

基于 JMX 和 J2EE 的网络管理系统

於成才, 黄 东

(东南大学 自动控制系, 江苏 南京 210096)

摘 要:网络规模不断扩大,接入设备越来越多,对网络管理提出了很高的要求。好的管理软件能有效对网络进行配置、优化、监控及告警。文中基于 J2EE 和 Java 管理扩展(JMS)的原理和结构,设计了一种全新的开放分布式网络管理模型,分析了这两种技术的优势及在系统中的应用,讨论了该系统的性能特点,并就如何在此系统上进行扩展给出了描述。与现有的管理系统相比,该系统具有高度的可扩展性、可移植性,能够满足用户需求的多变性。

关键词:Java 管理扩展;J2EE;分布式;代理;管理构件

中图分类号:TP393.07

文献标识码:A

文章编号:1673-629X(2006)12-0029-03

JMX and J2EE - Based Network Management System

YU Cheng-cai, HUANG Dong

(Automation Department of Southeast University, Nanjing 210096, China)

Abstract: Larger scale network, more and more connected equipments demand high efficient network management. Good management software can efficiently configure, optimize, monitor, alarm the network. Based on the structure and theory of J2EE and JMS, designed a network management model, analyzed the advantage of these two kinds of technology and their application in the system, furthermore, discussed its performance and how to extend it. Compared with current system, this model has high extensibility and transplantation, it can satisfy the variation of users' demand.

Key words: JMX; J2EE; distributed; agent; MBean

0 引 言

随着网络规模的不断扩大,网络复杂程度的不断增强,网络异构的数目越来越多,网络管理系统迫切需要一个先进的体系结构,用来满足用户需求的多变性,处理大量复杂的数据,实现很好的实时通信,进行有效的工作流管理,保证系统的高度可扩展性。许多分布式的管理技术已经应用于网络管理领域,比如 CORBA 等。这些管理技术总的来说存在以下弊端:

- 1)不能兼容各种不同的操作系统。
- 2)服务模块解偶性差,不适用于扩展和组件式安装。
- 3)客户端为“胖”性客户端,没有实现表现逻辑与业务逻辑的很好分离^[1]。

J2EE 是 SUN 公司推出的 Java 标准,用于实现基于 Java 的多层分布式系统。文中基于 J2EE 和 JMX 构建了一个大型分

布式的网络管理系统,实现了前台和后台之间的通信。整个系统采用 Java 语言实现,很好地解决了以上问题,具有较好的跨平台、通用性和可扩展性的要求。

1 J2EE 体系结构

J2EE 平台为开发分布式的多层应用提供客户端和服务端的支持。这样的应用的典型配置为:客户层提供图形用户接口,一个或多个中间层提供客户服务和业务逻辑,后台企业信息层提供数据管理。图 1 显示了组成一个

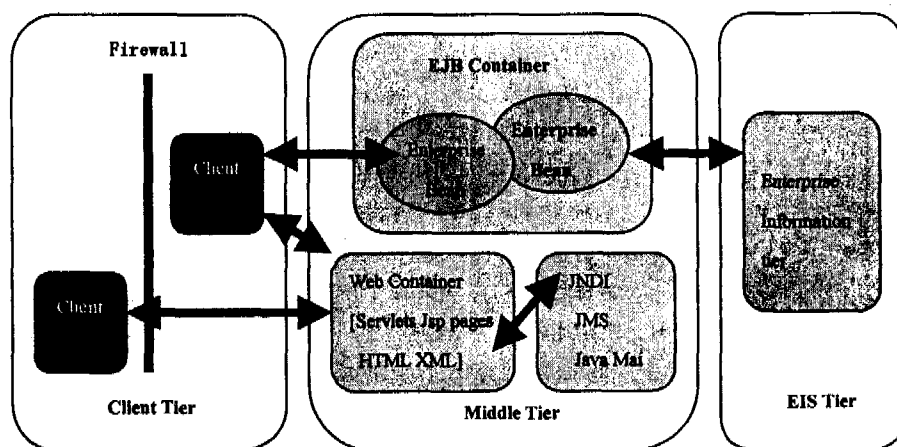


图 1 J2EE 体系结构^[2]

典型的 J2EE 环境的组件和相关的服务。

收稿日期:2006-03-05

作者简介:於成才(1980-),男,湖北荆州人,硕士研究生,研究方向为网络通信与网络管理;黄 东,副教授,硕士,研究方向为系统辨识与软件工程。

如图 1 所示, J2EE 提供了一个多层分布式应用模型, 这意味着应用程序的不同部分可以分布到不同的设备上。J2EE 体系结构定义了客户层(Client tier), 中间层(Middle tier)和后台层(EIS tier)。客户层支持不同的客户类型, 可以是企业防火墙内或外。中间层提供基于容器的服务, Web 容器和 EJB(enterprise java bean)容器分别为 Web 组件和 EJB 组件提供运行环境和服务支持, 这些组件用来解决特定应用的商务逻辑, 可以从组件供应商处得到, 也可以是自己开发的组件。后台的企业信息层通过标准的 API 进行访问。

J2EE 基于组件的开发模式的核心是容器的概念。容器就是为组件提供服务标准运行环境。组件能从任何实现 J2EE 平台的供应商中得到这些服务, 例如所有的 Web 容器为响应客户的请求提供运行支持, 执行请求处理(比如调用 JSP 页面或 Servlet 动作), 返回结果给客户。所有的 EJB 容器自动为 EJB 组件提供事务支持和生命周期管理及 Bean 查询等其它服务。容器也为访问企业信息层提供标准支持, 比如通过标准 JDBC API 访问关系数据库。另外容器提供了一种在部署和配置阶段选择应用程序功能的机制, 通过配置描述文件(XML files)组件在部署阶段能部署到特定的容器环境^[2,3]。

通过以上的分析, 可以看出 J2EE 在快速开发分布式应用方面具备很多优势, 表现在:

- 1) 简化的体系结构和开发过程。
- 2) 服务器、工具、组件的任意选择。
- 3) 容易与现有信息系统集成。
- 4) 为满足需求变化的可扩展性。

2 JMX 原理和体系结构

JMX(Java Management Extensions), 即 Java 管理扩展, 是一个为应用程序、设备、系统等植入管理功能的框架。JMX 可以跨越一系列异构操作系统平台、系统体系结构和网络传输协议, 灵活地开发无缝集成的系统、网络和服务管理应用。JMX 技术采用对资源的接口标准化, 以一种标准的方式实现对资源的管理, 能够在被管资源级别上提供智能化管理服务, 允许管理程序自动完成对设备的管理; 代理之间可以动态地交换信息, 减轻了网络管理程序复杂的事务, 同时能够减轻网络的负载。为了提升代理的功能, 新的服务可以动态地下载和插入到代理中。JMX 支持分布的模式, 即协议是独立的, 管理应用程序依赖 API 来访问代理, 而不是单一的协议。

JMX 体系结构中, 采用三层结构方法来减少网络管理系统的复杂程度。图 2 中给出了三层结构描述。

1) 资源封装层定义了如何实现 JMX 管理资源的规范。一个 JMX 管理资源可以是一个 Java 应用、一个服务或一个设备, 它们可以用 Java 开发, 或者至少能用 Java 进行包装, 并且能被置入 JMX 框架中, 从而成为 JMX 的一个管理构件(Managed Bean), 简称 MBean^[4]。管理构件可

以是标准的, 也可以是动态的, 标准的管理构件遵从 JavaBeans 构件的设计模式; 动态的管理构件遵从特定的接口, 比标准构件提供了更大的灵活性。

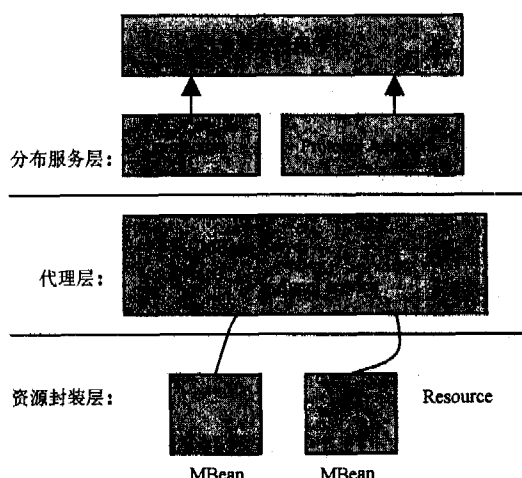


图 2 JMX 体系结构

2) 代理层是一个运行在 Java 虚拟机上的管理实体, 它活跃在管理资源和管理者之间, 用来直接管理资源, 并使这些资源可以被远程的管理程序所控制^[3]。代理层由一个 MBean 服务器和一系列处理被管理资源的服务所组成。

3) 分布服务层是为了管理部件而规定的一些接口, 包括一些适配器和连接器, 其作用就是使 MBean 服务器与管理应用程序能进行通信。协议适配器通过特定的协议提供了一张注册在 MBean 服务器的管理构件的视图。连接器还必须提供管理应用一方的接口以使代理和管理应用程序进行通信, 即针对不同的协议, 连接器必须提供同样的远程接口来封装通信过程。当远程应用程序使用这个接口时, 就可以通过网络透明地与代理进行交互, 而忽略协议本身。

3 基于 JMX 和 J2EE 的网络管理系统

3.1 系统结构

在该系统中管理应用程序功能由 J2EE 组件实现。表示子层用 Java swing 设计图形用户界面为用户提供 GUI 接口, 业务逻辑层由企业 Bean 组成, 下面的封装层以及代理层采用 JMX 标准结构设计, 如图 3 所示。

在资源封装层, MBean 定义了资源的管理接口并且将它们的变量和方法映射到接口的属性和操作上以便暴露给管理应用层, 比如管理构件中一个只读属性只有 get 方法, 而一个可读写的属性则提供了 get/set 方法。除了管理接口以便代理对其进行操作外, MBean 还定义了一种通知模型, 通知管理应用程序对属性变化或特殊情况做出反应。MBean 符合 JMX 规范, 因此可以在任何 JMX 兼容的代理中被实例化, 这就使得它们是可移植的。

代理中的核心部件是 MBean 服务器, 它相当于一个 MBean 实例的注册数据库并提供了通用的接口。通过这些接口, 管理应用程序对特定的 MBean 提出请求, 可以请

求得 MBean 的管理接口描述,因而可以知道这个 MBean 能够提供的关于被管资源的可管理特性。MBean 服务器包含了实现高级管理策略必须的服务:监视服务,用来监视管理构件的属性变化,并把这些变化通知给所有的监听者;定时服务,该服务可以在制定的时间和日期发出通告,也可以定期地周期性地发出通告,依赖于管理应用程序的配置;关系服务,定义并维持管理构件之间的关系;动态类加载,它可以从网络上的任何 URL 处下载并实例化管理构件,然后向 MBean 服务器注册,从而使得该构件接收管理应用层的管理^[4,5]。

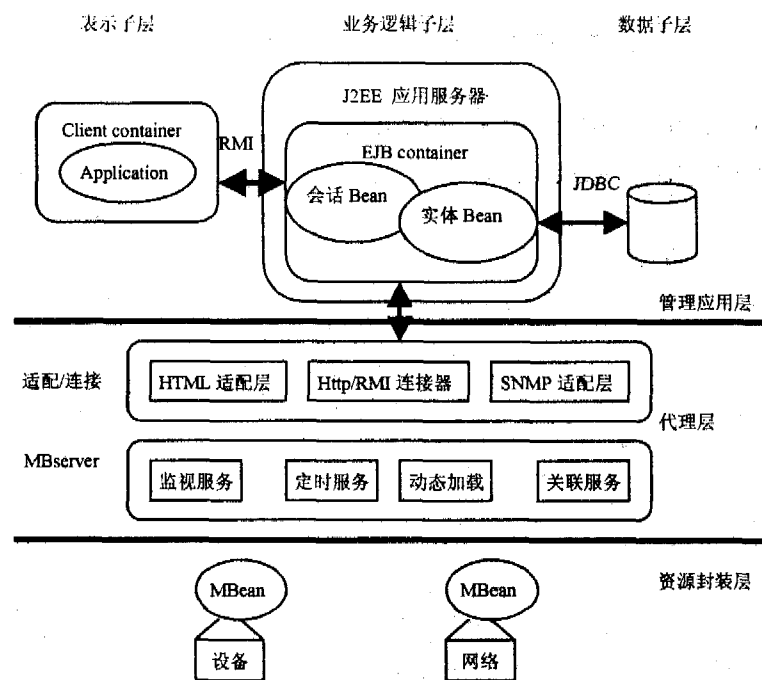


图 3 基于 JMX 和 J2EE 的网络管理体系结构

管理应用层通过连接器或协议适配器访问代理,由于这些适配器向管理层提供了统一的接口,因而使得管理与被管资源之间的交互变得简单和透明,这里管理应用程序由 Java 编写,通过 RMI 和 Java IIOP 连接器访问代理。管理应用层的会话 Bean 的远程端口暴露给客户(表示子层),所有客户端对服务器的请求都首先映射到会话 Bean,会话 Bean 本地调用实体 Bean 的商务方法和数据库进行交互,或者远程访问代理执行对 MBean 的操作。

可以看出,和传统结构相比,该系统具有以下优点:

- 1) 实现了分布式的管理,各层组件可以分布在不同的主机。
- 2) 模块化的设计,资源封装层、代理层、管理应用层各模块单独设计,较好实现了松耦合。
- 3) 可扩展性,可以动态添加被管理资源。可以开发不同的协议适配器件以便和现有的网络管理系统集成。
- 4) 可移植性,整个系统采用 Java 设计,可以移植到具有 Java 虚拟机的设备上。

3.2 系统性能

通过对网络和设备资源的封装,能有效地完成其性能

数据的采集任务,数据最终通过用户能够识别的信息以图形用户界面(GUI)的形式展现给用户,从而达到网络性能管理的目的。客户可以通过 GUI 对网络和设备进行一系列的配置和管理,这些任务下达后,GUI 模块把请求通过 RMI 转发给会话 Bean,会话 Bean 远程访问代理并执行对 MBean 的操作,从而达到网络配置的目的。由于 MBean 服务器为代理提供了智能化管理功能,代理能够监视设备的运行状况,当出现某些异常,比如连接失败、资源的某些属性超过规定的取值等,代理能够将这些信息及时反馈给管理应用层,从而很好地实现了网络监视的目的。所有的

这些配置、查询、异常信息都能通过实体 EJB 记录到数据库中。该系统还提供了虚拟网络管理的功能,即在网络断开的情况下,可以在客户端建立虚拟网络,网络中各网元的信息以及网络的状况被记录数据库中,客户可以像对实际的网络操作一样实现对虚拟网络的任何操作,如果这些操作满足需求,可以将这些配置参数反映到实际网络上。

3.3 系统扩展和伸缩

代理层的功能可以扩展。动态加载服务可以从任何位置加载和启动 MBean,因此可以开发新的类并请求代理加载和启动,从而扩展了代理的功能。同样代理的某些功能可以停用,以减少系统的开销。

被管资源可以动态添加。所有的与 JMX 兼容的组件能够集成到一个代理中,这些组件可以是任何新的设备和通信组件。

可以开发不同的连接器和协议适配器以便和现有的网络管理系统集成。已经实现的适配器包括 HTML 适配器,支持 Web 管理模式浏览访问代理。SNMP 协议适配器支持以 SNMP 协议为管理协议的应用程序对 MBean 进行管理^[5]。

管理应用层客户端可以改造成 Web 形式的配置界面,从而实现在任何一台具备上网功能的电脑上实现对网络的监控和管理。

4 结 语

可以看出,采用 J2EE 和 JMX 的体系结构设计的网络管理系统具有很多的优势,很好地实现了各层功能的划分,可以组件式的拆卸,各层模块解耦性好,易于开发和实现,可以满足客户需求的变化,达到系统高度的可扩展性。可以说,这种结构很好地解决了传统体系结构的弊端,将成为现代网络管理系统的主流结构。

参考文献:

- [1] 岑贤道,安常青.网络管理协议及应用开发[M].北京:清华大学出版社,1998.

(下转第 34 页)

的原语,它们可完成 Agent 交互中通信要求的基本操作。KQML 的保留通信原语见表 1。

表 1 KQML 的保留通信原语

功能分类	通信原语名称
会话类	ask - all, ask - one, ask - if, delete - all, delete - one, undelete, tell, untell, deny, insert, uninsert, stream - all, achieve, unachieved, advertise, unadvertised, subscribe
会话干预与机制原理	sorry, error, standby, ready, next, rest, discard
网络服务类	register, forward, broadcast, transport - address, broker - all, broker - one, recruit - all, recruit - one, recommend - all, recommend - one

部分功能说明:

* ask - all: sender 希望知道 receiver 中: content 为真的所有实例;

* delete - all: sender 请求从 receiver 的知识库中删除所有满足条件的记录;

* tell: sender 向 receiver 表明: content 在 sender 中为真;

* insert: sender 要求 receiver 向其知识库中写入: content; 等等。

2.2 KQML 的消息语法

KQML 语言的一般格式可参见文献[4], KQML 的消息语法, 采用类似于 LISP 的语法描述, 开头以一个通信原语(performative)作为第一行, 后跟参数名称(parameter name)和参数值(parameter value)。下面是一般的 KQML 消息语法描述。

(performative

```
:sender <word> // 消息发送方
:receiver <word> // 消息接收方
:reply_with <word> // 消息的 ID
:in_reply_to <word> // 所回答消息的 ID
:language <word> // 内容语言
:ontology <word> // 本体名称
:content <expression> // 消息内容
)
```

2.3 用 XML 封装 KQML 消息

由于 KQML 采用一种类似于 LISP 的语法来描述 ACL 消息, 所以 KQML 消息的内容本身可以用 XML 来编码, 用 XML 编码后更容易编写解析器, 而且用 XML 编码的消息还有利于在不同的操作系统平台上传输和解析。以下是将 KQML 消息直接封装成 XML 文档的一个例子[5]。

• Agent Peter 要发送的 KQML 消息为:

Advertise

```
:sender Peter :reply_with q1 :language KQML :ontology Kn
```

Kn

```
:content (Evaluate :language KQML)
```

```
:ontology P1 :reply_with q1
```

```
:content (val(torque motor 1)(sim-time 5))
```

• 用 XML 文件解析器将要发送的 KQML 消息生成如下的 XML 文档:

```
<! - advertise.xml - - >
```

```
<? xml version="1.0" encoding=" - "GB2312">
```

```
<KQML>
```

```
Advertise :sender Peter :reply_with q1 :language
KQML :ontology Kn
```

```
:content(Evaluate :language KQML) :ontology P1 :reply
-with q1
```

```
:content (val(torque motor 1)(sim-time 5))
```

```
</KQML>
```

3 结 论

ACL 是多 Agent 系统中 Agent 间相互协作、共同完成求解任务的基础和关键。文中提出并设计的基于 XML 和 CORBA 技术的 Agent 通信框架, 通过采用先进的 CORBA 规范来实现 KQML、利用 XML 来描述 KQML 消息, 给 KQML 消息的编码、解析和可扩展性带来了便利, 有利于解决 MAS 的平台异构性和互操作性问题, 为 Agent 通信机制提供了一个有效的技术实现方案。

参考文献:

- [1] Jennings N R, Sycara K, Wooldridge M. A roadmap of agent research and development[C]// Autonomous Agents and Multi-Agent System. Boston: Kluwer Academic Publishers, 1998: 275 - 306.
- [2] 石 慧, 徐从富. Agent 通信语言 KQML 的实现及应用[J]. 计算机工程与应用, 2005, 41(13): 94 - 97.
- [3] 魏晓斌, 周盛宗. Agent 通信机制探讨[J]. 计算机工程与应用, 2002, 38(5): 66 - 70.
- [4] Finin T, Labrou Y, Mayfield J. KQML as an agent communication language[EB/OL]. 1996. Draft; University of Maryland Baltimore County, <http://www.cs.umbc.edu>.
- [5] 邱 建. Internet 上多 Agent 之间通信框架的研究[D]. 武汉: 武汉理工大学, 2005.

(上接第 31 页)

- [2] Singh I, Stearns B, Johnson M, et al. Designing Enterprise Application with J2EE platform[M]. 2nd ed. London: Addison-Wesley, 2002.
- [3] Sun Microsystems, Inc. J2EE Specification[EB/OL]. 2005 - 06 - 20. <http://java.sun.com/J2EE/>.

- [4] Sun Microsystems, Inc. JMX Specification[EB/OL]. 2002. <http://java.sun.com/products/JavaManagement/>.
- [5] AdventNet Inc. AdventNet Agent ToolKit[EB/OL]. 2005. <http://www.adventnet.com/products/javaagent/help/snmp-agent/javadocs/index.html>.