J2EE 平台中分布式事务互操作的研究与实现*)

张 勇[1,2] 胡剑军[1,2] 陈宁江[1,2]

(中国科学院软件研究所软件工程技术研发中心 北京 100080)[1]

(中国科学院软件研究所计算机科学重点实验室 北京 100080)[2]

摘 要 RMI-IIOP 协议扩展了 J2EE 应用服务器的应用领域。符合 CORBA IDL 规范的非 Java 语言编写的 CORBA 客户端可以通过 IIOP 协议与 J2EE 应用组件之间实现互操作。在分布式事务环境中,CORBA 客户端通过 IIOP 协议调用应用服务器上的 EJB 组件时,应用服务器需要保证客户端的 CORBA 调用能正确到达 EJB 组件,并且事务上下文在客户端和服务器之间能够有效传播。本文基于应用服务器的 JMX 体系结构,设计了一个支持事务互操作的服务框架,实现了事务环境下基于 IIOP 协议对 EJB 的调用。它具有较好的灵活性和可重配性。并已经在中科院软件所研制的 OnceAS 应用服务器中得到了实现。

关键词 分布式事务,RMI-IIOP 协议,J2EE 应用服务器,事务互操作

Research and Implementation of Distributed Transaction Interoperability in J2EE Platform

ZHANG Yong[1,2] HU Jian-Jun[1,2] CHEN Ning-Jiang[1,2]

(Technlogy Center of Software Engineering, Institute of Sofware, Chinese Academy of Sciences, Beijing 100080)[1]

(Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)[2]

Abstract RMI-IIOP Protocol provides EJBs deployed in J2EE Application Server interoperability with CORBA objects implemented in various languages. To buid reliable applications, it is necessary to invocate EJB through IIOP Protocol in distributed transaction context. There are two key problems in J2EE Application Server to solve; dispatching correctly CORBA invocation to EJB, and propagating transaction context from client to server. This paper gives a flexible and reconfigurable transaction interoperation architecture based on JMX and interceptor technologies. The work shown in this paper has been implemented in ONce AS Application Server developed by Institute of Software, the Chinese Academic of Sciences.

Keywords Distributed transaction, RMI-IIOP, J2EE application server, Transaction interoperability

1 引言

RMI IIOP 协议是对 RMI(Remote Method Invocation)远程调用协议的扩展,是 J2EE 平台[1]的基本技术规范之一。通过 IIOP(Internet Inter-ORB Protocol)协议,符合 CORBA IDL 规范的非 Java 语言实现的 CORBA 客户端,可以调用 J2EE 应用服务器中的 EJB 组件[2]。在实际的应用中,为了提高可靠性,往往需要使用分布式事务机制,此时会出现 CORBA 客户端对 EJB 组件的事务性调用,实现分布式事务互操作。在分布事务环境中基于 IIOP 协议对 EJB 进行调用,J2EE 应用服务器需要解决好客户端请求正确转发给 EJB、事务上下文的传递等问题。

目前,主要的应用服务器产品(如 BEA WebLogic、IBM WebSphere、Oracle 9i/10g AS 等),都把基于 IIOP 协议对 EJB 组件进行事务性调用作为基本功能之一。但通过分析这些应用服务器的实现,发现它们存在一些不足,主要包括:对 IIOP 的支持不够灵活,作为 IIOP 整体服务的动态可部署能力、各部件的可动态配置能力不强。本文给出了一种基于 JMX[3]微内核管理体系结构的解决方案,所实现的分布式事务互操作体系具有较强的灵活性和可重配特性。

本文第 2 节分析了分布事务环境中基于 IIOP 协议实现互操作需解决的问题及本文的解决思路;第 3 节对系统的体系结构及各主要组成部分进行了介绍;第 4 节对实现过程中需解决的问题给出了关键实现;最后总结了本文工作的特点。

2 问题分析与设计思想

2.1 CORBA 调用的一般过程

在分析 J2EE 平台中基于 IIOP 协议实现事务互操作需要解决的问题以前,我们首先回顾 CORBA 客户基于 IIOP 协议调用的一般过程[4],如图 1 所示。

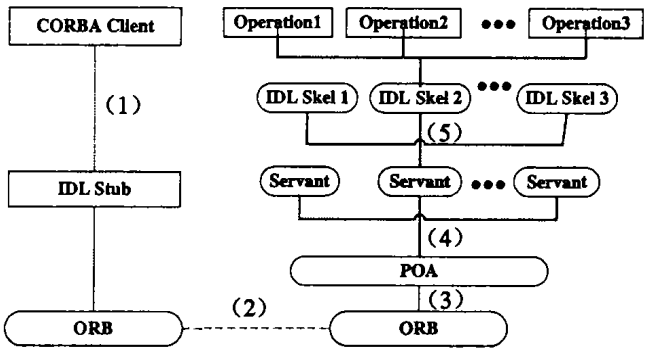

图 1 基于 IIOP 协议调用的一般过程

(1)客户端针对 IDL 存根(Stub)发出调用。

(2)请求调用通过客户端 ORB 传递到服务器端 ORB。

(3)服务器端 ORB 根据 IIOP 请求中 IOR(Interoperability Object Reference)的有关信息,定位对象适配器 POA

*)基金项目:973 计划,网构软件中间件平台模型和框架研究(2002CB312005);863 计划,网络环境的系统软件核心技术及运行平台(2001AA113010);863 计划,面向新型 ERP 的可重配 Web 应用服务器研究与应用(2003AA413010)。张 勇 博士研究生,主要研究领域为网络分布式计算;胡剑军 博士研究生,主要研究领域为网络分布式计算;陈宁江 博士研究生,主要研究领域为网络分布式计算。

(Portable Object Adaptor),并将请求转发给该 POA。POA 的主要责任是:管理 CORBA 对象的生命周期;生成 CORBA 对象引用;激活(Activation)和去活(Deactivation)注册在 POA 中的 CORBA 对象;负责将请求调用转发到合适的 CORBA 伺服器。为支持和管理不同种类的 CORBA 对象及不同风格的 CORBA 伺服器,服务器端的应用可以嵌套生成多个 POA。但至少要有一个 RootPOA,别的 POA 由 RootPOA 生成。

(4)POA 将请求调用转发到合适的 CORBA 伺服器。CORBA 伺服器是以某种语言实现的实际为 CORBA 调用提供服务的实体。

(5)当 CORBA 伺服器接收到请求调用后,利用 IDL 框架(Skeleton)将 IIOP 请求中的参数转换成为 CORBA 伺服器的操作能够接收的参数,然后将调用转发到具体的操作,以完成调用。

2.2 事务互操作的问题及解决方案

通过以上的分析可以看出,在 J2EE 平台中基于 IIOP 协议实现事务互操作,主要面临以下问题:

(1)CORBA 调用中实际提供服务的是 CORBA 伺服器,而在应用服务器中提供服务的是 EJB 对象,它不是 CORBA 对象。

(2) CORBA 客户端发出的调用是针对 IDL 接口生成的存根,而 EJB 提供的是 Java 语言描述的 EJB Home 接口和 Object 接口,而不是 IDL 接口。

(3)为使参与方协同完成分布式事务,需要基于 IIOP 协议在客户端和服务器端之间传播事务上下文。

针对上述问题,本文采用如下的解决思路:

(1) 为了使 EJB 为 CORBA 调用提供服务,本文为 EJB Home 接口和 Object 接口分别生成一个 CORBA 对象及实现该 CORBA 对象的伺服器。CORBA 伺服器在接收到调用请求后,重新构造一个调用请求(Invocation)转发给 EJB 容器(Container),由 EJB 容器负责完成实际的服务。

(2) 本文实现了 Java 语言到 IDL 语言的映射。这样可以将应用服务器中 EJB 组件 Java 语言描述的 Home 接口和 Object 接口映射成为 IDL 语言描述的形式,供 CORBA 客户发出针对 IDL 语言的调用。当客户端发出的针对 IDL 接口的调用到达 CORBA 伺服器时,然后再将针对 IDL 的调用还原回 Java 语言的形式。

(3)本文利用可移植请求拦截器机制[4],实现事务上下文的传播。

(4)为了使没有事务管理器的客户端能够进行分布事务管理,需要在应用服务器中提供 CORBA 事务服务,以使客户端能以 CORBA 方式请求事务服务。

3 系统的体系结构

在 OnceAS 应用服务器[5]中,实现了一个遵循 JMX 规范的可扩展管理框架。在该管理框架中,应用服务器中提供的各种服务,都以服务组件(MBean)的形式,注册到 MBean 服务器(JMX Server)中进行管理。管理框架负责服务组件生命周期的管理、服务之间依赖关系的管理及服务的部署和反部署等工作。在应用服务器中,与事务互操作相关的各个组件,包括 CORBA 服务、Cosnaming 名字服务、服务器端 ORB、EJB 容器等,都以 Mbean 的形式被纳入该管理框架。基于可扩展的管理框架,简化和方便了各项服务的开发和管理,并且使服务具有较好的模块化结构和较强的可配置性,能够支持服务的动态部署[6]。

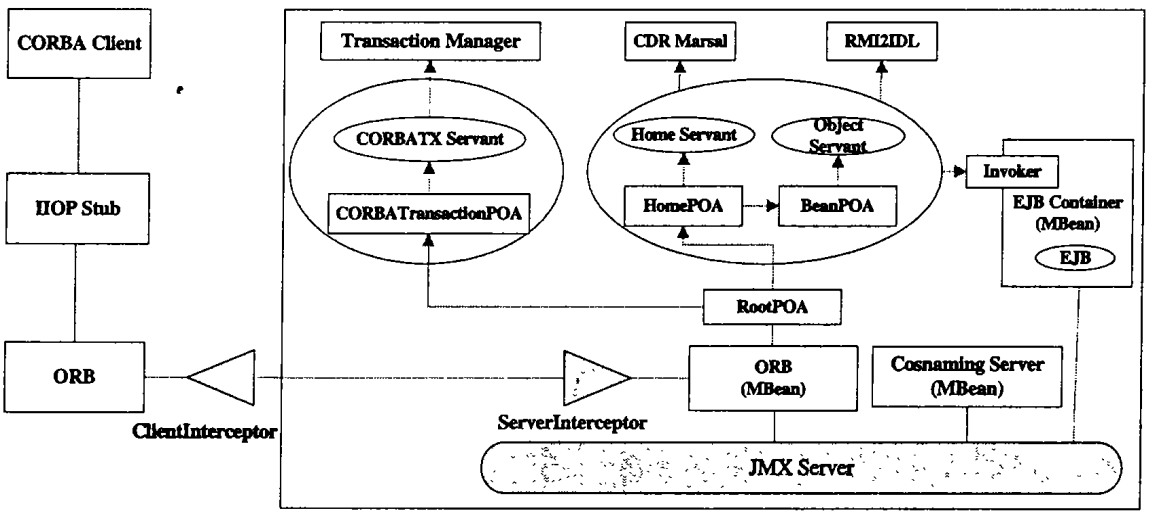

图 2 基于 IIOP 协议的事务互操作体系结构

图 2 给出了 OnceAS 中分布式事务环境下基于 IIOP 协议对 EJB 组件进行调用的整体结构,下面介绍其中的主要组成部分:

POA 本文主要实现了以下几种 POA:(1)HomePOA。具有持久生命周期策略,EJB Home 接口伺服器注册到 HomePOA 中;(2)BeanPOA。EJB Object 接口伺服器注册到 BeanPOA 中,其中用于注册会话 Bean Object 接口伺服器的 POA 具有暂态生命周期策略,用于注册实体 Bean Object 接口伺服器的 POA 具有持久生命周期策略;(3)CORBA 事务 POA。用于注册实现 CORBA 事务的伺服器,具有持久生命周期策略。因此,对会话 Bean 实例的引用是暂态的,对实体 Bean 实例的引用和 EJB Home 接口的引用是持久的。

CORBA 伺服器(Servant) 我们分别为 EJB 的 Home 接口和 Object 接口生成了 Home 伺服器(Home Servant)和 Object 伺服器(Object Servant)。Home 伺服器负责接收 HomePOA 发出的对 Home 接口的调用,Object 伺服器负责接收 BeanPOA 发出的对 Object 接口的调用,然后将调用转发给 EJB 容器,由 EJB 容器转发给实际的 EJB 实例。

RMI/IDL 映射器(RMI2IDL) RMI/IDL 映射器负责将 Java 语言描述的 Home 接口和 Object 接口转换为 IDL 语言

描述的接口，映射遵循 OMG 的 Java 语言到 IDL 语言映射规范[7]。

CDR 编码转换器(CDR Marshal) 基于 IIOP 通信时，采用 CDR(Common Data Representation)的编码方式[4]。CDR 编码转换器负责实现 CDR 编码格式的流(CDR Stream)与 Java 基本类型之间的转换。一个给定方法的存根策略(StubStrategy)知道如何将该方法的参数序列序列化(Marshal)进一个 CDR 输出流、从一个 CDR 输入流中反序列化(Unmarshal)方法的返回值和抛出的异常。一个给定方法的框架策略(SkeletonStrategy)知道如何从一个 CDR 输入流中反序列化方法参数序列，将方法的返回值和抛出的异常序列化进一个 CDR 输出流。

请求拦截器(Request Interceptor) 请求拦截器可以在请求传递路径上的特定拦截点拦截经过的请求和响应流，可以查询请求信息，并操纵从客户端传递到服务器的上下文信息[4]。我们设计了两类请求拦截器：一类是客户端请求拦截器，负责向与当前线程相关的 IIOP 请求中插入事务上下文信息；另一类是服务器端请求拦截器，负责从输入服务器端的 IIOP 请求中取出事务上下文信息，并用其构造事务，构造的事务将作为 CORBA 伺服器发出的 Invocation 调用的一个参数。

CORBA 事务服务 CORBA 事务服务的实现以应用服务器中提供的事务服务为基础。J2EE 为支持分布式事务提供了 JTA(Java Transaction API)规范和 JTS(Java Transaction Service)规范[1]。JTA 是一种高层的、与实现和协议无关的 API，应用程序和应用服务器使用 JTA 来访问事务；JTS 规范是支持 JTA 的事务管理器的实现规范，在高层 API 之下实现了 OMG OTS(Object Transaction Service)1.1 规范的 Java 映射[8]。OnceAS 应用服务器提供了 JTA 接口的实现和支持 JTS 的事务管理器。为了使服务器能接收 CORBA 客户事务调用请求，为客户端提供事务服务，我们将应用服务器提供的事务服务通过 MBean 形式封装成 CORBA 事务服务。

ORB 和 Cosnaming 的支持 为给事务互操作调用提供底层的 IIOP 通信支持，OnceAS 应用服务器端封装了一个第三方的 Java 版本的 ORB 实现。该 ORB 被封装成为一个 MBean 服务(ORBService)，被集成到应用服务器中。通过 ORBService，使应用服务器中所使用的 ORB 实现具有良好的可替换性，通过简单修改 ORB 服务的 XML 配置文件，就可以根据需要替换其它 ORB 实现。

此外，支持 IIOP 调用的 EJB 需要在 Cosnaming 名字服务器中注册名字，应用通过 Cosnaming 名字服务来查找和获得 EJB 实例。因此，OnceAS 中内置了一个符合 INS(Interoperability Naming Service)规范[4]的 Cosnaming 名字服务器，它也作为一个 MBean 进行管理。

4 系统实现的关键技术

4.1 EJB 组件部署到应用服务器后的处理过程

为了使 EJB 能够接收 CORBA 调用，当它们被部署到应用服务器上以后，应用服务器需要进行相关处理，主要的处理步骤如图 3 所示：

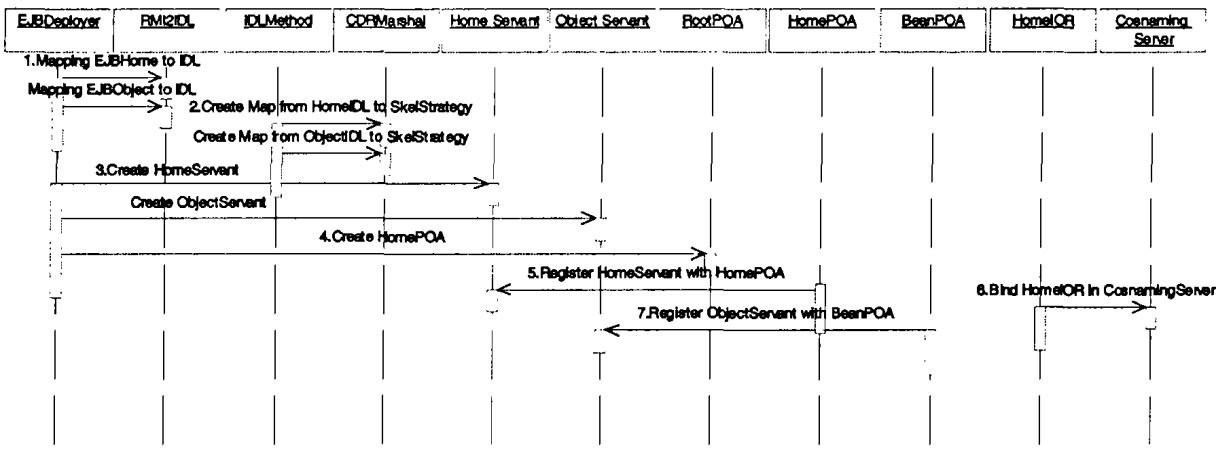

图 3 部署 EJB 组件的处理顺序图

1)应用服务器的 EJB 部署器(EJBDeployer)调用 RMI2IDL 映射模块，分析 EJB Home 接口和 EJB Object 接口，得到接口中每个方法的 IDL 名称(IDLMethod)。

2)利用 RMI2IDL 的映射结果，建立接口 IDL 方法名到框架策略的映射。分别为 EJB Home 接口和 Object 接口建立一个接口 IDL 方法名到框架策略的映射：HomeMethod2StrategyMap 和BeanMethod2StrategyMap。这两个映射分别供 Home 伺服器和 Object 伺服器对 EJB 接口中的方法进行序列化和反序列化时使用。

3)应用服务器的 EJB 部署器为 EJB 的 Home 接口和 Object 接口分别生成 Home 伺服器和 Object 伺服器。Home 伺服器和 Object 伺服器，需要实现 EJB 接口所对应的 IDL 接口。CORBA 伺服器采用了基于流的 ORB API(stream-based ORB API)的方式，接收来自于 POA 转发的通用的、与类型无关的请求，并把请求转换为一个通用的 Invocation 对象，然后将这个 Invocation 对象转发到 EJB 容器。基于流的方式与以 DSI (Dynamic Skeleton Interface)方式实现的 CORBA 伺服器相比，由于不用将操作参数和结果包装成 CORBA Any 类型，因此系统花费的代价较少[4]。

4)应用服务器通过 ORB，为 Home 伺服器生成一个具有持久生命周期策略的 HomePOA。

5)将 Home 伺服器注册到 HomePOA 中。

6)为 EJB Home 接口生成 CORBA 对象引用 HomeIOR (Interoperability Object Reference)，并将 HomeIOR 绑定到 Cosnaming 服务器中。

7)HomePOA 负责为 Object 伺服器生成 BeanPOA：为实体 Bean 的 Object 伺服器生成具有持久生命周期策略的 POA，为会话 Bean 的 Object 伺服器生成具有暂态生命周期策略的 POA。将 Object 伺服器注册到各自 BeanPOA 中，并激活，以备后续调用。

4.2 事务上下文的传播及请求拦截器的实现

本文基于可移植请求拦截器(Portable Request Interceptor)机制实现事务上下文的传播。拦截器的目的是对ORB的处理过程进行拦截,利用ORB服务在ORB处理中加入一些特定的处理过程。通过在ORB上设置拦截点,将拦截器注册到拦截点上,这样就可以拦截通过ORB的上下文信息。利用拦截器中实现的钩子(hook)方法,实现对上下文信息的特定处理。执行完拦截器中的方法后,然后再沿原来的传递路径继续向下执行。基于拦截器机制实现对事务上下文的传递,具有简单灵活的特点,而且实现的服务具有在不同ORB实现之间的可移植性[4]。拦截过程如图4所示。

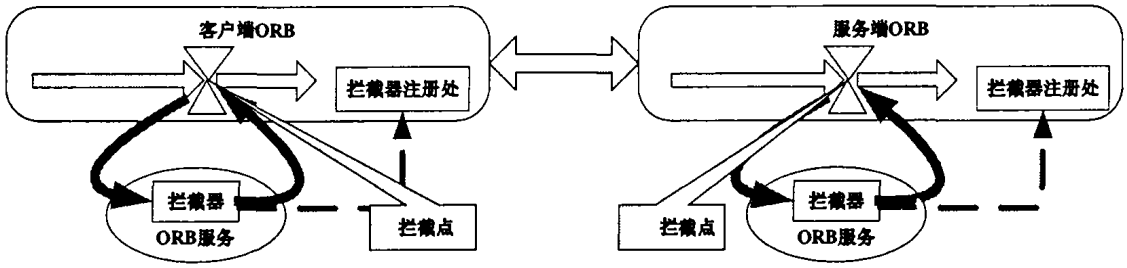

图4 基于可移植请求拦截器传递事务上下文的过程

4.2.1 *客户端请求拦截器的实现* 客户端请求拦截器负责向客户端发出的IIOP请求调用中,插入事务上下文。初始化客户端ORB时,需要使用客户端请求拦截器初始化器(ClientInterceptorInitializer),将客户端请求拦截器(IIOPTXClientInterceptor)注册到该ORB上。初始化的过程主要是给拦截器分配PICurrent和SlotID,其中PICurrent是用于服务上下文和请求(响应)服务上下文之间转换数据的槽表(slot table);slotID代表的槽具体存放与本拦截器相关的上下文信息。客户端请求拦截器的实现如图5所示,主要实现了下面两个方法:

(1)setOutgoingPropagationContext(PropagationContext tpc)方法,该方法将把要传递的事务上下文信息TPC设置到piCurrent的slot中。

(2)send_request(ClientRequestInfo ri),一个钩子方法,拦截器将修改服务上下文信息,使其包含事务上下文。

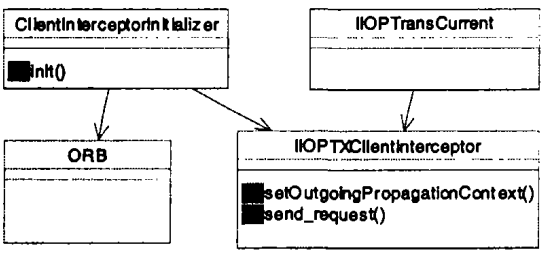

图5 客户端请求拦截器的实现

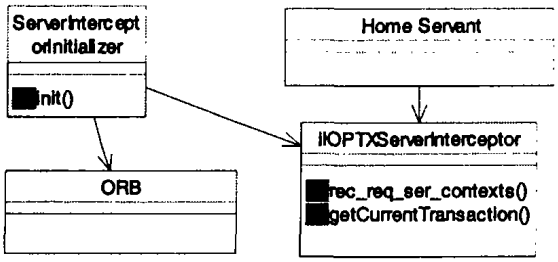

图6 服务器端请求拦截器的实现

4.2.2 *服务器端请求拦截器的实现* 服务器端请求拦截器,负责在服务器端取出客户端传递过来的事务上下文信息。与客户端一样,服务器端拦截器也要在ORB初始化时注册到服务器端的ORB上。服务器端拦截器的实现如图6所示,主要实现了下面两个方法:

(1)receive_request_service_contexts(ServerRequestInfo ri),一个钩子方法,拦截器得到输入请求上下文信息,并将信息转移到拦截器的piCurrent槽表中的槽slot中;

(2)getCurrentTransaction(),该方法将被CORBA伺服器调用。首先判断被拦截转移到槽表中的IIOP消息是否存在事务上下文信息TPC,若存在,则从TPC中取出事务的GloabID信息,利用GloabID信息,通过tpcImporter[1]重新构造事务Transaction,并返回。该方法将被EJB组件的CORBA伺服器调用,将获取的事务作为转发给EJB容器的Invocation调用的一个参数。

4.2.3 *实现事务互操作的过程* 一个EJB部署完成后,当CORBA客户对其发出事务性调用时,主要的处理过程如图7所示。

1. 客户端通过查找Cosnaming服务器获得接口的CORBA对象引用。

2. 通过IOR得到客户端存根。

3. 对存根发出调用。

4. 调用到达客户端ORB。

5. 客户端的事务请求拦截器将事务上下文(TXContext)插入到客户端ORB发出的IIOP请求调用中。

6. 服务器端的ORB接收到达的IIOP请求,服务器端请求拦截器拦截请求信息,取出事务上下文。

7. 由POA负责将客户端的请求转发给CORBA伺服器,如果是对Home接口的调用,则POA转发给Home伺服器,如果是对Object接口的调用,则转发给Object伺服器。下面以对Home伺服器的调用为例,对在伺服器中的处理过程进行说明。

首先按照HomeMethod2StrategyMap中存放的规则,对POA转发来的流式调用请求进行反序列化;然后根据客户端发出的对EJB Home接口的不同调用方法(如"_get_homeHandle","_get_EJBMetaData","remove__java_lang_Object"等)分别进行处理;其次,调用服务器端请求拦截器,获取客户端传递的事务上下文,并构造事务;最后,根据前几步的处理结果,构造一个通用的Invocation调用,形式为:Invocation(EJB对象ID,EJB方法名,方法参数,事务),并将In-

(下转第43页)

虑 Web 应用的 QoS 问题。Web 服务器集群环境下的 Web QoS 控制更是一个崭新的研究领域，它不仅要面临单一 Web 服务器环境下 Web QoS 控制的问题，而且需要从集群系统的角度来合理调度资源，满足应用对 Web QoS 的要求。

本文根据 Web 应用中 Session 的 HTTP 请求要求访问服务器的一致性，提出了在集群系统的分配器上采用基于 SessionID 的分配算法。后端的服务器采用改进访问控制策略。模拟实验结果表明，该控制模型对 Web 服务器集群系统的 QoS 具有比较满意的控制能力。

参 考 文 献

1 Cherkasova L, Phaal P. Session Based Admission Control: a Mechanism for Improving the Performance of an Overhead Web Server: [HP Laboratories Report No. HPL-98-119]. June, 1998. Http://www.hpl.hp.com/98/HPL-98-119.html 8/HPL-98-119.html

2 李波，李双庆，程代杰. Web 服务器集群中 TCP Handoff 技术及其实现. 计算机工程与应用. 2004

3 Valeria C, Michele C, Philip S Y. Dynamic Load Balancing on Web-Server Systems. IEEE Internet Computing, 1999(5/6): 28～39

4 Nochols K, et al. Definition of the Differentiated Service Filed in the IPv4 and IP v6 Headers. RFC 2474, Network Working Group

5 Mohit A, Darren S, Peter D, Willy Z. Scalable Content-aware Request Distribution in Cluster-Based Network Servers. In: Proc. of the 2000 Annual Usenix Technical Conference, San Diego, CA, June 2000

6 Vivek S, Mohit A. Locality-Aware Request Distribution in Cluster-based Network Servers. In: Proc. of ASPLOS-Ⅷ, ACM SIGPLAN, 1998. 205～216

7 Teo Y M, Ayani R. Comparison of Load Balancing Strategies on Cluster-based Web Servers. Transactions of the Society for Modeling and Simulation, 2001

（上接第 4 页）

vocation 调用传递给 EJB 容器，进行实际的对 EJB Home 接口的调用。方法调用执行完成，将调用结果或产生的异常按照原来路线返回客户端。在 Object 伺服器中的处理过程与 Home 伺服器类似，不再详细论述[9]。

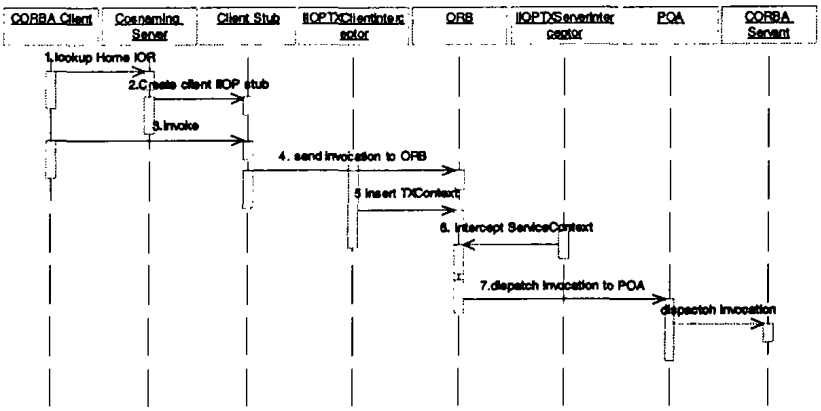

图 7 客户端发出调用后的处理过程

4.3 CORBA 事务服务的实现

为了给没有事务管理器的客户端提供一个划分事务边界和分布事务管理的接口，以将所有事务处理工作委托给服务器端处理，在系统中实现了 CORBA 事务服务，响应 CORBA 客户端对应用服务器的事务服务的调用[8]。CORBA 事务服务的实现以应用服务器内置的事务管理器为基础。

注册在事务服务 POA 中的 CORBA 事务伺服器负责提供实际的事务服务调用。CORBA 事务伺服器，提供了调用事务服务的一些标准操作，如：create_transaction，create_subTransaction，rollback，commit，get_Coordinator，get_terminator 等。

创建 CORBA 事务对象的过程如下：

(1)获得 ORB 服务的实例。(2)通过 ORB 获得 RootPOA。(3)为事务服务生成具有持久生命周期策略的事务服务 POA。(4)创建事务服务伺服器，得到事务工厂的 CORBA 对象引用。(5)将事务工厂的 CORBA 对象引用注册到 Cosnaming 名字服务器中，以供调用。

结束语 本文主要阐述了为支持分布式事务环境中基于 IIOP 协议访问应用服务器端的 EJB 组件所给出的一种解决方案，它解决了如下问题：将客户端请求转发到服务器端的 EJB，实现事务上下文在客户端和服务器端的传播，并且借助应用服务器的事务服务功能，为没有事务管理器的 CORBA 客户端提供了进行分布事务管理的 CORBA 事务服务。本文的工作主要有以下几个特点：(1)基于 JMX 规范的可扩展管理框架，使整个系统的模块化程度比较高，各个模块具有良好的可配置性，能够方便地替换 ORB 实现、Cosnaming 名字服务器、事务管理器等。(2)基于可移植请求拦截器传递事务上下文，使处理简单灵活，而且实现的服务具有在不同 ORB 实现之间的可移植性。(3)以流式方式实现的 CORBA 伺服器，使事务互操作的处理开销较少。本文讨论的工作已经在中国科学院软件研究所研发的 J2EE 应用服务器 OnceAS 中得到了实现。在本文工作的基础上，进一步的工作是要考虑在分布式环境中通过 IIOP 协议对 EJB 组件进行调用时的安全因素，即调用过程中事务上下文的安全传播及在服务端的验证等工作。

参 考 文 献

1 Sun Microsystems. Java 2 Platform Enterprise Edition Specification, v1.3. March 2001

2 Sun Microsystems. Enterprise JavaBeans Specification, Version 2.1. June 2003

3 Sun Microsystems. JMX Reference Implementation 1.2.1, Aug. 2003

4 OMG. The Common Object Request Broker: Architecture and Specification, version2.3.1, Oct. 1999

5 范国闯. J2EE 应用服务器关键技术研究：[博士学位论文]. 中科院软件所，2004

6 范国闯，等. WebFrame：一种多层次可扩展的 Web 应用服务器. 计算机学报，2004，27(4)：451～460

7 OMG. Java Language to IDL Mapping Specification, Version1.1, June 2001

8 OMG. Transaction Service Specification, Version1.2, May 2001

9 http://java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop/