文章编号:1001 - 9081(2006)07 - 1529 - 03

基于 Agent 的 Web Services 容器智能化研究

陈中育,吴建斌,王晓虎,叶荣华
(浙江师范大学 信息科学与工程学院,浙江 金华 321004)
(czy@ zjnu. cn)

摘 要:对 Web Services 容器的智能化进行研究,提出基于 Agent 的智能 Web Services 容器(IWSC),同时也给出了一种基于 IWSC 的 Web Services 动态组合机制。最后通过在局域网环境下基于 IWSC 的网上购书应用测试,表明 IWSC 与同类 Web Services 容器相比,在 Web Services 的智能动态组合方面具有明显优势。

关键词:Web Services;Agent;智能化;Web Services 容器
中图分类号: TP393.09 **文献标识码**:A

Research on the intelligence of Web Services container based on Agent

CHEN Zhong-yu, WU Jian-bin, WANG Xiao-hu, YE Rong-hua
(*College of Information Science and Engineering, Zhejiang Normal University, Jinhua Zhejiang* 321004, *China*)

Abstract: An Intelligent Web Services Container(IWSC) based on agent was put forward. Its software architecture was given. Based on IWSC, a mechanism of dynamic combination of Web Services was discussed. The simulation of ordering books based on IWSC indicates that IWSC has evident advantage on the intelligent dynamic combination of Web Services compared with similar Web Services containers.

Key words: Web Services; Agent; intelligence; Web Services container

0 引言

Web 应用体系经历了三个发展阶段。第一代 Web 应用体系结构模式是 Client/Server 模式。在此模式下,Client 端使用 SQL 语言通过 Web Server(Web 服务器)上的 CGI 直接访问数据库。此后,网络处理需求不断膨胀以及灵活性及适应性问题的出现,促使产生了第二代 Web 应用体系结构模式,即 Web Application Server(Web 应用服务器)模式。此模式中,Client 端通过 Internet 与 Web 服务器通讯,之后 Web 服务器再通过 Web 应用服务器上的 JDBC 接口访问数据库。

近年来,由于使用标准的 HTTP 和 XML 格式的规范化通讯技术以及面向服务计算(Service Oriented Computing, SOC)技术的发展,出现了第三代 Web 应用体系,即所谓"Web Services 平台体系"。此体系以 SOC 为基础,并以 SOAP, XML,WSDL 和 UDDI 分别作为其数据传输协议、消息传送格式、服务描述语言以及服务注册机构。Web Services 平台体系需要得到相应 Web Services 基础设施的支持。这个基础设施需要为 Web Services 提供一流部署、运行和管理环境。为此,文献[1]提出了"Web Services 容器"这一概念。文献[2]对这一概念作了进一步定义,认为它是一个存储 Web Services 的抽象层,类似于管理企业 beans 的 J2EE 容器。

目前的 Web Services 容器基本集中于简单静态调用,在智能化调用方面欠缺,不能满足组合 Web Services 和工作流支持的需求。而通过将 Agent 技术和 Web Services 容器相结合,可以提高 Web Services 容器的智能性。

1 基于 Agent 的智能化 Web Services 容器

1.1 IWSC 体系框架

基于 Agent 的智能化 Web Services 容器(IWSC)体系主要由接收/转发部件、Agent 工厂、合成服务执行器、Web Services 管理器和服务及 Agent 实例等组成,如图 1 所示。接收/转发部件负责响应用户的请求,并将服务的执行结果返回给用户。同时,它将根据消息判断是简单 Web Services 还是复合 Web Services 调用:如果是简单 Web Services,它将直接由 Web Services 管理器调用实例池或缓冲池中的 Web Services 实例,并将执行结果返回给客户;如果是复合 Web Services,它则调用 Agent 工厂产生相应的服务提供 Agent 来处理。所以,接收/装发部件也是 IWSC 的控制中心。

Agent 工厂主要实现服务提供 Agent 和服务外协 Agent 的创建和管理,可根据用户的需要为服务提供 Agent 建立执行规则,并由服务外协 Agent 与中介 Agents 协商,查询和调用远程 Web Services,从而实现 Web Services 的动态组合。而服务提供 Agent 可与 Web Services 管理器协作,实现调用本地 Web Services。所以,可将服务提供 Agent 和服务外协 Agent 作如下定义。

定义 1 服务提供 Agent(以 PA 表示)是一个二元组 $<S_{inter}, R>$,其中 S_{inter} 是容器所能提供的服务集合,R 是服务 Agent 与服务管理器之间的协作关系。

定义 2 服务外协 Agent(以 OA 表示)是一个三元组 $<S_{outer}, P, M>$,其中 S_{outer} 是已知的外协服务集合,P 是 S 中对

收稿日期:2006 - 01 - 04;**修订日期**:2006 - 03 - 29 **基金项目**:浙江省自然科学基金资助项目(Y104324,Y105092)
作者简介:陈中育(1964 -),男,浙江浦江人,副教授,主要研究方向:软件工程、计算机图形学; 吴建斌(1976 -),男,浙江衢州人,讲师,硕士,主要研究方向:软件工程、网络技术; 王晓虎(1974 -),男,浙江永康人,工程师,硕士,主要研究方向:软件工程、网络技术; 叶荣华(1971 -),男,浙江绍兴人,副教授,主要研究方向:软件工程.

应的服务提供 Agent 集合,M 是所知道的中介 Agent 集合。

合成服务执行器是组合 Web Services 的执行部件。当服务提供 Agent 根据客户要求形成相应的组合 Web Services 并第一次提交给合成服务执行器时,合成服务执行器将启动。它通过与服务提供 Agent 之间的协商实现组合 Web Services 的成功执行,从而满足客户需求。

服务管理器是 IWSC 的内部核心,主要作用是对 Web Services 进行部署、运行和管理。在 IWSC 内部,Web Servcies 主要有部署态和运行态两种存在形式。在部署态时,Web Services 以 Services 包(Archive,SAR)的形式存在,主要包含 Web Services 描述文件、Web Services 部署文件和 Web Services 程序文件等。在运行态下,对于经常被调用的 Web Servcies 实例,可以以缓冲池或实例池的形式存储。比如,对于无状态 Web Services,可以直接以一定的数量存在于实例池中。这样,当同时有多个请求到来时,可以实现并发调用,加快访问速度。而对于有状态 Web Services,由于状态的存在使得同一 Web Services 的不同实例不能重用,所以将以缓冲池的形式存在。当某一实例在规定时间内还没有被使用时,将会从缓冲池中删除或撤换,这样可以大大提高执行效率。

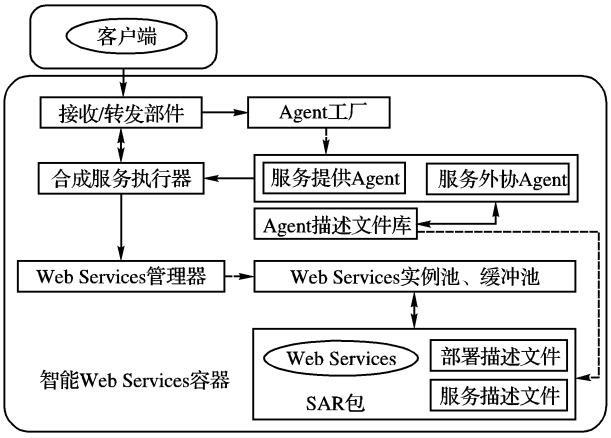

图 1 IWSC 体系结构

1.2 基于 IWSC 的 Web Services 动态组合机制

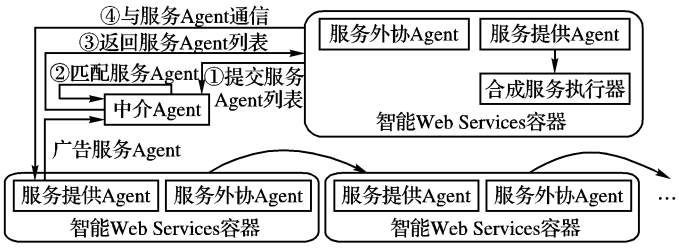

图 2 基于 IWSC 的动态 Web Services 组合机制

在前面所设计的 IWSC 基础上,下面将给出基于 IWSC 的 Web Services 动态组合机制。在组合之前,每个 Web Servcies 部署时,Agent 工厂会为新部署的 Web Services 生成一个对应的 Agent 实例,并将新部署的 Web Services 封装为此 Agent 的技能。此 Agent 就是我们前面所讲的"服务提供 Agent",服务提供 Agent 会自动将封装为其技能的 Web Services 发布到所知道的中介 Agents 中。

定义 3 组合 Web Services(简单表示为 S)是一个按照业务需求并以一定逻辑关系形成的服务集合 $\{W_1, W_2, \cdots, W_n\}$,其中单个服务元素 W 可以用一个四元组表示,$W = <D, OP, M_{in}, M_{out}>$。D 是服务的描述,OP 表示服务所提供的操作,$M_{in}$ 和 M_{out} 是服务执行时输入和输出的消息接口。

组合过程如下:

1) 当容器接收到关于组合 Web Services(以 S 表示)的请求后,将 S 的组合和执行作为容器的目标。

2) 若 $S \cap S_{inter} \neq \varnothing$,且 $S \subseteq S_{inter}$(S_{inter} 是容器所能提供的服务集合),则根据客户需求由容器的服务提供 Agent 与服务管理器协商形成满足相应的 S,并执行 S 将结果返回给客户。

3) 若 $S \cap S_{inter} \neq \varnothing$,但 $S \supseteq S_{inter}$,则意味着容器不能提供组合 Web Services(S)中所要求的部分服务 S_{part}。对于这些 S_{part},这时服务提供 Agent 需要与服务外协 Agent 进行协商。

4) 服务提供 Agent 与服务外协 Agent 协商,服务外协 Agent 通过查看以往记录,即已知的外协服务集合 S_{outer}。若 $S_{part} \subseteq S_{outer}$,则服务外协 Agent 从 P 中返回对应服务提供者给服务提供 Agent,服务提供 Agent 可据此形成相应的组合服务流程;否则,转 5)。

5) 对于那些容器内部不提供并且服务外协 Agent 也未知的服务,服务外协 Agent(OA)需要与中介 Agent(M)协商获取服务 Agent 列表。这个过程如图 2 所示。首先,OA 向 M 提交服务请求;M 匹配服务 Agent;M 将匹配的服务 Agent 列表返回给 OA;OA 与服务 Agent 进行通信,从而形成相应的组合服务 S。同时,OA 记录组合经验,已被下次使用。

6) 若 $S \cap S_{inter} = \varnothing$,则转 4)执行。

7) 若以上过程失败,则服务提供 Agent 将向客户返回一个失败消息,或者服务提供 Agent 主动找到另一个容器中的服务提供 Agent,并将服务委托给它。

当 Web Services 按照客户要求被成功组合时,服务提供 Agent 将提交给合成服务执行器去执行。合成服务执行器将服务映射到相应服务提供者,并在组合 Web Services 的执行过程中,合成服务执行器将检测是否异常。当出现异常时,合成服务执行器将与服务提供 Agent 协商,寻找可替代的服务提供者并再次尝试执行。所以,合成服务执行器与服务提供 Agent 之间的交互过程是一个不断协商的过程,也是组合 Web Services 流程的不断修复过程。

1.3 实现

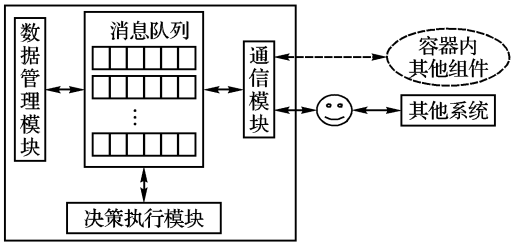

图 3 Agent 的组件组成(虚线部分为服务提供 Agent 的特有交互)

基于以上的实现框架和思想,我们实现了 IWSC 的简单原型系统。在这个原型系统里,服务外协 Agent 和服务提供 Agent 是智能化调用的关键部件,它们主要包括消息队列模块、数据管理模块、决策执行模块和通信模块。消息队列模块处理 Agent 中的消息缓冲和发送;数据管理模块负责数据映射,历史数据缓存以及其他数据的管理等功能;而决策执行模块则负责与 Agent 以及其他容器内部组件协作时的决策,以及根据决策采取相应的动作;通信模块处理与其他 Agent 之间的通信,对于服务外协 Agent 还负责与中介 Agent 之间的协作。其具体组成如图 3 所示。

在局域网环境下,我们在 IWSC 上模拟了网上订购图书

的过程。表 1 和表 2 是交互过程 IWSC 内服务提供 Agent 和服务外协 Agent 的部分消息记录。

表 1　服务提供 Agent 的消息记录

序号	消息动词	发送者	接受者	接收时间	消息内容	备注
00010	Ask	Supply-Agent	Out-Agent	2006/1/25-10:24:11	< event-item > < outer-s-agent > < function > orderbook </function > </outer-s-agent > </event-item >	请求外协
00011	Inform	Outer-Agent	Supply-Agent	2006/1/25-10:24:19	< event-item > < services-result > < item > < service-name > orderbook </service-name > < ser-param > bookname < ser-param > < ser-param > number < ser-param > </item > … </services-result > </event-item >	得到返回结果
…	…	…	…	…	…	…

表 2　服务外协 Agent 的消息记录

序号	消息动词	发送者	接受者	接收时间	消息内容	备注
00010	Ask	Outer-Agent	Mid-Agent	2006/1/25-10:24:11	< event-item > < s-agent-search > < function > orderbook </function > </s-agent-search > </event-item >	与中介 Agent 协作
00011	Inform	Mid-Agent	Outer-Agent	2006/1/25-10:24:13	< event-item > < result > < item > < agent-name > a-supply-agent </agent-name > < address > … </address > </item > … </result > </event-item >	得到对应 Agent 列表
00012	Ask	Outer-Agent	A-Supply-Agent	2006/1/25-10:24:16	< event-item > < service > orderbook </service > </event-item >	请求服务细节
00013	Inform	A-Supply-Agent	Outer-Agent	2006/1/25-10:24:18	< event-item > < service-name > orderbook </service-name > < ser-param > bookname < ser-param > < ser-param > number < ser-param > … </event-item >	返回服务名及参数
…	…	…	…	…	…	…

这次试验仅仅测试了 IWSC 对 Web Services 的简单智能化调用功能,在 Web Services 的复杂组合应用时仍存在很多问题,有待进一步研究。

2　相关比较与将来工作

IWSC 与目前现有的 Web Services 容器[3,6,16]相比较,基于 Agent 技术,增加了容器的智能特性,主要体现在以下几个方面:

1) 支持客户的多种 Web Services 调用方式。在 Web Services 的简单调用和组合应用两种调用方式下,IWSC 能够自适应调整。

2) 在分布式的环境下,支持 Web Services 的智能化组合。IWSC 能够根据客户请求,由服务提供 Agent 与服务外协 Agent 相互协作匹配服务,形成相应的组合 Web Services 流程。

3) 可以实现动态组合。在组合 Web Services 的执行过程中,IWSC 可以检测异常,如有异常则通过服务提供 Agent 与服务外协 Agent 找到可替代的服务提供者,修复服务流程。

除此之外,IWSC 还采用实例池和缓冲池两种实例存储机制,加快了 Web Services 的执行速度,提高了容器的性能。

参考文献:

[1] BERNHARD B. A Web Services Container [EB/OL]. http://www.sys-con.com/webservices/article.cfm?id=82, 2004.

[2] VENKATAVARADAN R, DHESIASEELAN A. Managing Web Services: A Container-Based Approach[EB/OL]. http://www.developer.com/java/web/article.php/2230731, 2005.

[3] IONA. Getting Started with Web Services[EB/OL]. http://iona.com/support/docs/e2a/asp/5.0/xmlbus /GettingStarted/GettingStarted.pdf, 2001.

[4] KLEIJNEN S, RAJU S. An Open Web Services Architecture[EB/OL]. http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=26, 2003-03.

[5] W3C. Web Service Management: Service Life Cycle[EB/OL]. http://www.w3.org/TR/wslc/, 2004-02.

[6] Systinet Corp. WASP Server for Java[EB/OL]. http://www.systinet.com/doc/wasp_jserver/waspj /index.html, 2004.

[7] WOOLDRIDGE M, JENNINGS NR. Intelligent Agents: Theory and Practice[J]. Knowledge Engineering Review, 1995, 10(2): 115-152.

[8] 刘大有,杨鲲,陈建中. Agent 研究现状与发展趋势[J]. 软件学报, 2000, 11(3): 315-321.

[9] 高济,王进. 基于 Agents 的软件合成框架 ABFSC[J]. 计算机学报, 1999, 22(10): 1050-1058.

[10] JENNINGS NR, FARATIN P, PARASONS LAS, *et al.* Automated negotiation: prospects, methods and challenges[J]. International Journal of Group Decision and Negotation, 2001, 10(2): 199-215.

[11] NISHIDA T. Social Intelligence Design for the Web[J]. IEEE Computer, 2002, 35(11): 37-40.

[12] WOOLDRIDGE M, JENNINGS NR. Intelligent Agents: Theory and Practice[J]. Knowledge Engineering Review, 1995, 10(2): 115-152.

[13] NWANA HS. Software Agents: An Overview[J]. Knowledge Engineering Review, 1996, 11(3): 205-244.

[14] 史忠植, 张海俊, 何清. 智能战略决策支持平台——多主体环境 MAGE[A]. 战争复杂性与信息化 战争模拟研讨会论文集[C], 2003. 88-96.

[15] 舒剑, 胡春明, 葛声, 等. Web Service 运行管理平台的研究与实现[J]. 计算机研究与发展, 2004, 41(3): 441-450.