Web 应用服务器可扩展热部署机制*)

胡建华[1,2] 范国闯[1,2] 陈宁江[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 应用服务器为基于组件的事务性 Web 应用提供运行环境和支撑平台。提供7×24小时不间断运行能力是 Web 应用服务器的一个重要特性,而静态部署降低了服务器的这种高可用性,增大了系统维护和管理的难度。目前主流 Web 应用服务器均支持热部署,但只支持固定应用类型,不具有可扩展性。为此,本文结合在自主研制 Web 应用服务器过程中的实际情况,设计了一种可扩展的热部署模型和机制。该机制在不中断 Web 应用服务器运行的情况下实现部署功能,并提供具有良好扩展性的部署系统结构,支持不断增多的新类型应用。该热部署机制已在中科院软件研究所自主研制的 Web 应用服务器 WebFrame2.0中实现,目前支持包括 Web Service 等多种应用类型的热部署。

关键词 Web 应用服务器,热部署,可扩展性

An Extensible Mechanism of Hot Deploy In Web Application Server

HU Jian-Hua[1,2] FAN Guo-Chuang[1,2] CHEN Ning-Jiang[1,2]
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080)[1]
(Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)[2]

Abstract The Web application server provides the environment and platform for component-based transactional Web applications. To run without shutting down is very important to the Web application server. However, deploying applications statically reduces this availability and makes it very difficult to be maintained and managed. At present, the existing Web application servers mainly support hot deployment, but the hot deploy mechanism is inextensible and can only support fixed types of applications. An extensible hot-deploy mechanism is proposed in this paper. This new mechanism can perform deployment without stopping and restarting the server, and it is also extensible to support new application types. This mechanism has been implemented in WebFrame2.0, a Web application server developed by Institution of Software of Chinese Academic of Sciences. As a feature of WebFrame2.0, the deploy mechanism supports multiple application types including Web Services applications.

Keywords Web application server, Hot deployment, Extensibility

1 引言

Web 应用服务器是为创建、部署、运行、集成和管理事务性 Web 应用提供一系列运行时服务的 Web 中间件。Web 应用服务器为应用组件提供运行环境和支撑平台,提供企业级的高伸缩性、高可用性、事务管理、消息服务、安全性等服务,支持与多种外部资源的集成[1]。

服务提供商以组件部署的方式向 Web 应用服务器提供服务的内容。组件部署包括三个方面的内容:组件加载、组件卸载和组件更新。组件加载是指在 Web 应用服务器中创建组件实例,使该组件中的业务逻辑能被访问;组件卸载是指删除 Web 应用服务器中组件的实例,该组件中的业务逻辑不再可以被访问;组件更新是指将一个已经创建实例的组件被修改后重新加载到服务器中。

为了保证服务的质量,Web 应用服务器需要7×24小时不间断运行。同时随着网络应用的发展,新的应用类型必然出现。如果采用静态部署的方法,Web 应用服务器中所有组件在 Web 应用服务器启动时加载并启动,并且只能在 Web 应用服务器停止时卸载。静态部署存在明显的缺点:一是资源的浪费,一个组件被加载后,直到服务器停机之前才能释放它所占用的资源;二是易干扰服务器的正常运行,静态部署使得为了重新启动一个发生错误或者变更的组件实例需要重启 Web 应用服务器,从而影响对其它组件实例的正常访问;三是灵活性差,不易支持新类型的应用。为解决静态部署存在的问题,提出了可扩展热部署的概念。热部署(Hot Deployment)是指在不中断 Web 应用服务器运行的情况下进行部署。目前主流的 Web 应用服务器均支持热部署,但只支持固定类型的热部署,为此本文将结合自行研究开发的 Web 应用服务器的实际需求,研究和设计了一种新的可扩展的热部署机制。它在实现一般热部署功能的基础上,实现了服务组件的热部署,提供了一种具有良好扩展性的部署系统结构,使 Web 应用服务器可以方便地支持新类型的组件部署。

本文第2节介绍 Web 应用服务器 J2EE 部署模型;第3节介绍了可扩展的热部署机制,说明系统模型和关键算法,并给

*)基金项目:国家自然科学基金项目(69833030);国家十五科技攻关计划(2001BA205A06);国家高技术研究发展计划(863计划)(2001AA113010;2001AA414020;2001AA414310;2002AA413610)。胡建华 硕士研究生,主要研究领域为分布式计算。范国闯,陈宁江 博士研究生,主要研究领域为分布式计算。

出基于该机制的一个实现；最后给出总结和进一步的研究方法。

2 J2EE 部署模型

Web 应用服务器支持的可部署文件包称为模块(Module)。从结构上来说，Web 应用服务器支持的模块有两种类型：基本类型和复合类型。基本类型是相对简单的基本应用，每一类规定了待部署文件包的目录结构、所包含文件的类型及命名规则，还规定了部署描述文件的格式以提供必要的运行环境配置参数。如果将若干基本类型模块，加上一个描述这些模块之间关系的部署描述文件，组成一个新的文件包，即是复合类型模块。多个复合类型模块加上合适的部署描述文件还可以组成更大的复合类型模块[2]。

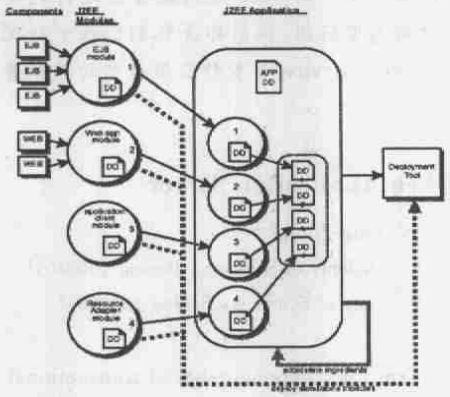

图1 部署模块的结构模型图

图1显示了可部署应用的结构，有4种基本应用类型：Web 应用(.war 文件)用于响应和处理浏览器发来的请求，它包括服务端的 Web 组件(servlet/jsp)、静态的网络资源(如 html 文件)以及客户端类(如 Applet 类)；EJB 组件(.jar 文件)，按照 EJB 规范为用户提供业务逻辑；资源适配器组件(.rar 文件)，使 Web 应用服务器与不同资源之间能够以统一的方式实现连接；应用客户端(Application client)，是针对服务器端的某个服务提供的一个轻型 Web 应用服务器。Web 应用服务器支持的复杂类型统称为企业应用(.ear 文件)。这些应用包可以单独部署，也可以组合成更大的模块部署。每种类型的应用的组成、结构和部署描述文件都需要遵守相应的 J2EE 规范。

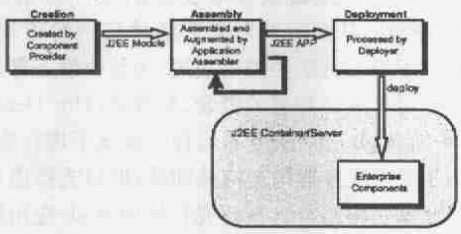

图2 创建可部署应用的步骤

创建一个可部署模块的过程如图2所示。在集成阶段，将创建的若干模块集成起来，形成一个组合的模块。在部署阶段，组件加载分为创建和启动两个步骤。组件创建是指运行环境根据组件类型和部署描述文件的内容，将组件中静态的文件装载到进程中，并分配不同的角色，从而创建组件的实例；组件启动是将组件实例绑定到名字空间，并告知管理器，该组件的业务逻辑可以被访问，以后对它访问会重定向到相匹配的方法。组件卸载跟组建加载恰好相反，它分为组件实例的停止和移出两个步骤：停止是通知管理器本组件实例停止服务；移出是将进程空间中的组件实例彻底删除，释放这个组件实例所占有的资源。

3 可扩展的热部署机制

3.1 热部署基本模型

为了能够支持组件的热部署，我们设计了新的部署器模型，该模型结构如图3所示。

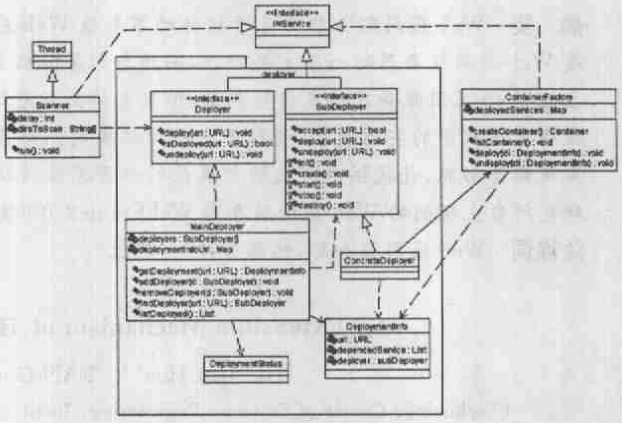

图3 热部署机制体系结构

IMService 是一个标识服务的接口，其实现类的实例可以被动态装载到 Web 应用服务器中并作为服务被调用。接口 Deployer 描述 MainDeployer 需要实现的操作。MainDeployer 实现 Deployer 接口，提供判断某个模块是否已经部署、激活加载和卸载的方法；SubDeployer 概括了具体部署器(ConcreteDeployer)的共有功能，具体处理组件实例的创建和管理，除了组件加载和卸载的方法外，还有组件实例初始化等方法。各个类职责分配如下：

扫描器(Scanner) 用于发现需要进行部署处理的组件模块。Scanner 每隔一定的时间(属性 delay)扫描指定的目录(属性 dirsToScan)，如果发现本次扫描到的信息与上次扫描的相比有不同，则说明有新的组件部署处理请求，于是激活部署服务处理这个部署请求。

主部署器(Main Deployer) 是部署子系统的入口，它具有四个作用：1)记录已经部署的应用，处理由 Scanner 发来的待部署组件的地址，判断它的状态(已经部署、修改或是卸载)；2)维护部署器列表，将新增的部署器增加到部署器列表中，或者从列表中删除不再可用的部署器；3)MainDeployer 本身并不在运行环境中加载或者卸载组件，而是在部署器列表中寻找合适的子部署器，将部署任务分配给它去完成，因此需要进行搜索；4)MainDeployer 对已部署的组件进行管理。通常一个 MainDeployer 需要多次调用相同或者不同的 Concr-eteDeployer 来完成一次部署。它依赖于接口 SubDeployer 的实现以及 DeploymentState 和 DeploymentInfo 记录部署的情况。

具体部署器(Concrete Deployer) 是处理具体可部署应用的部署器，它有三个作用：1)接受任务的能力，即能够识别需要自己处理的应用组件类型；2)为组件实例的创建作准备工作，包括分析模块中的文件，将需要的库文件加入到类路径中，根据部署描述文件配置属性；3)控制组件实例的运行，对于服务组件，能够暂停、终止或者删除它，对于应用服组件能

够停止和启动。ConcreteDeployer 依赖于 ContainerFactory 提供组件实例的运行环境和 DeploymentInfo 记录的部署信息。

部署状态(Deployment Status) 用于描述待部署应用的状态，对应部署过程的各个阶段，包括：部署信息初始化，子部署器初始化，模块已部署，模块部署失败，模块初始化，模块初始化失败，模块中组件实例启动，模块中组件实例启动失败，等等。

部署信息(Deployment Info) 封装组件部署所需要的信息，包括待部署应用的地址、需要使用的 ConcreteDeployer、应用的部署描述文件、所需库文件等。随着部署过程的进行，MainDeployer 和负责的 ConcreteDeployer 逐步设置各个属性。基于可部署应用的结构特点，DeploymentInfo 使用树型结构来描述模块的组合关系。

容器工厂(Container Factory) 与 ConcreteDeployer 是成对出现的，它为 ConcreteDeployer 部署的应用提供运行的环境，即容器。组件通过容器规定的协议和方法来访问其他组件和平台提供的服务。

图4显示了使用上述部署机制来部署一个应用的过程。

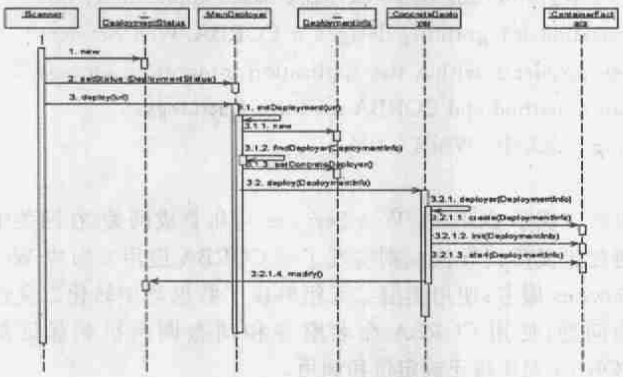

图4 部署过程

使用 SubDeployer 消除了 MainDeployer 和 ConcreteDeployer 之间的耦合[3]，同时各个支持简单类型应用的 ConcreteDeployer 本身是可插入的服务，并且相互独立，因此如果需要支持一种新的基本类型，只需要设计一个新的 ConcreteDeployer，针对这种基本类型的特点实现 SubDeployer 定义的方法，再将它注册到 MainDeployer 中即可。此时，用户可以使用已有的 ContainerFactory，或者自己编写。因此该模型具有很好的可扩展性。

3.2 关键算法和实现

在热部署模型的实现中，关键的问题主要有四个：1)如何动态发现待部署应用；2)如何为待部署应用找到合适的部署器；3)如何维护服务组件内部服务之间的依赖关系；4)组件实例如何动态创建。以下分别进行阐述。

Scanner 用来发现待部署的应用。具体实现时，可将 Scanner 实现为 Web 应用服务器进程下的一个线程，使得 Scanner 能够享有自己的 CPU 时间片，循环运行自己的 run()方法。在 Scanner 的 run()方法中，将上次扫描的结果与本次扫描的结果进行对比，一旦发现文件的变化，就发出部署请求。

MainDeployer 需要为待部署应用寻找合适的 ConcreteDeployer。待部署应用常常具有复杂的结构，MainDeployer 需要将它分解成基本类型的模块，才能用 ConcreteDeployer 加载到运行环境中。在分解的过程中，MainDeployer 还需要根据部署描述文件记录同一个模块的子模块之间的关系。MainDeployer 的处理算法如下：

```
Function MainDeployer_deploy(DeploymentInfo di)
                              Boolean deployed=false;
For I=0 to deployers.size
  Do if(deployers[I].accept(di))
        deployers[I].deploy(di)
        deployed=true;
        break;
     end if
  Endfor
  If (not deployed)
     DeploymentInfo[] diList=parseDocument(di.document);//解
        析部署描述文件
     For I=0 to diList.size
       MainDeployer_deploy(diList[I]);//递归调用
     Endfor
  Endif
End
```

Web 应用服务器内部的消息广播机制用于对组件实例之间依赖关系的维护。部署机制通过解析部署描述文件记录下每个组件实例所依赖的组件实例。一个组件实例只有依次经过创建、初始化和启动这三个步骤后才可用。前两个步骤都不要涉及其它组件实例的状态，但是组件实例在进入启动状态之前需要所依赖的组件实例都已经进入启动状态；否则该组件的部署器会进入阻塞状态，直到所依赖的组件实例启动后通过消息广播机制发出消息来唤醒它。同样如果某个组件实例停止了，它也会发出消息，部署机制会停止依赖于它的组件实例。

类的动态装载将静态模块中的文件动态装载到运行环境中，从而创建组件的实例。ContainerFactory 使用重载的类装载器，该类装载器利用 java 语言的特点[5]，搜集创建一个组件需要的文件和资源，让 java 的运行环境根据需要动态的创建各个类。通过重载类装载器，ContainerFactory 能够利用远程虚拟机上的资源创建类，从而实现远程部署。

3.3 可扩展示例：Mbean 服务热部署

基于上述热部署机制，我们实现了 JMX Mbean 服务组件的热部署。JMX 服务是可管理的 Bean，它实现一个特殊的接口并且遵守一定的模式[4]。JMX 服务部署器的实现类图如图5所示。其主要的类包括 ServiceDeployer 类和包 deployer.servicedeployer 中的类，它们联合充任热部署模型中的 ConcreteDeployer 的角色，将 Mbean 服务动态部署到 MbeanServer[4]提供的动态环境中。MbeanServer 是所有 MBean 服务创建、注册、接受访问的代理，它为 Mbean 服务提供动态装载等服务。部署时，将一个或者多个 MBean 服务加上部署描述文件打包成一个可部署的模块，称为 sar 文件，部署文件可以指定这些 Mbean 服务之间的依赖关系。

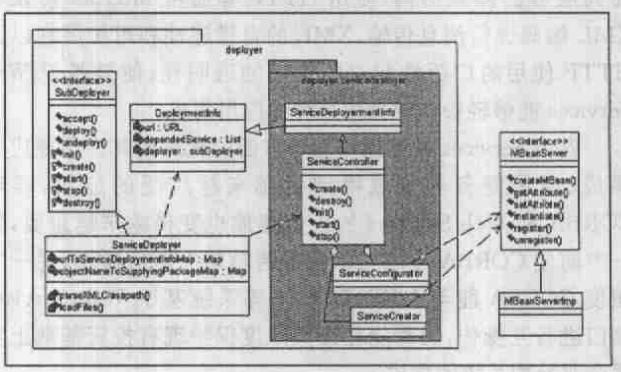

图5 JMX 服务部署器的实现

ServiceDeployer 根据待部署应用的文件扩展名(.sar)决

(下转第55页)

多利用数据挖掘发现数据中错误的方法，如聚类分析算法(如K-means、层次聚类等)、关联规则算法(如定量关联规则、比率关联规则和有序关联规则等)等。有些数据挖掘算法需要很长的时间，因此对于不同的应用场合，需要选择合适的算法。对于数据仓库数据源数据的清洗需要运行速度快的挖掘算法，而对于实时性要求较低的场合，如数据挖掘的数据清洗过程，可以选择运行时间相对较长的算法。

文[11]中提出数据仓库数据清洗方法应该满足：应该能够检测和删除所有主要的错误和不一致的情况，包括单数据源和多数据源集成的情况；应该有工具的支持，需要有限的手工查看和编程；应该很容易地扩展到其它数据源；数据清洗不应该孤立地进行，应该同数据转换结合起来，并利用元数据信息；数据清洗和数据转换的映射函数应该使用声明性语言描述；在数据仓库环境下，应该提供工作流支持，以便执行多数据源、大数据量的可靠有效的数据转换。

本文提出的模型提供了一个将数据集成、转换和清洗集成于一体的数据清洗框架，一方面指出了数据清洗的流程，另一方面给出了数据清洗需要的支持工具。该模型是以数据清洗的操作者为中心，交互式地进行工作流建模、自动地工作流调度和监视清洗工作。

结语 本文首先回顾了数据清洗的相关概念和技术。然后，提出了一个集成的以人为中心的数据清洗过程模型。该模型利用了工作流思想，将数据清洗的流程看作是一个工作流，流程中的每个任务或数据清洗的每一步都可以选择适当的技术和方法。并指出了数据清洗所需的工具集。该模型不仅适合于数据仓库环境下的数据清洗，也适用于数据挖掘过程的数据清洗。本文的进一步工作是实现数据清洗的可视化工作流定义工具、工作流调度和监视工具，设计实现各个工具箱，定义有关的数据清洗规则和策略。

参考文献

1 Wang R Y, Reddy M P, Kon H B. Towards quality data: an attribute-based approach. decision support systems, 1995, 13
2 Celko J, McDonald J. Don't warehouse dirty data. Datamation, 1995, 41(19): 42～45
3 Forino R, Data e. Quality: the data quality assessment, part 2. DM Review Online in September 2000. http://www.DMReview.com
4 Redman T. The impact of poor data quality on the typical enterprise. Communications of the ACM, 1998, 41(2): 79～82
5 Monge A E. Matching algorithm within a duplicate detection system. IEEE Techn. Bulletin Data Engineering, 2000, 23(4)
6 Marcus A, Maletic J I. Utilizing association rules for identification of possible errors in data sets: [The University of Mernphis' Technical Report CS-00-02]. 2000
7 Knorr E M, Ng R T. A unified notion of outliers. properties and computation. In: Proc. of KDD 97, 1997. 219～222
8 Orli R J. Data extraction, transformation, and migration tools part II. Available at: http://www.kismeta.com/ex2.html, 1996
9 Raman V, Hellerstein J M. Potter's Wheel: an interactive data cleaning system. In: Proc. of the 27th VLDB Conf. Roma, Italy, 2001
10 Brachman R J, Anand T. The process of knowledge discovery in databases: a human-centered approach. In Advances in Knowledge Discovery and Data Mining, MIT Press/AAAI Press, 1996
11 Rahm E, Do H H. Data cleansing: problems and current approaches. IEEE Techn. Bull. Data Engineering, 2000, 23(4)
12 Hernandez M A, Stolfo J S. Real-world data is dirty: data cleansing and the merge/purge problem. Journal of Data Mining and Knowledge Discovery, 1998, 2: 9～37
13 Simoudis E, Livezey B, Kerber R. Using recon for data cleaning. In: Proc. of KDD, 1995. 282～287
14 Tork R M, Schwarz P M. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In: Proc. 23rd VLDB, 1997
15 Wiederhold G. Mediators in the architecture of future information systems. IEEE Computer, 1992, 25(3): 38～49
16 Fox C, Levitin A, Redman T. The notion of data and its quality dimensions. Information Processing and Management, 1994, 30(1): 9～19
17 Korn F, Labrinidis A, Kotidis Y. Ratio Rules: A new paradigm for fast, quantifiable data mining. In: Proc. of the 24th VLDB Conf. New York, USA, 1998
18 Stephen H, Sudha R. Multiuser view integration (MUVIS): an expert system for view integration. ICDE, 1990. 402～409
19 Hernandez M, Stolfo S. The Merge/purge problem for large databases. In: Proc. of ACM-SIGMOD Conf., May 1995
20 Kukich K. Techniques for automatically correcting words in text. ACM Computing Surveys, 1992, 24(3): 377～439
21 Monge A E, Elkan C P. An efficient domain-independent algorithm for detecting approximately duplicate database records. Department of computer Science and Engineering, University of California, 1997

(上接第23页)

定是否部署这个应用。当一个 sar 文件放入 Scanner 的扫描目录以后，MainDeployer 会调用 ServiceDeployer 来部署这个应用。ServiceDeployer 根据 MainDeployer 传入的 DeploymentInfo创建一个子类 ServiceDeploymentInfo 的实例，以描述这个组件部署的信息。随后 ServiceController 实现服务的创建和控制，它的 ServiceCreator 属性创建服务，ServiceConfigurator属性进行服务创建后的配置。ServiceCreator 和 ServiceConfigurator 直接和 MbeanServer 进行交互，将待部署应用的服务动态地装载到 MbeanServer 提供的运行环境中。

结束语 WebFrame2.0是中科院软件所自主开发的遵循 J2EE 规范的 Web 应用服务器。基于本文的可扩展热部署模型，WebFrame 中实现了 J2EE 规范所规定的基本应用类型的热部署，增加了 Mbean 服务的热部署。此外，实现了远程部署的功能，将热部署系统设计为客户端和服务器端两部分，用户可以通过客户端将应用热部署到远端的服务器上。与传统的部署机制相比，本文提出的可扩展热部署机制具有如下特点：提高可用性，当部署一个组件时，系统不需要被中断；提高系统中组件的可维护性；增强 Web 应用服务器的可扩展性，有助于灵活的支持新的应用扩展；提高系统的灵活性，使可部署的服务富于变化。下一步的研究工作包括对模块结构的形式化检查，以减少动态装载时错误的发生。

参考文献

1 陈宁江，金蓓弘，范国闯. 多层企业应用的关键：J2EE Web 应用服务器. 计算机科学，2003，30(1)
2 Sun microsystems. Java (tm) 2 Platform Enterprise Edition Specification, v1.4, 2002.8
3 Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wsely, 1994
4 Sun Microsystems. Java™ Management Extensions Instrumentation and Agent Specification, v1.0, 2000.6
5 Gosling J, et al. Java Language Specification, Second Version. Addison-Wesley Pub Co; Sep. 1996