文章编号: 1001- 9081(2002) 04- 0006- 03

中间件与分布式计算

蒋雄伟, 马范援
(上海交通大学 计算机科学与工程系, 上海 200030)

摘 要: 首先阐述了中间件的重要性及其概念, 然后结合分布式计算模式的演变历史, 对中间件技术的发展过程进行了介绍和分析, 最后对中间件技术的发展趋势进行了探讨。

关键词: 中间件; 分布式计算; 分布式对象; Web Services
中图分类号: TP311.11 文献标识码: A

MIDDLEWARE AND DISTRIBUTED COMPUTING

JIANG Xiong-wei, MA Fan-yuan
(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China)

Abstract: First this paper explains the importance and concept of middleware, then reviews its development accompanied by the evolution of distributed computing model. Finally the future trends of middleware are discussed.

Key words: middleware; distributed computing; distributed object; Web Services

分布式计算技术的发展经历了两条不同的技术路线。在 20 世纪 80 年代, 人们试图在计算机网络上部署全新的分布式操作系统。尽管产生了许多技术成果和实验系统, 但却没有被用户和市场接受。面对现实情况, 在 20 世纪 90 年代, 人们开始探讨新的解决方案, 研究在网络计算平台上部署分布计算环境(也称为中间件), 提供开发工具和公共服务, 支持分布式应用。业界普遍遵循这一技术路线, 产生了一系列行之有效的技术和广为用户接受的中间件产品[1]。

1 中间件的概念

1.1 为什么需要中间件

在中间件产生以前, 开发者不得不直接解决许多很棘手的问题, 如多种操作系统、多种网络协议、多种数据库、性能、效率、安全等等。应用开发过程中大约 70% 的时间和精力用于解决这些与业务逻辑没有直接关系的难题。因此有必要将分布式应用软件所要面临的共性问题进行提炼、抽象, 形成一个可复用的软件部件, 供应用软件重复使用。

1.2 什么是中间件

中间件的应用范围十分广泛, 针对不同的应用需求涌现出了多种各具特色的中间件产品, 因此给中间件一个广义的定义或许比较恰当: 中间件是处于应用软件和系统软件(操作系统、网络协议、数据库等)之间的一个软件层, 它屏蔽了环境底层的复杂性, 提供给应用开发者统一的、功能强大的 APIs, 使应用开发者只专注于业务逻辑的开发, 快速地开发出可靠、高效的企业级分布式应用。

2 中间件的发展历程

中间件的发展历程是和分布式计算模式的演变过程紧密联系的。

2.1 中间件与两层 C/S 计算模式

中间件最初是围绕数据库访问模型(即两层 C/S 计算模式)发展起来的。在两层应用模型中, 一个"胖"客户直接访问某个数据库管理系统。SQL 标准提供了一种通用语言来访问数据库, 但是各数据库厂商对 SQL 进行的扩展又阻碍了这种通用性。随后, ODBC 建立了一个事实上的标准, 使得我们可以使用同一种语言与不同的数据库进行交互。ODBC 就是一种中间件, 称为数据库中间件。数据库中间件主要用于需要从多个异构数据库中获取数据的决策支持系统。

2.2 中间件与多层 C/S 计算模式

两层 C/S 分布式计算模式缺乏可伸缩性、可移植性、性能差、效率低。为了解决两层 C/S 模式存在的不足, 提出了三层或多层应用体系结构。在多层体系结构中, 业务逻辑从客户端分离出来移到中间层, 由中间层处理客户端调用, 访问数据库服务器。

为支持这种多层结构的应用模型, 出现了相应的中间件。早期的支持三层结构模型的中间件只是提供了一个通讯协议的较高层次的抽象, 开发者可以使用简单的 API 进行应用之间的通讯。但是, 作为一个真正实用的中间件, 还必须提供命名、安全、事务、灵活的通讯方式、面向对象、容错、负载均衡等等服务, 对这些服务的支持程度可以用来区分不同的中间件产品。

除了上述数据库中间件外, 目前市场上的中间件产品大致可分为远过程调用中间件、消息中间件、事务处理中间件和分布式对象中间件。需要指出的是这些中间件并不是可相互替代的, 而是各有所长, 可以单独使用也可以集成使用, 并且有逐步统一的趋势。

收稿日期: 2001-10-29

作者简介: 蒋雄伟(1974-), 男, 江苏昆山人, 博士后, 主要研究方向: 分布式计算、中间件技术; 马范援(1942-), 男, 教授, 博士生导师, 主要研究方向: 电子商务、中间件技术、计算机网络系统。

2.2.1　远过程调用(RPC)

RPC 的思想很简单, 就是把本地的过程调用扩展到分布式环境。RPC 机制是同步的, 因而要求客户方和服务方均要能正常工作。虽然有些恢复机制可以在一个远程调用阻塞时转而去调用另一个服务器上的过程, 但这种路由方式会影响效率。也有的 RPC 机制采用异步方式, 客户方非阻塞地调用远端的过程, 但这种方式不标准而且很难采用。因此 RPC 最适合于开发小型简单的请求/应答模式的应用。

Open Group 推出的分布式计算环境(DCE)是第一个 RPC 标准, 但由于 ORB 技术的冲击, 近年来 RPC 标准进展不大。

2.2.2　面向消息的中间件(MOM)

面向消息的中间件提供了灵活的应用通讯机制, 包括消息传递、消息队列和发布/订阅三种方式; 支持多种通讯协议、语言、应用程序、硬件和软件平台。因此面向消息的中间件非常适合于开发事件驱动的应用, 也常用于企业应用的集成。

面向消息的中间件至今仍没有统一的规范和标准, 这意味着基于消息中间件的应用是不可移植的, 也不能跨不同消息中间件产品进行互操作。

2.2.3　事务处理监控器(TP Monitor)

TP Monitor 可以说是最早的中间件, 最初的作用主要是支持并发用户, 管理客户端到数据库的连接, 保证事务完整性。目前的 TP Monitor 大都遵循分布式事务处理(DTP)标准, 支持两阶段递交, 在异构数据库事务性应用系统中得到广泛应用。

2.2.4　分布式对象

面向对象思想和分布式计算技术的结合形成了分布式对象计算模型。目前主流的分布式对象计算模型有:

• CORBA

CORBA 是最早出现的分布式对象计算模型, 1991 年 OMG 就颁布了 CORBA 1.0 标准, 目前的版本是 3.0。CORBA 的主要目标是解决面向对象的异构应用之间的互操作问题, 并提供分布式计算所需的一些其它服务。ORB 是 CORBA 平台的核心, 它用于屏蔽与底层平台有关的细节, CORBA 对象通过 ORB 进行交互。不同供应商的 CORBA 平台之间通过 IIOP 实现互操作[2]。

• DCOM

微软推出的分布式对象模型 DCOM 是 COM 的扩展, 它也主要是为不同网络环境中的分布式对象提供交互的标准。几乎所有运行着 Windows 的个人电脑都或多或少地使用了内建的 COM 支持, 并且大多数 32 位版本的 Windows 还支持 DCOM, 这使得在 Windows 环境下 DCOM 成为一个分布式对象标准的强有力的竞争者。

• Java RMI

虽然 Sun 的 Java 对于业界来说是相对较新的事物, 但是它所支持的平台无关性、安全性和面向对象特性却迅速在业界获得了广泛的认同。Java RMI 中制定了一个基于 Java 语言的体系标准, 遵循这个标准, 人们可以很容易地创建"Java 对 Java"的分布式应用程序。

3　中间件与基于 Web 的分布式计算

电子商务作为 Internet 的强大驱动力, 迫使分布式计算模式从局域网向 Internet 扩展。Internet 技术使得客户端无处不在, 但要使 Web 成为人们进行事务处理的平台, 还是需要中间件的支持。具体做法是在 Web 服务器和企业信息系统之间增加应用服务器, 形成多层 Web 应用。应用服务器实际上就是一个涵盖了多种中间件产品特征的综合中间件产品[3]。

3.1　CORBA

通过 CORBA 与 Java 的结合, 使基于 CORBA 的应用系统能扩展到 Web 上。

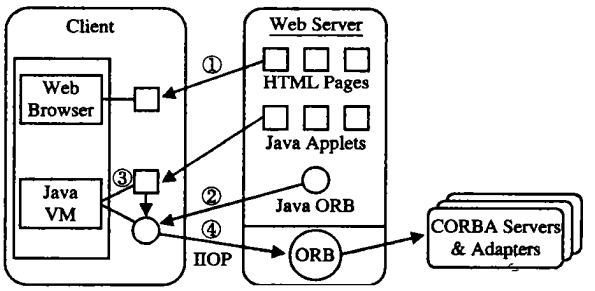

图 1　CORBA – Java 体系结构

如图 1 所示, Web 上的 CORBA 过程通常包括四步:

① Web 浏览器下载用 HTML 编写的网页文件, 这个文件中将同时包含与 Java Applet 的连接;

② Web 浏览器从 HTTP 服务器中下载 Java Applet;

③ Web 浏览器运行 Applet;

④ Applet 通过 ORB 及 IIOP 访问远程 CORBA 服务器中的对象。当然要实现这种访问, 必须使用具备 IIOP 功能的防火墙。

3.2　DNA

Microsoft 的 Web 应用解决方案称为 DNA(Distributed interNet Architecture), 其体系结构如图 2 所示。DNA 支持的客户端包括 CORBA 客户端、运行在浏览器中的 ActiveX Controls、独立应用、ISAPI 程序、ASP 和 HTML。客户端使用 Microsoft Active Directory 来定位中间层的组件, 并且使用 DCOM 来调用组件的业务方法。

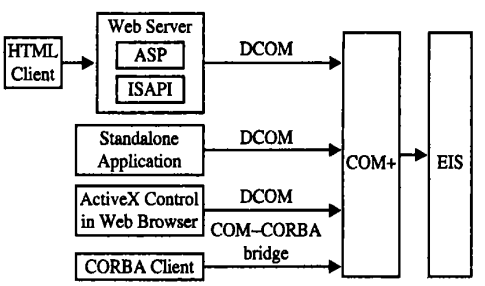

图 2　DNA 体系结构

DNA 的业务逻辑层由 COM+ 实现, COM+ 提供了一个比 MTS 更好的组件运行环境, 包括事务、安全、对象生命周期等等。COM+ 组件使用 ADO, OLE/DB 和 ODBC 来访问数据库。

3.3　J2EE

Sun 提出的 J2EE 体系结构是一个基于 Web 的分布式计算的完整解决方案, 如图 3 所示。J2EE 使用了 EJB Server 作为业务组件的运行环境, 在 EJB Server 中提供了分布式计算环境中组件需要的所有服务。J2EE 支持多种客户端的访问, HTTP 的客户端可以先向运行在 Web Server 上的 Java Servlet 或者 JSP 发出请求, JSP 中 Java 代码调用 EJB Server 中的 EJB, 以实现商业逻辑; 而其他的客户端通过 RMI/IIOP 直接访问 EJB Server 中的组件。

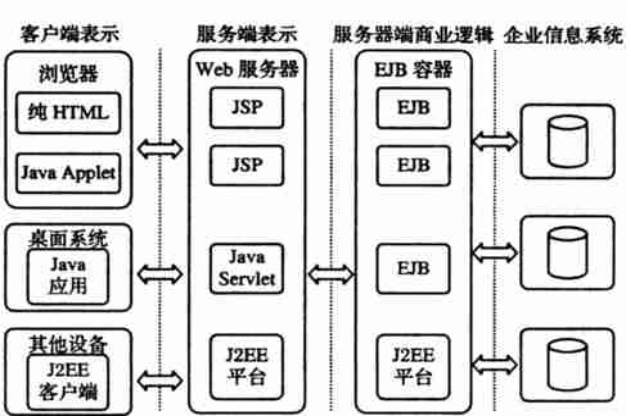

图 3 J2EE 体系结构

3.4 下一代分布式计算——Web Services

3.4.1 为什么需要 Web Services

上述三大中间件平台 CORBA、DNA 和 J2EE 都能很好地实现基于 Web 的分布式计算，但是这些系统有一个共同的缺陷：它们要求客户端必须使用特定的协议访问服务器端的对象。当各个公司需要相互合作或者扩展业务时，很难满足这样的要求，因为根本无法保证希望进行交互的双方采用的是相同的中间件平台。

提出 Web Services 的目标就是解决不同中间件平台上的服务之间的互操作性。

3.4.2 什么是 Web Services

Web Services 是一个自包含的、模块化的应用逻辑，可以用标准的 Internet 协议来访问。它不同于目前的组件技术，Web Services 不是通过特定的对象模型协议访问，例如 DCOM、RMI 或 IIOP，而是通过通用的网络协议和数据格式来访问，例如 HTTP 和 XML。用户可以使用任何语言在任何平台上调用 Web Services，只要他们能够创建并使用为 Web Services 接口定义的消息。

从一个 n 层应用体系结构的角度来看，Web Services 是一个对由其他中间件实现的服务的包装。Web Services 的组成如图 4 所示，包括一个处理请求的侦听器、一个用于公开业务逻辑所支持的方法的接口、相应的业务逻辑以及数据源。客户端只和侦听器进行交互，侦听器负责接收服务请求消息，然后解析消息，再把请求分派给接口中合适的方法。如果该服务返回一个响应，侦听器还负责把响应打包成一个消息发送给客户端。

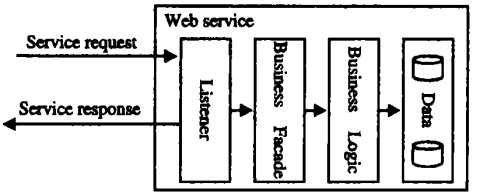

图 4 Web Services 的组成

3.4.3 Web Services 体系结构

Web Services 体系结构主要包括服务提供者、服务代理和服务请求者，如图 5 所示。服务提供者把服务“发布”给服务代理，服务请求者通过服务代理“发现”服务，并且“绑定”到该服务。

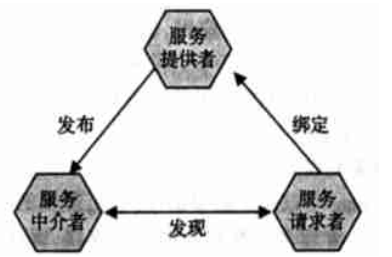

图 5 Web Services 体系结构

要支持这些操作，需要一个 Web Services 平台。基本的 Web Services 平台是 WSDL+ UDDI+ SOAP。WSDL 用于描述 Web Services，UDDI 用于发布和发现 Web Services，SOAP 用于访问 Web Services。目前 Web Services 还处于发展初期，许多标准和规范还在完善和制定中，要成为一门实用的技术还有很长一段路要走。

参考文献

[1] 王柏，王红熳，邹华. 分布计算环境[M]. 北京：北京邮电大学出版社，2000.

[2] Feiler J. Application Servers: Powering the Web- based Enterprise [M]. Acedemic Press, 2000.

[3] Slama D., Garbis J., Russell P. Enterprise CORBA[M]. Prentice by Hall PTR, 1999.

(上接第 5 页)

池中进行注册，服务中介者具有接受请求、发现服务、绑定服务的功能。

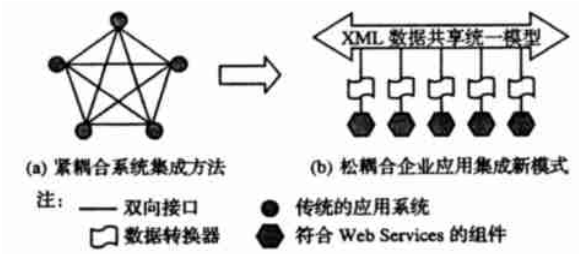

图 3

松耦合集成框架中的 Web 服务具有完好的封装性，是一种部署在 Web 环境下的对象，对于使用者而言，仅能看到该对象提供的功能列表，调用者不会感到 Web 服务发生的变迁，其它 Web 服务自然无须改变，这种松散的联合和动态的集成对建立无缝的跨平台互操作的信息共享与数据交换具有极大的优势。无疑，Web Services 将受到信息产业界各大系统集成及应用开发商的普遍支持。

基于 Web Services 的企业应用集成并不局限于企业内部，还可以面向敏捷供应链向企业上、下游扩展，服务提供者作为 Web 环境下的组件应用程序，物理位置可以在企业内部 Intranet，也可以在企业外部的 Internet，这样集成框架很容易与动态电子商务进行接轨。

3 结束语

本文深刻分析传统企业系统集成存在的问题，在研究 Web 环境下的组件技术的基础上，提出一种组件化松耦合企业应用集成框架，该技术易于面向敏捷供应链向企业上下游扩展，进而可与动态电子商务接轨。

参考文献

[1] 中国 XML 联盟. 查询和转换 XML[EB/OL]. http://www.xml.org.cn/, 2001- 08- 15.

[2] Kinsman C. Why Web Services? [EB/OL]. http://www.asp- zone.com/articles/ dk082100/ ck082100.asp, 2001- 08- 15.