基于 Web 服务的分布式应用的事务处理

都艺兵 徐大伟 （山东财政学院计算机信息工程学院 250014）

摘要：在次或对象层次中实现的事务特性。

关键词：本文中围绕一个基于Web服务的分布式应用的事务处理，讨论在Web服务环境中，如何实现原先在数据库层Web服务 事务处理

1 引言

随着 Internet 的飞速发展，企业应用环境逐渐向 Internet 环境转移，传统的企业应用模式逐渐被分布式应用模式取代。

由于 Web 服务是通过定义好的接口通过标准的 Internet 协议交互，并由一种标准的功能描述语言来描述，采用面向服务的体系结构(SOA)，具有封装性好、耦合度低、互操作性好、可集成性能强等特性，是实现分布式应用的最佳解决方案。目前 SUN 公司的 J2EE 和微软公司的.NET 两大类型的 Web 应用服务器均支持 Web 服务。

由于在分布式应用系统中，无法实现数据库集中，如果采用数据库集中，骨干网上的流量是无法接受的。所以数据物理上是分布的，但逻辑上是集中的。因此就需要有跨结点的、分布式数据同步机制来保证数据的一致性。

那么，如何在基于 Web 服务的应用系统中实现原来在数据库层次或对象层次中实现的事务特性?

在本文中围绕一个用 Web 服务实现的企业分布式应用系统，讨论在 Web 服务环境中，针对不同的应用需求，如何进行事务处理，实现数据库同步。

2 应用背景

某企业集团由集团总公司和在各地的分公司及销售网点组成，在为其开发的分布式应用系统中，各个分公司的服务结点都有各自的局部数据库，在该系统中各分公司的底层数据层的设计基本一致。为集中管理，在公司总部设一数据中心，对公司所有的数据进行集中管理。各分公司服务结点的数据是数据中心结点数据的子集。

这样做的目的：

(1) 数据中心也是一种服务结点，一方面向管理层提供各种统计报表，提供决策支持，对基础数据进行维护和管理，另一方面对外界提供公共服务。采用数据中心，就无需在运行时通过 Internet 频繁地访问各服务结点进行数据请求，大大提高了系统的执行效率。

(2) 当某个服务结点崩溃，需要重建时，可以从数据中心下载所有其管理和使用的数据。

这时，我们就需要解决数据库数据同步的问题。

① 数据中心的数据更新或分公司服务结点的数据更新时，如何保证其他结点数据的一致性?

② 若在一个事务中涉及多个服务结点的数据更新，如何实现事务处理?

由于整个系统是在分布式环境下，各个服务结点的连网状况比较复杂，很多结点无法一直保持在线的状态。因此，在系统的 Internet 骨干上设置了用于交换事务数据的交换中心。在交换中心上，为每个服务结点提供了 in/out 消息队列。只要某个服务结点连上了 Internet，就可以通过交换中心收发包含事务的消息。

系统的体系结构如图 1。

下面重点介绍系统的核心技术之一：事务处理。

3 事务处理

一种情况是事务处理在单个数据库中完成，为了保证系统中其他结点数据的一致性，需要将单个事务重复广播。具体实现：可以将该事务描述传递给交换中心结点，插入到相关的结点的事务队列中，其他结点从交换中心的事务队列，取下事务描述，在本地执行事务，从而异步地实现各服务结点数据的一致性。

由于系统的实现是基于 Web 服务的，而 XML 语言是 Web 服务之间传递信息的基本格式。所以系统首要解决的是用 XML 格式来表示数据库事务。

由于该系统各分公司结点是同构系统，使用相同的数据库结构，所以简化了系统的设计与实现。

每条事务消息包含一个事务的描述，由若干个操作组成，或者更新记录、追加记录，或者删除记录。

我们定义了一个 transaction 元素来表示一个事务消息。由于一个事务包含一组对数据库表的操作，所以在 transaction 元素下包含一个或几个表示一个操作的元素，我们用 operations 来表示。每个操作可以是保存数据(包

括追加和修改),也可以是删除记录,我们分别用 savedata 子元素和 deletedata 子元素表示。在 savedata 元素下可以包三个子元素:TableName,PrimaryKey,Fields,deletedata 元素的结构类似。

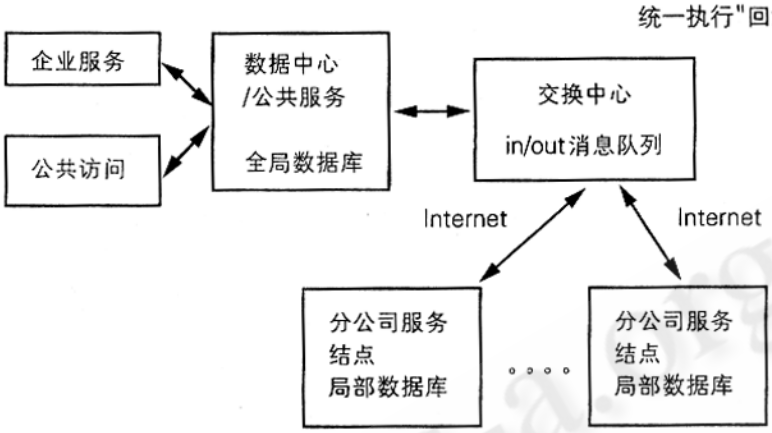

图 1　系统体系结构

有了 Schema 的模式定义文档,由于我们使用 Web 服务架构构建系统,在定义了事务消息的 XML 描述后,可以通过 SOAP 消息传输事务描述,在 SOAP 消息的 Body 中包含一组事务的描述。

为了简化系统设计,我们将每个事务消息用单个 SOAP 消息来传递。

具体的流程可以描述如下:

(1) 服务结点 A 完成了数据库事务 T,事务处理模块使用我们先前制订的 XML 模式将事务 T 表示为 XML 格式 T(XML)。

(2) 服务结点 A 的 SOAP Service 将 T(XML)包装在 SOAP Body 里面,组成 SOAP 消息,向交换中心发送,此时消息在交换中心中位于服务结点 A 的"out"消息队列中。

(3) 交换中心获得服务结点 A 的"out"消息队列中的事务消息,并分析出与之相关的需要实施事务同步的服务结点集合。例如服务结点 B(当然也可能是多个服务结点)。

(4) 交换中心将这则事务消息 T(XML)复制到服务结点 B 的"in"消息队列。

(5) 服务结点 B 上线之后,检查交换中心的消息队列,获得事务消息,在 SOAP Service 中将 T(XML)从 SOAP 消息中剥离并交给下层的事务处理模块。

(6) 服务结点 B 的事务处理模块将 T(XML)通过 XML 解析,转换成内部的事务,并实施运行。

前面的事务处理采用的仍然是完全基于数据库事务的数据同步方式,还不是真正的分布式事务。在分布式应用中,一个事务往往涉及多个服务结点,用前面的机制就无法解决数据一致性问题。对于这种情况,可以通过扩展两阶段提交协议来实现分布式事务处理。

两阶段提交:分布式系统中处理用户提交的事务时,事务管理器通常使用两阶段提交协议,保证所有操作所涉及到的分布式环境中的各个数据库中的相应数据被锁定,从而最后被正确地更新。如果因为某种原因操作不能完成,则要求统一执行"回滚"操作,回到事务开始前的状态。如果事务被成功执行,那么所有相关的数据库都被正确更新,此时这些更新应当被分布式环境中的所有数据库系统都了解。最后解除相关数据库相关数据的锁定状态,以便进行下一个事务的执行。

下面描述我们在系统中实现扩展的两阶段提交协议的过程。在这种机制中,我们需要在中央服务器上部署事务管理器,处于中央控制结点的位置。

① 事务管理器以事务发起者的身份开始工作,它首先在事务管理器日志中写入一条"Prepare Transaction"的记录,然后将相应的事务请求发送到所有相关的服务结点(需要执行事务的部分)上。

② 当每个参与事务的服务结点接收到"Prepare Transaction",服务结点可以按照自己的负载状况选择是"Accept"还是"Reject",并将"Accept"或"Reject"消息发回事务管理器。

③ 当事务管理器收到所有服务结点的响应后,如果全部都是"Accept"消息,那么继续下一步。如果至少有一个服务结点返回的是"Reject"消息,那么这个事务将不会真正开始,事务结束。

④ 当所有服务结点发回"Accept"消息后,事务管理器准备真正开始执行事务。它在事务管理器日志中写入一条"Begin Transaction"的记录,然后将相应的事务执行内容发送给所有相关的服务结点。

⑤ 各个服务结点将此事务写入自己的日志,然后锁定资源以供该事务使用,当资源准备完毕后,服务结点向事务管理器所在的计算机发送"Ready"消息。

⑥ 当事务管理器所在的中心控制结点接收到所有的"Ready"消息后,它在日志上记录"Commit Transaction",要求该事务被提交,并通知所有相关的服务结点。如果在一定的时间内,至少有一个服务结点没有返回"Ready"信息,那么它视该事务执行失败。将会在日志上记录"Rollback Transaction",并发送给所有服务结点,要求回滚事务,并将该事务结束。

⑦ 当这些相关的服务结点接收到"Commit Transaction"通知后,这些服务结点将自身承担的事务部分提交(Commit),并解除对资源的锁定,同时通知事务管理器执行完毕。

如果服务结点接收到"Rollback Transaction"通知

后,这些服务结点将自身承担的事务回滚(Rollback),并解除对资源的锁定。

在实现这种机制时,我们扩展了前一种方法中的事务模式定义,增加了两阶段提交所需要的控制元素,增加一个TwoPhaseCommit 元素,该元素包括以下子元素:serviceNode 元素、prepare 元素、begin 元素、rollback 元素和commit 元素。

· serviceNode 元素:描述服务结点的标识,该元素的值唯一标识了一个服务结点。

· prepare 元素:完成"Prepare Transaction"阶段的交互。事务管理器发出的"Prepare Transaction"消息是使用 prepare 元素及其子元素 request 来表示,同时如果发往服务结点 A,那么该消息中的 serviceNode 元素取值也应该是 A。而服务结点对"Prepare Transaction"消息的响应则是使用 prepare 元素及其子元素 response 来表示,具体的 response 元素的值是"accept"表示可以执行事务,如果是"reject"则表示不能执行事务。同样,服务结点 A 发回的响应消息,则消息中的 serviceNode 元素取值也应该是 A。

· begin 元素:完成"Begin Transaction"阶段的交互。事务管理器发出的"Begin Transaction"消息是使用 begin 元素及其子元素 request 来表示。事务管理器发出的描述事务内容的消息应当使用 begin 元素的子元素 trans_part 来表示,trans_part 中的内容与前面机制中描述的 transaction 元素是类似的。当服务结点接收到所有的事务消息后,即申请所有所需的资源,并使用 begin 元素及其子元素 response 来响应。如果资源申请成功,则 response 的值为"ready",否则为"failure"。

· rollback 元素:如果某个服务结点返回的 begin/response 的值为"failure",那么事务管理器就需要向每个服务结点发送 rollback 消息。

· commit 元素,如果每个服务结点返回的 begin/response 的值都是"ready",那么事务管理器就需要向每个服务结点发送 commit 消息。

4 小结

以上讨论了在 Web 服务环境中简单的事务广播和分布式事务扩展的两阶段提交模型的实现。

系统的缺陷及可进行的改进:

(1) 由于该系统中各服务结点基本上是同构系统,所以简化了系统的设计与实现。若各服务结点是异构系统和异构数据库,系统实现要复杂一些。

(2) 在文中没有讨论安全性事务控制。而在分布式基于 Internet 的环境中,安全控制是至关重要的。我们可以结合 SOAP 消息的安全扩展来实现安全事务控制。Web Services Security(WS-Security)是一个 SOAP Header Extension,为 Web 服务提供了一种保障服务安全性的语言。WS-Security 通过消息完整性、消息机密性和简单的消息认证来实现消息安全的目的。这些机制能够被用来适应现有的大量的安全模型以及加密技术。可以尝试使用这种技术来为事务控制提供安全性。

(3) 没有考虑对长事务的支持。比如在执行长事务时,该事务模型降低系统的并发度和性能,增大事务间冲突概率,导致事务等待时间延长,并且会明显增加事务死锁。在研究领域,已经提出很多改进的扩展事务模型,如嵌套事务模型、多层事务模型、Sagas、分支汇合事务模型、柔性事务模型等。

随着基于 Web 服务事务处理标准的发展和完善,系统的分布式事务处理也会不断地完善。

参考文献

1 (美)Jim Gray Andreas Reuter ,事务处理概念与技术,机械工业出版社, 2004.1。

2 孙瑛霖, 事务服务浅析 http://www-900.ibm.com/developerWorks/cn/java/l - transation/part1/index.shtml,2002-6。

3 范国闻等,Web 应用服务器研究综述, 软件学报,2003,