CORBA 中对象事务服务研究与实现*

杨 涛** 郑晓霞 刘锦德

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】 介绍了事务概念及其原子性、一致性、隔离性、持久性，阐述了保证事务 ACID 属性的两阶段提交算法；对 CORBA 中对象事务服务框架进行了研究，通过对规范中各个对象及其接口的分析，论述了 OTS 面向对象的分布式事务处理机制；最后给出了服务中事务工厂、控制器、协调者、资源等对象基于可移植对象适配器的实现。

关 键 词 事务； 两阶段提交； 对象； 对象事务服务； 对象引用

中图分类号 TP311

CORBA 作为当前分布式计算的重要规范，不仅解决了面向对象的异构应用之间的互操作问题，还提供了分布式应用所需的多项服务[1]。在 CORBA 的公共对象服务中，对象事务服务(Object Transaction Service，以下简称 OTS)将事务概念引入到分布式对象计算中。事务具有原子性、一致性、隔离性、持久性，对构建高可靠性应用，特别是要求并发访问共享数据的分布式应用起着关键作用，事务处理技术已经在银行、证券、电信等行业得到了广泛地应用。随着应用的普及和深化，传统的事务处理技术面临新的挑战，如系统规模无限扩大对性能要求更高，业务逻辑的快速变化希望开发周期更短。对象事务服务结合了分布式对象技术和事务管理技术，一方面具有事务管理器的高可靠性，同时具有 CORBA 中互操作性、可扩展性、可维护性等优点，提供了良好的分布式事务处理解决方案。

1 事 务

事务是可能包含多个计算任务的具有 ACID 属性的操作集合。事务是原子操作，或者整个执行，或者整个不执行，决不会部分地执行；事务具有一致性，它把数据从一个一致状态转换到另一个一致状态；事务是相互隔离的，一个事务不能看见另一个事务的工作过程；事务的结果是持久的，即使系统崩溃也能保持。

分布式应用可能会在一个事务中更改网络中多个计算机节点的数据；任一节点或节点间通信的失效都可能导致分布式事务失败。为了保证事务的完整性，分布式事务一般采用两阶段提交协议来完成事务。两阶段提交是一个分布式算法，它假设每个节点都有持久存储数据的能力，没有一个节点会永远崩溃，最终任意两个节点可以互相通信，在这些假设基础上保证所有的工作要么全做，要么全不做。两阶段提交中，一个主节点被指派为事务协调者，其他节点称为事务参与者。应用程序完成事务中所有操作后向协调者发出提交事务的请求，协调者驱动各参与者完成事务。整个提交过程分为两个阶段：1) 准备阶段，即协调者发送消息询问所有事务参与者，是否决定提交各自所做的工作，如果参与者可以提交自己的工作，就写入一条撤消日志和重做日志，并通知协调者可以提交事务，此时它不能够再单方面终止事务，否则就破坏了两阶段提交语义；若不能提交，就返回事务失败信息。事务的最终结果取决于事务中每个参与者的投票；2) 提交阶段，如果所有参与者都同意提交事务，协调者写入一条提交记录到日志中，通知参与者提交事务，各参与者释放事务中保持的锁和资源，如果任一参与者决定终止事务，那么协调者通知各参与者利用撤消日志回滚事务。如果在两阶段提交过程协调者或任一参与者崩溃，那么只要该部分重新启动，

2001 年 8 月 24 日收稿

* 信息产业部预研基金资助项目

** 男 24 岁 硕士生

系统都能够利用日志从失败中恢复，保证事务完成。两阶段提交存在一些问题，如它是阻塞的；不能处理两个以上站点同时出错；偏向于撤消而非提交事务等。尽管如此，它在分布式事务处理中得到了普遍应用，早期得到业界广泛支持的 X/OPEN DTP 模型和目前的 CORBA 对象事务服务都采用它来进行分布式事务提交。

2　对象事务服务框架

对象管理组在1999年定义了对象事务服务1.1版规范，规范结合事务概念和对象模型阐述了面向对象的分布式事务处理，其框架如图1所示[2]。

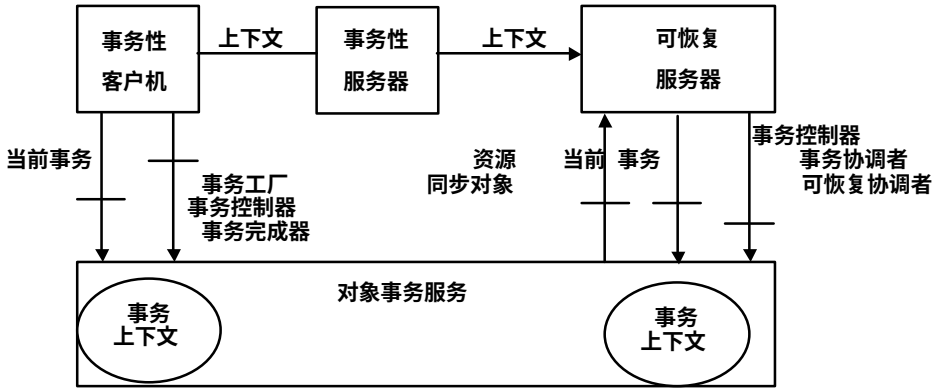

图1　OTS结构框图

OTS定义了多个CORBA对象(以下“ 对象 ”均指CORBA对象)接口，每个对象实现一定的功能，相互协作共同完成事务处理。从图1可以看到，OTS没有定义事务接口，应用程序使用事务完成器和事务协调者接口来实现对事务的操作。事务完成器包括结束事务的commit和rollback方法,分别用于提交和撤消事务。事务协调者接口提供了事务的大部分操作，包括向事务注册事务资源、同步对象，查询事务状态，判断事务间关系等。将逻辑上的事务接口划分为事务完成器和事务协调者，是为了向编程实体提供粒度更细的控制。比如说，可能不允许事务中涉及到的每个线程都有权提交一个事务；当接口细分后，OTS就可以限制其它参与者访问事务完成器对象的权限，只允许事务发起者提交事务。

对事务的事务完成器和事务协调者的访问是通过事务控制器接口的get_terminator 和get_coordinator实现的。事务控制器本身不对事务进行任何操作，它只返回该事务的事务完成器和事务协调者 对象的引用，可视为这两个对象的简单封装。一个事务的事务控制器对象是当系统发起新事务时产生的。事务工厂创建新事务会返回事务的事务控制器对象引用。应用程序只要获得了事务的事务控制器，就可以访问对应的事务完成器和事务协调者对象，从而实现事务的任何操作。

事务作为一个导致系统从一个有效状态转变到另一个有效状态的逻辑操作,其结果最终将对系统的状态产生持久的影响。事务性对象描述了事务的这种特性，它指其操作能够在事务中改变系统持久数据的对象；它可以是本身就包含这些数据，也可以通过其它对象间接引用持久数据。事务性对象是一个空接口，它只是向OTS表明这个对象的操作将会受到事务的影响，一般应用程序需要从此接口派生新接口来实现最终应用的业务逻辑。提供事务性对象实现的服务器称为事务性服务器。调用事务性对象操作的应用程序称为事务性客户。特别地，如果这个客户创建了新事务，则称为事务发起者。

一种特殊的事务性对象称为可恢复对象。它参与了事务的提交，包含了代表系统状态的数据，随着事务对系统状态的改变这些数据会在事务提交时被更新。它拥有持久存储能力，能够根据日

志从失败中恢复。事务性对象可以在操作中引用可恢复对象来保存和更改持久数据，因此事务性对象不一定是可恢复对象。可恢复对象在事务提交或回滚时其状态会发生持久性改变，因此它要以某种方式参与事务的完成；这是通过向某特定事务注册资源对象来实现的。资源定义了 prepare、commit、rollback 等操作，OTS 可通过这些接口完成两阶段提交。另外，为了处理一些特殊情况，比如很多应用程序通过数据库来管理数据，因此有些可恢复对象只保留一些瞬态数据，依靠数据库来保存持久数据，这样可恢复对象需要在准备提交之前和事务完成后调用数据库的一些接口。为了让可恢复对象知道事务提交的开始和结束，通知这些对象，因此 OTS 规范定义了一个“同步”接口，可恢复对象将同步对象注册后，当 OTS 准备提交事务之前，将会调用它的 before_completion 方法，而在事务完成后，将调用 after_completion 方法。这样就给了事务参与者一个调用数据库接口的机会。

3 服务实现

OTS 作为 CORBA 中提供分布式事务处理的服务，其实现遵从 CORBA 公共服务的一般原则，如接口和实现分离，对象引用按接口分类，客户端只依赖与接口，与实现无关。另外本文还假设底层有良好的支持可移植对象适配器的对象请求代理[3]。根据规范，OTS 实现应当能够对事务进行管理，包括创建、提交、撤消等，在系统失败的时候能够保证事务完整性，允许应用程序参与事务，为开发人员提供面向对象的编程接口。

3.1 事务工厂

在 OTS 中，事务工厂负责创建新事务，返回事务的事务控制器对象引用，应用程序可以根据事务控制器完成事务上的一切操作，实现对事务的完全控制。它是应用程序访问 OTS 的入口点，因此 OTS 应该以某种形式公布事务工厂的对象引用，如用 resolve_initial_reference，或使用名字服务，交易服务等。

事务工厂在创建事务时要为新事务分配一个事务ID，事务ID结构otid_t包括三个字段[2]：

```
long formatID;
long bqual_length;
sequence <octet> tid;
```

其中 tid 是一个八元组的序列，其内容没有限制，但为了唯一标识一个事务，它必须是全局唯一的。使其唯一的一个生成方法是采用 OSI CCR 中的命名方法。字段 formatID 表示 tid 的生成方法。如果采用了 OSI CCR 命名方法，formatID 应该设为零，否则 formatID 应该大于零。如果 formatID 小于零，则表示该 ID 为空。字段 bqual_length 表示 tid 中事务分支标识的长度[4]。

3.2 事务控制

OTS 中对每个事务的提交、回滚等操作是通过该事务的事务控制器，事务完成器，事务协调者对象实现的 如果为每一个事务都创建相应 CORBA 对象的伺服程序，那么每个事务都会在 OTS 中存在三个对象实例，每个实例都会包括事务标识符，状态等事务相关信息，当系统中事务数量较多时，无疑将会占用大量的系统资源[5]。为了提高效率，我们使用可移植对象适配器中默认伺服器来实现事务控制器，事务完成器和事务协调器.

事务工厂创建事务时，用一个 transaction 类封装事务的有关信息，但并不在服务器端真正创建相应的事务控制器对象的伺服程序，它仅仅创建一个对象引用。POA 可以根据对象标识符直接创建对象引用，不用实例化 CORBA 对象。由于事务标识符在系统内唯一标识了一个事务，因此我们可以将事务 ID 转换为对象标识符，并以此为参数创建相应的事务控制器对象引用。因此事务工厂创建事务流程如下：

1) 产生一个事务标识符

2) 新建 transaction 类的实例，其中包括了事务标识符，状态等信息；

3) 将事务标识符转换为对象标识符；

4）调用 control_POA 的 create_reference_with_id 创建对象引用并返回此引用；

要注意的是，由于使用默认伺服程序时POA对所有的对象请求调用同一个伺服程序进行处理，因此为了使默认伺服程序能够正常工作，必须为事务控制器创建单独的POA，并且其IdAssignmentPolicy要为USER_ID, RequestProcessingPolicy为USE_DEFAULT_SERVANT。

当应用程序通过事务控制器的对象引用调用其方法如get_terminator时，服务器端ORB将根据对象引用找到所属的控制器对象适配器control_POA，并调用其默认伺服程序的方法。事务控制器对象的方法如get_terminator应实现如下：

1）调用POACurrent的get_object_id()得到当前对象引用的对象标识符；

2）调用Terminator_POA的create_reference_with_id创建对象引用并返回此引用；

get_coordinator实现类似。

当应用程序使用事务完成器来提交某事务时，事务完成器伺服程序可如下实现，以commit方法为例：

1）调用POACurrent的get_object_id()得到当前对象引用的对象标识符；

2）将对象标识符转换为事务标识符；

3）根据事务标识符在事务工厂维护的transaction列表中查找对映事务；

4）提交该事务

事务的控制器，完成器和协调器实现如图2所示。

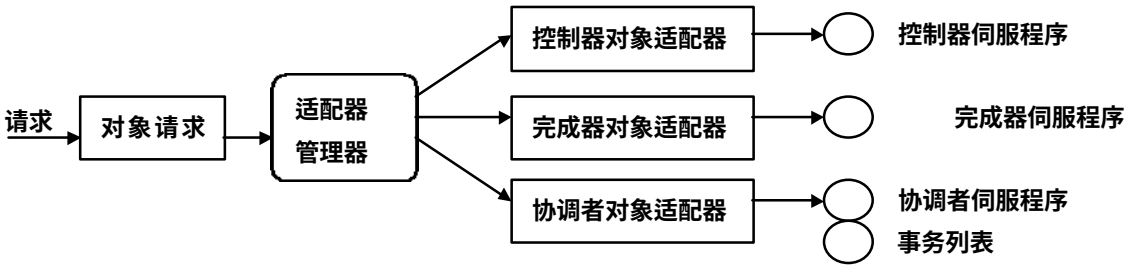

图 2　控制器、完成器和协调者的实现

3.3　事务完成

当应用程序完成事务中所有操作后，将请求 OTS 提交事务。OTS 作为事务协调者驱动各个参与者执行两阶段提交。可恢复对象要参与事务的提交，它应当实现资源接口的 prepare, commit, rollback, forget, commit_one_phase 等方法，并且通过该事务的事务协调者的 register_resource 方法向 OTS 注册该资源。因此，OTS 只需通过 IDL 编译器生成资源框架类，而从框架类提供资源实现是应用程序的责任。要注意的是，可恢复对象和资源的实现要遵循两阶段提交的语义。可恢复对象应使用事务日志来记录事务处理过程，在失败情况下能够恢复。资源对象的 prepare 应当准备提交可恢复对象中的操作，并返回 VoteReadOnly、VoteCommit、VoteRollback 之一。Rollback 方法采用一定机制撤消事务对可恢复对象的改变，如可以在事务中第一次更改可恢复对象时作一个备分；Rollback 允许被多次调用，包括系统失败之后；如果它已经忘记了该事务就什么也不做，但不会出错。Commit 方法应该使事务中对可恢复对象的更改永久保存下来。另外，事务参与者还必须具有处理系统失败的能力。当可恢复对象注册资源时将会得到一个恢复协调者 RecoveryCoordinator 对象的引用。恢复协调者包括 replay_completion 方法；如果资源已经被 prepare,但两阶段提交失败，资源都应调用 replay_competion 方法完成事务。

在事务完成中，OTS 负责调用注册在某事务上的资源对象的 prepare 等方法，协调各参与者执行两阶段提交，状态转换如图 3 所示。它首先调用各个资源对象的 prepare 方法，询问各参与者是否决定提交事务，然后等待判定结果；如果所有的资源都同意提交，那么它再调用各个资源的

commit 方法提交各部分操作。如果一个或多个资源撤消事务，那么 OTS 调用其它资源的 rollback 方法，通知它们回滚事务。

资源在接收到 prepare 调用后，如果它决定退出，则进入 Abort 状态，不在与事务有关系。如果它决定提交事务，则等待 OTS 的下一条指令。下一步根据事务失败还是成功，可能接收到提交或撤消命令。另外，由于 OTS 规范还定义了同步接口，因此 OTS 还要负责在提交前后调用已注册的同步对象的方法。

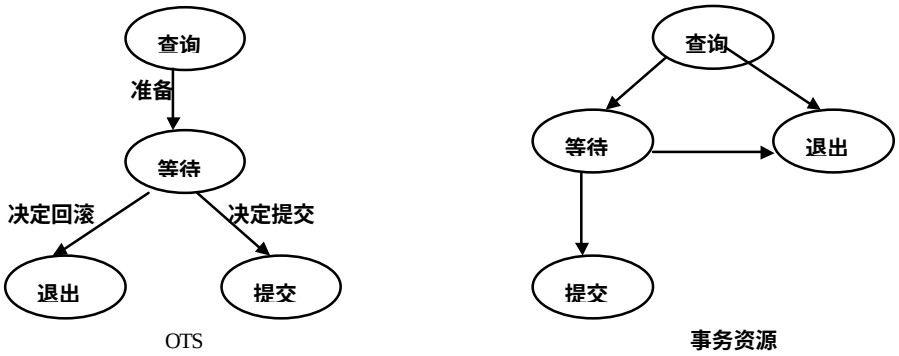

图 3 两阶段提交状态转换

3.4 事务上下文传播

当应用程序开始一个新事务后，如果它调用事务性对象的操作，该操作将作为事务的一部分进行处理，即服务器代表客户机执行了事务操作。客户端事务到服务器上的传播是通过事务传播上下文实现的。事务上下文指明了当前事务及其可能的父事务列表，每个事务的标识符和事务完成器、事务协调者对象引用。

OTS 规范定义了两种事务传播方式：隐式和显式。隐式传播自动将对事务性对象的调用请求与客户端事务上下文关联，应用程序不直接暴露于事务.在显式模型中，应用程序使用事务工厂来创建事务，并将得到的事务控制器对象引用作为操作的一个参数传递给服务器，它实际上就是普通的带有事务参数的对象操作。

与显式传播不同，隐式传播通过 Current 伪对象来实现事务与线程关联。Current 代表了当前线程的事务。应用程序调用 Current 对象的 begin()开始新的事务，客户机的 OTS 库把这个新事务和当前线程关联。如果这个线程进行远程激发，OTS 库将截取输出的请求，检查目标对象接口是否从事务性对象继承。若是，OTS 就把事务上下文加入到输出的请求中，并且当请求到达服务器端后重建事务/线程关联，这样服务器端执行请求的线程就能够代表客户机最初开始的事务执行操作。要注意的是，对象适配器不需要为每个请求初始化事务上下文，如果应用程序希望某对象的操作作为事务的一部分，那么该对象的 IDL 接口一定要从事务性对象派生。

Current对象能够执行对事务的大部分操作，如提交、回滚、查询事务当前状态等。它的接口是伪接口，对它的调用就相当于访问普通的C++对象。它的功能可以通过封装其它对象如事务工厂、事务完成器、事务协调者等实现。如begin方法可以通过先得到事务工厂的引用，然后远程调用事务工厂的create方法来创建事务。提交事务可以先获得事务的事务完成器引用，然后通过它来提交。从上面可以知道，事务上下文中包括了事务完成器和事务协调者引用，只要能够访问事务上下文，就可以实现对事务的全部操作。当前线程与事务上下文的关联可以利用线程局部存储区(Thread Local Storage，以下简称TLS)来实现。UNIX和WIN32中的多线程程序都可以利用TLS来保存与线程相关的数据。WIN32提供了TlsAlloc, TlsGetValue，TlsSetValue，TlsFree，UNIX提供了pthread_key_create，pthread_key_de lete， pthread_getspecif ic，pthread_setspecific等函数实现TLS的分配、释放、存取等。在事务上下文传播中对象适配器为请求执行线程分配TLS，初始化上下文。

然后执行线程可以通过Current读取TLS得到上下文中的事务完成器和事务协调者等，实现对事务的操作。

4　结 束 语

按照上面的思想，本文在基于 POA 的对象请求代理上实现了一个完全遵循 OTS 规范的 OTS 服务，并且已经在与国腾集团的软件合作项目中得到应用，功能和性能均得到了一定的检验，说明上述实现方案是可行的，对人们进一步研究和实现对象事务服务具有一定的参考价值。值得指出的是，除了事务处理外，如果要作为一个完整的产品，对象事务管理器还包括负载平衡、安全等内容，如何将这些技术与 OTS 融合，是下一步要研究的内容。

参　考　文　献

1　Object Management Group. CORBA services: Common Object Services Specification, 2000

2　Object Management Group. Object Transaction Service,2000

3　Object Management Group. The Common Object Request Broker: Architecture and Specification,1999

4　X/Open. Distributed Transaction Processing: The XA Specification,1991

5　Michi Henning, Steve Vinoski 著．基于C++ CORBA高级编程，徐金捂，徐 科，吕志民，等译．北京：清华大学出版社，2000

6　Hu Jian, Tang Xuefei, Liu Jinde. Research of CORBA-baseddistributed real-time fault-tolerant system. Journal of University of Electronic Science and Technology of China, 2000.29(2):193~196[胡 健，唐雪飞，刘锦德．基于CORBA的实时容错系统的研究．电子科技大学学报,2000.29(2):193~196]

7　Su Sen, Tang Xuefei, Liu Jinde. Object-oriented interoperability technology. Journal of University of Electronic Science and Technology of China, 1998.27(1):90~94[苏 森，唐雪飞，刘锦德．面向对象的互操作技术．电子科技大学学报，1998.27(1): 90~94]

Research and Implementation of Object Transactions Service Based on POA in CORBA

Yang Tao　Zheng Xiaoxia　Liu Jinde

(College of Computer Science and Engineering UEST of China　Chengdu　610054)

Abstract　This paper introduces the concept of transaction and its properties, including atomicity, consistency, isolation and durability. The two-phase commit algorithm to guarantee the ACID properties is described. The architecture of object transaction service in CORBA is studied. It also addresses the object oriented mechanism of distributed transactions processing by analysis of the objects and interfaces in the specification, Based on portable object adapter the implementation of most interfaces including transaction factory, control, coordinator and resource is presented.

Key words　transaction;　two-phase commit;　object;　object transaction service;　object reference