一种支持多协议的开放互操作框架

吴 翔, 黄 罡, 郑子瞻, 王千祥, 梅 宏
(北京大学信息科学技术学院 软件研究所, 北京 100871)

摘 要: 在不改变支撑环境基础结构的前提下, 根据不同应用需求定制或扩展不同互操作协议, 是中间件异构问题的解决方案之一. 本文提出了一种支持多协议的开放互操作框架, 其核心在于将构件化的思想引入框架的设计以及协议的实现中, 将协议的主要功能封装为构件, 通过框架增加、删除、替换相关构件, 就能实现互操作协议的定制与扩展. 论文实现了互操作框架以及主流互操作协议, 并给出性能评测.

关键词: 互操作; 中间件; 构件; J2EE
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2003) 12A-2115-04

An Open Interoperability Framework Supporting Multiple Protocols

WU Xiang, HUANG Gang, ZHENG Zi-zhan, WANG Qian-xiang, MEI Hong
(*Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*)

Abstract: The heterogeneity between middlewares can be solved by customization or extension of interoperability protocols according to diverse application requirements without changing existing infrastructures. In this paper, we present an open interoperability framework. The basic idea is to introduce componentization into design and implementation of interoperability framework and protocols. We encapsulate the main functions of protocols into components so that the customization and extension can be achieved by using this framework to add, delete and replace corresponding components. This paper implements this framework and several mainstream protocols and then evaluates the performance.

Key words: interoperability; middleware; component; J2EE

1 引言

随着 Internet 的发展以及中间件应用的推广和深入, 中间件技术逐步完善, 版本不断更新, 一些新的中间件技术也开始出现. 中间件的繁荣, 解决了硬件平台、操作系统以及编程语言的异构问题, 但中间件自身也存在异构性, 即采用不同中间件技术的应用之间无法直接交互. 由于定义一个适合所有应用领域、运行平台、编程语言的中间件不切实际, 因此, 不同中间件之间的异构问题不可避免[1]. 中间件的异构性, 主要表现为互操作协议体系的异构. 中间件通过屏蔽底层实现细节为应用提供了强大的互操作能力. 但不同的中间件采用不同的互操作协议, 它们之间往往不可直接互通. 现有的一种解决方案是异构中间件采用相同互操作协议进行交互. 但由于不存在统一的互操作技术, 因此中间件需要具有面向领域定制与扩展互操作协议的能力. 通过分析现有中间件平台的体系结构, 本文提出一种可定制和扩展的互操作框架, 提供中间件在不改变支撑环境基础结构的前提下根据不同应用需求定制或扩展互操作协议的能力, 使运行于其上的构件可以按需采用不同的互操作协议同其它中间件上的构件进行交互, 从而解决中间件的异构问题.

2 概述

2.1 基于中间件的互操作基本要素

目前主流的分布计算中间件技术体系可分为三个部分: (1) 互操作协议体系, 包括负责传输层消息的编码与连接管理的通信协议, 定义对象调用的接口文法的接口规约, 以及解决客户与服务对象之间特定于通信协议的地址发布与获取问题的服务定位; (2) 端点, 是完整实现接口规约的编程语言实体的运行空间, 负责装载、调用、释放接口实现并为之提供系统级服务; (3) 适配器, 负责互操作框架与端点之间的关联. 现有分布计算中间件技术体系往往自成一体、不可分割, 互操作协议体系、适配器、端点相互依赖、相互影响, 相应的中间件核心模块甚至产生功能交织或重叠. 这些现状都不利于我们对中间件的互操作部分进行修改, 从而导致对新协议的支持以及特定于领域的协议定制难以实现, 间接导致中间件异构问题难以解决. 总之, 中间件的一体式技术体系是主要症结所在, 只有打破这种一体式技术体系, 形成一个灵活、可扩展而且相对独立的互操作框架, 才能解决互操作协议的扩展和定制问题.

2.2 开放互操作框架

图 1 所示的开放互操作框架打破了传统中间件的一体式技术体系. 首先, 将互操作协议体系从端点分离, 采取这种方式, 端点可以动态地同任何中间件已经支持的互操作协议体系结合, 位于端点中的构件在开发时可以不关心运行时的互操作协议和传输协议, 使构件开发具有良好的透明性. 其次, 在此基础上, 将互操作协议封装到适配器中, 构件可以更换适配器, 采用不同的适配器, 构件就可以支持不同的互操作协

收稿日期: 2003-10-27; 修回日期: 2004-01-15
基金项目: 国家重点基础研究发展规划(973 计划) 项目(No. 2002CB31200003); 国家自然科学基金(No. 60233010, 60125206, 60103001); 国家高技术研究发展计划(863 计划) 课题(No. 2001AA113060); 教育部科学技术研究重大项目(No. MAJOR0214)

议. 其三, 打破一体式框架后, 构件可以同时支持多种互操作协议, 而不同的互操作协议使用不同的命名服务, 提供不同的地址引用格式, 因此还需集成各种命名服务器. 我们的方案通过命名服务联盟提供了解决异构构件发布与定位问题的机制. 以上三种策略打破了传统中间件中端点与互操作协议间的紧耦合关系, 更进一步, 我们发现还可以把传输协议从互操作协议分离, 松散互操作协议与底层传输协议的紧耦合关系.

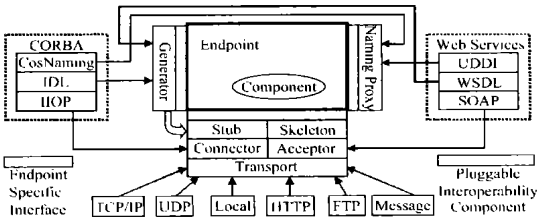

图 1 开放的互操作框架

开放互操作框架支持互操作协议和传输协议的即插即用, 与现有中间件产品的互操作框架相比, 具有良好的灵活性和可扩展性. 在框架中扩展一个新的互操作协议, 只需以下工作: (1)生成程序(*Generator*): 针对新的互操作协议中的接口规约开发生成程序. 生成程序的作用是为端点中的构件自动生成客户桩和服务器骨架. 客户桩和服务骨架分别是客户端和服务器端的适配器. 客户桩将特定于客户端程序设计语言的方法调用转换成特定于互操作协议的消息或者相反. 服务骨架将特定于互操作协议的消息转换成 *PKUAS* 所定义的消息, 或者相反. (2)命名代理(*Naming Proxy*): 将新的互操作协议的命名服务集成到命名代理中. (3)连接器和接收器(*Connector/Acceptor*): 不同的互操作协议, 对连接的管理和消息的格式规定都有所不同, 所以要针对每个互操作协议开发各自的连接器和接收器. 连接器和接收器都负责发送和接收特定于给定互操作协议的消息, 将特定于互操作协议的消息转换成特定于底层传输协议的消息(比如 *TCP* 和 *HTTP*), 以及连接管理. (4)传输器(*Transport*): 把新的互操作协议所需的传输协议增加到传输器中, 传输器的作用是以阻塞或非阻塞的方式发送和接收消息、处理超时和抛出异常来标识网络上的错误以及封装网络地址.

中间件的发展趋势是基础设施体系的构件化[2], 将互操作部分从中间件的其它部分剥离出来形成一个独立的框架, 与一体式的结构相比, 除能解决中间件的异构性问题外, 也符合了这种趋势, 为将来中间件实现反射[3]功能打下基础.

2.3 开放命名服务

不同的互操作协议采用不同的命名服务, 互操作地址往往只能存储在特定的名字服务器中, PKUAS 只能调用而不能修改这些名字服务器, 因此, PKUAS 通过命名服务联盟 JFS (JNDI Federation Server), 使用 Proxy 机制集成了这些名字服务器. 首先, 用户必须将待集成的名字服务器注册到全局唯一的 JFS 服务器中, JFS 服务器为每个注册的名字服务器实例化相应的 Proxy 插件, 该 Proxy 通过遍历将名字服务器中所有的名字转存到 JFS 服务器中, 不同的是, 与这些名字绑定的是 Proxy 引用, 而不是互操作地址. 因此, JFS 服务器在查找到名字绑定后并不直接返回, 而是委托 Proxy 查询相应的名字服务器, 然后才返回结果.

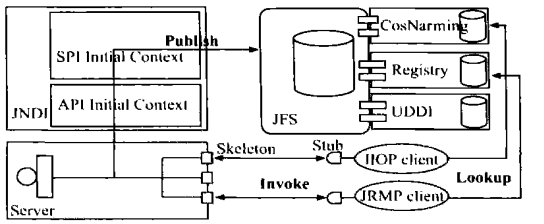

图 2 开放命名服务

3 框架的实现及其性能

3.1 PKUAS 概述

PKUAS 是一个符合 J2EE(Java 2 Enterprise Edition) 规范的构件运行支撑平台, 目前已通过 J2EE[4] 蓝图程序 JPS v1.3 的测试[5]. 为了能够明确标识、访问和操纵系统中的计算实体, 中间件必须具备构件化的基础设施体系[6]. 基于这个思想, PKUAS 设计了基于微内核、高度构件化的平台体系结构, 将平台自身的实体划分为四种类型: 容器系统、公共服务、工具和微内核.

3.2 开放互操作框架在 PKUAS 中的实现

CORBA(Common Object Request Broker Architecture)[7] 可扩展传输框架(XTF: Extensible Transport Framework) 支持多传输协议在 ORB(Object Request Broker) 中的即插即用. PKUAS 的开放互操作框架借鉴并扩展了这个思想, 不仅支持多传输协议 (TCP/IP, SSL 和 HTTP), 也支持多互操作协议(例如 IIOP: Internet Inter-ORB Protocol、JRMP: Java Remote Method Protocol 和 SOAP: Simple Object Access Protocol) 的即插即用, 充分体现了框架的灵活性和可扩展性. 图 3 所示的是框架的核心——开放通信服务部分在 PKUAS 上的实现.

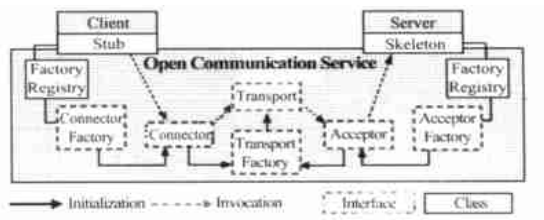

图 3 开放通信服务

客户端要调用服务器端的服务, 也就是进行一次互操作, 实际的流程是: (1) 客户端首先委托客户桩去调用服务器端的操作. (2) 客户桩到工厂登记处查找特定于所使用的互操作协议的连接器工厂, 连接器工厂为这次调用创建一个连接器. 客户桩使用这个连接器管理连接, 并将特定于互操作协议的消息转换成特定于底层传输协议的消息. (3) 传输器工厂为连接器创建特定于传输协议的传输器. 传输器是一个可配置的构件, 负责通过底层的传输协议发送和接收消息. (4) 在服务器端, 在 PKUAS 启动或者应用被部署和重新部署的时候, 首先到工厂登记处查找特定于所使用的互操作协议的接收器工厂, 接收器工厂建立所需的接收器, 接收器负责侦听特定的端口(比如 SOAP 的接收器通常侦听 80 端口). 接收器是开放互

操作框架中最主要的配置构件,即是说,当创建一个新的接收器时,与之相关的接收器工厂就会使用一系列与传输相关的参数来配置接收器.(5)接收器侦听到调用请求以后,将特定于传输协议的消息转换为特定于互操作协议的消息,发送给服务器端骨架.(6)服务骨架将特定于互操作协议的消息转换成PKUAS所定义的消息,然后直接调用服务对象的相应方法.(7)如果需要返回结果,调用的结果按上面流程相反的方向返回.

按照上述实现,在增加新的互操作协议时,不仅需为该互操作协议开发生成程序(负责自动生成客户桩、服务器骨架),将互操作协议所使用的名字服务器注册到JFS中去以外,还需要开发特定于该协议的连接器、接收器、传输器以及负责创建它们的工厂.对协议自身有自主开发与集成现有产品两种形式.PKUAS自主开发了IIOP、SOAP和LOCAL协议,集成了JDK的JRMP协议.下面分别以IIOP和JRMP协议的扩展为例来具体说明开放互操作框架在PKUAS中是如何实现的.

3.3 IIOP协议的支持

要将IIOP协议纳入互操作框架中,关键在于开发特定于IIOP协议的接收器、连接器以及底层特定于TCP/IP协议的传输器.为便于接收器、连接器以及传输器的创建,我们加入了接收器工厂、连接器工厂和传输器工厂类.在这三种可配置构件中,接收器是最重要的一部分.出于性能的考虑,PKUAS为其上部署的每一类EJB都生成了一个接收器对象,不同的接收器对象负责侦听不同的端口.接收器对象接收到通信请求后,起一个新的GIOPDecoder线程来接收IIOP消息,然后GIOPDecoder对象负责对IIOP消息进行解码,将客户端调用的参数传给服务器端骨架IIOPSkeleton,IIOPSkeleton使用这些参数来构造PKUAS内部格式的消息对象,然后发送给相应的EJB对象处理.在服务器端实现了IIOP接收器及其工厂后,再将IIOP接收器工厂纳入到服务器端的工厂登记处之中.

在服务器端增加接收器的同时,客户端也需要负责管理连接以及解释IIOP消息语义的连接器.我们针对IIOP协议建立了连接器工厂,客户端每次请求同服务器端建立连接,IIOP连接器工厂就会为这次连接创建IIOP连接器对象.IIOP客户桩使用IIOP连接器发送和接收IIOP消息.类似服务器端,客户端需要把IIOP连接器工厂纳入客户端的工厂登记处.客户端和服务器端要通过TCP/IP进行IIOP互操作,还需要开发特定于TCP/IP的传输器以及负责创建该传输器的传输器工厂.

要将IIOP纳入互操作框架,除了需要对上述的核心通信部分进行修改,还需要有能根据EJB构件的接口产生特定于IIOP协议的客户桩和服务器端骨架的生成程序.图4中的Java-to-IDL Compiler和IIOP Compiler合在一起起到了生成程序的作用.Java-to-IDL Compiler根据EJB构件的Home接口和Remote接口生成相应IDL接口.IDL接口表明了客户端和服务器程序之间的一个契约,描述了EJB构件可以被调用的方法及其参数.IIOP Compiler根据IDL接口生成支持编排和反编排这个接口所需的参数的客户桩和服务器骨架.

3.4 JRMP互操作协议的支持

互操作框架对JRMP协议的支持基于底层JDK关于JRMP的实现.因此,与IIOP协议相比,框架对JRMP的扩展支持要简单得多.如图5示,扩展所涉及到的关键技术包括特定于JRMP协议的生成程序,JRMP接收器以及代表实际EJB的伪EJB类等.当一个应用在PKUAS上部署时,如果该应用存在支持JRMP协议的EJB构件,特定于JRMP协议的生成程序(EJB-JRMP Compiler)将根据这个EJB构件为其Home接口和Remote接口分别生成代表它们的FakeEJB类,并把跟Home接口相关的FakeEJB对象通过JFS服务器注册到RMI Registry中去,RMI Registry被集成到统一命名服务之中.在运行时刻,FakeEJB作为代表实际运行EJB的伪EJB类,同JDK自带的JRMP底层设施进行交互,从中得到客户端调用的参数以及事务、安全上下文,以此构造PKUAS内部格式的消息对象,然后发送给相应的具有实际业务处理能力的EJB对象处理.

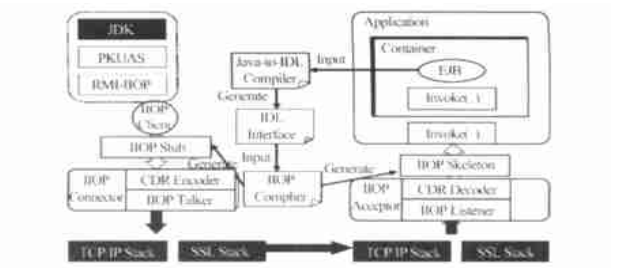

图4　IIOP实现框架

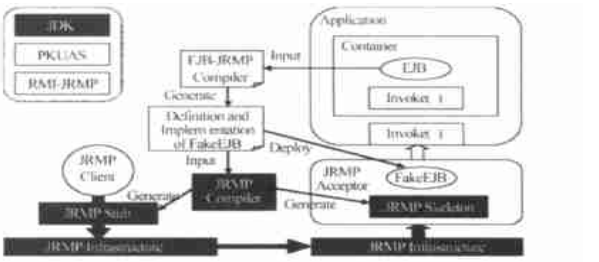

图5　JRMP实现框架

3.5 性能评测

本文提出的支持多协议的开放互操作框架在PKUAS上得到实现,并通过了对几种主流互操作协议的测试.测试所使用的平台是PIII 800MHz,512M SDRAM的PC,操作系统是Windows 2000 Professional,构件运行支撑平台采用PKUAS以及当前最成功的开源J2EE应用服务器JBoss[8](JBoss采用3.2.1版本,以all模式运行).测试程序由服务器端的无态会话EJB和客户端的独立Java程序组成.Java客户端使用各种互操作协议发送特定字节数的字符串消息到服务器端,服务器端的无态会话EJB接收消息并返回一个确认消息.每类调用重复测1万次,以此减小测试结果的随机性.

图6中PKUAS上IIOP的调用延时比JBoss的IIOP调用延时稍慢,主要由于JBoss缺省对IIOP协议做了不合规范的优化,如通过引用而不是值的方式传递对象[9].PKUAS上JRMP调用延时要明显优于JBoss的实现.此外,从图中可以看出,当参数的大小从一个字节增加到4096个字节,PKUAS使用IIOP要比使用JRMP多消耗124%~228%的调用时间.产生这种差别的主要的原因是PKUAS上的IIOP是使用纯JAVA来实现

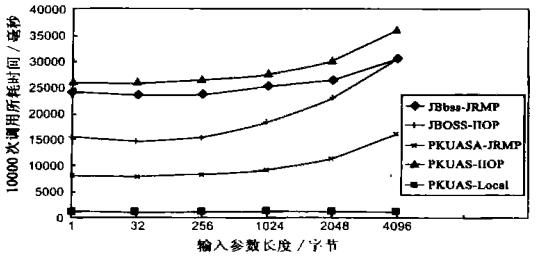

图 6 性能测试(1)

的,而 JRMP 使用的是本地代码实现,后者效率更高. Local 调用不需要网络传输数据以及编排和反编排参数,且通过引用而不是值来传递参数,因此效率最高. 图 7 是 PKUAS 上 SOAP 协议实现与 JBoss 的 SOAP 协议实现的性能比较,JBoss 通过集成开源 Web Services 引擎 AXIS 来支持 SOAP 协议. PKUAS 的 SOAP 实现性能优于 JBoss 的实现. 在通过 SOAP 调用 EJB 时,JBoss 利用 Servlet 接收 SOAP 消息并转换为对 EJB 的 RMI 调用,PKUAS 也支持这种模式(PKUAS-SOAP For Servlet). 特别地,PKUAS 允许不经过 Servlet 而直接调用 EJB(PKUAS-SOAP For EJB),此时仅需处理 HTTP 请求中的 SOAP 消息,且无需二次调用,因而性能最优.

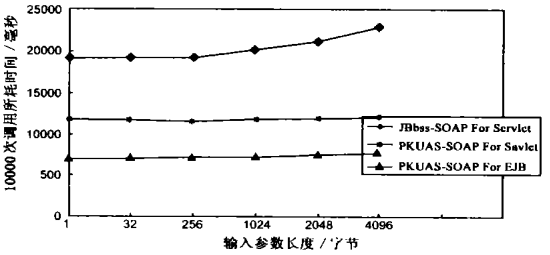

图 7 性能测试(2)

4 相关工作

与本文方案类似的主要包括 XTF, CAROL[10] 和 FlexiNET[11]. CORBA 可扩展传输框架(XTF)仅是一个传输协议的框架,不支持多种互操作协议. CAROL 是一个开放源码的互操作框架的实现,它本质上是一个互操作协议的切换器. CAROL 使 J2EE 服务器可以同时被多种协议的客户端访问,但它不具有运行时刻动态改变 J2EE 构件所支持的互操作协议的能力. FlexiNet 是一个致力于中间件配置与应用部署问题的 Java 中间件平台,它本质上是一种反射式中间件[12]. FlexiNet 的主要不足在于协议的定制与扩展较为复杂,尤其是无法集成已有的互操作协议的实现. 在中间件异构问题的其它解决方案中,Web Services 受到了前所未有的广泛支持,由此产生了一种新的异构构件互操作问题解决方案,即,异构构件的互操作协议统一采用 Web Services[13]. 但目前采用 Web Services 技术的实际应用不多,这个技术也难以适用于所有的领域.

5 结束语

本文详细论述了互操作框架在 PKUAS 上的具体实现,并讨论了几种主流互操作协议在框架上的实现,最后给出性能评测,证实了这种方法的可行性与实用性. 进一步的工作包括:扩展集群互操作协议,进一步完善框架. 实现框架的反射性,在运行时刻中允许改变构件所使用的互操作协议.

参考文献:

[1] W Emmerich. Software engineering and middleware: A roadmap [A]. ICSE-Future of SE Track 2000 [C]. New York: ACM Press, 2000. 117 – 129.

[2] Adrian Colyer, Gordon Blair, Awais Rashid. Managing Complexity in Middleware [EB/OL]. http://www.cs.ubc.ca/~ycoady/acp4is03/papers/colyer.pdf, Boston, Massachusetts, 2003.

[3] Coulson, G. What is Reflective Middleware [EB/OL]. http://boole.computer.org/dsonline/middleware/RMarticle1.htm, 2000.

[4] SUN Microsystems. Java 2 Platform Enterprise Edition Specification, Version 1.3, Proposed Final Draft 4 [S].

[5] 黄罡, 王千祥, 曹东刚, 梅宏. PKUAS: 一种面向领域的构件运行支撑平台 [J]. 电子学报, 2002, 30(12A): 39– 43.

[6] Costa F M. Combining meta-information management and reflection in an architecture for configurable and reconfigurable middleware [D]. Lancaster, UK: Lancaster University, 2001.

[7] Object Management Group (OMG). The Common Object Request Broker: Architecture and Specification Revision 2.6 [S].

[8] JBoss Group. JBoss [Z]. http://www.JBoss.org.

[9] Emmanuel Cecchet, Julie Marguerite, Willy Zwaenepoel. Performance and scalability of EJB applications [A]. Proceedings of OOPSLA'02 [C]. Washington: ACM Press, 2002. 246– 261.

[10] Guillaume Riviere, Simon Nieuviarts. Carol 1.5.1 [Z]. http://carol.objectweb.org/.

[11] Hayton R, ANSA Team. FlexiNet Architecture [Z]. Technical Report, 1999.

[12] 黄罡, 王千祥, 梅宏, 杨芙清. 基于软件体系结构的反射式中间件研究 [J]. 软件学报, 2003, 14(11): 1819– 1826.

[13] Douglas C Schmidt, Steve Vinoski. Object Interconnections: CORBA and XML-Part 3: SOAP and Web Services [EB/OL]. http://www.cuj.com/experts/1910/vinoski.htm?topic=experts&topic=experts, September 2001.

作者简介:

吴 翔 男, 1979 年 9 月生于四川, 北京大学信息科学技术学院硕士研究生, 主要研究方向为软件构件和分布计算技术. Email: wuxiang@cs.pku.edu.cn.

黄 罡 男, 1975 年 10 月生于湖南, 博士, 北京大学信息科学技术学院讲师, 主要从事软件工程、软件构件和分布计算技术等方面的研究. 电话: 010-62757801-1, Email: huanggang@cs.pku.edu.cn.