StarBus－OTS 系统的设计与实现

韩伟红　黄子中　贾　焰　王志英　（湖南长沙 国防科技大学计算机系　410073）

摘要：基于分布式对象技术的 OTS(对象事务管理)服务有效地保证了分布式事务的原子性和永久性，目前已经成为分布式事务的一种有效的解决方案。本文以基于分布式软件构作平台 StarBus 实现的 OTS 服务为基础，详细阐述了 OTS 服务的系统结构和实现的关键技术。

关键词：对象事务服务 OTS　两阶段提交 2PC　事务管理器 TM　资源管理器 RM

一、引言

事务处理是计算机传统的两大应用领域之一，随着数据库技术的不断成熟和发展，事务处理技术的应用范围也越来越广泛。进入 80 年代，随着网络技术的发展和网络基础设施的不断完善，在分布式计算环境下的分布事务应用已经成为事务发展的趋势。进入 90 年代，以 Internet/Intranet/Extranet 为代表的计算机技术的飞速发展，事务处理技术又面临新的挑战：系统规模的无限扩大，至少要支持上万个在线 Client 用户，以及系统的服务对象将超越 1 百万个；更高的吞吐率的要求，1996 年的吞吐率要求为 57(百万/分钟)，2000 年吞吐率的要求将为 100(百万/分钟)；更短的应用开发周期，充分利用已有的，新功能模块能够象构件一样动态的加载等等。

始于 90 年代初的分布式对象技术，已经发展成为当今分布异构环境下建立应用系统集成框架和标准构件的核心技术，出现了以 OMG 的 CORBA 和 Microsoft 的 COM/DCOM 为代表的应用互操作技术标准。CORBA 作为一种被广泛承认的技术标准在跨平台性、安全性、可扩展性、技术的完整性和连续性等方面具有技术优势，是一种技术领先的标准。

StarBus 是我们开发的系统集成中间件产品，它采用分布对象技术，遵循 CORBA 2.0 标准。利用 StarBus 开发分布式应用能够有效地控制分布式软件开发的复杂性，无需考虑如何管理和访问异地对象，集中精力设计实现应用对象，从而大大减少分布式应用开发的工作量。同时如将 StarBus 作为标准“软总线”，则易于设计可重用性、移植性和互操作性好的对象构件，从而极大地提高了分布式应用系统的效率。

分布式对象技术为解决事务处理所面临的问题提供了良好的解决方案，分布式对象技术与事务处理技术相结合的对象事务管理(Object Transaction Service: OTS)技术因此诞生。该技术不仅将分布式对象技术的良好的程序设计模型和程序设计方法，以及高可扩展性、高可管理性和方便的遗留系统的集成方法引入了事务处理之中，而且拓宽了分布式对象技术的应用范围，为分布式对象技术提供了一个大有作为的应用领域。本文主要介绍基于 StarBus 开发的对象事务服务系统 StarBus－OTS 的设计与实现方法。

二、系统组成

1. OTS 应用的结构

OTS 应用系统主要由三部分组成：Transaction Originator(TO)，Recoverable Server(RS)，Object Transaction Service(OTS)，如图 1 所示。TO 是事物的发起者，在事物处理的过程中可能会使用 RS，所以也把 TO 称为 Transaction Client；RS 是由一组 Recoverable Object 组成，Recoverable Object 可以由 TO 调用；OTS 提供保证事物

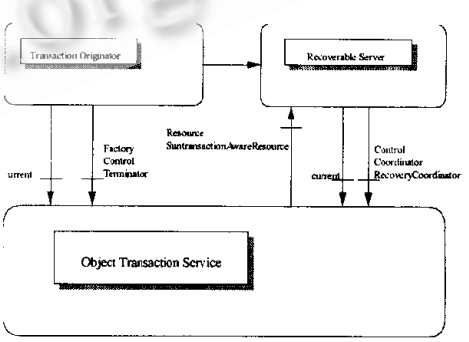

图 1　OTS 应用系统结构图

原子性和持久性的方法。其中 OTS 是系统组成部分，而 RS 和 TO 是应用组成部分。图中 OTS 提供的接口的基本功能如下：Current 提供操作允许 OTS 用户显式地管理事务与线程之间关系的接口；TransactionFactory 提供操作允许 Transaction Originator 开始一个事务；Control 可以实现显式地管理和传播事务上下文；Terminator 支持对事务的结束和回滚操作；Coordinator 为事务的参与者 Recoverable Objects 提供了相应的操作；RecoveryCoordinator 支持在特定情况下的恢复过程；Resource 提供了 OTS 对 resource 所要求的操作；Synchronization 对不支持 XA 协议的对象的同步操作；SubtransactionAwareResource 支持嵌套事务中针对子事务资源的操作。

2. OTS 系统结构

OTS 主要由 Transaction Factory，Control，Terminator，Coordinator，RecoveryCoordinator 等几部分组成，图 2 是 OTS 的系统组成图。从图 2 可以看出，OTS 中的 Transaction Factory，Control，Terminator 等三个对象主要与 Transaction Client 相关，而 Coordinator 和 RecoveryCoordinator 则主要与 Recoverable Server 相关。

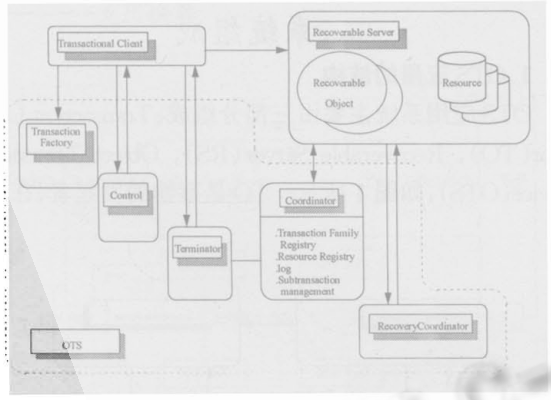

图 2 OTS 系统组成图

三、系统逻辑流程与功能

OTS 主要由 TransactionFactory，Control，Terminator，Coordinator，RecoveryCoordinator 等几部分组成，下面我们用一个应用的例子说明这些组成部分的功能及系统的逻辑流程。图 3 是 OTS 系统的组成结构及 OTS 完成事务处理的过程图。基于 OTS 服务的事务处理过程如下：

1. Transaction Originator 通过向 TransactionFactory 发出 create 请求开始一个事务，返回一个 Control 对象唯一标识事务。

2. 通过 Control 对象，Transaction Originator 可以拿到这个事务的 Coordinator，请求相应的事务服务。

3. Transaction Originator 激活 Recoverable Object，并把 Coordinator 作为参数传递给 Recoverable Object。

4. Recoverable object 在第一次使用资源时向 Coordinator 注册，同时将 RecoveryCoordinator 对象返回给 Recoverable Object。

5. Transaction Originator 通过调用 Control 对象的 get-terminator 方法得到 Terminator 对象。

6. Transaction Originator 通过调用 Terminator 对象中的方法结束事务，Terminator 与 Coordinator 协调完成 2PC 协议以保证事务的原子性和持久性。

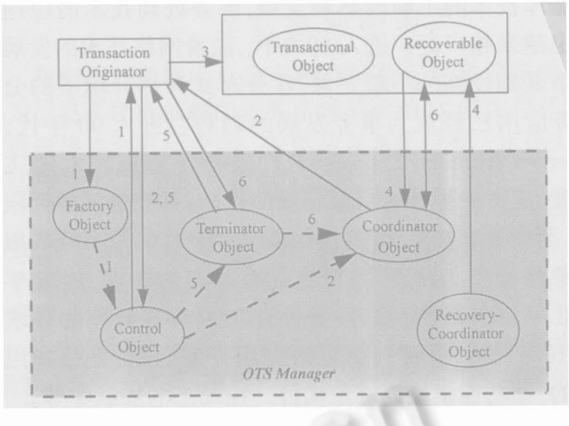

图 3 OTS 系统的组成及功能

四、关键技术

OTS 系统实现过程中要解决的几个关键问题是事务的初始化、事务上下文的传播以及终止事务时如何完成 2PC 协议，下面分别讨论这几个关键问题的解决方法。

1. 事务初始化

基于 OTS 产生一个新事务，首先 Transaction Client 用 OTS 中 TransactionFactory 对象提供的方法 create 初始化一个新事务。TransactionFactory 对象接到这个请求以后产生一个 Control 对象并把它返回给 Transaction Client，通过 Control 对象可以得到两个重要的实体：Coordinator 和 Terminator，通过 Control 对象中的接口 get-coordinator 和 get-terminator 获得。

2. 事务传播

事务执行过程中可以激活多个 Recoverable Object，

所有执行 RO 都和这个事务共享同一个事务上下文，因此需要进行事务上下文的传播。我们采用滚动式发展策略，首先实现一个没有自动传播事务功能的 OTS，以后再逐步加入这些功能。因此在 Recoverable Object 的每一个事务型的方法调用接口中都把 Coordinator 作为最后一个参数。Transaction Client 通过 Control 获得 Coordinator，之后把它作为参数传给 Recoverable Object，实现了事务的传播。Recoverable Object 可以向 Coordinator 注册资源。

3. 事务终止时 2PC 的实现

当 Transaction Client 决定提交或退出一个事务时，事务就终止了。TC 首先通过 Control 对象的 get-terminator 接口得到 Terminator 对象的引用，之后可以用 Terminator 对象的 commit 操作提交事务，或用 rollback 操作回滚事务。当 TC 要求提交事务时，OTS 是利用 2PC 协议来保证事务的原子性和持久性。由于事务执行过程中涉及的所有资源都已注册到 Coordinator 中，所以 2PC 协议由 Coordinator 实现。2PC 协议的执行过程如图 4 所示，说明如下：

(1)Coordinator 激活一个事务执行过程中用到的所有资源的 prepare 操作。

(2)当资源的 prepare 操作被激活以后，资源就检查自己的状态看是否能够提交事务，并向 Coordinator 投票。

(3)Coordinator 搜集所有资源的投票，如果有一个资源投 VoteRollback 票，则 Coordinator 决定 rollback 事务，并激活所有资源的 rollback 操作；否则 Coordinator 决定提交事务并激活所有资源的 commit 操作。如果资源投的是 VoteReadOnly 票，则无论 Coordinator 作什么决定，都不再向那个资源发出请求。

(4)资源投 VoteCommit 票后就等待 Coordinator 作出决定 commit 或者 rollback 事务，但如果资源投的是 VoteRollback 票，则它可以自行 rollback 事务而不必等待 Coordinator 的决定。

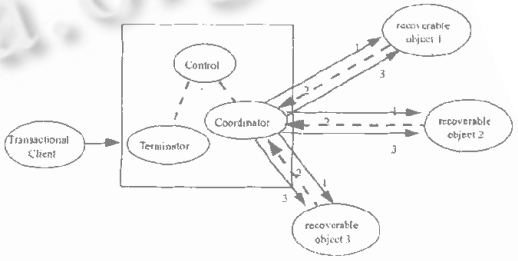

图 4　2PC 协议的执行过程

五、小结

从应用的角度来看，OTS 服务保证了在分布式环境下的分布事务的原子性和永久性；从分布式系统的角度来看，OTS 服务面向分布式环境，他具有开放性和可扩展性等优点。所以 OTS 服务是一种技术先进而且应用前景良好的服务构件，可以广泛地应用于多个领域。

参考文献

[1] David Bell and Jane Grimson,《Distributed Database System》, Addison－wesley, 1992。

[2] Won Kim,《Modern Database System》, Addison－Wesley Publishing Company, 1995。

[3] Object Management Group. 1991. The Common Object Request Broker : Architecture and Specificaton. OMG Docment no. 91.12.1.

[4] 李昭原等,《数据库技术新进展》,清华大学出版社, 1997。

(来稿时间:1999 年 5 月)