StarOSGi：一种OSGi分布式扩展中间件

史殿习[1] 吴元立[2] 丁 博[1] 尹 刚[1] 王怀民[1]
（国防科技大学计算机学院 长沙 410073）[1] （解放军第309医院信息科 北京 100091）[2]

摘 要 随着应用范畴从单个结点扩展到普适计算、企业计算等分布式环境，OSGi技术需要提供对远程服务访问的支持。在保留其面向服务、动态性、轻量级等已有优点的前提下，如何为OSGi技术体系提供有效的分布式扩展，是研究者所面临的重要挑战。现有OSGi分布式扩展研究工作存在着对编程模型具有明显侵入性、不支持与非OSGi系统互操作等共性问题。以CORBA中间件技术为基础，以非侵入性、通用性和良好互操作性为目标，提出了基于CORBA的OSGi分布式扩展模型，进而基于CORBA动态调用和Java反射技术设计了一个支持OSGi分布式扩展的中间件StarOSGi。该中间件保持了OSGi原有面向服务的编程模型和轻量级特点，能够透明地将集中式的OSGi应用转变为分布式应用，并且支持OSGi应用与遗留CORBA应用的互操作，具有明显的性能优势。

关键词 OSGi，服务，分布式扩展，中间件
中图法分类号 TP311 **文献标识码** A

StarOSGi：A Distributed Extension Middleware for OSGi

SHI Dian-xi[1] WU Yuan-li[2] DING Bo[1] YIN Gang[1] WANG Huai-min[1]
(School of Computer, National University of Defense Technology, Changsha 410073, China)[1]
(Department of Information, The 309th Hospital of Chinese PLA, Beijing 100091, China)[2]

Abstract As the application domain has been expanded from a single node to distributed environments such as pervasive computing and enterprise computing, the OSGi technology should provide support for remote service access. On the premise of keeping the existing advantages such as service-oriented, dynamics and light-weight, it is a great challenge for the researchers to provide effective distributed extension for the OSGi technology. Existing works in this domain has a set of deficiencies, such as the invasiveness to original OSGi programming model and lacking the support of interoperating with non-OSGi systems. This paper proposed a CORBA-based Distributed OSGi Model, which chooses CORBA middleware technology as a foundation and aims at non-invasiveness, generality and interoperation. Based on this model, this paper designed and implemented an OSGi distributed extension middleware-StarOSGi, which implements the remote service invocation capability with CORBA dynamic invocation and Java reflection technology. It can turn the central OSGi applications into distributed environment transparently while keeping the service-oriented programming model and lightweight feature of OSGi. Furthermore, StarOSGi supports the interoperation between OSGi and CORBA applications and also has performance advantages over existing works.

Keywords OSGi, Service-oreinted, Distributed extension, Middleware

动态化、模块化、面向服务的优势使得OSGi(Open Services Gateway Initiative)规范[1]在各个领域得到了大量应用，包括Eclipse[2]，Websphere[3]，microServices[4]，JBoss AS 5.0[5]以及Spring Application Platform[6]等在内的大量软件工程项目都采用了基于OSGi的架构。然而，随着OSGi技术应用范围的扩展，其本身缺乏对分布计算支持的局限性也逐渐显现出来，这主要是因为OSGi技术最初定位于服务网关领域，只为单个JVM(Java Virtual Machine)内的Java应用提供了一个动态的模块化系统。如何对OSGi进行分布式扩展，使其具有支持分布式处理的能力，进而有效地支持异构的、大规模的分布式计算应用，成为OSGi进一步发展亟需解决的问题。

针对这一问题，OSGi联盟制定了OSGi分布式扩展规范，即RFC 119[7]规范，近年来学术界也在OSGi分布式扩展技术相关领域展开了相关研究，并取得了Newton[8]，R-OSGi[9]以及D-OSGi(Distributed OSGi)[10]等一系列具有代表性的研究成果。这些成果使用不同的技术和方式对现有OSGi进行分布式扩展，使其具有支持分布式处理的能力。然而，现

到稿日期：2010-02-26 返修日期：2010-05-28 本文受国家核高基重大专项课题(2009ZX01043-001)，863国家重点课题(2007AA010301)资助。
史殿习(1966－)，男，博士，副研究员，CCF会员，主要研究方向为分布计算、中间件、普适计算等，E-mail：dxshi@nudt.edu.cn；吴元立(1985－)，男，硕士生，主要研究方向为分布计算；丁 博(1978－)，男，博士生，主要研究方向为分布计算、普适计算等；尹 刚(1975－)，男，博士，助理研究员，CCF会员，主要研究方向为分布计算、中间件、信息安全等；王怀民(1964－)，男，博士，教授，CCF会员，主要研究方向为分布计算、网络安全等。

有工作仍然处于探索阶段，其不足集中表现为：对 OSGi 编程模型存在一定的侵入性，底层分布式实现技术会影响分布式 OSGi 应用的开发方式；不支持与非 OSGi 系统的互操作，特别是与企业计算领域大量遗留系统的互操作；在一定程度上破坏了 OSGi 的最小执行环境，限制了分布式 OSGi 应用的适用领域。

CORBA[11]是国际对象管理组织 OMG 制定的分布对象计算规范，其具有独立于平台、独立于操作系统以及独立于编程语言等特性，遵循 CORBA 规范的 CORBA 中间件如 Orbix[12]，StarBus[13]等已被广泛应用于电信、金融、交通、国防等领域。针对目前现有的 OSGi 分布式扩展研究中存在的问题，我们以 OSGi 分布式扩展规范 RFC 119 为基础，以非侵入性、通用性和良好互操作性为目标提出了基于 CORBA 的 OSGi 分布式扩展模型，进而设计实现了基于 StarBus 的 OSGi 分布式扩展中间件 StarOSGi。StarOSGi 的特点是采用 CORBA 的 DII/DSI（Dynamic Invocation Interface/Dynamic Server Interface）技术和 Java 反射技术实现远程服务方法调用，能够透明地将集中式的 OSGi 应用转变为分布式应用，既保持了 OSGi 原有面向服务的编程模型和轻量级特点，又支持了 OSGi 应用与 CORBA 应用的互操作。

本文第 2 节对相关工作进行了研究和分析；第 3 节重点描述了基于 CORBA 的 OSGi 分布式扩展模型；第 4 节描述了 StarOSGi 的设计与实现，重点描述了其分布式扩展机制及其实现原理；第 5 节给出了基于 StarOSGi 的应用实例，同时与 D-OSGi 进行了性能对比；最后，对全文进行了小结，指出了需要进一步研究的工作。

1 相关工作

OSGi 规范是由 OSGi 联盟最初针对嵌入式领域提出的一个开放的、管理本地网络设备服务的规范，本质上是一种面向服务的动态模块化系统。在 OSGi 中，模块被称为 Bundle[14]，是能够对外提供服务的构件，由普通的 JAR[15]文件加上额外的元信息描述构成的。OSGi 规范明确定义了 Bundle 的边界、Bundle 之间的交互方式以及 Bundle 的动态部署方式等。OSGi 框架为 Bundle 提供了一个标准执行环境，并且能够为 Bundle 及 Bundle 中的服务提供动态的生命周期管理。

OSGi 分布式扩展是指将分布式计算技术与 OSGi 框架融合，使之能够支持跨多虚拟机的异构分布式处理。目前已有的一些代表性成果包括 Newton，R-OSGi 以及 D-OSGi 等。

Newton 借鉴了 Spring 的分离关注点思想，即将领域模型和基础设施分离，以提供一个轻量级分布式构件开发模型。Jini[16]则是其远程基础设施的基石。同时，Newton 使用 SCA（Service Component Architecture）[17]来描述构件装配模型，并且提供了动态的 SCA 实现，以加载和管理部署在分布不同 JVM 中的 SCA 构件。但是 Newton 的构件模型并不完全遵循 OSGi 规范，Jini 的服务描述方式与 OSGi 的服务描述并不一致，因而是一种侵入式最强的 OSGi 分布式扩展方式。

R-OSGi 是 EclipseCon2007 大会上发布的实现 OSGi 分布式处理的项目。R-OSGi 遵循 OSGi 规范，使用对接口进行字节码分析的方式来动态产生服务代理 Bundle，以实现远程服务的透明访问，并使用 SLP（Service Location Protocol）[18]实现远程服务发现。但是 R-OSGi 采用了基于消息的特定的远程互操作协议，无法满足与基于标准互操作协议（如 IIOP，SOAP）的非 OSGi 系统交互的需求。同时，R-OSGi 在服务使用上不透明，服务消费者要实现 R-OSGi 自定义的 DiscoveryListener 接口，对 OSGi 编程模型有一定侵入性。

D-OSGi 是 RFC 119 规范的一个参考实现。D-OSGi 使用 Web Service 来实现 OSGi 分布式扩展，使用基于 HTTP 的 SOAP 协议来实现远程服务的方法调用，并通过 WSDL 来发布服务。但是基于文本的 SOAP 协议会带来较大的性能开销，且 D-OSGi 目前采用静态 XML 配置文件的形式来实现远程服务发现，这不满足 OSGi 环境中对服务的动态性要求。

2 基于 CORBA 的 OSGi 分布式扩展模型

OSGi 联盟 RFC 119 规范是针对 OSGi 分布式扩展的规范，其目的是给出一个通用的解决方案以支持最基本的分布式处理特性和功能，包括服务发现和与外部环境的交互，并使得熟悉 OSGi 的开发人员不需要太多额外的学习就可以轻松地开发分布式 OSGi 应用。为有效支持分布式处理，RFC 119 规范对 OSGi 框架和编程模型进行了相应的改变，在 OSGi 框架中，增加了服务钩子[19]能力以达到对服务使用信息的监控；在编程模型上，增加远程服务属性标识和配置分布式应用所需要的元信息；在 OSGi 框架上引入了两个额外的系统级模块：DSW（Distribution Software）和 Discovery Service，其中，DSW 是一个软件实体，通过使用多种已有的分布式软件系统，使得 OSGi 平台能够支持对其他地址空间或其它机器服务的绑定和注入；Discovery Service 也是一个软件实体，通过使用多种已有的服务发现系统，使得 OSGi 平台能够支持跨地址空间、跨机器节点服务的注册和查找。

OSGi 分布式处理扩展的关键是结合 OSGi 框架特点，尽可能地保持原有的 OSGi 编程模型，采用成熟的分布式技术来实现远程服务的发现和调用，为集中式的 OSGi 框架透明地、无侵入式地增加分布式能力。CORBA 是一种成熟的分布式中间件技术，具有独立于平台、独立于操作系统和独立编程语言等特性。使用 CORBA 技术来实现 OSGi 的远程服务调用不仅能够支持分布在网络中不同机器节点的 OSGi 应用之间的远程服务调用，同时由于其底层采用国际标准 IIOP 协议，也支持与企业计算领域中大量 CORBA 遗留系统的互操作。因此，针对现有 OSGi 分布式扩展研究方面存在的不足，本文使用 CORBA 技术作为 OSGi 分布式扩展的架构基础，以非侵入性、通用性和良好互操作性为目标提出了一种基于 CORBA 的 OSGi 分布式扩展模型——CDOM（CORBA-based Distributed OSGi Model），如图 1 所示。

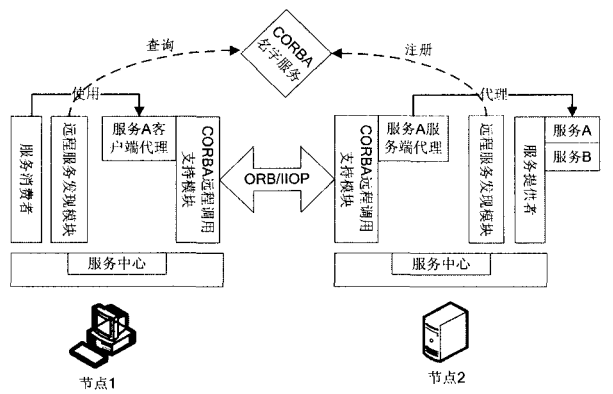


图 1 基于 CORBA 的 OSGi 分布式扩展模型

CDOM 模型的核心是 CORBA 远程调用支持模块 CORBA-DSW(CORBA-Distribution-Software)和远程服务发现模块 CORBA-NDS(CORBA-Naming-Discovery-Service)。

(1)CORBA-DSW 远程调用支持模块负责监控本地服务中心的服务注册、查询和注销情况,根据监控情况动态生成远程服务的服务端代理或客户端代理,服务端代理和客户端代理提供了基于 CORBA 的远程服务调用能力。服务端代理的作用是当接收到来自对象请求代理 ORB(Object Request Broker)的 CORBA 调用请求后,负责将该请求转化为 Java 调用请求,然后根据 Java 调用请求调用服务,并将服务的处理结果由 Java 形式再转变为 CORBA 形式后返回给 ORB。客户端代理的作用是实现位于其他节点上的远程服务的接口,将该接口中方法的 Java 调用请求转换成对远程服务的服务端代理的 CORBA 调用请求。客户端代理与服务端代理的生成、运行和终结是动态、透明的,其生存周期与其所关联的服务的生存周期相关。

(2)远程服务发现模块 CORBA-NDS 用于辅助 CORBA 远程调用支持模块为本地 OSGi 框架提供远程服务发现能力,用于支持远程服务的注册、查询、改变和注销。

3 StarOSGi 的设计与实现

基于第 2 节提出的 CDOM 模型,以我们自主开发的遵循 CORBA3.0 规范的分布式中间件 StarBus 为基础,本文设计并实现了一个支持 OSGi 分布式扩展处理的中间件 StarOSGi(StarBus-based Distributed OSGi System)。StarOSGi 通过对服务中心的监控、动态代理生成技术以及基于 CORBA 名字服务的远程发现服务来保持 OSGi 原有面向服务的编程模型,通过 StarBus 中实现的标准互操作协议 IIOP 来保持 StarOSGi 的领域通用性,并支持 OSGi 应用与 CORBA 应用的互操作。

3.1 StarOSGi 总体架构

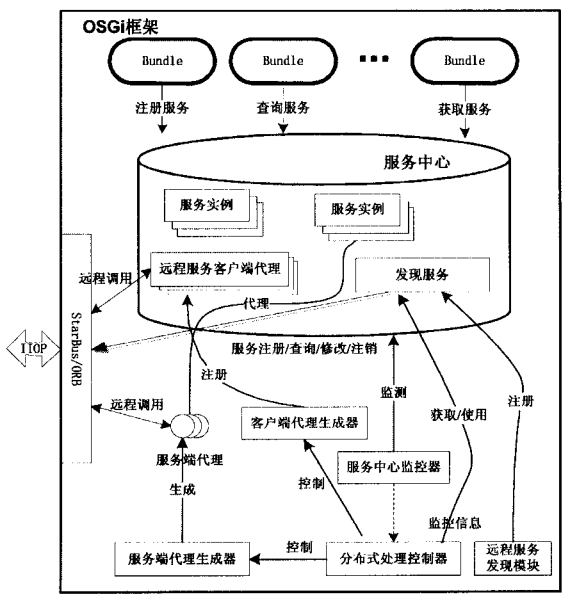


图 2 StarOSGi 对 OSGi 的扩展结构

StarOSGi 对 OSGi 的分布式扩展结构如图 2 所示。StarOSGi 对 OSGi 的分布式扩展主要包括远程调用支持模块和远程服务发现模块两个部分,其中远程调用支持模块进一步分为分布式处理控制器、StarBus/IIOP、服务中心监控器、客户端代理生成器和服务端代理生成器等几个部分。StarBus 中的对象请求代理 ORB 及标准的互操作协议 IIOP 是 StarOSGi 支持 Bundle 之间以及 Bundle 与非 OSGi 应用之间进行互联、互通及互操作的基础;StarOSGi 的核心是服务注册中心监控机制、服务端代理和客户端代理的动态生成机制以及远程服务发现机制等,这些机制之间相互协作完成服务的注册、发现以及代理的动态生成,从而完成应用之间的相互访问。

3.2 服务中心的监控机制

为了使集中式的分布式应用可以透明地转化为分布式应用,需要对本地 OSGi 平台的服务注册中心的服务注册和服务使用情况进行监控,以捕获远程服务注册事件或者对远程服务的使用需求这类服务信息,根据监控的结果来决定下一步的动作,如服务端或客户端代理的动态产生和远程服务查询等。StarOSGi 使用 ServiceTracker 和服务钩子机制来实现对本地 OSGi 服务中心的监控。从远程服务提供者的角度来讲,远程服务与普通的本地服务的唯一区别就是远程服务在服务注册时其属性标识为可被远程访问的,在 StarOSGi 里远程属性用"osgi.remote=true"标识,且符合 RFC 119 规范。StarOSGi 对于已注册的、正在注册的和以后将要注册的远程服务都要能够对其进行识别,以区分在本地服务中心注册的非远程服务,并为远程服务做进一步的服务端代理生成以及远程服务注册等处理。服务钩子提供对服务注册中心的服务查询和请求信息的监控能力,当监控到服务使用者查询的服务不在本地时,将激活远程服务发现,获取远程服务信息并创建本地的客户端代理,并把其注册到本地,本地服务注册中心再通知服务使用者其所需服务已满足。

3.3 代理动态生成机制

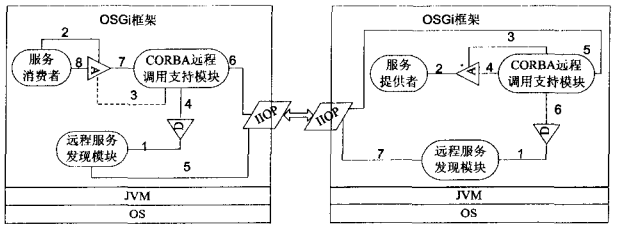
StarOSGi 通过动态生成代理的方式提供透明的远程服务使用,并支持与 CORBA 系统的互操作。Java 反射机制[20]以及 CORBA 的 DII/DSI 是实现动态代理生成的关键支撑技术。客户端代理是通过 Java 动态代理生成机制完成的,即利用 java.lang.reflect.Proxy 类与服务端代理所采用的 java.lang.reflect.InvocationHandler 类相结合来实现的,并在方法实现中通过 DII 机制来实现 Java 请求与 CORBA 请求的转换。服务端代理的动态生成是通过 CORBA DSI 机制及 CORBA 与 Java 的语言映射来实现的,其中 DSI 机制可以支持服务方动态地将客户方请求转发给具体的对象实现。

3.4 远程服务发现

OSGi 使用集中式的服务发现机制,需要远程服务发现机制来实现透明的分布式扩展,而远程服务注册与查询的时机取决于 StarOSGi 对服务注册中心的监控。StarOSGi 通过 CORBA 名字服务来实现远程的服务发现,CORBA 名字服务维护着服务元信息到服务端代理所对应的 CORBA 对象引用的映射关系。对开发人员来说,远程服务发现与本地的服务注册中心提供的查询能力看起来一样,实现了透明的远程服务发现。

3.5 远程服务的发布和使用

本节通过发布远程服务和使用远程服务两个方面来阐述 StarOSGi 实现原理,如图 3 所示,服务 A 是可被远程访问的服务,通过远程属性进行标识。



(a)使用远程服务　　(b)发布远程服务

图3　StarOSGi的实现及工作原理

服务提供者发布远程服务时，StarOSGi的工作原理如图3(b)所示。首先，远程服务发现模块向本地服务中心注册DiscoveryService服务(在图中用D表示)(1)，服务提供者向本地服务中心注册服务A(2)，并在注册时附带的服务属性中标明服务A可被远程访问，接着CORBA远程调用支持模块监控到属性标识为可被远程访问的服务A注册事件(3)，通过本地服务中心获取服务A(4)，为服务A生成服务端代理(5)，并进一步通过服务中心获取DiscoveryService(6)，调用DiscoveryService的服务注册方法将服务A的相关信息和为服务A生成的服务端代理位置信息注册到CORBA名字服务上(7)。

服务消费者使用远程服务时，StarOSGi的工作原理如图3(a)所示。首先，远程服务发现模块向本地服务中心注册服务DiscoveryService(1)，服务消费者通过ServiceTracker的方式来表示对服务A的使用请求(2)，CORBA远程调用支持模块通过服务中心监控器监控到服务消费者对服务A的使用请求(3)，然后通过服务中心获取DiscoveryService(4)，调用DiscoveryService的服务查询方法查询到服务A在网络中其他节点上存在，获取服务A服务端代理的位置信息(5)，接着根据服务A的服务端代理位置信息和服务A的接口信息为服务A动态生成服务A的客户端代理(6)，将客户端代理注册到本地服务中心(7)，最后，服务消费者通过本地服务中心获取服务A的客户端代理(8)。

4　应用实例与性能测试

StarOSGi在保持OSGi原有的编程模型和轻量级特点的基础上，可以将集中式的OSGi应用透明地转变为分布式应用，并支持OSGi应用与CORBA应用的互操作。为了验证StarOSGi的有效性和性能，我们在其上开发了体感鼠标的应用，同时将已有的RMI的标准测试集JavaParty/KaRMI[21]移植到OSGi框架上，在网络环境下对StarOSGi与D-OSGi做了性能对比测试。

4.1　应用实例

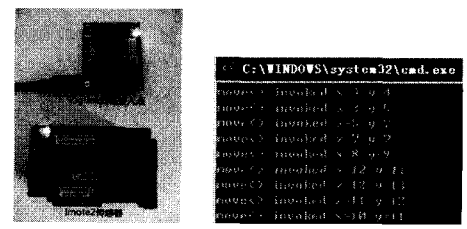
体感鼠标是指通过人的3D动作来控制电脑鼠标的光标移动，这样就可以不再局限于桌面鼠标的2D移动模式，而直接通过手势、肢体运动等来实现新型的人机交互。本实验通过Imote2传感器的重力感应功能实现了一款简易的体感鼠标。

Imote2传感器是一款先进的无线传感器节点平台，通过自身的重力传感器能够感知自身在空间中X，Y，Z轴方向上加速度的变化。本应用通过相应算法将物理世界中的Imote2传感器在角度发生偏移时的加速度信息转化为通过网络与之互联的PC机上鼠标光标的移动，从而实现通过人的3D动作来控制远程电脑上鼠标光标移动的目标。设计实现该应用的目的是验证分布在网络中基于StarOSGi的两个OSGi应用之间的互联、互通和互操作，包括透明的服务远程发现和服务方法的远程使用，同时验证StarOSGi能够适用于嵌入式领域，保持了OSGi技术跨平台的通用性。

应用包括MouseInterface，Mouse和MouseController三个bundle。MouseInterface部署在Imote2传感器和PC机上，包含对PC机上光标控制的服务接口Mouse.java，并在元文件中导出Mouse.java所在的包以供其所在的OSGi框架上的其他bundle使用；Mouse bundle部署在光标受控PC机上，通过提供服务MouseImpl来控制本机光标。本应用的网络连接环境如图4所示，光标受控PC机通过USB模拟网络与ZigeBee网络接入点传感器连接，体感鼠标传感器通过无线ZigBee网络与ZigBee网络接入点传感器连接，这样就建立了从光标受控PC机到体感鼠标的网络连接。CORBA名字服务运行在光标受控主机上。

图4　体感鼠标网络连接示意图

本应用案例的实际运行截图如图5所示。运行结果为光标受控主机的光标在屏幕上根据实验者手中体感鼠标传感器的角度的改变而相应的移动。通过对本实验的分析可以得出以下结论：StarOSGi能够在保持OSGi原有的编程模型的情况下实现OSGi框架之间远程服务的发现与使用；StarOSGi在传感器平台上运行良好，能够适用于资源受限的嵌入式领域。



(a)传感器运行截图　　(b)光标受控主机运行状态截图

图5　体感鼠标演示截图

4.2　性能测试

由于缺乏OSGi分布式扩展标准的性能测试程序集，本节使用了Javaparty/KaRMI测试程序集。这一测试程序集以不同大小和复杂度的参数来调用远程方法，获取远程方法调用时间的平均值，以评估和对比不同RMI实现的性能。我们将这一测试集移植到OSGi上，将之实现为分布在不同节点上的OSGi性能测试服务端和客户端，并使用同样遵循RFC 119规范的D-OSGi作为性能对比对象。

实验使用了两台PC机，客户端硬件环境为主频1.5GHz的Intel Pentium M处理器、1GB DDR内存，服务端硬件环境为主频2.33GHz的Intel Core2 Duo处理器和1GB DDR内存。客户端与服务端通过百兆以太网连接，ping的平均网络延迟为217μs。CORBA名字服务运行在客户端PC机上。

图6展示了StarOSGi与D-OSGi远程服务方法调用时间对比。其中横坐标为方法名，pingVoid，pingInt，pingRetInt和pinIntInt等方法的参数为简单数据类型，pingObj4，pingObj32

(下转第189页)

from workflow logs[C] //Proceedings of the Sixth Intenational Conference Extending Database Technology. 1998:469-483

[3] vander Aalst W M P, Weijters A J M M, Maruster L. Workflow Mining:discovering process models from event logs[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 53(3): 245-264

[4] Herbst J, Karagiannis D. Workflow mining with InWoLve [J]. Computers in Industry, 2004, 53(3): 245-264

[5] Schim G. Mining exact models of concurrent workflow [J]. Computers in Industry, 2004, 53(3): 265-281

[6] Zhao Weidong, Dai Weihui. Role-Activity Diagrams Modeling Based on Workflow Mining [C]//2009 WRI World Congress on Computer Science and Information Engineering. Volume 4, 2009: 301-305

[7] She Jianchun, Yang Dongqing. Process Mining: Algorithm for S-Coverable Workflow Nets [C]//Second International Workshop on Knowledge Discovery and Data Mining(WKDD 2009). Jan. 2009: 239-244

[8] 李嘉菲，刘大有，于万钧. 过程挖掘中一种能发现重复任务的扩展 α 算法[J]. 计算机学报, 2007, 30(8): 1436-1441

[9] 吕静，陈未如，刘俊，等. 并发分支模式挖掘[J]. 计算机科学, 2004, 31(10): 270-276

[10] 陈翔，夏国平，李涛. 基于 Petri 网的工作流模型合理性研究[J]. 北京理工大学学报, 2004, 24(12): 1074-1078

[11] 袁崇义. Petri 网原理与应用[M]. 北京：电子工业出版社, 2005: 213-259

(上接第 165 页)

和 pingObj64 方法的参数则是不同长度的复杂参数；纵坐标表示远程服务方法调用的平均时间，单位为 μs。测试结果显示 StarOSGi 在简单类型数据调用上与 Apache CXF 的 D-OSGi 相比具有较明显的性能优势，这是由于 Apache CXF 的 D-OSGi 是采用 SOAP 协议来实现远程互操作，相对于 StarOSGi 采用 IIOP 协议，SOAP 协议带来了额外编码和 XML 解析开销。在复杂参数传递过程中，StarOSGi 与 Apache CXF 的 D-OSGi 性能相差不大。

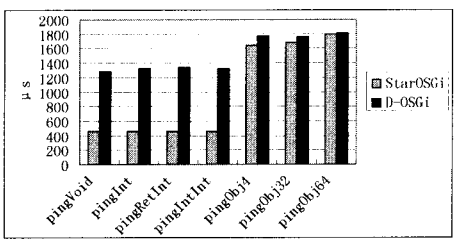


图 6 StarOSGi 与 D-OSGi 远程服务方法调用性能对比

结束语 OSGi 分布式扩展是 OSGi 在企业计算领域面临的重要问题。本文提出了一种基于 CORBA 的 OSGi 分布式扩展模型，基于该模型设计实现了一个支持 OSGi 分布式扩展的中间件 StarOSGi，与现有的 Newton，R-OSGi 以及 D-OSGi 相比，其特点在于能够保持 OSGi 面向服务的编程模型和轻量级特点，可以将集中式的 OSGi 应用透明地转变为分布式应用，并支持 OSGi 应用与 CORBA 应用的互操作，而且在性能上具有一定的优势。基于 StarOSGi 开发了体感鼠标等应用，并进行了对比测试，验证了 StarOSGi 的能力及有效性。下一步我们将对 StarOSGi 进行进一步的完善和改进，研究通用的服务发现协议，进一步扩大 OSGi 分布式扩展中服务发现的通用性。

参 考 文 献

[1] OSGi Alliance. OSGi Service Platform-Release 4[R]. 2005

[2] Gruber O, Hargrave B J, McAffer J, et al. The Eclipse 3.0 Platform: Adopting OSGi Technology[J]. IBM Systems Journal, 2005, 44(2)

[3] Sadtler C, Ganci J, Griffith K, et al. IBM Webshpere Product Overview. Redbook paper[R]. IBM, 2003

[4] Pratistha D, Nicoloudis N, Cuce S. A micro-services framework on mobile devices[C]// Conference on Web Services. Nevada, USA, 2003

[5] Fleury M, Reverbel F. The JBoss Extensible Server[C]// ACM/IFIP/USENIX International Middleware Conference. Riode Janeiro, Brazil, June 2003

[6] Walls C, Breidenbach R. Spring in action[M]. Manning Publications Co., Greenwich, CT, 2007

[7] OSGi Alliance. RFC 119 Specification[R/OL]. http://www.osgi.org/ Specifications/HomePage, 2009

[8] Paremus. The Newton Project[CP/OL]. http://newton.codecauldron.org, 2006

[9] Rellermeyer J S, Alonso G, Roscoe T. R-OSGi: Distributed Applications Through Software Modularization[C]// Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference. 2007

[10] Alliance A. Apache CXF Project[CP/OL]. http://cxf.apache.org/ distributed-osgi.html, 2009

[11] Object Management Group. The Common Object Request Broker: Architecture and Specification. Second Edition[R]. 1995

[12] Baker S. CORBA distributed objects: using Orbix [M]. New York: Addison-Wesley Publishing Co., 1997

[13] Wang Huai-min, Wang Yu-feng, Tang Yang-bin. StarBus+: Distributed Object Middleware Practice for Internet Computing[J]. 计算机科学技术学报：英文版, 2005(4)

[14] Marples D, Kriens P. The open services gateway initiative: An introductory overview [J]. IEEE Communications Magazine, 2001

[15] Sun Microsystems. Java JAR File Specification[R/OL]. http://java.sun.com /j2se/1.4.2/docs/guide/jar/jar.html, 2009

[16] Waldo J. The Jini architecture for network-centric computing [J]. Communications of the ACM, 1999, 42(7): 76-82

[17] Beisiegel M, Blohm H. SCA Service Component Architecture-Assembly Model Specification v. 1.0[R]. Open Service Oriented Architecture collaboration, Mar. 2007

[18] Veizades J, Guttman E, Perkins C. RFC 2165: Service Location Protocol[R]. IETF, 1997

[19] OSGi Alliance. RFC 126 Specification[R/OL]. http://www.osgi.org/ Specifications/HomePage, 2009

[20] Horstmann C S, Cornell G. Core Java 2, 7th Ed[M]. 2007(1): 132-145

[21] Philippsen M, Zenger M. Javaparty-transparent remote objects in Java[C]// ACM 1997 Workshop on Java for Science and Engineering Computation. June 1997