分布事务处理中的资源管理*)

张 昕 丁 柯
(中国科学院软件研究所 计算机科学重点实验室 北京 100080)
(中国科学院软件研究所 软件工程技术中心 北京 100080)

Resource Management in Distributed Transaction Processing

ZHANG Xin DING Ke
(Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)
(Software Engineering Technology Center, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)
Email: zhangxin@otcaix.iscas.ac.cn

Abstract The environment of distributed transaction involves a number of resource managers, and we should do operations on resources such as connection and enlistment during a transaction. Connecting to a resource will cost a lot of time, so connection pool mechanism is put into use on account of improving performance. And the resource connections in the connection pool need to be scheduled properly. In this paper, we specify how to implement resource enlistment and delistment in a distributed transaction, and present an algorithm of resource connection reusing and scheduling. Now the resource connection reusing and scheduling algorithm has been successfully employed in the implementation of the distributed transaction manager ISTX 1.0 system.

Keywords Distributed transaction, Resource management, Resource enlistment, Connection pool

1 引言

在分布事务处理环境中，资源管理器(数据库管理系统，可靠消息队列以及事务性文件系统等是最常见的资源管理器[1])扮演着数据管理的核心作用，因此，分布事务处理中的资源管理的研究显得极为重要。

分布事务总是涉及多个资源管理器，它需要记录有哪些资源管理器参与了该事务，因此，所有参与此事务的资源管理器都必须在事务中注册，使得应用程序能够使用此资源；同时，资源管理器也需要知道自己在哪些事务中被注册。这样，分布事务管理器在提交事务时能够通知这些资源管理器，它们共同完成两阶段提交协议[2]。

通常，连接资源管理器是一项耗时的操作，而分布事务管理器中的连接数量是有限的，如果频繁地请求获取新的连接，将会极大地影响性能，这在 Web 服务器环境中可能会出现。为了降低开销和提高性能，应该尽量复用已经建立的连接。连接池机制(connection pooling)作为一种优化手段，能够较大地提高性能[3]。分布事务管理器创建资源连接池，用它来存放已经建立好的但暂时不用的资源连接，资源连接在使用完毕后并不立即释放，而是存入连接池中，可以为以后的连接请求复用。

连接池有一定的大小，只能存放一定数量的资源连接，因此对于池中的资源需要进行适当的调度，将长时间未被使用的资源连接调出连接池，使池中的资源连接均保持较高的使用频率，不恰当的调度算法有可能极大地降低系统性能。

我们在分析现有资源管理技术的基础上，研究了分布事务中的资源注册和资源调度的方法，并在 ISTX 1.0 中给出了实现。

2 分布事务中的资源连接和资源注册

分布事务涉及多个资源管理器。在 J2EE 框架中，事务和资源之间的连接，通常是通过应用服务器来进行的。连接池作为提高性能的一种手段，也是应用服务器的一个组成部分，由应用服务器进行实现和管理。事务和资源的连接，在具有连接池的环境中，都变成和连接池相连。图 1 表示了一个由应用服务器管理的具有连接池的资源连接。

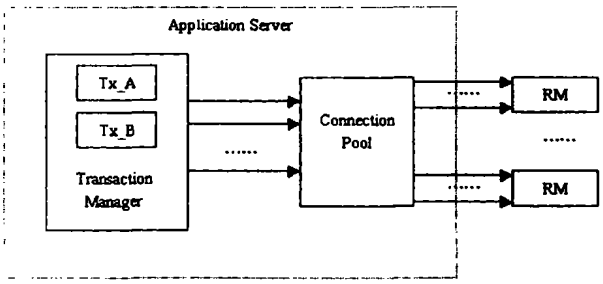

图 1 资源连接

资源注册(enlistment)和注销(delistment)也是由应用服务器来完成的。为了能够调用资源管理器，完成分布事务，应用服务器需要将事务中使用到的资源在事务中进行注册，并在使用完毕后注销。资源注册有如下两个目的[2]：

· 向事务管理器通报参与到事务中的资源管理器，并允许事务管理器通过 connection 对象向此资源管理器告知其已经连接到某个事务中。

· 使得事务管理器可以根据不同的事务对资源进行分组，便于事务管理器管理 TM 和 RM 之间的两阶段提交协议。

*)本文研究得到国家 863 高科技发展计划资助项目(编号 2001AA414020)、国家重点基础研究发展规划 973 资助项目(编号 G1999035807)、国家自然科学重点基金资助项目(编号 69833030)的资助。张 昕 博士生，主要研究领域为软件工程和网络分布计算。丁 柯 博士生，主要研究领域为分布式计算，分布事务处理技术。

在JTA框架中，资源注册的具体实现过程为：对于每个应用程序使用到的资源，应用服务器调用Transaction.enlistResource方法，并用XAResource对象来表示所需要注册的资源。javax.transaction.xa.XAResource接口是X/Open规范定义的XA接口的Java映射，它定义了资源管理器和事务管理器之间的关系，并由一个资源适配器(resource adapter)实现XAResource接口，负责事务和资源的连接。Transaction.enlistResource方法最终会使事务管理器调用XAResource.start方法，通知资源管理器，将事务和相应的资源进行连接。

Transaction.enlistResource方法有一个参数XAResource，用于通知事务，此资源在事务中注册，而XAResource.start方法有以下两个参数：

1. XID，即事务ID，唯一表示事务，用于通知资源已经在此事务中注册；

2. 标志位，用于表示事务的状态，事务管理器负责将此标志位传递给资源管理器。

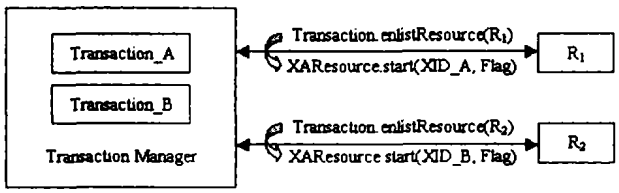

图2 资源注册

图2表示资源R_1在事务A中注册，R_2在事务B中注册的过程。

如果已经有其他的XAResource对象注册到此事务中，事务管理器则会调用XAResource.isSameRM方法来判断新注册的XAResource对象和原来的XAResource对象是否代表同一个资源管理器的实例。这样做的目的是出于性能优化的考虑，因为资源的注册和注销可能比较费时[4]。如果二者代表同一个资源，则资源管理器将新注册的资源和原来的XAResource对象归入同一组，确保同一个资源管理器只会在事务完成时收到一次"请准备提交"的消息；如果二者不代表同一个资源管理器，事务管理器建立一个不同的事务分支ID(transaction branch ID)，确保新的资源管理器在事务完成准备提交时会收到所需的通知。

Transaction.delistResource方法用于将指定的资源从事务上下文中注销。此方法有以下两个参数：

1. XAResource对象，代表已注册的资源。

2. 标志位，指明资源注销的原因，有以下几种可能：事务被挂起(TMSUSPEND)；事务的某些操作失败(TMFAIL)，例如资源连接失败；正常情况下的资源释放(TMSUCCESS)。

delistResource请求会使事务管理器通知资源管理器，结束事务与XAResource之间的连接，事务管理器用XAResource.end方法传递合适的标志位到资源管理器。

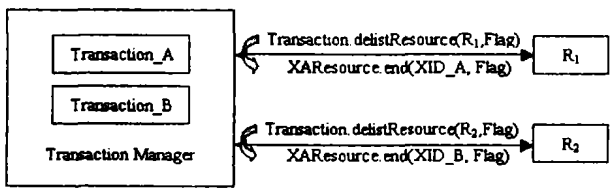

图3 资源注销

图3表示资源R_1从事务A中注销，R_2从事务B中注销的过程。

事务管理器为每个参与到事务中的资源管理器获取一个XAResource接口，并且使用start方法将事务和资源进行连接，用end方法使两者断开连接。

XAResource.start方法以及XAResource.end方法都由各个资源适配器实现。事务在和资源进行连接时，需要申请一个connection对象，通过connection对象和资源进行连接。资源适配器需要保证在任意一个时刻，一个connection对象最多只和一个事务相连。

如果有多个事务和同一个资源相连接，则需要由事务管理器在适当的时候调用XAResource.start和XAResource.end方法，进行事务上下文切换。每次当资源被另外一个事务使用时，都需要调用XAResource.end方法，断开前一个事务和此资源的连接，然后调用XAResource.start方法，调入当前事务上下文。如果在一个connection对象还和另外一个事务相连时调用XAResource.start方法，则会产生异常，表1描述了不同情况下调用各个方法时事务状态之间的转换。

表1 事务连接状态

XAResource方法	XAResource事务状态		
	未连接T_0	已连接T_1	连接挂起T_2
start()	T_1		
start(TMRESUME)			T_1
start(TMJOIN)			T_1
end(TMSUSPEND)		T_2	
end(TMFAIL)		T_0	T_0
end(TMSUCCESS)		T_0	T_0
Recover()	T_0	T_1	T_2

其中T_0表示资源没有和任何事务相连，T_1表示资源和某个事务正常连接，T_2表示资源连接被挂起，表中的空白则表示抛出异常。

3 连接池调度

性能是事务处理的一个关键方面，评价一个事务处理系统性能的重要指标有反应时间以及吞吐率等。为了提高系统性能，出现了很多优化措施，连接池就是其中非常重要而且有效的一种。

3.1 连接池使用的目的

在获取资源连接时，每个对新的资源连接的请求都会导致很大的开销。譬如，一个典型的JDBC数据库连接请求的底层途径如下：首先，Java应用程序调用getConnection方法，由JDBC请求一个来自JVM的Socket连接；然后JVM需要检查底层调用的安全方面；如果权限允许，调用穿过主机网络接口，到达局域网上；调用可能还需要穿过防火墙，到达Internet或广域网，然后最终到达目的子网络，在那里它可能需要穿过另外一个防火墙，然后到达数据库主机；数据库服务器处理新的连接请求，此时认证服务器可能需要通过查询来确定是否有适当的许可和认证；然后数据库初始化新的客户机连接，包括所有内存和操作系统开销；返回调用被送回JDBC客户机(在那里它必须穿过所有防火墙和路由器)；JVM收到返回调用，然后创建适当的connection对象；最后，请求的Java应用程序收到connection对象。从上述过程中可以很明显地看出，请求一个新的connection对象会带来大量的系统开销和很多潜在的问题[5]。其他的资源连接过程和代价与这个例子类似，因此，请求一个新的资源连接会耗费较大

的系统代价，如果连接操作比较频繁，系统性能将因此极大降低，出于性能优化的考虑，在很多实现中引入了连接池的方法。我们可以把连接当作对象或者设备，连接池中有许多已经建立的连接，本来需要与资源进行连接的地方，都改为和池相连，池临时分配连接供访问使用。资源连接在使用完毕后将连接交还，存入连接池而不是删除，以后对此资源的连接请求，直接从连接池中查找是否有已经建立好的连接，如果有，则直接复用。

连接池只能存放一定数量的资源连接，因此，需要对池中的资源进行适当的调度，将不经常使用的连接调出，经常使用的连接调入。以下是在 ISTX 1.0 的实现中使用的连接复用和调度算法。

3.2 连接复用和调度算法

一个好的连接复用和调度算法应该使得应用程序尽可能地复用已有连接，并且使得连接池中的资源连接都能保持较高的使用频率，提高系统性能。本文提出了一个连接复用和调度算法，并已经应用于 ISTX 1.0 中，取得了很好的效果。

3.2.1 *主要数据结构* 在此算法中，每一种资源，都有一个对应的连接池，存放此类资源的连接。资源连接存放在一个向量表(Vector)中，向量表中每一个元素代表一个资源连接，资源连接由 PoolManager 管理，所有资源连接的申请均需通过应用服务器调用 PoolManager 获得。

可用资源连接分成两类：active 资源和 pooled 资源，active 资源表示正在使用的连接，而 pooled 资源表示处于连接池中的资源，为以后的连接复用使用。

其中，ResourceLimit 表示可用资源的上限；ActiveLimit 表示 active 资源的上限；MinSize 表示连接池中需要保持的可复用的资源连接的数量(即最小值)；CheckInterval 表示定期检查连接池的时间间隔，用毫秒表示；ReleaseFactor：从 0 到 1 之间的实数，表示每次检查时需要从池中释放的百分比；TimeoutException：超时异常，申请资源连接时等待时间过长，则抛出此异常。

3.2.2 *算法执行过程描述* 图 4 的各个执行步骤描述了算法中连接复用和连接池调度的具体过程：

应用程序向应用服务器请求一个资源连接(图 4 中标号为 1)，然后应用服务器调用 PoolManager (1.1)，接着由 PoolManager 调用 canActivate 方法(1.1.1)，此方法判断是否能从连接池中复用以前建立的连接，将结果返回给应用服务器(1.1.1.1)，如果结果为 Yes，则从 ResourcePool 中取得一个连接，传给应用程序(1.1.1.1.1)；如果池中没有已经建立的连接，或者 active 资源的使用数量已经达到 ActiveLimit 上限，则不能使用池中的连接，此时，应用程序会被放入一个 WaitingApplication 等待队列(1.1.1.1.2)，直到 TimeoutException 异常抛出或者分配到资源连接为止。

应用程序在资源使用完毕后，向应用服务器报告资源使用完毕(2)。应用服务器判断等待队列中是否有其他应用程序在等待此资源，如果有，则唤醒队列中的第一个程序，将资源连接分配给这个应用程序(2.1)；如果没有，调用 PoolManager(2.2)，然后 PoolManager 调用 release 方法释放该类资源连接(2.2.1)。

PoolManager 定期(即 CheckInterval 的时间长短)调用后台线程 releasedPooled(3)，检查连接池使用情况，池中的资源如果使用频率不高，则会逐渐从连接池中被移走。直到最终池中的资源到达最小值，即 MinSize 为止，时间长短由 ReleaseFactor 和 CheckInterval 共同确定。例如，ReleaseFactor 为 0.2，CheckInterval 为 10 秒，则所需时间为 10/0.2=50 秒。

连接池中资源连接的数量低于 MinSize 时，则由 PoolManager 生成新的资源连接(4)，以减少资源使用高峰期时的系统负担，提高系统性能。PoolManager 需要判断是否能生成新的资源连接，如果资源的使用数量已经到达上限 ResourceLimit，则不能生成新的连接。

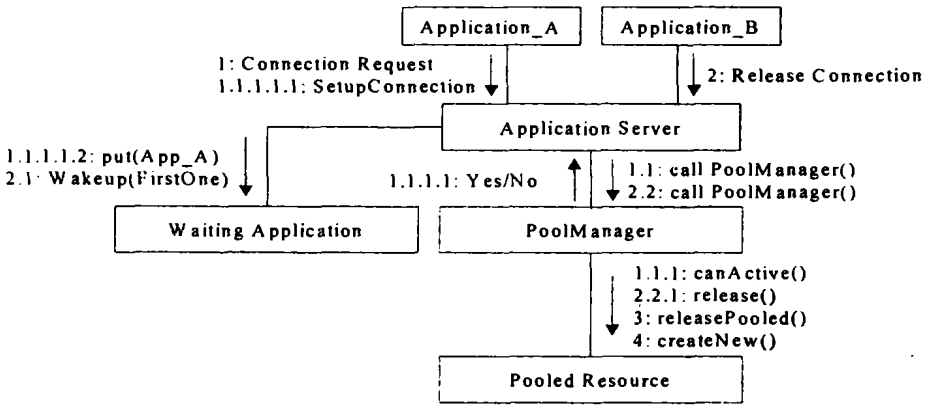

图 4 算法执行过程描述

结论 分布事务中的资源管理包括事务和资源的连接，资源管理器在事务中的注册和注销，连接池的管理以及调度算法等多方面内容。本文分析了 J2EE 框架下的资源注册和注销的过程，在此基础上设计和实现了一个连接复用和调度算法，并已经将其应用到分布事务管理器 ISTX 1.0 中，取得了很好的效果。

参考文献

1 Bernstein P A, Newcomer E. Principles of Transaction Processing. Morgan Kaufmann Publishers, Inc. 1997

2 Sun Microsystems. Java Transaction API (JTA) Specification Version 1.0.1, April 29, 1999, URL: http://java.sun.com/produces/jta.

3 Bergsten H. Improved Performance with a Connection Pool. URL: http://webdeveloperjournal.com/columns/connection_pool.html.

4 BEA Systems, Inc. WebLogic Server XA Resource Provider Requirements (XAResource Enlistment and Delistment). URL: http://edocs.bea.com/wls/docs61/jta/resource.html#1063434

5 IBM Inc. 用 JDBC 管理数据库连接. URL: http://www-900.ibm.com/developerWorks/education/java/j-jdbc/tutorial/j-jdbc-6-1.html

6 Oscar Sànchezi Vilar. Automatic Web Publishing. Trinity College Library, 2000