基于软件体系结构的反射式中间件研究*

黄 罡+, 王千祥, 梅 宏, 杨芙清

(北京大学 信息科学技术学院 软件研究所,北京 100871)

Research on Architecture-Based Reflective Middleware

HUANG Gang+, WANG Qian-Xiang, MEI Hong, YANG Fu-Qing

(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: 86-10-62757801 ext 1, Fax: 86-10-62757801 ext 8, E-mail: huanggang@cs.pku.edu.cn

http://www.sei.pku.edu.cn/belljointlab

Received 2002-11-04; Accepted 2003-04-16

Huang G, Wang QX, Mei H, Yang FQ. Research on architecture-based reflective middleware. *Journal of Software*, 2003,14(11):1819~1826.

http://www.jos.org.cn/1000-9825/14/1819.htm

Abstract: As Internet provides an open and dynamic runtime environment for distributed applications, one of the most important challenges to the next generation of middleware is how to allow the observation and manipulation of the runtime states and behaviors internal of middleware platform. Reflective middleware is able to open up the platform implementation to meet such requirements. There are three limitations in current research and experiments on reflective middleware: they focus on parts or individual entity of a system, lacking a global view; they only address the reflection of the underlying platform, ignoring the reflection of the application; they mostly experiment on CORBA with less work on J2EE. The design and implementation of a reflective component operating platform, called PKUAS, is presented. Based on its componentized structure, PKUAS introduces software architecture as the global view of the whole reflective system. As a J2EE-compliant application server, PKUAS can reflect both the underlying platform and EJB components. Moreover, this paper demonstrates the usage of reflection with the PKUAS management tool and evaluates PKUAS through comparison with other reflective middlewares. The work presented in this paper addresses the aforementioned limitations and improves the applicability of reflective middleware.

Key words: component; middleware; reflection; software architecture; J2EE

摘 要: Internet 为分布应用提供了一种开放、动态的运行环境,这要求分布应用的主要基础设施中间件能够支持

* Supported by the National Natural Science Foundation of China under Grant Nos.60233010, 60125206, 60103001 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA113060 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB31200003 (国家重点基础研究发展规划(973)); the Major Project of Science and Technology Research of Ministry of Education of China under Grant No.MAJOR0214 (国家教育部科学技术研究重大项目)

第一作者简介: 黄罡(1975－),男,湖南常德人,博士生,主要研究领域为软件工程,分布计算.

运行时查看并调整平台内部状态和行为,由此产生了反射式中间件.目前的反射式中间件研究与实践存在3个不足:注重系统局部或单个实体的反射而缺乏全局视图;注重中间件平台内部功能的反射,而对上层应用的反射不够;集中于CORBA平台,而对J2EE平台的反射性研究较少.介绍了一个反射式的J2EE应用服务器PKUAS.基于构件化的平台内部体系结构,PKUAS引入软件体系结构作为全局视图以实现反射体系对系统整体的表示和控制,作为J2EE应用服务器,PKUAS可反射底层平台以及上层EJB构件.同时,以PKUAS实时监控工具为例,探讨了如何利用基于软件体系结构的反射体系管理整个系统,并给出了PKUAS与其他几种反射式中间件的比较.该工作有效地解决了现阶段反射式中间件研究的不足,提高了反射式中间件的实用性.

关键词: 构件;中间件;反射性;软件体系结构;J2EE

中图法分类号: TP311 **文献标识码**: A

作为分布应用开发与运行的主流平台,分布计算中间件屏蔽了底层环境的异构性和复杂性,允许分布应用以“黑盒”方式复用其中封装的支撑机制.但是,Internet的发展促使应用环境从封闭、静态转变为开放、动态,这就要求软件系统具有足够的灵活性,以适应开放、动态的运行环境.开放中间件平台内部细节,允许上层应用以“灰盒”甚至“白盒”方式复用中间件的功能,是中间件满足上层应用灵活性要求的有效途径之一.由此产生了新的研究热点——反射式中间件[1~6].

“反射性(reflection)”的概念首先在人工智能领域出现,然后被引入计算机的其他领域[7].将反射性引入中间件能够以可控的方式开放平台内部的实现,从而提高了中间件的定制能力和运行时的适应能力.反射式中间件就是一种能够通过与系统运行状态和行为具有因果关联(causal-connected)的系统自述(self-representation)来监测并调整系统状态和行为的中间件系统[5].目前,反射式中间件的研究与实验大多集中于CORBA,主要是因为CORBA提供了许多便于实现反射性的设施,如接口池、截取器等[8].OpenCorba是一种基于反射式语言NeoClasstalk的反射式CORBA,通过元类(meta-classes)将ORB内部特征分离并单独实现,从而允许系统在运行过程中监测并调整这些内部功能单元,如监视并调整远程对象调用的行为、在服务器进行类型检查等[2].dynamicTAO是实时ORB软件TAO的扩展,其反射能力体现在对系统服务的配置与重配置上,即通过配置管理器将特定的策略(如安全策略、调度策略等)及其实现机制关联起来[3,4].mChaRM是一种反射式的RMI分布调用机制,它将方法调用看成是通过一个逻辑信道传输的消息,在该逻辑信道上实现一个关联对象,以监测信道中传输的消息并动态增加一些额外的消息处理功能[5].

现有反射式中间件的研究与实践有3个不足:1) 注重系统局部或单个实体的反射,缺乏全局视图,增加了使用和管理反射式中间件的难度;2) 注重中间件平台内部功能的反射,对上层应用的反射不够;3) 往往集中于CORBA平台,较少考虑J2EE平台的反射性.本文介绍了一个支持反射机制的J2EE应用服务器PKUAS.基于构件化的平台内部体系结构,PKUAS引入软件体系结构作为全局视图以实现反射体系对系统整体的表示和控制.同时,充分利用EJB模型的特点增强对上层应用的反射.

1 PKUAS概述

PKUAS是一个符合J2EE规范的构件运行支撑平台,支持3种标准EJB容器,包括无态会话容器、有态会话容器和实体容器,并支持远程接口和本地接口,提供IIOP,JRMP,SOAP以及EJBLocal互操作机制,内置命名服务、安全服务、事务服务、日志服务、数据库连接服务;通过了J2EE蓝图程序JPS v1.1的测试[9].

为能够明确标识、访问和操纵系统中的计算实体,反射式中间件必须具备构件化的基础设施体系[6].如图1所示,基于Java虚拟机,PKUAS将平台自身的实体划分为如下4种类型.

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3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.

公共服务: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,必须在命名服务中加以注册.

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,部署工具既可热部署整个应用,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作出相应调整.

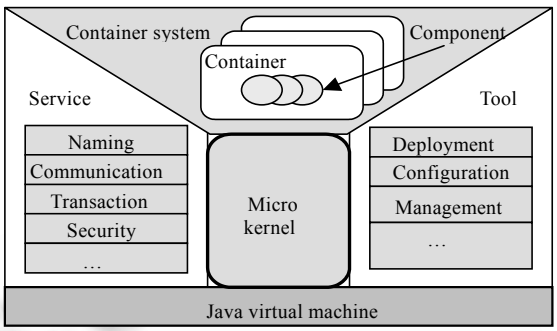

Fig.1 Componentized platform structure
图 1 构件化的平台结构

微内核:上述 3 类实体统称为系统构件,微内核负责这些系统构件的装载、配置、卸载以及启动、停止、挂起等状态管理.PKUAS 微内核符合 Java 平台管理标准 JMX(Java Management eXtensions),继承了 JMX 可移植、伸缩性强、易于集成其他管理方案、有效利用现有 Java 技术、可扩展等优点[10].其中,容器系统、服务、工具等被管理的系统构件组成资源层,通过 MBean 接口对外提供与管理相关的属性和操作.负责注册资源的 MBeanServer 和管理资源的插件组成管理层.MBeanServer 对外提供所有资源的管理接口,允许资源动态地增加或删除.管理插件则是执行其他管理功能的 MBean,如 PKUAS 实时监控管理工具的核心功能就是通过管理插件实现的.

2 PKUAS 反射体系

反射体系通常具有 3 个基本元素:元模型、元数据、元协议.元模型通过层次结构定义了系统中各个实体之间的反射关系,即上层实体反射下层实体.其中,最底层称为基层(base-level),基层中的计算实体实现了系统的业务功能.其他层次均称为元层(meta-level),元层中的计算实体实现了系统的反射功能.元层实体封装的数据称为元数据(meta-data),描述了系统的运行状态与行为.而元协议(meta-protocol)则是访问和操纵元数据的一组规则,具体表现为元层实体对外提供的操作.作为反射体系的核心,元数据决定了反射的内容.而元模型定义了内容的组织方式,元协议定义了内容的表现方式.

2.1 元数据

大多数反射式中间件的元数据侧重于单个实体接口的描述,忽略了系统的整体描述,而这种整体信息对管理和演化系统尤为重要[5].软件体系结构(software architecture,简称 SA)是对系统整体结构设计的刻画,包括全局组织与控制结构、构件间通信、同步和数据访问的协议、设计元素间的功能分配、物理分布、设计元素集成、伸缩性和性能、设计选择等[11].显然,如果将 SA 描述的信息导出并转换为元数据,应能弥补目前反射式中间件整体描述能力的不足.因此,PKUAS 的元数据是由设计阶段的 SA、部署阶段的部署信息以及运行阶段的上下文导出并累积而成,如图 2 所示.

软件体系结构:文献[12]介绍了描述风格规约、构件规约、连接子规约和体系规约这 4 种规约的软件体系结构描述语言 ABC/ADL.体系结构风格是一组构件和连接子模板的词汇表、一组这些构件和连接子模板如何连接的约束以及用来确定系统整体属性的语义模型的集合.构件是系统中的功能单元和计算实体,而构件规约包括构件模板规约(构件必须是基于体系结构风格中的构件模板)、属性规约(描述构件的版本、安全和负载等额外的特性)、接口规约(描述构件的接口信息)和内部结构规约(用来定义复合构件,即由多个构件和连接子组成的构件).连接子描述了构件之间的交互关系,连接子规约的内容与构件规约类似.体系结构规约描述了系统的具体配置,由构件实例、连接子实例和配置声明组成.ABC/ADL 借鉴面向对象中类与对象的关系区分了构件

定义和构件实例、连接子和连接子实例.配置声明表示了运行时刻实例之间的拓扑结构.

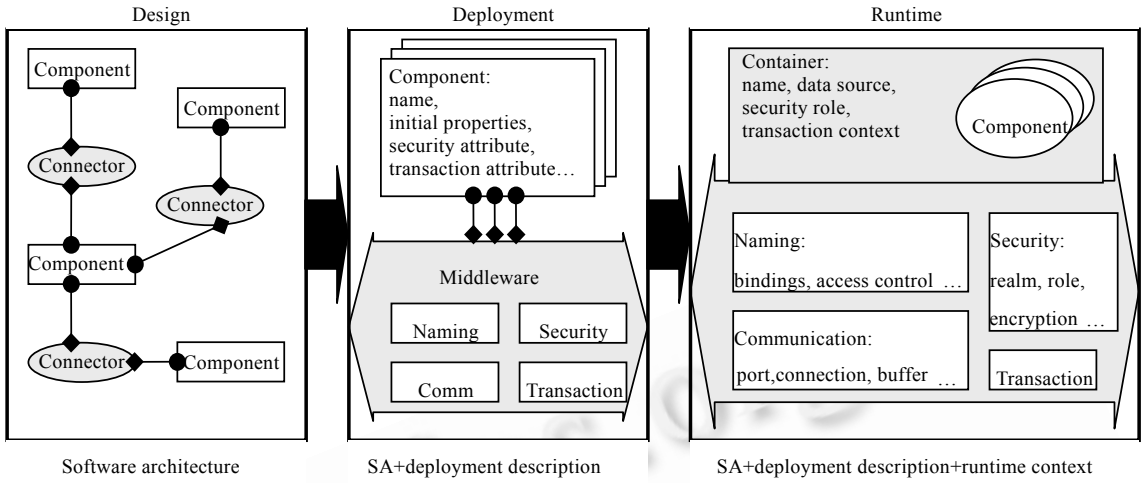

Fig.2 Aggregation and transformation of metadata in PKUAS

图 2 PKUAS 元数据的累积与转换

部署信息:EJB 规范定义的部署信息,允许构件提供商、应用集成商和部署者声明那些部署工具需要却没有包含在 EJB 代码中的信息[13,14].其中,构件提供商负责提供一个或多个可用构件,必须在部署信息中声明构件的名字、构件的文本描述信息、构件的实现类、构件的接口名、构件类型、构件的可重入性、构件的事务类型(由构件自己管理还是容器管理事务)、持久属性(如容器管理的或构件自己管理的持久性、主码、容器管理的持久域、容器管理的持久关联)、构件的环境变量入口、构件所需资源的引用、构件依赖其他构件的引用、安全角色的引用等.应用集成商负责将来自不同提供商的所有构件组装成目标应用,可修改构件提供商声明的构件的文本描述信息、构件的环境变量入口、构件依赖其他构件的引用等,同时,应用集成商还可以在部署信息中声明构件引用的绑定、安全角色及其与构件提供商声明的安全角色引用之间的关联、安全标识(使用调用者的主体或使用缺省的安全主体)、构件方法的许可、构件的事务属性(如不使用事务、优先使用客户端事务上下文、优先使用服务器端事务上下文、若有事务就报错)等.部署者负责将构件提供商或应用集成商提供的 1 到多个构件部署到选定的 EJB 服务器中,因而在必要时可根据 EJB 服务器具体实现的特征修改或补充部署信息,如确保所有构件引用有效、资源可用等.

运行上下文:平台的配置信息(如安全、事务、名字、通信等服务的配置信息)、资源的使用情况(如数据库连接池的总容量和可用容量、传输协议使用的端口、线程并发数量)、构件运行过程中累积的信息(如构件实例事务上下文的状态、构件被访问次数的统计、构件产生例外的统计、构件实例的数量等)、服务运行过程中累积的信息(如名字服务中所有的名字绑定、安全服务产生的审计信息、服务产生的例外统计、通信服务中特定端口接收和发送的数据量统计)组成了系统的运行上下文.

上述 3 个阶段产生的元数据之间均存在冗余.对于冗余或冲突的数据,应由后阶段产生的信息覆盖前阶段.这种覆盖关系保证了运行阶段的元数据能够尽可能完整、精确地描述运行系统的状态和行为.实质上,部署信息和运行上下文均可看成部署阶段和运行阶段对设计阶段 SA 的精化,因此,PKUAS 的元数据本质上是运行系统软件体系结构的精确描述,而 PKUAS 的元模型和元协议的设计也以适合软件体系结构为基准,因此,PKUAS 是一种基于软件体系结构的反射式中间件.

2.2 元模型

现有反射式中间件大多将上层应用的计算实体视为基层实体,而将中间件平台自身的计算实体(如各种服务)视为元层实体[2,3,5].但是,中间件封装的分布系统开发过程中遇到的共性问题及其解决机制也属于分布系统

的业务功能,如事务服务、安全服务、通信功能等.换言之,按照基层实体的定义,中间件平台内部实体也应属于基层.从软件体系结构来看,服务和组成上层应用的 EJB 都是构件,通信服务提供的各种通信原语则是连接子,如远程方法调用、本地调用、消息通信等.在如图 3 所示的 PKUAS 元模型中:

基层:由构件和服务组成.这些基层实体实现了基于 PKUAS 应用系统的核心业务功能.

元层:各种容器和各种服务 MBean 是直接反射基层实体的元层实体,管理部署信息和运行上下文导出的元数据,并维护这些元数据与基层实体状态和行为的因果关联.如容器实现构件元数据与构件生命周期的因果关联;名字服务 MBean 允许用户浏览名字服务中存储的所有信息,或设置给定信息的安全机制(如只有管理员用户才能修改和删除信息);而通信服务 MBean 则可显示系统支持的互操作协议(如 JRMP、IIOP、SOAP),传输协议(如 TCP/IP、HTTP、SSL),目前网络资源状况(使用端口、通信缓存、连接并发使用的线程数量)或增加和删除某种协议、调整网络资源等.容器系统 MBean 负责反射各种容器,主要维护软件体系结构导出的元数据.而服务器 MBean 是反射容器系统 MBean 和服务 MBean 的顶级元层实体.目前 PKUAS 反射体系主要作用于单个服务器运行实例,因此,服务器 MBean 仅作为整个 PKUAS 反射体系的入口,在今后考虑多个节点上运行的多个服务器实例时,服务器 MBean 的功能会有所增强.

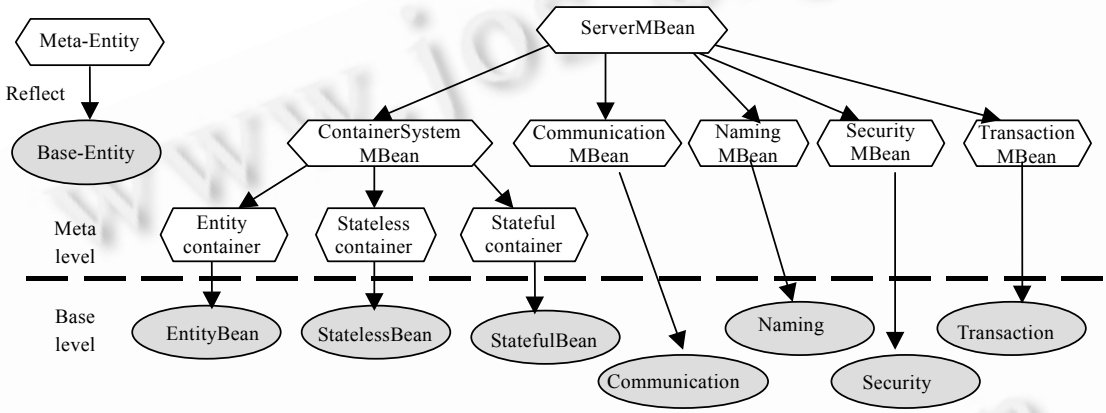

Fig.3 Meta model of PKUAS

图 3 PKUAS 元模型

2.3 元协议

PKUAS 反射元协议可以从基层实体的类型和因果关系的实现划分成 4 种类型,见表 1.

Table 1 Types of meta-protocol in PKUAS

表 1 PKUAS 元协议分类

	Reflection of service	Reflection of component
Structural reflection	Enumeration, addition, removal and replacement	Enumeration, addition, removal and replacement
Behavioral reflection	Dynamic configuration	Change of status for a given instance

2.3.1 服务的反射元协议

服务 MBean 负责服务元数据的访问和修改,并执行装载、启动、停止、释放、替换、动态配置等动作.

服务的结构性反射:即装载、卸载或替换一个 MBean,可通过 JMX 定义的标准接口 javax.management.MbeanServer 提供的 registerMBean( )和 unregisterMBean( )实现[10].

服务的行为性反射:服务配置信息的获取和修改可通过 MBean 提供的 getMetadata( )和 setMetadata( )两种标准操作完成.如 setMetadata("state","start|stop|release")将启动、停止或释放指定 MBean 反射的服务;getMetadata("whole_metadata")返回指定服务的所有元数据;getMetadata("implementation_class")查看服务的实现类;setMetadata("implementation_class","pku.as.naming.NamingService")装载指定的名字服务,如果随后某时刻再次执行该操作,且第 2 个参数指向另一个实现类,名字服务 MBean 就会替换先前的名字服务;NamingMBean.getMetadata("whole_bindings")返回名字服务中所有注册的名字及其值;NamingMBean.

SetMetadata(“authentication”,“true|false”)设置名字服务是否需要对客户进行认证.

2.3.2 构件的反射元协议

容器负责构件元数据的访问和修改,并维护元数据与构件状态和行为的因果关联.PKUAS 构件反射元协议目前关注于在线替换单个构件.构件在线替换的原则是:正在运行过程中的实例不能替换,即实例处于空闲态时可替换;一次事务中的构件实例不能替换(一次事务往往包含对同一个构件实例的一系列调用,而事务的开始和结束均由构件实现或客户控制),即当实例的事务对象为空时可以替换[15].如图 4 所示为一个有态会话构件的在线演化流程(其他类型构件的在线演化与之类似).当容器接收到 setMetadata(“implementation_class”,“…”)的构件替换指令时,首先阻塞请求处理接口对实例管理器 getInstance( )的调用,即暂时不允许任何空闲实例队列中的实例变为运行态;然后将空闲实例队列中所有事务对象为空的构件实例替换成新实现类的实例;随即释放所有阻塞的 getInstance( )调用.此时没有替换的实例将在 free( )中陆续完成,即在实例从运行态转变为空闲态时,若该实例需要替换且事务对象为空,则替换新实现后再进入空闲实例队列.因此,所有旧构件实例的事务全部结束标志着在线演化的完成.

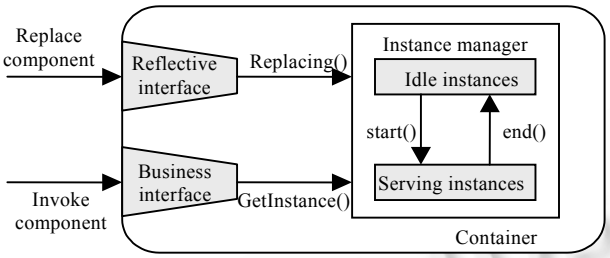

Fig.4 Online replacement of a stateful EJB

图 4 有态会话 EJB 的在线替换

3 实例研究

图 5 为 PKUAS 实时监控管理工具的浏览器界面.该工具通过一个管理层 MBean 查询和控制 PKUAS 反射体系.左半部分的树型列表显示服务器中已部署的应用、待部署的应用、安装的服务等,此时选定了一个已部

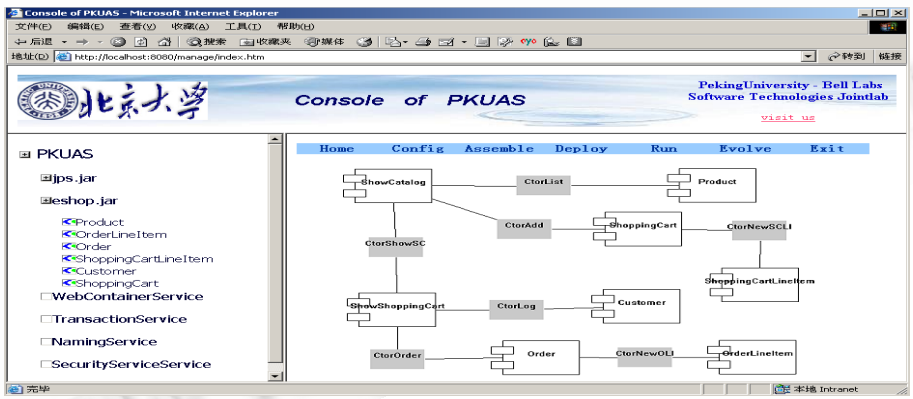

Fig.5 Representation of software architecture of eShop

图 5 网上商店的软件体系结构显示

署,名为 eShop 的简单的网上电脑配件销售系统;右半部分上方罗列了主要的应用控制指令,如 Home(返回管理工具主页)、Config(配置应用支持的互操作协议、端口等)、Assemble(组装应用)、Deploy(部署应用)、Run(控制应用生命周期,如启动、挂起、恢复、停止等)、Evolve(演化单个构件)、Exit(释放反射选定应用占用的资源),右半部分下方则显示了 eShop 的软件体系结构.目前只能以位图图像显示采用 ABC/ADL 工具设计的 SA,仅允许通过左边的树型列表监控单个构件.PKUAS 与 ABC/ADL 的集成工作还在进行中,将支持直接对 SA 的各种图元进行操作.在图 6 中选定了名字服务,其详细信息包括名字服务的实现类、名字服务的 URL、名字服务是否使用安全认证机制、采用何种安全协议(如 IIOP-SSL)、采用何种认证机制以及名字服务中存储的所有绑定的名字及其值类型(如 EJB 的远程引用是 IOR,JDBC 等资源的引用是 Java 本地引用).

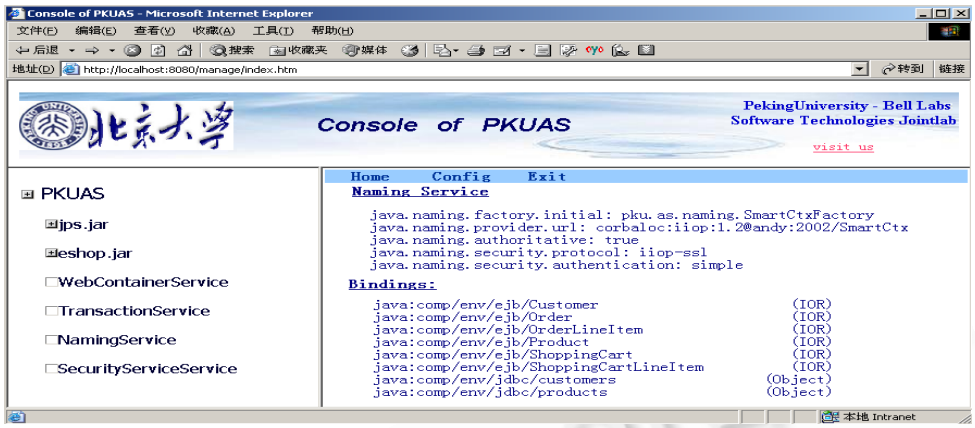

Fig.6 Management of naming service based on reflection

图 6 基于反射性的命名服务管理

4 相关工作

文献[6]提出了一个反射式中间件基准框架.我们针对其中的6条准则比较了PKUAS与其他反射式中间件,见表2.

Table 2 Comparison results between PKUAS and other reflective middlewares

表 2 PKUAS 与几种反射式中间件的比较

Benchmark	OpenCORBA	dynamicTAO	mChaRM	PKUAS
Componentized platform structure	Not explicitly defined	Micro kernel	Independent communication module	Micro kernel
Location of middleware	Meta-Level	Meta-Level	Meta-Level	Base-Level
Granularity	Service	Service	Communication channel	Component and service
Pervasiveness	Several aspects	Any aspects	Communication aspect	Any aspects
Safety and security	Not defined	ORB Consistency management and access control	Consistency related to communication channel	Architectural consistency and access control
Meta-Level complexity	Separated reflection of aspects	Separated reflection of aspects	Not defined	Architectural reflection of all aspects

其中: 1) 构件化平台体系:平台包含的服务和构件在运行时可标识,这是实现良好反射性的基础;2) 元模型层次划分:必须清晰、严格划分元模型的层次,其中的关键是中间件处于元层或基层;3) 反射粒度:是否能够反射构件、服务或整个系统;4) 普适性:是否可反射中间件的各种视图(如编码、同步、传输协议、安全等);5) 反射的安全性:个体或局部的反射可能会影响整个系统的正确性和一致性,某些重要的反射还需要进行访问控制等保护措施;6) 元模型的复杂度管理:元层及其实体增加了整个系统的复杂度,需要简单、有效的管理元模型的方法.由于 PKUAS 基于构件化的平台体系结构,引入软件体系结构作为全局视图,从而实现了有别于其他反射式中间件的特点,包括中间件属于基层、反射粒度良好、普适性强、安全性优、元模型具有统一的管理等.

5 结束语

中间件通过“黑盒”复用的方式屏蔽了分布环境中操作系统、网络、编程语言的异构性,简化了分布应用的开发和管理.与封闭、静态的传统运行环境相比,Internet 为软件系统提供了一个开放、动态的运行环境.如何提供足够的灵活性以适应这种开放、动态的运行环境,是下一代中间件的主要目标之一,而开放平台内部细节的反射式中间件是切实可行的途径.本文介绍了一个反射式的构件运行支撑平台 PKUAS.首先讨论了中间件平台内部功能的构件化方法,介绍了基于微内核的 PKUAS 平台体系.以软件体系结构隐含的信息为核心的反射元数据,弥补了现有反射式中间件对系统整体结构的元数据描述能力的不足.结合 J2EE 服务器的特征设计了等同

对待平台功能和上层应用的反射元模型,设计实现了访问和操纵元数据的反射元协议,利用 J2EE 服务器定义的 EJB 构件模型的特点,增强了中间件对构件的反射能力.以 PKUAS 实时监控工具为例,探讨了如何利用反射体系管理整个系统.最后,基于一个反射式中间件基准框架给出了 PKUAS 与其他几种反射式中间件的比较.

今后的工作重点包括:反射元数据的标准化定义以及反射元协议的丰富;在实时监控工具中集成 ABC/ADL 图形化工具模块[12],以支持直接操纵软件体系结构中的各种图元;整个反射体系的性能优化.

References:

[1] Coulson G. What is reflective middleware? Distributed Systems Online Journal, IEEE Computer Society, http://boole.computer.org/dsonline/middleware/RMarticle1.htm, 2000.

[2] Ledoux T. OpenCorba: A reflective open broker. In: Cointe P, ed. Proceedings of the 2nd International Conference on Reflection. LNCS 1616, Heidelberg: Springer-Verlag, 1999. 197~214.

[3] Kon F, Román M, Liu P, Mao J, Yamane T, Magalhães LC, Campbell RH. Monitoring, security, and dynamic configuration with the dynamicTAO reflective ORB. In: Sventek JS, Coulson G, eds. IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing. Vol 30. Heidelberg: Springer-Verlag, 2000. 121~143.

[4] Klefstad R, Schmidt DC, O'Ryan C. Towards highly configurable real-time object request brokers. In: Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. IEEE Computer Society, 2002. 347~447. http://computer.org/proceedings/isorc/1558/1558toc.htm.

[5] Cazzola W. Communication-Oriented reflection: A way to open up the RMI mechanism [Ph.D. Thesis]. Milano: Università degli Studi di Milano, 2001.

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

[7] Demers FN, Malenfant J. Reflection in logic, functional and object-oriented programming: A short comparative study. In: Proceedings of the Workshop on Reflection and Meta-level Architectures and Their Applications in AI (IJCAI'95). 1995. 29~38. http://citeseer.nj.nec.com/106401.html.

[8] Object Management Group. The common object request broker: architecture and specification, revision 2.6. 2001. http://www.omg.org/technology/documents.

[9] Huang G, Wang QX, Cao DG, Mei H. PKUAS: A domain-oriented component operating platform. Acta Electronica Sinica, 2002,30(12A):39~43 (in Chinese with English abstract).

[10] SUN Microsystems. Java Management Extensions Instrumentation and Agent Specification, V1.0. 2000. http://java.sun.com/products/JavaManagement.

[11] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall, 1996.

[12] Mei H, Chang JC, Yang FQ. Software component composition based on ADL and middleware. Science in China(F), 2001,44(2): 136~151.

[13] SUN Microsystems. Java 2 Platform Enterprise Edition Specification, Version 1.3, Proposed Final Draft 4. 2001. http://java.sun.com/j2ee.

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

[15] Wang QX, Chen F, Mei H, Yang FQ. Using application server to support online evolution. In: Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2002). IEEE Computer Society, 2002. 131~140.

附中文参考文献:

[9] 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.电子学报,2002,30(12A):39~43.