点执行,参加者节点不含客户机,以保证事务执行的高效性、安全性与可靠性。对以往事务处理模型作如下改进: 因为客户机是不可靠节点,协调者关于整个分布事务的运行记录不能存于本地,否则客户机出现永久故障时,参加者无法从中获得事务处理信息。解决办法是将记录保存在属于可靠节点的某个服务器上,它可以是该事务的一个参加者,我们称该服务器为记录服务器。它在事务进行过程中,只与协调者保持联系。如果事务进行过程中发生故障,参加者将利用属于可靠节点的记录服务器上的运行记录决定故障恢复策略。

现给出一个客户机/服务器环境下分布式数据库应用的实例。它是上述事务处理模型的具体实现,从中还可以了解与本模型对应的事务提交协议的组成与特点。设本次事务涉及两个参加者 dbserver1与 dbserver2,它们是两个独立运行 DBMS的服务器,协调者记录服务器为 recorder,客户机节点可能有多个,其中某个客户机节点应用程序中的协调者用 RPC实现的程序框架如图 2所示。

从图 2可以看出,分布事务中的 send/wait原语与事务处理 SQL语句的执行只需一次 RPC调用即可完成,它具有丰富的语义。首先相当于协调者执行 send原语,将事务操作命令发往某个参加者,该参加者此时执行 wait原语,接收命令。然后参加者调用本地过程,它们一般以存储过程的形式工作,可以是系统的或用户定义的。调用完毕,该参加者执行 send原语,将执行结果返回协调者,对应协调者再以 wait原语形式接收结果,该结果是本次 RPC调用的返回值。这些值代表参加者对于协调者所发事务命令的投票值,在不同上下文中可能代表 ready、not-ready、ack等语义。用 RPC形式实现分布事务处理,将以往多个 send/wait原语及多个事务处理操作合并为一条语句,提高了抽象层次与结构化程度,使程序简洁、清晰、易于维护。RPC以函数调用形式实现,适合于第三代过程语言环境。这些都将显著简化包含分布事务处理的应用开发。

图 3是与图 2对应的协调者状态转换图。从中可以看出该协议对 2PC协议进行了改进,当协调者处于 will commit状态,将显式增加一个写 commit到记录服务器的等待状态,并根据返回值决定是否继续事务。图 2中以 prepare为参数的 RPC调用表示开始两段提交的第一段,返回值代表 ready或 not-ready,以 commit为参数向记录服务器发出的 RPC调用将 commit写入运行记录,进入两段提交的第二段。若写记录成功,表示本次事务完成,不论出现什么故障,参加者都将提交事务。

从图中还可以看出,该协议也包含 PA协议,当协调者根据 RPC返回值决定异常终止事务时,立即用调 abort子程序,首先将异常终止写进运行记录,并不要求写操作一定成功。再向所有参加者发出异常终止命令,也不要求所有参加者一定确认收到命令。然后立即退出,并完全"忘记"本次事务。显然,分布事务基于上述操作仍能正确结束,因为各节点遵循这样一个原则: 对于分布事务中没有信息可参考的恢复查询的正确回答是"异常终止"。这也是 PA协议的基础,故图 2协议蕴含了 PA协议。

3 模型的故障恢复策略

如图 2所示,从事务开始到图中标(1)处的这段时间内,无论协调者、参加者出现节点故障,还是它们之间的连续发生中断,参加者能够根据本地事务记录作出异常终止事务的决定,恢复到故障前状态,不出现阻塞。事务执行到图 2中(2)处,假设协调者所在客户机节点发生永久不可恢复故

障,参加者无法再接收到来自协调者的事务命令。等待超时后,该参加者将根据本地事务记录中关于信息服务器的信息访问它,得知本次事务尚未提交,该参加者可作出异常终止本次事务的决定,恢复到事务开始前状态。若此时记录服务器也发生故障,参加者将无法获得有关本次事务的信息,不能作出决定,将暂时处于阻塞状态。但由于记录服务器属于可靠节点,一段时间后将从故障状态中恢复过来,这时参加者就可以脱离阻塞,故所有参加者最终可以一致地结束本次事务。

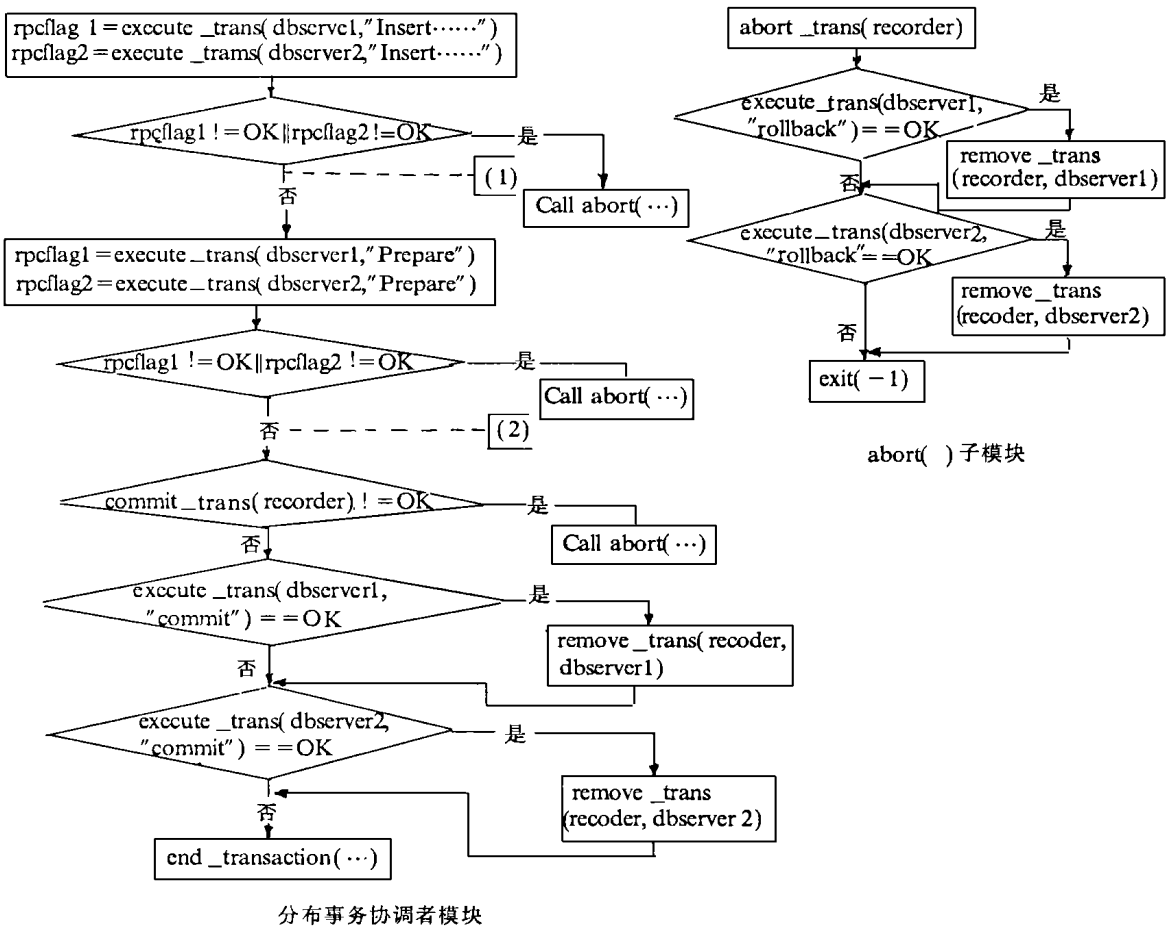

图 2 客户机节点协调者程序框图

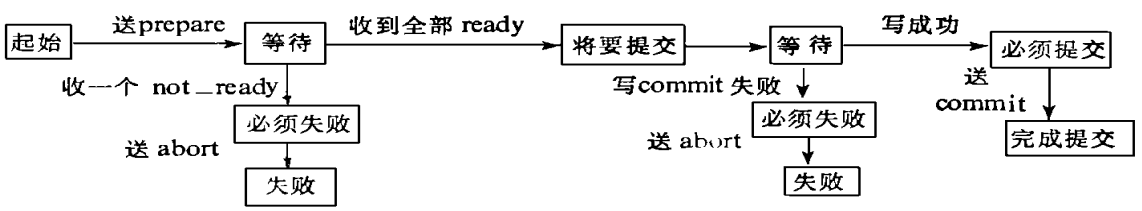

图 3 协调者状态转换图

3PC对此类型故障的处理方法与上述方法不同,它需要附加一个阶段完成提交,出现上述故障后,参加者之间先要相互通信,如果都未收到协调者的提交命令,则要按某种规则选一个参加者作为新的协调者继续完成事务。所以,与前述方法相比,3PC要求节点之间交换更多的报文,恢复时的处理机制也比较复杂。OC协议这样解决该问题:在开始阶段不直接进入两段提交,先经过一个转换阶段,在事务执行树中将协调者角色动态地从不可靠的客户机节点转移到某个可靠的服务器节点,并构造新的执行树,以此提高分布事务的可靠性。但是,OC协议中计算分布执行树,动态转移协调者并构造新树的算法本身就很复杂,并且要求每次开始新事务时,就要从头计算一遍,所以开销大,比较难于实现。

本文中事务处理方法之所以能用于 2PC基本相同的开销解决了协调者故障造成的阻塞问题,关键在于改变了以往事务处理模型中对于运行记录的习惯处理方法。在其他故障情况下的恢复问题与前述方法类似,在此从略。

4 结 束 语

本文提出的分布事务处理模型及用 RPC实现的协议在 Sybase System 10开放式客户机/服务器环境下得到具体实现,操作系统为 Netware 3.12,客户机是中文 Windows 3.1,服务器 DBMS为 Sybase SQL Server,结果表明前述模型及协议可以较好地完成客户机/服务器模式下的分布事务处理。当出现该模式下最容易发生的协调者节点故障时,仍可确保事务最终以一致的方式结束,解决了 2PC、PA等两段协议在此情况下可能出现的阻塞问题,同时相对于能够解决阻塞的 3PC、OC等协议在事务执行效率、实现复杂性、执行开销等方面都具有较明显的优点。

参 考 文 献

1　Pano L, Ramamrithan K. Synthesis of extended transaction methods using ACTA. ACM Trans on Database System, 1994, 19(3): 450~491

2　Ceri Stefano, Pelagati Giusepe. Distributed database, principles and systems. New York: McGram-Hill, 1984

3　Mohan C, Lindsay B, Obermark R. Transaction management in the R* distributed system. ACM Trans on Database Systems, 1986, 11(4): 378~396

4　Rothermel K, Pappe S. Open commit protocols tolerating commission failure. ACM Trans on Database Systems, 1993, 18(2): 289~332

5　陈建容,严隽永.分布式数据库设计导论,北京:清华大学出版社,1992: 99~165

A New Kind of Distributed Transaction Protocol in Network Environment

Liu Yu　　Cai Xiyao

(Software Engineering Institute, XiDian University　Xian　710071)

Abstract　A distributed transaction processing model and its realizing protocol based on RPC are proposed for client/server applications in current network environment. Analysis and experiment show that it not only satisfies specific properties and requirements of client/server application, but also overcomes the blocking in the two phase commit protocol caused by coordinator breakdowns, and avoids shortcomings of more expenses and difficult to realize in three phase commit protocol.

Key words　client/server;　distributed transaction processing;　remote procedure call

编辑　徐培红