基于StarBus的对象事务服务的设计与实现*

徐海渊 吴泉源

(国防科技大学计算机学院 长沙 410073)

摘 要 在分布式计算环境中, 对象事务服务(Object Transaction Service简称OTS) 对于提高分布式系统的开发效率具有重要意义。设计与实现一个高效的对象事务服务的关键在于克服网络阻塞的问题, 本文着眼于一种基于面向对象传值语义的设计方法, 可以有效地减少网络阻塞, 适合于企业级应用。

关键词 事务, 原子性, 隐含上下文, 智能代理, 回滚, 提交

分类号 TP393

Design and Implementation of the StarBus-Based Object Transaction Service

Xu Haiyuan Wu Quanyuan

(Institute of Computer, NUDT, Changsha, 410073)

Abstract In the distributed computing environment, the Object Transaction Service (OTS) is useful to be really effective for devoloping the distributed system. The implementation design of the OTS is critical for the service to reduce network traffic jam. This paper focuses on a new implementation design on top of the object-oriented pass-by-value semantics which can deal with the problem so as to reduce network traffic jam and be applied to enterprise applications.

Key words transaction, atomicity, implicit-context, smart-proxy, rollback, commit

事务是一个编程范型, 用来简化构造时可靠实用, 尤其是适用于并发访问共享数据。起初事务从数据库应用中发展而来, 目的是使得每个活跃进程都感觉在单独使用共享数据, 并且即使出现异常也能保证数据的一致性。后来, 事务概念被扩展到分布式计算当中形成分布式事务。今天, 普遍认为事务是构造可靠分布式应用的关键。随着面向对象技术的日益成熟和普及, 事务与对象结合形成对象事务概念。对象事务服务能有效地支持控制分布式事务的范围和实现, 允许单一原子事务包含多个对象, 允许对象改变与事务关联的内部状态和协调事务完成等功能, 在商业及其它领域具有广泛的应用。StarBus是我院“863计划”项目——“分布式Client/Server计算机系统集成平台与应用系统”的最新研究成果, 是采用分布对象技术, 遵循CORBA2.0标准的系统集成中间件, 成功地提供了分布式对象环境中的StarBus软总线支持, 并在此基础上提供了名字服务、安全服务、查询服务、持久服务、时间服务、事件服务等通用服务, 能够有效控制分布式软件开发的复杂性, 减少分布式应用系统开发的工作量。为进一步提高分布式应用系统开发的效率, 有必要在StarBus中加入对象事务服务的实现。对象事务服务的企业级实现的困难在于: 如何减少网络阻塞以提高系统性能。

本文在已有“863计划”课题成果StarBus的基础上借鉴IONA公司的对象事务服务的部分实现思想, 提出了基于面向对象传值语义的解决方案和设计方法, 有效地解决了网络瓶颈问题, 扩展了应用的前景。

1 基本概念

1.1 事务

事务是一个由涉及一个或多个对象的一系列操作组成的, 进行一致性与可靠性计算的原子单元。事

* 国家863计划课题资助
1998年6月25日修订
第一作者: 徐海渊, 男, 1971年生, 硕士

务具有原子性、一致性、隔离性和持久性的特征。原子性是指：要么完全成功，要么就像没有发生。一致性是指：事务涉及的所有对象保持状态一致。隔离性是指：在事务完成之前其中间状态不为其它事务所见。持久性是指：在事务完成后其影响永久存在。

1.2 对象事务服务

对象事务服务涉及以下几个概念。事务客户指发出事务定界范围内的方法调用的任意程序；事务对象指行为受到事务范围内调用影响的对象，典型地包含或直接地指能被请求修改的永久数据；恢复对象指数据受到事务范围内调用影响的对象，恢复对象包含于事务对象，通过登记一个资源对象到事务服务中来参加事务服务协议；事务服务器指一组行为受事务影响但没有自身的恢复状态和恢复资源的对象集合，不参加事务的完成，但能强迫事务回滚；恢复服务器指数据受提交或回滚事务影响的对象集合。

1.3 事务服务模式

事务服务模式主要包括平面事务模式、嵌套事务模型和开放嵌套事务模式。平面事务模式是指：事务的所有参与者均处在同一个平面上，其中任何的异常都会导致整个事务回滚。嵌套事务模式是指：一事务可以包含任意数目的子事务。这种递归定义最终形成了事务树并导致以下相关：①开始相关：子事务必须后于父事务开始而先于父事务完成；②提交相关：下层事务提交成为上层事务提交的条件，一个嵌套事务能最后提交当且仅当顶层事务提交；③离开相关：父事务离开则子事务离开而不管子事务是否提交，子事务离开父事务却不一定离开但必须进行策略选择（忽略子事务的失败，或重新执行子事务，或试图执行一个别的子事务，或如果子事务的失败致命的，则父事务离开），它提供了调整回滚范围的有效机制。开放嵌套事务模式是对嵌套事务模式的扩充并能做到对后者的兼容。

2 对象事务服务的接口定义和实现模型

2.1 基本接口定义

(1) 当前接口：为使用对象事务服务而定义的伪对象接口。客户调用开始和提交操作来开始和结束事务。StarBus 软总线透明地将伪对象的上下文加入请求当中，并传给所有的参与者。上下文含有唯一识别事务的标识符和状态信息。客户调用回滚操作离开事务，调用挂起操作来挂起事务以阻止事务上下文随消息传递，当需恢复时调用恢复操作。获取控制操作返回控制对象用于直接操作事务服务。顶层事务可调用定时操作来规定子事务执行的时限。

(2) 控制接口：允许显示地管理或传播事务上下文。获取终止器操作时返回一个终止器对象，用于结束事务。获取协调器操作返回一个协调器对象，用于资源参与事务。

(3) 终止器接口：提供提交和回滚操作来结束事务。

(4) 协调器接口：由事务服务实现。恢复对象用它来协调事务的参与者。恢复服务器调用登记资源操作参与事务（对于嵌套事务模式，可调用登记子事务资源操作参与子事务）。创建子事务操作用于生成嵌套事务模式下的子事务。直接回滚操作用于离开整个事务。哈希操作返回当前事务的句柄。在调用登记资源操作前恢复服务器用哈希值判断当前资源是否已登记。

(5) 资源接口：由参加两阶段提交协议的一个恢复对象实现。对象事务服务用两阶段提交协议来协调事务的提交或回滚，集中决定提交，但给各参与者投票的权力。在第一阶段，调用所有参与事务的资源对象的准备操作，每一个恢复对象返回一个投票结果（同意提交、同意回滚或同意只读）。根据投票结果，事务服务决定提交或回滚。如果协调器只有一个登记资源，则调用单阶段提交操作来声明进行单阶段提交。

(6) 子事务资源接口：用于嵌套事务模式下的恢复实现。

(7) 事务对象接口：用于标志事务对象。

2.2 对象事务服务的实现模型

图 1 模型中两阶段提交协议的实现过程，首先是所有的顶层事务向其登记的资源对象发准备操作。这些资源中有普通的资源对象，也有资源代理。对于后者，操作则被传给远程的智能代理。智能代理

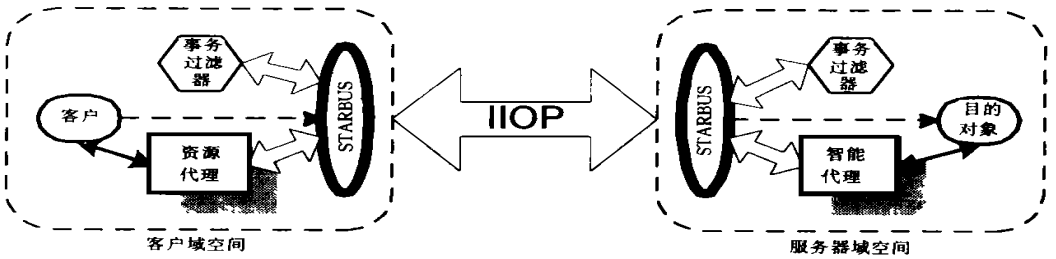

图1 对象事务服务实现模型

Fig. 1 Implementation model of OTS

将操作发给所属资源。同样，这些资源中有可能有新的资源代理，如此直到所有资源都收到准备的请求。然后，智能代理收集所属资源的应答。如果所有资源同意提交，则智能代理投票同意提交，否则智能代理通知所属资源回滚并向上级协调器发回滚应答。最后，如果顶层协调器所属资源均投票同意提交，则顶层协调器同意提交，整个事务完成，否则整个事务回滚。其中事务过滤器用于实现在调用操作过程中的事务传送，提供责任链模式，是程序员定制StarBus软总线行为的系统挂钩。事务过滤器负责使层次事务结构平面化为数据流，并将这些信息附加在远程调用中。在远程调用被传送之前，StarBus软总线调用过滤器的输出请求接口，决定本次调用是否在事务范围之内。如果是，则过滤器将对象参数链传给所有的在该事务层次之上的控制对象。当调用到达目的端时，StarBus软总线调用目的端过滤器的接收请求接口来恢复事务的层次结构并创建所有传送控制对象的事务智能代理，同时事务上下文也被传给目的对象。目的对象完成操作后，StarBus软总线调用过滤器的输出应答接口，由过滤器决定是否在应答中附加信息。StarBus软总线将应答传回客户端，并调用客户端过滤器的接收应答接口来检查消息中是否有附加信息。最后StarBus软总线将应答返回客户。

事务智能代理通过继承控制类使得StarBus软总线能识别智能代理，通过继承实现类在局部空间内重新实现了被传送对象的准确行为。因此，智能代理是在局部范围内解析操作的事务实现。当使用这种插入技术时，智能代理成为局部空间的协调器，因此，它必须向上级协调器登记拥有的资源。这样智能代理又成为上级协调器的资源对象，并需要支持对象事务服务中的资源接口。

3 对象事务服务的设计方法

3.1 设计目标

能支持平面事务模式和嵌套事务模式；支持遗留应用封装；支持模式互操作性；支持网络互操作性；支持灵活的事务传播协议；支持TP监视器。

3.2 面向对象的传值语义

对于非对象类型，当调用具有传值语义时，意味着在调用中参数的实际值被传递。对于对象类型，调用具有传值语义则意味着：实际参数值是对象引用而不是对象本身。在对象管理组(OMG)规定的接口定义语言(IDL)到C++的映射中，远程对象接口的引用对应一个C++智能指针。这个智能指针的行为类似指针，并在用于访问远程对象时能执行一些动作的对象。因而在IDL到C++的映射中，一个远程对象接口的引用看上去就像一个局部C++对象指针，并且任何与这个局部C++指针的互操作都导致一个对远程对象的调用。尽管这种方式对多数环境是适合的，但在一些有更高要求的场合，如高延迟、低带宽的传输网(如国际互连网)上，需要减少网络阻塞来提高系统性能。网络传输的这些限制引出了另一种不同于远程过程调用的范型——远程程序。远程程序通过将客户/服务器远程通信变为局部互操作的方法解决了远程调用带来的性能问题。这样不再是两个远程对象跨越网络进行通信，而是其中一个迁移到另一个对象所在空间中，然后开始局部通信。对编译型语言，可采用对象拷贝(而不是类迁移)的模型。通过拷贝对象实例可以建立松散远程程序的形式，这一技术隐含接收后已具

备所拷贝对象的实现类, 或在系统支持动态链接库时存在动态链接的事实。所拷贝的是具体对象实例的内部状态而不是实现类。这种方法的优点在于使对象间的互操作局部化而减少远程通信。

3.3 StarBus 软总线服务

StarBus 软总线服务的范围从底层技术 (如引用解析、消息处理等) 直到支持高级服务 (如事务、安全等)。实际上有些 StarBus 服务需要在远程调用时上下文隐含传递, 这需要 StarBus 软总线支持在请求与答复的过程中上下文的透明加入。识别和传递隐含上下文的前提是, 隐含上下文被说明为 IDL 的数据类型, 且 StarBus 软总线提供回调机制来允许服务在请求与答复过程中提供和接收隐含上下文。本实现方案定义了一个伪对象接口来实现 StarBus 软总线与事务服务之间的互操作。根据这种技术, StarBus 软总线在请求过程中充当客户的角色调用事务服务, 以实现上下文的隐含处理。定义的接口模型包含发送、接收两个接口。发送接口提供发送请求和接收应答操作, 由 StarBus 软总线在客户端发出请求之前和接收答复之后调用。接收接口提供接收请求和发送应答操作, 由 StarBus 软总线在服务方接收请求之后和发出答复之前调用。要实现基于 StarBus 软总线的传值语义, 需要由传值服务支持发送回滚操作和接收回滚操作。传值服务通过调用识别接口的识别发送和识别接收操作, 表明希望得到 StarBus 软总线的支持。StarBus 软总线回调机制流程如下:

在客户方

(1) 发送远程调用前 StarBus 软总线调传值服务的发送请求操作并提供对象引用清单。

(2) 传值服务决定哪些对象引用代表要拷贝对象并进行登记, 然后调相应的获取对象状态操作。

(3) 要拷贝对象响应请求返回内部状态信息 (也可能一个机同调用中包含多个要拷贝对象)。

(4) 传值服务依靠相关操作判断上述过程的完成, 最后传值服务建立对象信息数据结构并将其加入到隐含上下文, 然后 StarBus 软总线将其传给服务方。

在服务方

(1) 分发请求之前, StarBus 软总线调用服务方传值服务的接收请求操作将隐含上下文中对象信息数据结构提供给传值服务。

(2) 传值服务根据对象信息建立远程对象与局部拷贝之间的关联。

(3) 在生成局部拷贝之后, 传值服务必须改变对象引用使之指向局部拷贝。

(4) 传值服务调用局部拷贝对象的设置对象状态操作来初始化对象的内部状态。然后, StarBus 软总线获得控制并将请求分发给目的对象。

4 结束语

按照以上基于 StarBus 对象事务服务的设计思想, 我们已经完成了第一阶段的设计与实现任务, 基本实现了 StarBus 相关的技术支持, 相信最终正式版本的问世必将对 StarBus 的应用前景产生深远影响。在此感谢王怀民、周立两位博士在本文写作中给予了很多宝贵意见。

参考文献

1 E Grasso. Passing Objects by Value in CORBA. OMG discussion paper. doc. n. 96-07-03, July 1996

2 J. Gray, A. Reuter. Transaction Processing: Concepts and Techniques Morgan Kaufmann Publishers. 1993

3 朱海滨, 蔡开裕, 樊爱华, 宋辉. 分布式系统原理与设计. 长沙: 国防科技大学出版社.

4 R Orfali, D Harkey, J Edwards. The Essential Distribute Objects Survival Guide. John Wiley & Son. 1996

5 E Gamma, R Helm, R Johnson, J Wlissides. Design Patterns. Addison Wesley. 1994

6 E Grasso. An Extended Transaction Service for Telecom Object Request Brokers. in Proc. 2nd International Workshop on Distributed Object Oriented Computing, Frankfurt, Germany, 1996

7 F Moss. Nested Transactions. MIT Press. 1985