广西科学院学报
Journal of Guangxi Academy of Sciences
2006,22(4):266～268,271
Vol. 22,No. 4 November 2006

一种可动态重配的 JDNS 名字服务器
A Dynamically Reconfigurable JDNS Naming Server

陈宁江[1],李 都[2]
CHEN Ning-jiang[1],LI Du[2]

(1.广西大学计算机与电子信息学院,广西南宁 530004;2.中国科学院软件研究所软件工程技术中心,北京 100080)
(1. School of Computer, Electronics and Information, Guangxi University, Nanning, Guangxi, 530004, China; 2. Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing, 100080, China)

摘要:针对现有名字服务器的不足,利用 Java 语言提供的动态代理、拦截器等机制,提出一种可动态重配的名字服务器。这种服务器使名字服务具有较好的动态可配置性和适应性。

关键词:名字服务器 反射 拦截器 RMI 重配性
中图法分类号:TP393 **文献标识码**:A **文章编号**:1002-7378(2006)04-0266-03

Abstract: To deal with the shortcomings of current naming servers based on RMI mechanism, a dynamically reconfigurable naming server called JNDS is presented by use of save dynamic proxy and interception mechanism. It makes naming server to have better reconfiguration and adaptability.

Key words: naming server, reflection, interceptor, RMI, reconfiguration

名字服务器提供了分布式系统中实体(如用户、机器、组件和对象等)的名字、属性、位置和管理等信息的统一描述方法,这种方法可以将分布式系统中的实体与其名字相关联,从而可以通过名字方便地找到对应实体[1]。目前应用服务器的名字服务器大多使用基于 Java RMI 机制[2]实现的,这种方式的特点是实现简单有效。但是,随着应用服务器需求的发展,Java RMI 本身的局限性使得基于 Java RMI 机制的名字服务器不能很好地满足应用对运行时的可配置性、灵活性和服务质量等非功能性的需求[3]。为此,我们综合引入具有反射能力的 Java 动态代理[4]、拦截器[5]等机制,提出了一种名字服务器模型——JDNS(Java Dynamic Naming Server),这种名字服务器可以使名字服务具有更好的动态可配置性和适应性。

1 JDNS 名字服务器系统模型

JDNS 名字服务器的系统模型如图 1 所示。它在基于 Java RMI 名字服务器的系统模型中增加了一个代理层,JDNS 名字服务器的客户端和服务器端程序都不再直接和基于 Java RMI 名字服务器的系统交互,而是和代理层交互。代理层包括如下几个组件:

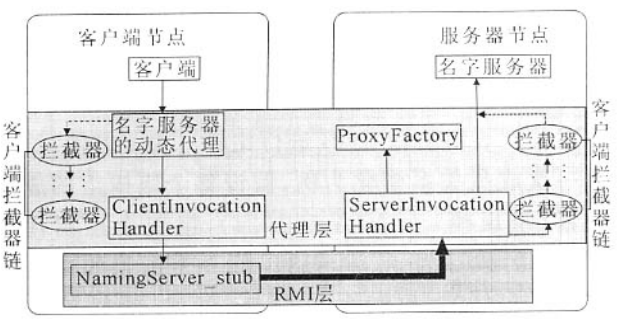

图 1 JDNS 系统模型

(1)拦截器链。服务器端和客户端分别设置两组拦截器对象,它们构成一个链状结构。用户请求/响应在拦截器链中经过,每个拦截器根据拦截到的信息来处理某一功能。客户端拦截器链位于客户端,用

收稿日期:2006-07-17
作者简介:陈宁江(1975-),广西南宁人,博士,主要从事网络分布计算和中间件研究。

于在客户请求中封装特定的信息和处理逻辑。位于服务器端的服务器端拦截器链对到达的客户请求进行预处理后,再传给名字服务器;在服务器的响应被返回给客户之前,在响应中加入特定的信息。常用的拦截器有安全校验拦截器、负载平衡策略拦截器、缓存策略拦截器等。各个拦截器既可以在系统启动时根据配置文件静态设置,也可以在运行时进行动态装卸。

(2)名字服务器的动态代理(NamingServerDynamicProxy)。由名字服务器中的 ProxyFactory 在运行时生成,并通过 NamingServer_stub 传到客户端。名字服务器的动态代理的主要作用有两个方面:(i)它将客户端的请求传递到服务器端,并将返回值返回给客户;(ii)它负责激活拦截器链,将客户端的请求依次经过各个拦截器的处理,之后再通过 NamingServer_stub 调用名字服务器上对应的方法;名字服务器的动态代理还可封装用于创建和配置拦截器的信息。

(3)调用句柄(InvocationHandler)。ServerInvocationHandler 是服务端的请求调用入口,并负责管理服务器端的拦截器。当客户请求调用到达服务器时,将先进入 ServerInvocationHandler;ServerInvocationHandler 将请求通过服务器的拦截器链的处理,然后再在 NamingServer 中进行普通的名字服务操作。客户端的调用句柄 ClientInvocationHandler 是客户端请求处理的入口,名字服务器的动态代理发起的方法调用都会先进入这个对象,然后再转发给客户端的拦截器链。此外,调用句柄还具有拦截器工厂的功能,根据配置来创建新的拦截器,并将之加入拦截器链中。

2 JDNS 名字服务器的动态配置

JDNS 名字服务器中所有的拦截器都遵循统一的 AInterceptor 接口,其中的主要属性和方法如表 1 所示。用户可以使用 JNDS 提供的缺省的拦截器,也可以遵循 AInterceptor 接口自己编写拦截器,然后通过配置加到客户端或者服务器端的拦截器链中。拦截器的信息和链结构在 JDNS 的配置文件 jdns_config.xml 中设置。JDNS 通过对拦截器链的动态配置实现名字服务器功能模块行为和参数的调整。拦截器链的配置包括:(1)在系统启动时,将自动读取 jdns_config.xml 中的信息来决定要装载和启动的拦截器,装载的先后顺序与配置的顺序一致;(2)在运行时,JDNS 系统的拦截器链结构进行修改,调用句柄可以在运行时生成新的拦截器实例,并将其加入到拦截器链中,也可以根据拦截得知的调用功能需求,在拦截器链中增加或者删去若干拦截器。相对于传统的名字服务器,JNDS 的优势在于其能够实现运行时的动态配置。

表 1 AInterceptor 接口的主要属性和方法

属性/方法名	说明
属性:nextInterceptor	记录拦截器链上与本拦截器相邻的下一个拦截器的信息
方法:init()	初始化拦截器的设置
方法:start()	将拦截器加入到拦截器链中
方法:stop()	将拦截器从拦截器链中删除
方法:setNext()	设置拦截器链的下一个拦截器
方法:getNext()	获得拦截器链的下一个拦截器
方法:invoke()	invoke 方法负责执行本拦截器所要进行的处理,并激活下一个拦截器

用户为 JDNS 制定各种动态重配策略。这些策略规定了该名字服务器在运行环境(包括当前的环境状况和服务器的工作情况)下将要使用的拦截器链的配置。名字服务器在响应客户端的请求而生成一个服务器的动态代理时,都会先检查名字服务器的运行环境,再根据给定策略,决定对当前的拦截器链进行修改或重配。

JDNS 典型的动态重配过程见图 2,主要步骤包括:

(1)从 Proxy_stub 获得一个从名字服务器的动态代理对象,该对象中包含了 ClientInvocationHandler 的信息;

(2)客户端的调用请求(InvocationContext 封装了请求信息)通过名字服务器的动态代理对象进入 ClientInvocationHandler 对象,获取客户端拦截器链的信息;

(3)名字服务器的动态代理向客户端拦截器链发出调用;

(4)客户端拦截器链中的各个拦截器依次处理客户请求,在 InvocationContext 中加入各自的信息;

(5)NamingServer_stub 传递调用到服务器端;

(6)ServerInvocationHandler 接受客户请求,将 InvocationContext 传递给服务器端拦截器链,服务器端拦截器依次取出 InvocationContext 中的信息,进行处理;

(7)各个服务器端拦截器处理完请求后,DynamicNamingServer 进行相关的名字操作;

(8)ProxyFactory 根据需要生成代理实例,随同

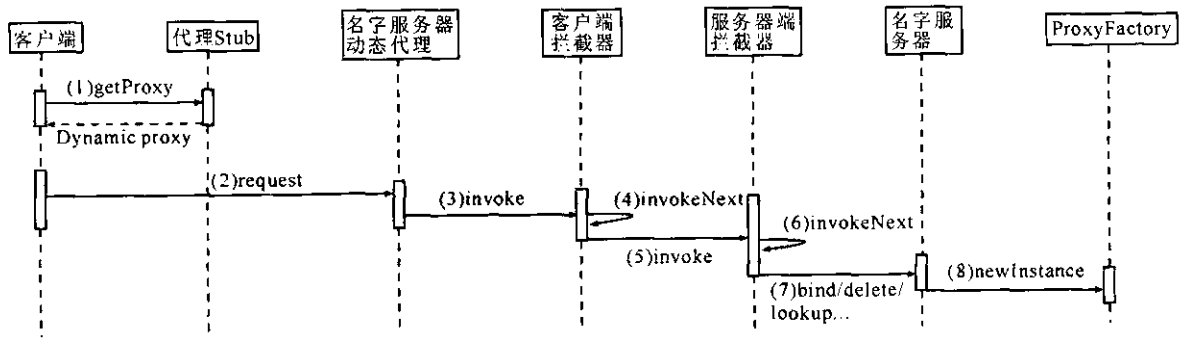

图 2 JDNS 名字服务器的动态配置

名字服务操作结果传到客户端，至此，一个请求调用过程完成。

JDNS 的另一种动态重配过程是当新的一个请求到来时，ProxyFactory 根据新生成的代理对象，封装适当的服务处理逻辑和客户端拦截器信息；ClientInvocationHandler 读取代理中的信息，若有重新配置或创建拦截器的请求，则重新配置或生成新的拦截器链，从而也实现了名字服务器的运行时动态重配。

3 JDNS 名字服务器应用实例

JDNS 名字服务器可以为失效恢复、负载平衡、安全控制等多种功能提供灵活性的支持。我们以名字服务的负载平衡为例说明 JDNS 名字服务器的应用。传统的 Java RMI 名字服务器在运行时只能采取单一的负载平衡策略，而 JDNS 名字服务器则可以实现负载平衡策略的动态配置。

以下给出 jdns _ config. xml 的主要配置内容，其中设置了一个客户端拦截器（RR _ LBInterceptor）和一个服务器端拦截器（LoadMetricInterceptor），前者负责根据给定的负载平衡算法选出要将客户请求发往哪个服务器，后者负责获取和计算本机器的负载信息。客户端存在包含负载平衡逻辑的代理 LBProxy，它确定要将客户请求发往哪个拦截器。我们定义了两种面向不同负载状况的负载平衡策略，即面向轻负载情况的 Round-Robin 平衡策略和面向重负载情况的负载权重优先策略，将它们分别包装在 RR _ LBInterceptor 和 LWF _ LBInterceptor 中。LBProxy 是负责 JDNS 动态负载平衡的动态配置过程如下：

(1)LoadMetricInterceptor 定期计算当前负载度量 *currentLoad* 的值：$currentLoad = f(M_{cpu}, M_{memory})$ 其中，M_{cpu} 和 M_{memory} 分别为 CPU 和内存占用度量值，f 表示综合二者因素的度量函数，例如，$f = \alpha \cdot M_{cpu} + \beta \cdot M_{memory}$，其中 $\alpha + \beta = 1$；

(2) 设 T_{Load} 为负载状况的阀值，ProxyFactory 根据 *currentLoad* 的值，决定客户端需要使用的 LBInterceptor，即：若 $currentLoad < T_{Load}$，则将 RR _ LBInterceptor 封装进 LBProxy；若 $currentLoad > T_{Load}$，则将 LWF _ LBInterceptor 封装进 LBProxy；

(3) ClientInvocationHandler 接收到 LBProxy 后，若当前客户端拦截器链中不包含 LBProxy 中的拦截器，则生成新的拦截器实例，替换原来的拦截器。

```
<ClientInterceptors>
    <interceptor>
<! —初始配置为 RR _ LBInterceptor  —->
<name>RR _ LBInterceptor </name>
<code>com. once. jndi. RR _ LBInterceptor. class
</code>
  </interceptor>
</ClientInterceptors>
<ServerInterceptors>
    <interceptor>
    <name>LoadMetricInterceptor</name>
    <code>com. once. jndi.
LoadMetricInterceptor. class</code>
    </interceptor>
</ServerInterceptors>
<Proxies>
  <proxy>
    <name>LBProxy</name>
    <parameter>LoadThreshold = 50
    </parameter>
  </proxy>
</Proxies>
```

（下转第 271 页）

[3] VASAKI PONNUSAMY, ETTIKAN KANDASAMY KARUPPIAH, ROSNI ABDULLAH. Anycast Group Membership Management Protocol[M]. IEEE, 2003: 1052-1056.

[4] WANG YUE, ZHANG LI, WEI YAN. Research on IP Anycast Secure Group Management: Asia Pacific Advanced Network 2003[C]. Korea: [s. n.], 2003: 49-55.

[5] 陈恺,许勇. 安全多播中基于成员行为的 LKH 方法[J]. 软件学报, 2005, 16(4): 601-608.

[6] SUVO MITTRA. Iolus: A Framework for scalable secure Multicasting, ACM SIGCO MM'97 [C]. Cannes: [s. n.], 1997: 277-288.

(责任编辑:凌汉恩　邓大玉)

(上接第 268 页)

4　结束语

我们针对基于 Java RMI 机制的名字服务器存在的不足进行了改进,提出了一个可动态重配的名字服务器 JDNS,它通过使用动态代理和拦截器等机制,使名字服务器在灵活性、可配置性和可扩展性等方面具有优势,主要表现在:(1)具有动态适应能力,由于拦截器链结构可以动态修改和动态代理在运行时生成,因此 JDNS 名字服务器可以根据环境状况和应用需求进行调节;(2)JDNS 名字服务器的功能模块可以动态裁减,它在运行时决定加载哪些拦截器和拦截器链的结构如何组织,从而有助于名字服务器提供按需服务能力;(3)具有良好的可扩展性。用户通过实现 AInterceptor 接口和动态代理,可以将自行实现的功能引入到名字服务器中。

使用运行时动态生成的代理代替预编译生成的存根,会给名字服务器带来一些开销。但是,考虑到 JDNS 名字服务器的应用场合和所获得的效益,这种性能上的开销是可以接受的。JNDS 名字服务器已经在中科院软件研究所开发的 OnceAS 应用服务器[6]中得到应用,为 OnceAS 服务器的动态重配和服务质量保障能力提供了一个良好的基础。

参考文献:

[1] COULOURIS G, DOLLIMORE J, KINDBERG T. 分布式系统概念与设计[M]. 金蓓弘,译. 第 3 版. 北京:机械工业出版社, 2004.

[2] SUN Microsystems Inc. Java Remote Method Invocation Specification, JDK 1. 3[EB/OL]. [2006-07-17]. http://java. sun. com/products/jdk/rmi/.

[3] 范国闯,钟华,黄涛,等. Web 应用服务器研究综述[J]. 软件学报, 2003, 14(10): 1728-1739.

[4] SUN Microsystem Inc. Java Dynamic Proxy Classes [EB/OL]. [2006-07-17]. http://java. sun. com/j2se/1. 3/docs/guide/reflection/proxy. html.

[5] SCHMIDT D C, STAL M, ROHNERT H, et al. Pattern-Oriented software architecture: patterns for concurrency and distributed objects [M]. Volume 2. NY: Wiley & Sons, 2000.

[6] ZHANG W B, YANG B, JIN B H, et al. Performance tuning for application server OnceAS[M]//CAO JN, YANG LT, Guo MY, et al eds. Parallel and Distributed Processing and Applications. Berlin: Springer-Verlag, 2004: 451-462.

(责任编辑:凌汉恩　邓大玉)