中间件及其在三层客户机/服务器模型中的应用*

宋晓梁 刘东生 许满武
南京大学计算机科学与技术系,计算机软件新技术国家重点实验室 江苏·南京(210093)

摘 要 本文介绍了中间件的五种类型及其各自的特点,并讨论了中间件在三层的客户机/服务器模型中应用的一个实例。

关键词 中间件,客户机/服务器模型

MIDDLEWARE AND ITS APPLICATION IN A THREE TIER C/S MODEL

Song Xiaoliang Liu Dongsheng Xu Manwu
Computer Science and Technology Department,
National Key Laboratory for Computer Software, Nanjing University, Jiangsu·Nanjing 210093

Abstract In this paper, we discuss the five kinds of the middleware and give a sample of the middleware's application in a three tier C/S model.

Keywords Middleware, Client/ server model

随着以网络计算为中心的应用系统规模的扩大和软硬件结构的日趋复杂多样,客户端和服务器端的负担也日益繁重,并且传统软件的移植性、互操作性和重用性也都不能满足现在的性能需求,为此人们提出了一种介于客户端和服务器端的软件——中间件。

中间件(Middleware)作为前端客户机和后端服务器之间的一个中间层,为应用程序处理提供了如下功能,它一般包含应用逻辑,负责接收客户端的应用请求,对请求做出响应处理后将请求交给后端服务器,并负责将服务器的处理结果返回给客户端。

从概念上讲,很早就已经有了中间件的雏形,在主机环境下的 TP Monitor 就是一种中间件。但是,只有客户机/服务器以及 downsizing 的概念提出之后,中间件的概念才被人们广泛地关注。在客户机/服务器环境下,一般将中间件放在位于客户机和服务器之间的中间层,负责应用逻辑的处理,从而使客户端变得精干。与此同时,中间件还可以放在客户机和服务器之间的多层应用服务器中。中间件已经成为了联结分布式计算环境中各个相对独立的系统的胶合剂。

中间件具有如下特点:

易于集成 中间件能无缝地连入应用开发环境中,应用程序可以很容易地定位和共享中间件提供的应用逻辑和数据。

易于移植 中间件使与平台有关的细节对于应用程序来说是透明的,因此可以在不改变应用程序代码的情况下改换计算机底层硬件、操作系统或通信协议。

易于演进 中间件实现的功能对应用程序来说是透明的,所以可以对局部进行改进而不会影响到系统的其它部分。

高可靠性 中间件应该是可靠的,需要提供接管和恢复功能,保证事务及关键性业务不被丢失。

易于使用 中间件能和同构或异构环境下的多种数据源通信,同时它能管理数据间的公共逻辑约束。它将用户从复杂的平台、网络、数据库选择中解放出来。

1 中间件的分类

根据中间件所起的作用及采用的技术,我们大致可将其分为以下五种。

1.1 基于数据库的中间件

基于数据库的中间件是所有中间件中最普遍、最成熟的一种。基于数据库的中间件允许应用程序同本地或异地的数据库进行通信。它提供了一系列

收稿日期: 1999-04-23(修改稿) 宋晓梁 硕士研究生。主要研究方向:新型程序语言设计。
* 国家高新技术研究发展(863)计划及江苏省自然科学基金资助。

应用程序接口 API,通过中间层而不考虑操作系统及网络来访问数据库。并且在很多情况下,API 被隐藏在开发工具之中而不被开发者了解。

ODBC、JDBC 都是基于数据库的中间件标准。通过ODBC 访问数据库的方式是绝大多数应用程序使用数据库的方式,它通过使用驱动程序(driver)来提供数据库的独立性,驱动程序与具体的数据库有关,它是一个用以支持 ODBC 函数调用的模块(通常是一个 DLL),应用程序通过调用驱动程序所支持的函数来操作数据库,若想使应用程序操作不同类型的数据库,就要动态地链接到不同的驱动程序上。ODBC 具有良好的数据库独立性,它可以避免应用程序对不同类型数据库使用不同的 API,通过 ODBC 可以使得数据库的更改变得非常容易,因为对应用程序来说这只需改换一下驱动程序。JDBC 定义了一个 Java 开发的 CLI。JDBC 实际上就是一系列用于特定数据库的 Java 类库,它源于 ODBC 体系结构。

ODBC 的结构图如下:

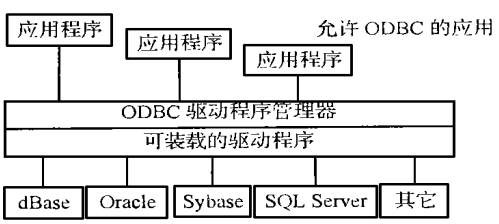

图 1 ODBC 结构图

现在,Microsoft 又提出了 OLE- DB。OLE- DB 提供了不同数据源的统一的访问点。OLE- DB 的目标是提供通过 OLE Automation 来访问多种数据库,或在应用程序和数据库之间提供一个 COM 层,通过 COM 层的对象访问数据库。

在基于数据库中间件领域中,目前还提出了应用分割技术,即将用户的一些应用逻辑放到中间层,为客户机"减肥",这也为 NC(Network Computer)等的引入打下了基础,并增强了应用程序的处理性能、安全性和并发性。目前,很多数据库前端开发工具都支持应用分割技术。

但是,在基于数据库的中间件模型中,数据库作为信息的中心存储单元,中间件负责数据间的同步及点到点通信。这种方式不适合于高性能应用处理,因为它需要大量的数据通信,同时,当网络发生故障时,系统将不能正常工作。

1.2 基于 RPC 的中间件

RPC 已经存在很长一段时间了,它沿用了用户熟悉的编程模式,从程序员的角度出发,RPC 十分容易理解——程序代码调用远端过程并将结果返回。当使用 RPC 时,只需要编写很少的网络程序代码,大部分代码由 IDL(Interface Define Language)生成。

RPC 应用不仅可以调用在远方节点上的子程序,甚至可以在不同操作系统环境下运行。它使程序员不必考虑网络的细节,仍可采用自己熟悉的 Call/ Return 语法。

RPC 一般采用 Call/Return 模式,多用于应用程序之间的通信,而且采用同步方式。RPC 程序之间的同步通信一般采用 Request- Wait- Reply 方式,因此,对小型简单的不需要采用异步通信方式的应用比较适合,但对大型复杂的应用不太适合,因为它需要程序员考虑网络或系统的故障、处理多个网络连接、可移植性、缓冲及流量控制和进程之间的同步等多种问题。

1.3 基于 TP Monitor 的中间件

TP Monitor 是一种复杂的中间件产品,它为应用处理提供了一种通信机制,它允许开发者在 TP Monitor 环境中定义事务服务。TP Monitor 位于客户机和数据库服务器之间,采用三层或多层模型。客户通过 Transaction RPC(TRPC)机制在 TP Monitor中调用事务,TP Monitor 运行事务来连接数据库,并将处理结果返回给客户端。TP Monitor 提供一系列服务,如应用管理、管理控制和应用之间消息传递等。常见的属性包括全局事务协调、分布式两阶段提交、资源管理器支持、协调故障恢复、高可用性、安全性、网络负载平衡等。

在 TP Monitor 中,事务有一个明确的起止点,如果事务失败,TP Monitor 可以回滚事务,不会使系统处于不完整、不一致状态。TP Monitor 同时可以复用数据库请求。因为每个客户调用事务,而不是直接和数据库进行连接,因此 TP Monitor 可以协调数据库请求,传统的 Connection- Per- Client 的限制(在客户机/ 服务器环境中)可以去掉,如 100 个客户可能只需要 10 个数据库连接。并且 TP Monitor 还可以在同一个事务中读写异构数据库中的信息,并保持异构数据库的完整性。

常见的 TP 产品有:BEA 的 Tuxedo、IBM 的 CICS、NCR 的 TopEnd、Microsoft 的 MTS 等。

1.4 基于 ORB(Object Request Broker)的中间件

基于 ORB 的中间件主要是采用面向对象的技术,ORB 可以看作是与编程语言无关的面向对象的 RPC 应用。它的成员函数可以采用类似 Object- function() 方式调用远端的对象。目前,ORB 存在两个彼此竞争的标准:CORBA ORB 和 DCOM ORB。

当使用 ORB 时,IDL(Interface Define Language)用于定义对象之间的接口,它类似于 RPC 中的 IDL 定义过程的接口。ORB 特别适用于对象接口变化不频繁,不会导致代码经常被重新编译及链接的情况。ORB 的总体框架图如图 2 所示。

理想的要求是,一个 ORB 应赋予每个对象(与其它对象特别是远程的对象进行通信时)以下的分布透明性:

位置: 无论其它对象是否处于同一个计算机中;

访问路径: 与其它对象交换消息经过的途径;

重定位: 其它对象从一台计算机移至另一台;

数据表示: 其它对象相应的数据格式;

通信机制: 使用何种进程间的通信机制和规程;

调用机制: 其它对象的方法是如何执行的。例如: 进程、线程和动态链接库等的细节;

存储机制: 其它对象任何可以或不可使用存储的情况;

机器类型: 任何机器型号上的不同;

程序语言: 其它对象是以何种语言实现的;

操作系统: 任何操作系统上的不同;

安全机制: 其它对象本身所具有的访问控制机制。

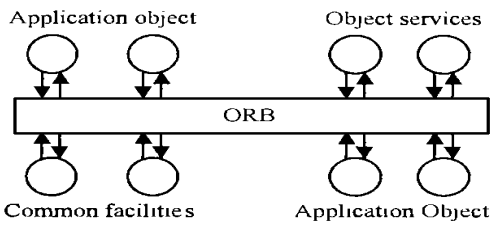

图 2　ORB 的总体框架图

对以上各方面的任何变更都无须将这个特定对象重新编译(或重新链接、重新加载等等), 其整体效果是, 对任何一个对象的实现动态地进行更改, 都将不会影响到其它的对象, 无论它们是服务者还是请求者。

1.5　基于消息的中间件 MOM

基于消息的中间件 MOM 提供了一个完整的处理环境, 允许开发者及用户连接不同系统之间的数据和代码, 或采用一致的界面进行应用处理的互连。MOM 提供了一个高层应用接口, 为不同系统提供操作核心。MOM 产品的工作主要是通过将信息以消息的方式在程序间传递来完成。

MOM 一般可以分为两种形式: 消息传递(Message Passing) 和消息队列(Message Queuing) 。

消息传递在建立大型的分布式应用中比较常见。其主要的模式是广播/订购(Publish- Subscribe) 方式。采用该方式, 应用程序既可以订购, 也可以广播。该通信模型提供了位置透明性。程序只需要简单地将消息以主题方式发送出去, 由中间件来负责将消息传递给所有订购该主题的程序。MOM 主要通过 agents 技术来实现 Publish- Subscribe 方式应用。当程序广播消息时, 首先与一个代理进行连接, 将消息传递给代理。代理负责路由消息给相应的程序。由于代理可以实现消息的动态路由功能, 因此, 该方式能够提供较好的容错性能, 但它缺乏 MOM 的异步特性, 不太适合长时间网络断开的情况。

消息队列方式允许程序无需直接建立起连接即可发送和接收消息。程序只须简单地将消息发送给消息队列, 由消息队列负责消息的传递, 对应用程序完全透明。消息队列采用异步方式, 为信息提供了一个安全的存储方式, 特别适用于不是直接连接的应用, 如移动用户、发送方或接收方进程可能处于不活动状态的应用。它的缺点是需要一些配置工作, 性能不是很高, 而且如果队列丢失, 整个系统将受到影响。

MOM 可以克服基于 RPC 的中间件的限制, 提供基于消息的异步通信机制, 因此 MOM API 调用不会阻塞应用程序, 同时 MOM 不会占用大量的网络带宽, 可以跟踪事务, 通过将事务存储在磁盘上, 可以恢复系统及网络故障。

常见的 MOM 产品有: DEC 的 MessageQ、IBM 的 MQSeries、Microsoft 的 MSMQ。

2　中间件优点及具体应用

传统的客户机/服务器模式是一种双层模型。双层模型的物理实现方式为: 一台桌面电脑当做客户机使用, 而一台网络服务器则用于容纳后端数据库引擎。在双层模型里, 程序逻辑在客户机与服务器这两种物理位置之间分担, 应用程序的商业逻辑必须物理性地驻留于客户机端, 或在后端 DBMS 里以触发器或存储过程的形式实现。这种双层的客户机/服务器方案的优点在于实现数据访问相当简单, GUI 可与数据源直接约束在一起, 数据维护的所有细节都可以得到自动控制。但是它也存在着很多的不利因素。尽管数据访问得到了简化, 但却缺乏灵活性, 我们通常无法对自己与数据源的交互作用进行完全的控制, 因为这种控制是自动的。很明显, 额外的管理会消耗客户机资源, 并可能导致应用程序的性能下降。

双层客户机/服务器模型存在以下三个方面的限制因素:

不可伸缩　双层模型无法超越客户机的物理界限, 而且服务器也会禁止这种模型的伸缩。

不能管理　由于不能封装商业规则, 也不能对这些规则进行集中配置, 所以常用程序的重用也不方便。

性能较差　因为将图形界面与数据源绑定在一起, 所以会消耗客户机的主要系统资源。这对客户机来说是一个沉重的负担, 会导致系统性能的下降。

正是由于双层客户机/服务器方案存在诸多限制, 所以又提出了一种新的三层服务器模型。三层客户机/服务器模型以构建分割式应用程序为基础。对一个应用程序进行分割以后, 可将代码划分为不同的逻辑组件, 在三层的客户机服务器模型中, 这些逻辑组件分为三个逻辑层: 用户服务、业务服务和数据服务, 它们共同组成一个应用程序。我们把这种

三层的设计模型称为“服务模型”。

三种服务的属性如下:

用户服务 提供信息和功能、浏览定位, 保证用户界面的一致性和完整性;

业务服务 共享的业务政策, 从数据中生成业务信息, 保证业务的一致性;

数据服务 数据的定义、永久数据的存储和检索, 保证数据的一致性。

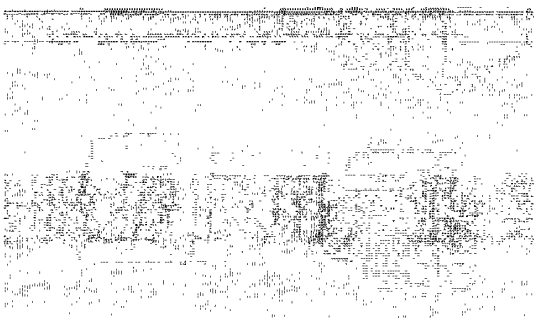

图 3 服务模型结构图

使用服务模型, 可以把应用程序的需求分解成明确定义的服务。在定义了服务之后, 需要进一步创建具体的物理构件来实现它们。构件是一个或几个服务在物理上的封装, 可以通过构件的接口获得这些服务。构件可以是. exe 或. dll 文件、数据库触发器和存储过程的集合, 或者任何几个其它物理软件实体。构件是由它所提供的服务以及它和其它构件的相互作用来定义的。根据性能和维护的需求、工作量、网络带宽以及其它的因素, 可以在网络上灵活地部署这些构件。这些构件总是通用的, 并且遵守公开的接口标准, 所以它们可以被重用, 并能被多个应用程序所共享, 外界所能知道的就是它们的接口。实际上这些物理构件就是基于 ORB 的中间件。

这种基于构件的三层客户机/ 服务器的优点可概括为以下四个方面:

可重用性 许多应用程序可共享和重用封装在构件中的功能。

灵活性 从桌面计算环境到功能更强的网络服务器, 随处都可分配工作, 这有利于协调性能和网络带宽。

可管理性 将大型复杂的工程细分为简单、安全的构件工程。

易维护性 将业务逻辑部署在中央服务器上, 而不是分散在用户桌面上, 这有助于处理各种变化, 并缩短解决方案的往返时间。

在某电视台资料管理及业务处理信息系统中, 我们采用了上述三层的服务模型: 在客户机上的客户服务程序, 在构件服务器上的业务服务程序和在数据服务器上的数据服务程序。客户服务程序主要处理表示逻辑, 也就是以某种方式同用户进行交互, 它基本上是一种图形用户界面。业务服务程序主要完成业务规则的控制和对数据库的访问, 并且业务服务程序是以构件的形式提供给客户服务程序调用的。由于业务服务程序是以构件的形式给出的, 所以一个构件能被多个客户服务程序调用, 例如节目资料检索构件就能让其它各个子系统的客户服务程序调用, 以便客户服务程序在检索结果集基础上进行其它处理。数据服务程序主要就是各种智能数据库所能提供的功能, 例如存储过程、存储查询等等。

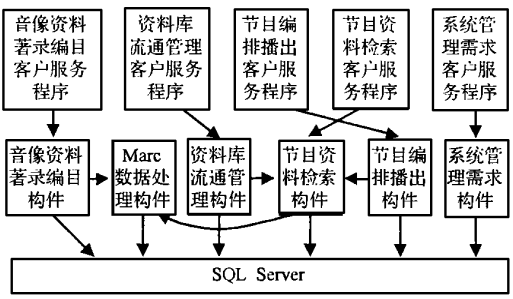

图 4

该系统的结构如图 4 所示。在这个三层客户机/ 服务器模型中, 我们把各种业务规则都封装在各自相应的构件中, 这样业务规则的实现和客户使用的图形用户界面的实现就相分离了(在传统的两层客户机/ 服务器模型中, 两者是同时在客户机应用程序中实现的), 当一方要进行改变时, 只对需改变的一方进行改变。与此同时, 在同一层上的构件之间也可以相互调用, 这样也增加了代码的可重用性, 减少了编程量, 而且因为构件也是相互独立的, 所以当需求改变时也只需对相应的构件进行改变。如此一来就大大减少了系统维护的负担, 并提高了系统的适用性。

参考文献

[1] Ron Ben- Natan. CORBA- A guide to common object request broker architecture. McGraw- Hill, 1995

[2] D. Box. Q&A ActiveX / COM. Microsoft Systems Journal, 1997; (3): 93- 105

[3] A. Birrel, B. J. Nelson. Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, 1984; 2 (1): 39- 59

[4] 黄 俊, 许满武, 陆剑锋. 中间件 LSM 及其在物资运输调配系统中的应用. 微型计算机, 1997; 17(增刊 2): 78- 80

[5] Cynthia McFall, IBM Corporation. An Object Infrastructure for Internet Middleware IBM on Component Broker. IEEE Internet Computing, 1998; 2(2)

[6] Object Management Group. CORBA 2. 1 Update Sheet, 1997. 8