Web 应用服务器的可扩展管理模型及其实现*)

董伟川[1,2] 金蓓弘[1,2] 张 锋[1] 范国闯[1,2] 林世彪[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 Web 应用服务器是为事务性 Web 应用提供一系列运行时服务的分布式系统。它既要管理多种不同类型的资源，又要集成以往成熟的网络管理协议和系统，因此，如何统一有效地管理 Web 应用服务器的资源、应用和服务成为实现 Web 应用服务的一个难点。为此，本文基于 JMX(Java™ Management Extensions)技术，提出了一种可扩展管理模型，该模型具有较好的灵活性和开放性，可扩展能力强。同时，本文还给出了实现该模型的若干关键技术包括可扩展管理内核的实现技术、资源远程管理技术、服务可扩展管理技术。目前该模型已在我们研制的 Web 应用服务器 WebFrame2.0中成功实现。

关键词 可扩展管理模型，JMX，分布式系统，Web 应用服务器

An Extensible Management Model and its Implementation for Web Application Servers

DONG Wei-Chuan[1,2] JIN Bei-Hong[1,2] ZHANG Feng[1] FAN Guo-Chuang[1,2] LIN Shi-Biao[1,2]
(Technology Research Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)[1]
(Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)[2]

Abstract Web application server is a distributed system that provides runtime services for transactional Web applications. It must be capable of managing many types of resources. Also it needs to integrate the former mature network management protocols and systems. So how to manage Web resources, applications and services becomes a difficulty of implementing a Web application. To solve the problems, this paper presents a JMX (Java™ Management Extensions) based extensible management model with the advantage of flexibility, openness and extensibility. Then the key technologies on how to implement the model, including the implementation of the extensible management kernel, the remote management of resource and the extensible management of service are described. We have implemented the model in WebFrame2.0 application server which is a J2EE application server.

Keywords Extensible management model, JMX, Distributed system, Web application server

1 引言

Web 应用服务器是在 Web 计算环境下产生的新一代中间件，为运行和管理事务性 Web 应用提供一系列运行时服务(如消息、事务、安全、应用集成等)，具有传统事务监控器的高伸缩性、高可用性、高可靠性等特性，被认为是事务性 Web 应用的操作系统[1]。在传统的分布式应用中，系统与网络管理通常需要安装遵循某种管理协议的管理软件，如 SNMP、CMIP、CIM/WBEM、TMN 等，但这些管理软件都存在不足，例如，SNMP 存在一些安全漏洞，网络入侵者很容易获取正在通过网络传递的各种信息，甚至可以关闭某些终端；CMIP 协议所占用的网络系统资源相当于 SNMP 的十倍，如果不进行大规模的改造，很少有网络系统能够全面支持 CMIP；CIM/WBEM、TMN 不支持不同管理协议之间的互操作性，并且只能对事先已知的资源进行管理，缺乏灵活性和可扩展管理能力[2]。另外，这类管理软件价格昂贵。

JMX 是 SUN 公司针对这些已有企业网络管理存在的问题于1999年在 Java 平台上提出的一个可扩展、低成本的网络管理规范，致力于解决分布式系统的管理问题[3~5]。JMX 对被管理资源进行了抽象，提供底层基本类的集合，开发人员在保证大多数公共管理类的完整性和一致性的前提下，可以进行高效、灵活的扩展，以满足特定网络管理应用的需要；不仅如此，JMX 可以跨越一系列不同的异构操作系统平台、系统体系结构和网络传输协议，开发无缝集成的系统、网络和服务管理应用，在分布式网络环境中充当中间件的作用，并能够平滑地将管理方案应用到已存在的管理体系中。由于 JMX 继承了 Java 语言的特性，因此 JMX 还具有兼容性、灵活性和快速升级能力[3~5]。我们基于 JMX 规范提出并设计了 Web 应用服务器的可扩展管理模型，该模型充分利用了上述 JMX 的优点，并在 Web 应用服务器中起了微内核的作用。

本文第2节介绍 Web 应用服务器可扩展管理的基础 JMX；第3节阐述为什么提出可扩展管理模型并给出该模型；第4节给出实现可扩展管理模型的三个关键技术；第5节给出 Web 应用服务器 WebFrame2.0在集群环境下的管理系统的设计和实现。

2 可扩展管理的基础——JMX

我们提出的可扩展管理模型基于 JMX1.0规范，JMX 体系结构分为三层：装置层(Instrumentation Level)；代理层(Agent Level)；分布式服务层(Distributed Services Level)，参见图1。这三个部分之间通过 RMI 进行通信。JMX 规范还提供了一套 Java API 用于访问已有的标准管理协议，这套 API 通常称为附加的管理协议 API(Additional Management Protocol APIs)。

2.1 装置层

装置层提供了实现 JMX 可管理资源的规范。所有的应用

*)基金项目：国家重点基础研究发展规划973项目(2002CB312005)；国家高技术研究发展计划863(2001AA113010；2001AA414020；2001AA414310)。董伟川 硕士研究生，主要研究领域为软件工程技术和分布式计算。

程序、服务、设备、用户等都是 JMX 可管理的资源。JMX 将这些资源表示成特定的 Java 对象即 MBean(例如,Standard MBean 或 Dynamic MBean),并通过对 MBean 的管理实现对资源的管理[6]。装置层还给出了 Java 管理对象 MBean 的定义方法和一套通知(Notification)机制,通知机制使得 MBean 产生和传递组件之间的通知事件,并在 MBean 之间进行信息的传递。

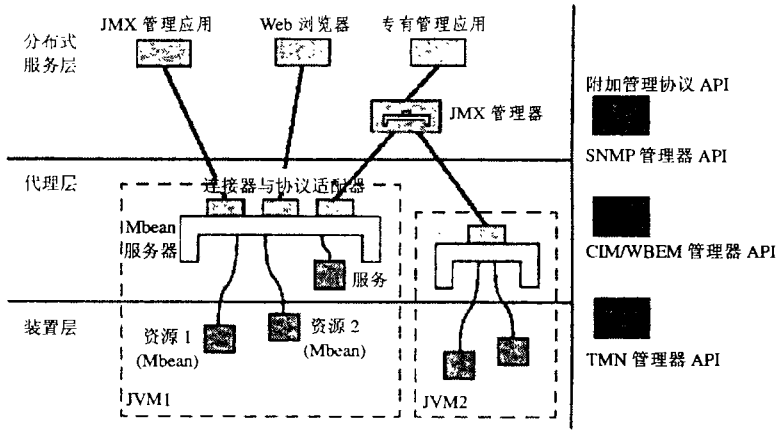

图1 JMX 体系结构图

2.2 代理层

代理层充当管理决策开发者,提供了实现代理的规范。代理直接控制被管理资源或者使它们能被远程的管理应用来访问。代理和它所控制的资源经常位于同一台 JVM(Java 虚拟机)。这一层利用装置层来管理一个 JMX 可管理资源。一个 JMX 代理包含至少一个管理服务器 MBean Server 和一系列 MBean 的服务,外加至少一个通信连接器或者适配器(connector/adaptor)。在应用服务器中,从管理的角度看,MBean Server 就是一个总线,所有服务都是在它之上集成并被它管理,服务间通信也是靠 MBean Server 来完成。

2.3 分布式服务层

分布式服务层提供实现 JMX 管理器的规范,提供了实现 JMX 管理职能的接口。这一层定义了在代理层之上的管理接口和组件。这些组件发布从高层管理平台到 JMX 代理的分布式管理信息。这一层还提供安全机制。

附加的管理协议 API 提供了与其他的管理系统交互的途径。一个应用程序使用这些 API 可访问传统的管理系统,并将这些系统的属性转换成 JMX 可管理资源,这样就可以用符合 JMX 规范的管理应用通过 JMX 代理管理传统系统。

3 可扩展管理模型

提出可扩展管理模型的一个重要原因是希望利用并集成已有的成熟的管理协议和 J2EE 服务,前者如 SNMP、CMIP 和 TMN;后者如 JNDI、JMS、JDBC 等等。尽管引言中提及网络管理协议存在缺陷,但它们都是代表着比较成熟的网络管理技术。其中 SNMP 用于互联网络设备的网络管理框架[7,8],采用管理进程/代理进程模型,管理协议在应用层或者 TCP/IP 模型的"处理层"上运行。SNMP 是一个静态的模型,是基于对象的,采用无连接、不保证可靠的 UDP 传输服务[8]。SNMP 随着 Internet 的发展成为网络管理的主要工具。同 SNMP 相比,CMIP 采用一个完全面向对象的模型,具有分布式、等级制的对象结构,应用 OSI 协议栈中面向连接、可靠的传输服务。TMN 框架是一套完整的规约,考虑的是电信网络中网络元素和资源的管理与控制,给定一个标准接口,管理信息可以通过已知和可控的方式进行访问,TMN 框架结合了 SNMP 和 CMIP 的优点并向层次化、集成化、Web 化和智能化发展。总之,SNMP、CMIP、TMN 各有所长。

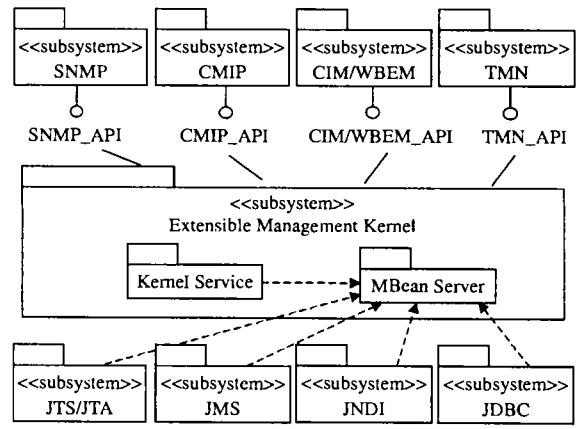

图2 可扩展管理模型

为了利用已有这些成熟的网络管理框架和 J2EE 服务,我们提出可扩展管理模型(见图2)。模型的核心是可扩展管理内核(Extensible Management Kernel),网络管理资源和 J2EE 服务受其管理。具体说,对于管理协议,模型提供 SNMP、CIM/WBEM 和 TMN 管理器的 API,这样就可以很灵活地把 SNMP、CMIP 等网络管理协议集成到该模型当中,从而可以和这些管理系统进行交互。因为该模型可扩展,所以如果还需要集成其他协议时,只需要再定义相应的管理接口或者 API,应用程序便可以使用这些 API 可访问其他管理系统,并将这些系统的属性通过 JMX 转换成被管理资源。可扩展管理内核主要包含遵循 JMX 规范的 MBean Server 和核心服务(Kernel Service)包。对于 J2EE 服务、应用和管理,可在模型内核的 MBean Server 上注册成为 MBean,成为被管理对象。而核心服务包提供一些公共服务的 MBean。上述模型在 Web 应用服务器 WebFrame2.0中实现时,应用服务器中的所有服务、应用和管理都是由 MBean Server 进行管理。由核心服务包提供的 MBean 是独立于 WebFrame 定制服务的,它们是实现 WebFrame 需要使用的共同服务的 MBean,按照 JMX 规范编写,通过在配置文件中的 Mlet 标签来标识,启动时必须加载;依赖于 WebFrame 定制服务的 MBean,它们需要根据应用的具体需求来配置和加载需要的服务,其编写要遵循 WebFrame 服务的模式,也是在启动 WebFrame 时同时加载的。多数系统核心服务如 JNDI、事务、消息服务等都是通

过这种方式启动。

采取上述管理模型，由于提供了一个标准的方式管理资源，因此可达到低成本管理的目的；被管理的资源如网络管理协议和 J2EE 服务都可以作为独立的子系统插入到模型中的，这种管理模式意味着较强的伸缩性和灵活性。可扩展管理模型比 JMX 更有特色之处在于它能够成为一个可配置和管理的软件的核心，从而给基于组件集成这种开发模式提供了一个统一的框架。

4 关键技术

4.1 可扩展管理内核

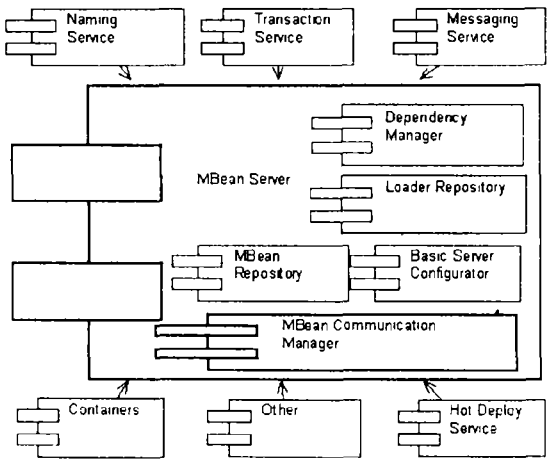

图3 Web 应用服务器的可扩展管理内核

Web 应用服务器提供两类服务，一类是核心服务，如 Web 应用运行时所需的事务(Transaction)、消息(Messaging)、名字(Naming)、容器(Container)和热部署(Hot Deploy)等服务，另一类是为上述第一类服务提供服务的基本服务如服务的装载、卸载以及服务之间的通信和交互等。为了提高系统的可扩展性、灵活性和开放性，我们把这两类服务分开，如图3所示，将上述第二类服务封装在 Web 应用服务器微内核即管理服务器 MBean Server 中，第一类服务通过系统编程接口向 MBean Server 注册。所有的服务均以 MBean 的形式由 MBean Server 进行统一管理和调度，服务之间通过 MBean Server 进行通信。MBean 是 Web 应用服务器所有服务最基本的单位，它约定服务从创建、初始化、运行到停止整个过程所有的接口，描述一个服务自身的元信息。MBean Server 由一系列实现 MBean 接口的组件构成，包括服务器基本配置组件(Basic Server Configurator)、MBean 注册库(MBean Repository)、组件装载器库(Loader Repository)、服务依赖性管理器(Dependency Manager)和服务间通信管理器(MBean Communication Server)。服务器基本配置组件是 MBean Server 首先启动的服务，其主要功能是利用 XML parser 解析 XML 格式的系统文件，读取系统的基本配置信息如环境变量、库路径、操作系统信息等。组件装载器库保存多个组件装载器，不同组件装载器装载服务所需的物理资源如二进制文件、设备资源等，通过过滤条件装载不同类型的组件。MBean 注册库保存所有成功注册到 MBean Server 的服务，这些服务所需资源均由上述组件装载器装载，MBean Server 自动初始化每个服务。服务依赖性管理器主要维护所有服务与服务、服务与资源之间的相互依赖关系。例如 Web 应用服务器的数据访问服务依赖于名字服务，当启动数据访问服务时，服务依赖管理器首先检查名字服务是否启动，如果没有启动，它会通知 MBean Server 启动名字服务。同样的方式，当关闭名字服务，服务依赖管理器首先检查自己的依赖链，如果还存在未停止的依赖服务，它会通知 MBean Server 停止这些依赖服务。所有服务都以 MBean 的形式注册在 MBean Server，MBean Server 只暴露(Expose)这些服务的可管理接口，并不直接暴露服务的对象引用。服务间通信采用消息传递的方式，如果一个服务 A 想调用另一个服务 B 的方法，服务 A 需要向 MBean Server 发送一个调用消息，该消息包含 B 服务的名字、方法名和方法参数值，然后 MBean Server 将调用的结果返回给服务 A。通信管理器主要负责上述消息的编码(Marshalling)和解码(Unmarshalling)、服务间的事件通知。

MBean Server 是 WebFrame 可扩展管理框架的基础。所有的应用程序、服务、设备、用户等被管理资源均是 MBean Server 能管理的资源。它们以 MBean 形式存在。MBean Server 就是为管理和控制 MBean 提供服务的。MBean Server 包含了创建、注册、删除和访问 MBean 所必需的方法。所有可管理的 MBean 是以插件方式插入到 MBean Server 中的，通过这种方法达到可扩展管理的目的。已经注册到 MBean Server 中的 MBean 组件即为可管理组件。MBean 对应的属性、操作即可通过连接器/适配器进行远程访问控制。

4.2 资源远程管理

对于在远端而非本地 JVM 上的管理应用，MBean Server 可以通过两种方式来实现：其一，使用连接器(connector)把代理和远端 JMX 管理应用连接起来，即把代理中的连接器服务器和应用端连接器客户连接起来，但是这种方式必须要求二者有相同的远程接口；其二，通过某个特定协议的适配器(Adaptor)将代理和远端 JMX 相连，这种方式由于采用协议因此可以基于不同的远程接口。在 WebFrame2.0中我们采用的是后者，即适配器模式[9]来实现的。

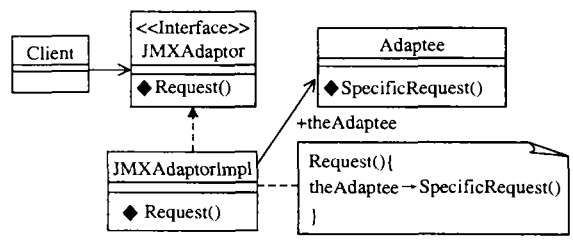

图4 WebFrame 中 JMX 适配器结构图

采用适配器模式可以使得一个接口(adaptee 的接口)与其他接口兼容，适配器模式给出了多个不同接口的统一抽象。WebFrame 中设计了一个 AdapterImpl 类，对一个 adaptee 类进行私有集成，这样适配器就可以用 adaptee 的接口了。图4是 WebFrame 中适配器模式的工作原理：其中 JMXAdaptor 是给客户端使用的一个接口；Adaptee 用来定义已经存在的接口，这个接口需要匹配；JMXAdaptorImpl 则对 Adaptee 和 JMXAdaptor 这两个接口进行匹配。针对 SNMP 和 TMN 协议，如果实现了 SNMPAdaptor 和 TMNAdaptor，就可以集成支持 SNMP 和 TMN 的管理系统。

4.3 服务可扩展管理

构造应用服务器中系统服务的通用方法是对每个服务分别进行开发、编译、执行。但是，这种静态方法把一个特定服务的实现和它的配置紧紧结合在一起，使得软件体系结构缺乏可适应性、难以扩展。这种静态方法会产生以下问题：(1)在开发步骤中，需要过早确定服务的配置，而程序员也许在开发阶段还无法知道发布这些服务最好的方式；(2)当要修改或终止某一服务时，也许会对其他的服务造成影响。因为在服务静态的实现和部署中，每个服务的实现和它的初始化配置是紧密结合的，所以难以做到在不影响其他服务的情况下对某一服

务进行修改;(3)系统资源无法得到更有效的利用,采用静态的方法往往需要对每一个服务都给予一个进程,从而会浪费宝贵的系统资源。因此在我们的应用服务器里引入管理资源的生命周期和 Component Configurator 设计模式[10]的概念(见图5),提供一种系统服务可扩展机制,使得在整个生命周期的任意时刻能把服务整合到应用服务器中。该模式主要由如下几个部分构成:(1)Service 接口定义了含有 abstract 类型的几个函数,包括 init(),fini(),suspend(),resume(),info()等。这个接口使得应用程序可以动态对每个服务进行配置和管理。在 WebFrame 应用服务器中,每个服务被包装成一个 MBean,每个 MBean 都需要实现这个 Service 接口。(2) Concrete Service 实现了 Service 的接口以及服务本身提供的功能。在 WebFrame 中,每个 Concrete Service 被包装成一个 MBean。(3) Service Repository:所有的服务在 Service Repository 中进行注册。通过它可以对所有的服务进行集中式管理。如图5所示,在服务配置(Service Configuration)阶段,Service Configurator 通过调用服务的 init 方法对一个服务进行初始化。一旦这个服务被成功地进行初始化,Service Configurator 把它加入到 MBean Server 的 Service Repository 中。Service Configurator 使用 Service Repository 对所有已经加载的服务进行管理。一旦服务被加载入系统中,服务就被运行,进入服务处理(Service Processing)阶段。在这个过程中,Service Congifurator 可以暂停或者继续服务的运行。当一个服务不再需要时,Service Configurator 把服务终止,进入服务终止(ServiceTermination)阶段。Service Configurator 调用服务的 fini()函数进行一些终止前服务所需要进行的操作。一旦服务被终止,它就被从 Service Repository 中删除。

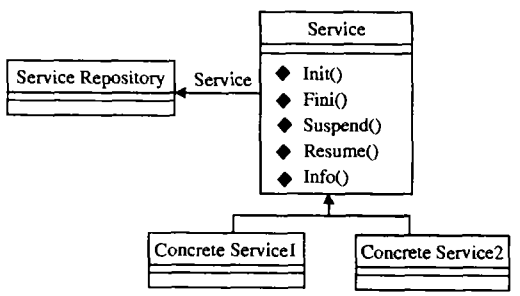

图5 系统服务可扩展模式

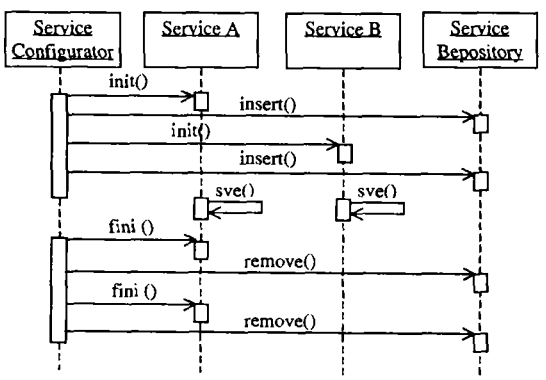

图6 系统服务扩展模式图

在 WebFrame2.0中,当 MBean Server 启动完毕后,将从一个系统服务配置文件(WebFrameSysService.xml)中读取所有服务信息,然后按照图6描述的方式将服务加载到应用服务器中。用户可以在服务配置文件中修改服务的配置属性,定制或者替换用户所需的服务,甚至还可以添加用户级的系统服务。例如,用户为了使用其自定义的目录服务,为此,需要替换名字服务实现,用户只需要更改名字服务相关配置属性(如组件名、包名、二进制文件路径、名字服务端口等)即可。

5 Web 应用服务器集群管理

WebFrame2.0可在集群环境下运行。相应地,系统管理也支持集群的管理。MBean Server 是可扩展管理的基础。管理员在 Web 客户端发出对某台机器的哪个部分进行管理的请求,客户程序把请求发送给管理服务器,管理服务器发消息给集群管理器,集群管理器监听到管理服务器发出的消息,返回给管理服务器一个所管服务器(Managed Server)的列表,管理服务器根据这个列表去查找那个具体的服务器,获取该服务器的属性值或者对它的服务的属性值进行设置和修改,然后将操作结果返回给基于 Web 进行管理的管理员。

WebFrame2.0管理服务器的体系结构如图7所示。

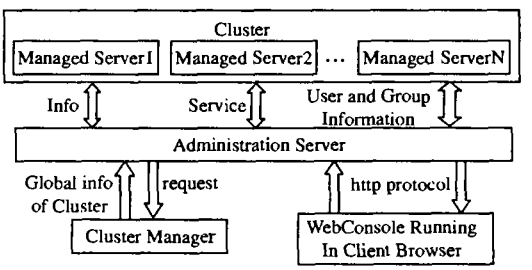

图7 WebFrame2.0管理服务器的体系结构

总结 本文简要分析了 JMX 的结构,并基于 JMX 提出了 Web 应用服务器的可扩展管理模型。该模型具有以下特点:(1)基于 JMX 规范,能集成已有网络管理协议,从而在分布式网络管理中起着重要作用。(2)提供灵活的可扩展的、定制服务的功能,能够管理 Web 应用服务器的不同类型的资源、应用和服务等。(3)在分布式系统中起着内核的作用,所有服务都可以作为插件插入到管理服务器中被管理。在我们自主研制的 J2EE 应用服务器 WebFrame2.0中已成功实现了该管理模型。下一步工作将进一步研究如何尽可能降低管理系统的额外开销,从而提高 Web 应用服务器的整体性能。

致谢 非常感谢中科院软件所软件工程技术研发中心应用服务器小组全体成员,特别是黄涛研究员,给予本文建设性的指导。

参 考 文 献

1 Dietzen S. Web Application Server Architecture. HPTS Panel 2002
2 Birrell, Nelson, Owicki. 网络对象、数字设备公司系统研究中心技术报告. 1994
3 Sun Microsystems, Inc. Java™ Management Extensions Instrumentation and Agent Specification, v1.0. 2002. http://www.sun.com/jmx
4 Sun Microsystems, Inc. JMXdatasheet. 2002
5 Sun Microsystems, Inc. Java™ Management Extensions White Paper. 2002
6 AdventNet Agent Toolkit Java/JMX Edition 4.2, 2002. http://www.adventnet.com/products/javaagent/help/jmx/mbeans
7 Miller M A. Internet Management With SNMP. China Water Power Press, 2001(in Chinese)
8 Harnedy S. Total SNMP (Second Edition). Publishing House of Electronics Industry (in Chinese)
9 Wu J. Distributed System Design. China Machine Press, 2001(in Chinese)
10 Schmidt D C, Jain P. Service Configurator. A Pattern for Dynamic Configuration of Services. In: Proc. of the Third USENIX Conf. on Object-Oriented Technologies and Systems. Portland, Oregon, June 1997