PKUAS:一种面向领域的构件运行支撑平台

黄 罡,王千祥,曹东刚,梅 宏
(北京大学信息科学技术学院软件研究所,北京 100871)

摘 要: 基于中间件的构件运行支撑平台是实现基于构件的软件复用的关键.现有中间件技术侧重于解决大多数领域开发分布系统均会遇到的共性问题,忽视了单个应用领域内的共性问题.在下一代中间件技术追求的主要特性中,开放性与灵活性能够有效增强中间件对领域特性的支持.本文介绍了一个开放、灵活、面向领域的构件运行支撑平台,重点研究了平台内核、可扩展互操作框架以及元编程机制,并结合电力领域探讨了面向领域的定制与扩展方法.

关键词: 构件; 中间件; 应用服务器; 面向领域
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2002) 12A-1938-05

PKUAS: A Domain-Oriented Component Operating Platform

HUANG Gang, WANG Qian-xiang, CAO Dong-gang, MEI Hong
(*Software Institute, Peking University, Beijing* 100871, *China*)

Abstract: The critical challenge to component-based software reuse is the component operating platform based on middleware. The existing prevalent middleware has focused on the common problems in the development of distributed systems in almost every application domain, but ignored the commonality in a given domain. The openness and flexibility, which are the well-recognized important characteristics of next generation middleware, can enhance middleware of supporting the domain-orientation much more. This paper presents an open, flexible and domain-oriented component operating platform, called PKUAS. This paper describes the micro kernel interior of the platform, interoperability framework and the meta-programming mechanism in detail. Moreover, this paper studies how to customize and extend PKUAS for the electric power systems.

Key words: component; middleware; application server; domain-orientation

1 引言

基于构件的软件复用是学术界和产业界实践软件复用思想的主要途径之一,其关键是基于中间件的构件运行支撑平台.中间件技术已成为软件界研究的热点,国内外已有不少标准和实用产品推出.但是,中间件如何有效地支持基于构件复用的软件开发方式,仍是研究者们为之努力的目标.中间件抽象出分布系统的共性问题并封装了相关的支撑机制,如分布通信机制、安全问题、事务性等.纵观现有主流的面向对象中间件技术(如 J2EE: Java 2 Platfrom Enterprise Edition、CORBA: Common Object Request Broker Architecture、COM: Component Object Model),可以发现:现有中间件很好地解决了在大多数领域开发分布系统均会遇到的问题,却难以有效支持领域特性(单个应用领域内的共性问题)的抽象及其支撑机制的封装,具体表现为:一,难以有效满足特定的业务需求,如系统对实时性和可靠性的要求;二,导致某些支持领域特性的功能重复开发.如果复用这些功能模块能够提高系统开发效率;三,某些通用功能可能用不上或者不合用,导致用户为这些功能付出额外成本.学术界与产业界都注意到这一问题,如 OMG (Object Management Group)制订了 CORBA 针对实时领域的扩展以及针对嵌入式系统的定制,某些 J2EE 产品针对客户关系管理系统等应用领域进行了优化.但是,随着应用的深入和领域的增加,这些方案难以满足领域特性的变化.因此,中间件技术如何有效支持领域特性的封装(我们将具有这种能力的中间件称为面向领域的中间件)成为中间件应用深入和普及的关键问题之一.而现有中间件缺乏面向领域扩展能力的根本原因在于其开放性与灵活性不足.

2 体系结构

2.1 设计原则

为了提供良好的开放性与灵活性,PKUAS 的设计基于如下原则:

(1)基于面向对象中间件:Gartner Group 将目前的主流中间件划分为数据访问中间件、远程过程调用中间件、事务中间

收稿日期:2002-06-06;修回日期:2002-10-10
基金项目:国家杰出青年科学基金课题(No.60125206);国家自然科学基金(No.60043002,60103001);国家高技术研究发展计划(863 计划)(No.2001AA113060);教育部重大项目(No. 重大 0214)

件、消息中间件和对象中间件.由于面向对象技术具有对构件的自然支持,因此,对象中间件是构件运行支撑平台的必然选择.

(2)结合 J2EE 与 CORBA:PKUAS 的目标应用环境是 Internet,这要求 PKUAS 必须具备跨异构平台的能力、多样化的客户端表现能力、强大的互操作能力以及良好的开放性.在现有主流对象中间件中,J2EE 提供了良好的体系结构与完整的构件模型[2],而 CORBA 则提供了强大的跨异构平台的能力与互操作能力[3].因此,PKUAS 符合 J2EE 体系结构、支持 EJB 构件模型并集成 CORBA 互操作协议.

(3)支持 ABC 方法:ABC 提供了一种基于构件复用的从高层规约到最终实现可运行系统的系统化的开发方法[4],而特征驱动、软件体系结构指导、中间件支撑的构件组装与运行环境(简称 ABC 环境)是支持 ABC 方法的主要工具.其中,ABCTool 应用于 ABC 的软件体系结构分析与设计、基于 SA 的构件组装阶段,主要功能包括构件池的管理、基于软件体系结构的图形化建模、自动合成系统的 OOD 模型.作为 ABC 环境的部署和运行平台,PKUAS 应该允许用户在使用 ABCTool 完成系统建模和组装后,简捷、直接地将系统部署到 PKUAS 中.

2.2 体系结构

借鉴操作系统的微内核思想,PKUAS 通过抽取一组最基本功能形成一个内核,将平台内部的其它功能封装在各个相对独立的模块内,允许用户根据领域特征定制与扩展这些功能模块,在系统启动阶段由内核装配成领域特定的构件运行支撑平台,如图 1 所示.基于 Java 虚拟机,PKUAS 将平台自身的计算实体划分为四种系统构件:

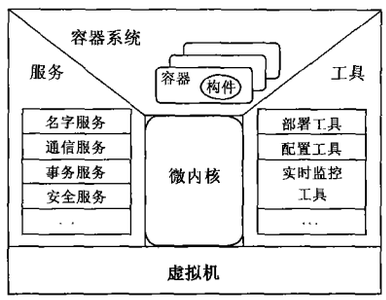

图 1　构件化的 PKUAS 平台内部体系

(1)容器系统:容器是构件运行时所处的空间,负责构件的生命周期管理(如类装载、实例化、缓存、释放等)以及构件运行需要的上下文管理(如命名服务上下文、数据库连接等).在 PKUAS 内置的四种 EJB 容器中,一个容器实例管理一个 EJB 构件的所有实例,而一个应用中所有 EJB 构件的容器实例组成一个容器系统.这种组织模式有利于实现特定于单个应用的配置和管理,如不同应用使用不同的通信端口、认证机制与安全域.

(2)服务:实现系统的非功能性约束,如通信、安全、事务等.由于这些服务可通过微内核动态增加、替换、删除,因此,为了保证容器或构件正确调用服务并避免服务卸载的副作用,必须提供服务功能的动态调用机制.对于供容器使用的服务,必须开发相应的截取器作为容器调用服务的执行点.对于供构件使用的服务,必须在命名服务中注册.

(3)工具:辅助用户使用和管理 PKUAS 的工具集合,主要包括部署工具、配置工具与实时监控工具.其中,部署工具既能热部署整个应用,也可热部署单个构件,从而实现应用的在线演化;配置工具允许用户配置整个服务器或单个应用;而实时监控工具允许用户实时观察系统的运行状态并做出相应调整.

(4)微内核:负责上述系统构件的装载、配置、卸载以及启动、停止、挂起等状态管理.与操作系统微内核不同的是,PKUAS 内核并不负责系统构件之间的通信,从而避免了整个系统性能的降低.

3 关键技术

与其它 J2EE 产品相比,PKUAS 的特点在于开放、灵活的定制与扩展能力.其关键技术主要包括平台内核、互操作框架和元编程机制.

3.1 平台内核

不同领域的应用既可能采用不同类型的容器或容器的不同实现,也可能采用不同的分布通信协议,还可能使用不同的服务或同类服务的不同实现.这要求 PKUAS 平台自身必须具有良好的结构,以允许用户根据领域特点定制和扩展容器、服务、分布通信机制.通过分析现有 J2EE 实现的体系结构及其支持构件运行的基本功能,PKUAS 抽象出一个平台内核,其主要功能就是有效地管理容器、服务、管理工具等系统构件.与操作系统微内核负责其它模块之间通信不同,PKUAS 内核并不负责系统构件之间的通信,以免过多降低性能.系统构件之间的通信或者采用 Java 语言提供的对象调用机制,或者采用 PKUAS 提供的通信服务实现.

PKUAS 内核符合 Java 平台管理标准 JMX(Java Management eXtensions),继承了 JMX 可移植、伸缩性强、易于集成其它管理方案、有效利用现有 Java 技术、可扩展等优点[5].如图 2 所示,PKUAS 内核可分为两层:

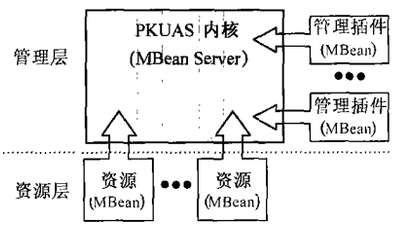

图 2　PKUAS 内核

(1)资源层:由可被 JMX 管理的系统构件组成,如容器系统、服务、管理工具等.这些被管理的构件通过 MBean 接口,对外提供与管理相关的属性和操作.MBean 可通过 Java 事件机制向外通知自身状态的改变或对其它资源的需求,同时,MBean 也可通过 Java 事件机制接收通知并做出相应的管理动作.此外,每个 MBean 还拥有一个 Metadata 对象,以描述与管理接口相关的信息.

(2)管理层:由负责注册资源的 MBeanServer 和管理资源的插件组成.MBeanServer 对外提供所有管理资源的接口,隐藏了这些资源的对象实例引用,从而允许管理资源动态的增加或删除.管理插件则是操纵注册资源 MBean 接口的构件,也是一个 MBean,因此允许用户增加新的管理功能并动态增加到系统中.典型的管理功能包括类的动态装载、监视、定时器、资源关联等.

3.2 互操作框架

技术和市场的普及与发展决定了单一的互操作技术不可能适应所有的应用领域,这意味着不同领域的构件运行支撑

平台需要不同的互操作技术.目前主流的分布中间件技术体系可分为三个部分(如表 1 所示):互操作框架——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;容器——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;适配器——负责互操作框架与容器之间的关联.

表 1 典型的分布中间件技术体系

中间件	互操作框架			容器	适配器
	协议	接口	命名		
CORBA	GIOP/IIOP	IDL	COS Naming	POA	IDL Stub/Skeleton
RMI	JRMP	Remote	Registry	Container	RMI Stub/Skeleton
Web Services	SOAP	WSDL	UDDI	不确定	不确定

现有分布中间件技术体系往往自成一体、不可分割,互操作框架、适配器、容器相互依赖、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,导致一个平台往往只能固定使用一种互操作技术.为了根据不同领域需求集成不同互操作技术,甚至允许多种互操作技术在同一个平台中同时生效,PKUAS 提出了一种开放的互操作框架,如图 3 所示.通过微内核,PKUAS 松散了互操作框架、适配器、容器之间的依赖关系,PKUAS 开放互操作框架主要由如下部分组成:

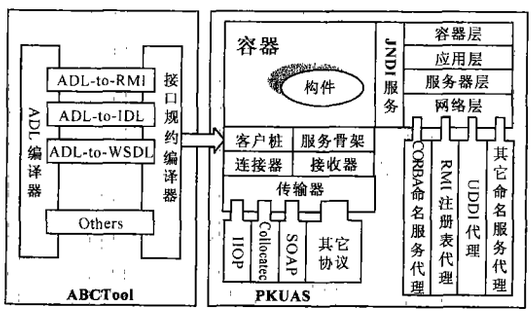

图 3 PKUAS 开放互操作框架

(1)开放通信服务:负责消息的传输.现有的通信协议扩展技术往往针对特定互操作协议下传输协议的扩展[6],而 PKUAS 的通信服务允许用户增加新的互操作协议或新的传输协议,扩展能力更强.PKUAS 开放通信服务包括如下主要构件:

(a)客户桩和服务骨架:客户桩根据接口规约转换对象调用与具体互操作协议消息(如 GIOP 消息或 SOAP 消息),并指导连接器与目标构件的接收器建立连接.而服务骨架根据接口规约转换具体互操作协议消息与 PKUAS 内部消息(容器只需理解该消息格式而无需关心底层互操作协议消息格式).为了便于用户开发,桩和骨架均由接口规约编译器自动生成.

(b)连接器与接收器:是通信服务的主要构件,负责管理连接(根据不同的消息种类,如,IIOP 的 Request, Reply, CancelRequest, CancelReply 等),网络连接复用(IIOP 既可以使用多个 IP 端口,也可使用指定的 IP 端口,而 SOAP 往往只能使用 80 端口),以及远程引用的解析(连接器根据远程引用定位并与服务器端的接收器建立连接)与构造(接收器负责构造全局唯一的网络地址,而容器负责构造容器内部唯一的构件标识).

(c)传输器:对应底层网络的一个连接,负责将互操作协议消息以底层传输协议的消息格式传输.传输器松散了互操作协议与传输协议之间的关系,用户通过实现新的传输器就能够使用不同的传输协议传输相同的互操作协议消息,如使用 TCP/IP 或 SSL 传输 IIOP 消息,简化了某些领域互操作框架的定制与扩展.为了提高性能,PKUAS 提供了 RMI-Collocated 互操作技术,允许那些驻留在同一个 PKUAS 实例的构件不通过底层网络而直接交互.而 RMI-Collocated 的关键在于实现一个不采用底层网络通信机制的传输器.

(2)开放命名服务:不同的互操作技术定义了不同的地址引用格式,使用不同的命名服务进行发布与查找.为了便于扩展以及提高性能,PKUAS 将命名服务的实现划分为四个层次.其中:

(a)容器层/应用层/服务器层:分别存储特定于单个构件/应用/服务器的数据,如容器层存储构件属性的初始值、调用构件的别名等,应用层存储 LocalEJB 的互操作地址以及,服务器层存储服务器的配置信息、服务器内所有构件的 RMI-Collocated 互操作地址和其它互操作地址(缺省为 IOR: Interoperable Object Reference)等.

(b)网络层:PKUAS 通过 Proxy 机制集成位于不同服务器上的命名服务实例,以及不同互操作技术提供的命名服务实例.首先,用户必须将待集成的名字服务器注册到全局唯一的命名联盟服务器(JFS)中,该服务器为每个注册的命名服务实例提供相应的 Proxy 插件,该 Proxy 通过遍历将命名服务实例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 查询相应的命名服务实例,然后才返回真正可用的互操作地址.另一方面,为了允许其它构件访问 PKUAS 构件,服务器初始化过程中将根据组装与部署阶段的信息,为那些被其它构件调用的 PKUAS 构件构造特定于客户端的互操作地址,并注册到 JFS 服务器,而 JFS 服务器通过 Proxy 将该名字绑定注册到相应的命名服务实例,因此,其它构件通过自己的命名服务实例就可获取 PKUAS 构件的互操作地址.

(3)开放接口规约编译器:在 ABC 方法中[4],软件体系结构分析与设计阶段的构件接口规约统一采用 ADL 描述,在构件组装阶段由 ABCTool 自动生成到不同接口规约语言的映射(如 ADL 到 IDL、WSDL、或 RMI 接口规约的映射),在构件部署阶段由相应的接口规约编译器自动生成特定于底层通信协议的客户桩和服务骨架.PKUAS 定义了接口规约编译器的扩展接口,方便用户针对新的互操作技术扩展相应的编译器,而 ABCTool 也允许扩展 ADL 到其他接口规约语言的映射机制.与其他 J2EE 产品提供独立的部署工具不同,PKUAS 将部署工具嵌入 ABCTool,使得后者支持系统分析、设计、组装、部署等全部的开发阶段,而 PKUAS 仅仅负责系统的运行.

综上所述,对于 PKUAS 而言,增加新的互操作协议需要开发相应的通信服务构件集合、命名服务代理、以及接口规约

编译引擎.而增加新的传输协议则仅需实现新的通信服务传输器构件即可.

3.3 元编程机制

现有中间件通过提供通用的、对开发者透明的分布系统特性,如分布性、安全性、事务性等,极大简化了分布应用的开发.随着网络环境与应用领域的发展,分布系统的适应性越来越重要,开发者往往需要在不修改现有设计与实现的前提下调整分布系统的行为[1].元编程(meta-programming)机制作为一种有效的解决方案被引入中间件,如 CORBA 中的 Smart Proxy、POA、DII 等[7].元编程机制通过松散系统行为与资源、公用特性的紧耦合关系,提高系统的适应能力.为了提高平台对领域变化性的适应能力,PKUAS 提供了截取器(interceptor)和可编程客户桩(Smart Proxy)两种元编程机制.如图 4 所示,截取器位于容器内部,在调用请求进入目标构件之前被激活.多个截取器可组成一个截取器链,根据请求中包含的信息依次执行特定功能,如认证、授权、审计、日志等.由于构件意识不到截取器的存在,因此可在不修改构件的前提下通过增加截取器来改变系统运行时的行为.截取器对于 PKUAS 服务的扩展至关重要,当增加供容器使用的新服务时,只需增加相应的截取器就可在容器运行时调用新增的服务.与 CORBA 截取器相比,PKUAS 的截取器类似请求截取器,作用于服务器端.而客户端可通过可编程客户桩实现类似功能,即,自己编写一个客户桩以增加新的客户端功能或改变缺省客户桩的行为.

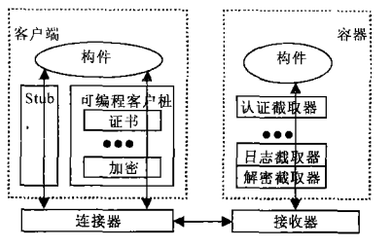

图 4　PKUAS 元编程机制

4 实例研究

在提供基础框架与设施的前提下,如何针对领域特性进行定制与扩展是 PKUAS 的研究重点之一.本节将探讨特定于电力领域的 PKUAS 通信模式扩展方法.

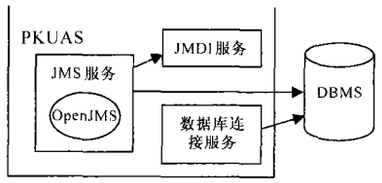

图 5　OpenJMS 集成框架

PKUAS 内置的 IIOP 协议支持同步的请求/应答模式,而电力领域需要的异步请求/应答模式和异步发布/订阅模式.为此,PKUAS 将集成一个符合 JMS(Java Message Service,定义了标准的 Java 应用消息传输机制,提供异步点到点通信与发布/订阅模式[8])规范的产品 OpenJMS[9].如图 5 所示,在不修改代码的前提下,集成 OpenJMS 包括如下三个主要工作:

(1)封装 JMSServer:OpenJMS 提供了一个可独立运行的中间服务器 JMSServer,用来缓存消息并管理通信的参与者,实现消息的异步传输和多播.OpenJMS 支持 RMI、TCP、SSL、HTTP、HTTPS 等多种底层传输协议,缺省的传输协议为 TCP.如图 6(a)所示,PKUAS 将其封装成一个 JMS 服务,即,通过一个符合 JMX 规范的 MBean 来启动、停止、配置 JMSServer.而安装 JMS 服务仅需在 PKUAS 服务配置文件中增加如图 6(b)所示内容.

(2)集成 JNDI 服务:使用 JMS 和管理 OpenJMS 均需要使用 JNDI.由于部署在 PKUAS 中的构件只能访问 PKUAS 提供的 JNDI 服务,因此,需要将 OpenJMS 内置的 JNDI 服务替换成

```
/*pku\as\message\JMSSer.java片断 */
public class JMSServer extends Service implements JMSMBean {
public void start Service() {
String [] jms_args={ " -config ", config_file};
//config_file 从 services.conf的相应属性获得.
org.exolab.jms.server.JmsServer.main(jms_args);
//启动 OpenJMS .
}}
```

(a) JMSMBean 的实现

```
/*PKUAS 的服务配置文件 (services.conf)片断 */
<SERVICE CLASS="pku.as.message.JMSServer">
<ATTRIBUTE NAME="OPENJMS_HOME"
VALUE="..\OpenJMS/>
<ATTRIBUTE NAME="config"
VALUE="../OpenJMS/conf/pkuas_jms.xml"/>
</SERVICE>
```

(b) JMSMBean 的配置

```
/*OpenJMS 配置文件 (pkuas_jms.xml)片断 */
<ServerConfiguration host= "localhost "
        embeddedJNDI= "false "/>
//指定 PKUAS 提供的 JNDI服务.
<JndiConfiguration>
<property name= "java.naming.factory.initial "
value= "pku.as.naming.SmartCtxFactory "/>
</JndiConfiguration>
```

(c) JNDI 服务的集成

```
/*OpenJMS 配置文件 (pkuas_jms.xml)片断 */
<DatabaseConfiguration>//  指定 MySQL 数据库.
<RdbmsDatabaseConfiguration
driver= "org.gjt.mm.mysql.Driver "
url= "jdbc:mysql://localhost/pkuas "
user= "openjms "password=" openjms "/>
</DatabaseConfiguration>
```

(d) 数据库的集成

图 6　集成 OpenJMS 关键步骤

PKUAS 的 JNDI 服务.这一工作可通过修改 OpenJMS 的配置文件实现,如图 6(c)所示.

(3)集成数据库:OpenJMS 通过将消息存储到数据库来实现消息的持久性.因此,需要将 PKUAS 使用的数据库配置成 OpenJMS 使用的数据库.如果 PKUAS 使用的数据库是 MySQL,则 OpenJMS 的配置文件需要做如图 6(d)所示的修改.

5 相关工作

众多中间件产品大多提供丰富的构件和灵活的定制能力,如 IBM Websphere Application Server[10]和 BEA WebLogic[11],而 Sun ONE[12]在产品中内置了针对不同领域实现的构件集合,允许用户通过定制形成领域特定的中间件.这种方案中的可定制构件由厂商提供,不能从根本上支持领域特性.某些产品提供了有限的扩展能力,如,OpenEJB[13]意识到容器必须是可替换的,因此利用统一接口将容器与底层服务器分离,支持容器的增加、删除和替换,但 OpenEJB 仅支持安全与事务两种基本服务以及简单的 TCP/IP 通信机制,不允许用户定制和扩展;Jboss[14]考虑到服务的可扩展性,将服务和容器封装成 MBean,在容器中提供基于截取器的元编程机制,而分布通信机制与容器绑定在一起,导致两者均无法定制和扩展;iPortal[15]间接利用 CORBA 的可扩展传输协议规范实现了面向 IIOP 的传输协议扩展机制,却无法支持其它互操作协议的扩展,其内置的服务和容器可定制但不可扩展.

6 结束语

现有主流构件运行支撑技术难以有效支持领域特性(单个应用领域内的共性问题)的抽象及其支撑机制的封装,主要原因在于缺乏足够的开放性与灵活性,而这正是下一代中间件必须具备的特性.本文介绍了一个面向领域的构件运行支撑平台 PKUAS,在深入分析现有分布中间件技术体系的基础上,抽取出一个内核以松散平台内部各个系统构件之间的紧耦合关系,定义了一个开放的互操作框架,简化互操作协议或传输协议的扩展.为了提高平台的适应能力并解决服务的动态调用问题,引入了基于截取器和可编程客户桩的元编程机制.

目前 PKUAS 的缺省配置是一个 J2EE 应用服务器:支持三种标准 EJB 容器,包括无态会话容器、有态会话容器和 BMP (Bean Managed Persistent)实体容器,并支持 EJB Remote 接口和 EJB Local 接口;提供 RMI-IIOP、RMI-SOAP、RMI-Collocated 互操作机制,以及命名服务、安全服务、事务服务、日志服务、数据库连接服务;已通过了 J2EE 基准程序 JPS 1.2 的测试.进一步的工作主要包括:通过单个或多个构件的热部署提高应用的在线演化能力;与基于软件体系结构的建模工具 ABCTool 集成;通过反射机制提高平台的动态适应能力;实现电力领域特定的构件运行支撑平台的定制与扩展方案.

参考文献:

[1] W Emmerich. Software engineering and middleware: A roadmap[A]. ICSE-Future of SE Track 2000 [C]. 2000. 117 - 129.

[2] OMG. Common Object Request Broker Architecture Specification[S]. v2.6, http://www.omg.org, 2001.

[3] SUN Microsystem. Enterprise JavaBeans Specification [S]. Version 2.0, Final Release, http://java.sun.com/j2ee, 2001.

[4] Hong Mei, Jichuan Chang, Fuqing Yang. Software component composition based on ADL and middleware[J]. Science in China(F), 2001, 44(2): 136 - 151.

[5] SUN Microsystem. Java Management Extensions Instrumentation and Agent Specification [S]. v1.0, http://java.sun.com/products/Java-Management, 2000.

[6] C O'Ryan, F Kuhns, D C Schmidt, et al. The design and performance of a pluggable protocols framework for real-time distributed object computing middleware[A]. IFIP/ACM Middleware 2000 Conference[C]. Pallisades, New York, 2000. 372 - 395.

[7] N Wang, D C Schmidt, et al. Evaluating meta-programming mechanisms for ORB middleware[J]. IEEE Communications Magazine special issue on Evolving Communications Software: Techniques and Technologies, 2001, 39(10): 102 - 113.

[8] SUN Microsystem. Java Message Service Specification [S/OL]. Version 1.0.2, Final Release, http://java.sun.com/products/jms/docs.html, 1999.

[9] Jim Alateras, Tim Anderson, Jim Mourikis. OpenJMS user guide[Z/OL]. http://www.exolab.org/openjms, 2002.

[10] IBM webSphere [Z/OL]. http://www-4.ibm.com/software/webservers/appserv/.

[11] BEA webLogic [Z/OL]. http://www.bea.com/products/weblogic/server/.

[12] Sun open net environment (ONE) [Z]. http://wwws.sun.com/software/sunone/index.html.

[13] OpenEJB[Z/OL]. http://www.openejb.org.

[14] Jboss[Z/OL]. http://www.jboss.org.

[15] IONA iPortal [Z/OL]. http://www.iona.com/products/ip_ipas_home.htm.

作者简介:

黄 罡 男,1975 年 10 月生于湖南株州,北京大学软件研究所博士研究生,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.E-mail: huanggang@cs.pku.edu.cn.

王千祥 男,1970 年 2 月出生于山东莱州,博士,北京大学软件研究所副教授.主要研究领域为软件工程,网络计算环境.Email: wqx@cs.pku.edu.cn.