基于服务架构的多 Agent 企业应用集成模型的研究

冯文辉[1] 李吉桂[2] 张永华[3]

(广东技术师范学院计算机科学学院 广州 510665)[1] (华南师范大学计算机学院 广州 510631)[2]

(广州大学商学院 广州 510006)[3]

摘 要 分析了常见的企业应用集成方案的现状和缺点,提出了一种基于服务架构的多 Agent 企业应用集成模型,最后设计实现模型原型,成功将某电子商务系统与 ERP 系统进行集成。

关键词 面向服务架构,企业应用集成,多代理

Research of Multi-agent Implemented EAI Model with SOA

FENG Wen-hui[1] LI Ji-gui[2] ZHANG Yong-hua[3]

(School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China)[1]

(School of Computer, South China Normal University, Guangzhou 510631, China)[2]

(School of Business, Guangzhou University, Guangzhou 510006, China)[3]

Abstract Analyzing the existing enterprise application integration solutions' application and their shortcomings, this paper presented a service-oriented architecture based, multi-agent implemented enterprise application integration model. Finally, we implemented the model, integrating an e-commerce system and an ERP system successfully.

Keywords SOA, EAI, Multi-agent

1 引言

随着计算机信息技术的发展、Internet 的普及,大多企业都开发了适合自己的各种企业信息管理系统,这些管理信息系统在企业某些部门或业务的管理上发挥了重要作用。但是企业现有的这些管理信息系统,因为投入的时间、部门、开发公司不同,实现技术不统一等原因,造成系统各自独立运行,数据不能共享,各自业务流程不能自动衔接,企业内部出现了自成体系的信息化孤岛,企业与企业之间的电子商务活动更加难以实现。

为了解决以上这些问题,产生了企业应用集成(Enterprise Application Integration,简称 EAI)[1]技术,企业应用集成用以解决企业现有多种应用系统的互连互通、数据共享、业务流程统一的集成问题。

2 现有企业应用集成方案的分析

纵观企业应用集成发展过程,传统的企业应用集成解决方案主要有以下 4 种[2]:

1. 点对点集成解决方案

早期企业启用的应用系统个数较少,一般多采用的是点对点的系统集成方案。点对点的解决方案是指对需要进行信息交互的两个应用之间进行集成。它的缺点:需要手工编写异购应用系统和平台的接入,代码量大;应用系统之间耦合度高,不易扩展;接入接口不可复用,对技术人员要求高;当接入应用系统较多时,开发效率降低,维护成本提高。

2. 消息代理解决方案

消息代理解决方案是一种在数据源与目的地之间移动数据使信息处理流畅的软件技术方案,数据源与目的地包括已有的应用、文件、数据库、对象(如 CORBA, COM)、硬拷贝输出及 Web 客户端等。它的缺点:消息代理与应用系统之间的交互通过适配器来连接,消息代理或者应用系统发生改变,适配器都要随之改变,应用与集成代理的结合过于紧密,可移植性较差;没有业务流程管理机制,不易扩展。

3. 集线器型解决方案

集线器型解决方案是利用中间件工具控制数据的传输和交换,控制数据和事务的业务规则也集中到了该中间件中,如应用服务器、对象请求代理、消息代理等。这种集成解决方案的缺点:部门级的应用,难于解决企业全局的业务流程整合;业务流程复用性差,不便扩展。

4. 基于消息中间件型解决方案

消息代理中间件解决方案是使用消息代理中间件提供应用集成所必需的数据的递送、收集、翻译、过滤、映射和路由等功能,屏蔽不同的硬件平台、数据库、消息格式、通信协议之间的鸿沟与差异,提供应用到应用之间的高效、便捷的通信能力。它的缺点:每个应用都对应多个队列,管理维护不便;依赖于消息中间件,不同的消息中间件存在兼容和移植问题;没有充分利用现有的硬件资源;没有解决业务流程的复用和扩展问题。

这 4 种传统的应用集成方案都复杂、昂贵,并且不灵活,难以快速适应基于企业现代业务变化不断产生的需求。

近年来随着 XML 和 Web 服务的发展,出现了 Web 服务集成[3](WSI)。Web 服务集成解决了上述传统的集成方法的

到稿日期:2008-04-29

缺点，但是这种方式也有一些局限：(1)在创建数据，服务和流程模型时考虑得不多，可能导致在不同的 Web 服务中同一个词被赋予不同的含义。(2)应用是通过传输层或中间件 APIs 直接发送 SOAP[4] 消息，这为更换传输协议或中间件增添了困难。(3)无法在多个不同的 Web 服务集成项目中使用同一种安全架构，这可能将为单点登录等设施增加复杂性等等。

伴随着 Web 服务技术的成熟以及面向服务架构概念重新被赋予重要意义，面向服务架构的集成技术给企业应用集成带来了新的解决方案。企业服务总线[5](Enterprise Service Bus，ESB)的概念是从面向服务体系架构(Service-Oriented Architecture，SOA)发展而来的企业应用集成解决方法。企业服务总线是一种提供了开放的、基于标准的消息机制。它通过简单的标准适配器和接口，来完成粗粒度服务和其他组件之间的互操作。企业服务总线的缺点是天额外中转性能消耗，中央 ESB 的性能瓶颈和高可用性问题。

3 基于服务架构多 Agent 企业应用集成模型

鉴于以上现有的企业应用集成方案的缺点，本文提出了一种基于服务架构的多 Agent 企业应用集成模型。该集成模型以服务为基础集成架构，使用智能 Agent 来完成不同企业应用之间的集成。这种集成模型可以工作在分布式环境下，能较好地解决面向服务架构的企业服务总线集成方案的中央式 ESB 的性能瓶颈和高可用性等问题。

3.1 基于服务架构的多 Agent 企业应用集成模型结构

基于服务架构的多 Agent 企业应用集成模型中包含一个集成管理服务端和多个集成点。集成管理服务端和各个集成点在系统中处于对等的位置并共同构成分布式的企业应用集成模型结构。为了为企业应用集成提供全局路由信息和集中式的管理，本模型引入集成管理服务端，它负责同步和保存全局的消息路由信息，并实现对各个集成点的集中式管理。集成点则完成应用请求处理过程并直接将处理结果返回给应用请求者。这种集成模型利用集成管理服务端将任务分配给多个集成点来完成，解决了面向服务架构的企业服务总线中央式 ESB 的性能瓶颈的缺点，而且多个集成点同时工作，能使得集成效率提高，同时也增加了集成方案的容错性。利用智能 Agent 技术来实现模型的两个核心部分，使得模型更具智能性、实用性、健壮性。模型设计了两类 Agent：管理 Agent 和集成 Agent。它们分别用以完成集成管理服务端和集成点的各项功能。

基于服务架构的多 Agent 企业应用集成模型主要由集成管理服务端(管理 Agent)、多个集成点(集成 Agent)、UDDI[6] 和适配器 4 个组成部分。模型结构图如图 1 所示。

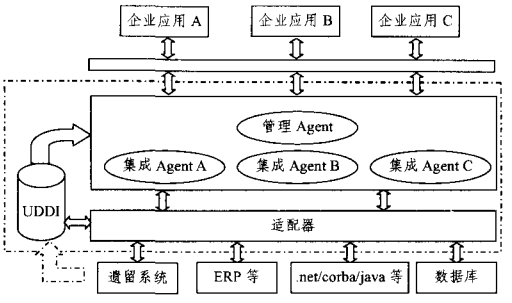

图 1 基于服务架构的多 Agent 企业应用集成模型结构图

模型中 4 个主要组成部分的功能：

(1)集成管理服务端(管理 Agent)：主要负责接检验收来自企业应用访问请求者的身份，选择集成点，集成点的管理与出错处理及恢复，集成平台的安全管理，服务质量等。

(2)集成点(集成 Agent)：主要负责接将接收到的管理 Agent 处转发的企业访问请求标准化，在知识库中查找对应的解决方案后，形成服务编排，调用服务，将结果信息非标准化后返回给对应的企业应用。

(3)适配器：适配器将企业遗留系统，ERP 或者 CRM 和数据库转换成标准的 Web Services，以便基于服务架构多 Agent 企业应集成平台及外部系统访问。

(4)UDDI：UDDI 用来发布使用适配器转换成功的 Web Services 和其他 Webservices 等。

3.2 管理 Agent 和集成 Agent 的组成及工作机制

在多 Agent 企业应用集成模型中，管理 Agent 和集成 Agent 是模型的重要组成部分。

管理 Agent 实现验证访问身份安全，集成点的选择、管理、出错处理及恢复。管理 Agent 主要包含的功能模块有身份验证、集成点分配模块、管理模块、集成点信息表、服务质量等。管理 Agent 详细功能模块示意图如图 2 所示。

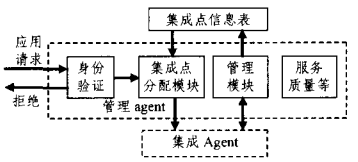

图 2 管理 Agent 功能模块示意图

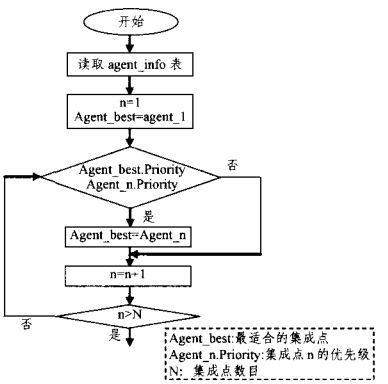

图 3 集成点选择算法流程图

管理 Agent 的工作机制：管理 Agent 在接收到应用请求后，首先分析请求应用的身份，身份验证通过后，管理 Agent 查询当前集成点信息表(集成点信息表包含集成点的路由地址，集成点当前任务分配状态等)，获取当前各集成点的状态，然后集成点分配模块根据集成点的状态来分配请求任务，集成 Agent 选择当前最适合的集成点的流程图如图 3 所示。集成点分配模块根据优先级来实现集成点分配，该分配模块在集成平台初始化时先将各个集成点分配同样的优先级，当集成点被分配一个任务后其优先级降低 1 级，当任务完成后再将优先级加 1 级，每次选择优先级最高的集成点进行分配。管理 Agent 选择适当的集成点后，将应用请求转发给该集成点。当集成 Agent 完成应用请求结果返回后，管理模块修改

(下转第 290 页)

以影响了置信区间。

结束语 本文通过对网络流量的分析，提出了一个实时检测 DDoS 攻击的检测模型，此模型具有算法简洁、易于实现等特点，适用于实时检测。通过实际数据检验表明，此方法可快速准确判别服务器是否受到 DDoS 攻击。由于该算法使用了正常流量的先验知识，从而提高了算法的适应能力，解决了突变正常流量与异常流量无法区分的难题，提高了服务器可能受到 DDoS 攻击的预报准确率。

我们下一步将研究如何利用路由器的流量分析与控制功能，建立集检测、控制为一体的全自动网络安全管理系统。

参 考 文 献

[1] Background on DDoS. http://www.ddos.com/index.php?content= products/ background.html December, 2007

[2] Rohrmair G T, Lowe G. Using data-independence in the analysis of intrusion detection systems. Theoretical Computer Science, 2005, 340: 82-101

[3] Carl G, Kesidis G. Denial-of-Service attack detection techniques. IEEE Internet Computing, 2006, 10(1): 82-89

[4] Li Ming. An approach to reliably identifying signs of DDOS Flood attacks based on LRD traffic pattern recognition. Computers & security, 2004, 23: 549-558

[5] Li Ming. Change trend of averaged Hurst parameter of traffic under DDOS flood attacks, Computers & Security, 2006, 25 (3): 213-220

[6] Carl G, Brook R R, Rai S. Wavelet based of Service detection. Computers & Security, 2006, 25: 600-615

[7] Hamdi M, Boudriga N. Detecting Denial-of-Service attacks using the wavelet transform. Computer Communications, 2007, 30: 3203-3213

[8] 肖志新，杨岳湘，杨霖. 基于小波技术的网络异常流量检测与实现. 计算机科学，2006，33(10)

[9] Feinstein L, et al. Statistical approaches to DDoS attack detection and response // DARPA information survivability conference and exposition proceedings. 2003, 1: 303-14

[10] 魏向荣，李之棠. DDoS 的协方差检测模型. 通信学报，2006，27 (11A): 72-75

[11] 李德全. 拒绝服务攻击. 电子工业出版社，2007: 1-16

[12] 帕普里斯 A，等. 概率、随机变量和随机过程 . 西安交通大学出版社，2004: 220-222

(上接第 287 页)

集成点信息表中相关项；当集成 Agent 出错时，管理 Agent 根据错误状况，选择重新分配请求或者恢复集成 Agent 重新执行。

集成 Agent 利用消息传递机制实现不同企业应用的集成，它是模型的核心组成部分。集成 Agent 所包含的功能模块有：集成 Agent 核心引擎、标准(消息标准机制)、服务编排、知识库等。集成 Agent 的核心引擎消息转换模块则包含消息接收器、消息发送器、消息连接器、消息转换器等模块。消息路由则包含输入路由和输出路由两个部分。其详细功能模块示意图 4 所示。

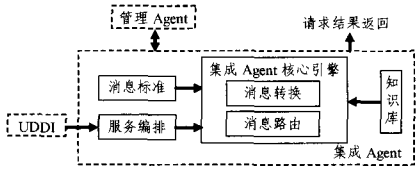

图 4 集成 Agent 功能模块示意图

集成 Agent 的工作机制：集成 Agent 接收到管理 Agent 的转发的应用请求消息，消息转换模块读取应用请求消息，并将消息标准化。集成 Agent 分析应用请求类型，根据集成 Agent 知识库(包含集成 Agent 以前编排的服务的历史记录)中包含的集成 Agent 提供常用服务，选择对应的 Web Services，若没有对应的服务则通过服务编排模块生成对应的 Web Services，并将其加入到知识库中。调用 Web services 获得返回结果，最后将结果非标准化发送给应用请求者。

3.3 模型实现

为了证明基于服务架构的多 Agent 企业应用集成模型的设计思想，进行了原型平台的实现。利用该原型平台对某企业电子商务协商系统和 ERP 进行集成[7]。平台以 Web Services 为基础，将企业 ERP 系统中的生产计划模块、库存管理模块中某些功能通过 SOAP/XML 适配器转换为 Web Servi-ces 发布到企业私有 UDDI 注册中心。管理 Agent 和集成 Agent 之间的通信采用 JMS[8] 异步通信机制。管理 Agent 的集成点分配模块初始化时将各个集成点分配同样的优先级 20，每次选择优先级最高的集成点进行分配。集成 Agent 的核心引擎消息转换模块则实现了针对不同协议(JMS, SOAP, HTTP 等)的消息接收器、消息发送器、消息连接器和消息转换器。消息路由实现了输入路由和输出路由两个部分。利用 BPEL4WS[9] 表示的 Web Services 业务流程作为集成 Agent 知识库。通过对电子商务系统与 ERP 系统之间的数据访问，检验了智能企业应用集成平台模型的可行性，结果显示模型能较好地实现异构应用之间互连互通、数据共享、业务流程统一的集成。

结束语 结合基于以服务为集成架构的设计理念，本文给出了一种基于服务架构的多 Agent 企业应用集成模型，用智能 Agent 技术设计了模型中两个重要的部分：集成服务端和集成点，最后实现了模型的原型，原型能实现异构应用的连接、数据共享、业务流程统一。在未来的工作中，我们将从以下几个方面逐步完善模型：集成 Agent 核心引擎静态调用 Web Services，为了增加集成中心的灵活性考虑采用动态调用的方式实现；管理 Agent 对应用请求仅仅做简单的身份验证，应解决消息在网络传递过程中安全性不高等问题。

参 考 文 献

[1] Mann J. Approaches to Enterprise Application Integration [EB /OL]. http://eai.ebizq.net, 2004-08-14

[2] SOA-解决中小企业应用集成的良药. http://www.sagapio.com/techsubject/soaplan.htm, 2006-04-1

[3] Newcomer E, Lomow G. Understanding SOA with Web Services 中文版. 电子工业出版社，2006

[4] W3C. SOAP Version 1.2 Part 1 Messaging Framework —— W3C Recommendation[EB/OL]. (2003-06). http://www.w3.org/TR/soap12-part1/

[5] Keen M, Bishop S, Hopkins A, et al. http://publib blboulder-1ibmlcom /Redbooks1nsf /Red PieceAbstracts/ sg246346. html? Open Patterns: Im Plementing an SOA using an Enter Prise Service Bus EB /OL1200417125

[6] UDDI.org White Paper. The Evolution of UDDI. http://www.uddi.org/specification.html. 7/19/2002

[7] 冯文辉. 面向服务架构的协商系统与企业 ERP 集成研究. 华南师范大学硕士学位论文. 2007

[8] Monson-Haefel R, Chappell D A. Java Message Service. O'Reilly. 2001

[9] Chafle G, Chandra S, Mann V, et al. Decent ralized Orchest ration of Composite Web Services. WWW2004, New York, USA, 2004: 134-143