Client/Server 体系结构的发展

欧阳明 (清华大学计算机系 100084)

随着 PC LAN 发展到 90 年代,有两个趋势使得文件共享模式最终发展到了 Client/Server 模式。首先是 PC LAN 应用的发展及其用户数目的增加,因 LAN 上传输的是文件而受到了带宽的限制。多用户的 Xbase 应用只能在用户较少的情况下才能提供较好的性能。其次是 GUI 界面的出现并在 PC 机界占据了统治地位,促使原有的字符用户界面向 GUI 转变。

一、两层 Client/Server 结构

与这种要求相适应,应用软件体系结构和技术发展到了 Client/Server 模式,也就是两层的 Client/Server 结构,它用一个真正的数据库服务器取代了文件服务器,这样网络对于客户请求的响应就不再是一个文件,而是一个查询的结果集合了,这无疑能显著地减少网络流量。其次,数据库服务器大都是支持存储过程和触发器的,这样,服务器就可编程实现业务规则,从而带来了系统整体性能的提高。

两层的 Client/Server 结构有一些很好的特性。首先它可以在 RAD(Rapid Application Development)方法下进行开发,能高效地开发出小规模应用;其次是它有大量的理解这一结构的编程开发人员,如使用 Visual Basic、PowerBuilder 和 Delphi 的开发者。总之,两层 Client/Server 体系在解决工作组范围的应用上已被证明是非常有效的,但对于企业级的大规模应用或分布于 WAN 上的应用,两层结构就遇到了诸多问题。

在大的企业环境下,首先遇到的就是两层 Client/Server 中数据库服务器中的性能问题,DBMS 需要为连接到服务器的每个客户维持一个线程,即使客户和服务器间没有工作,它们之间仍然要不断交换数据包以保持连接。如果网络连接出了问题,客户端必须重新发起一个会话,重建连接。当用户数目上升时,DBMS 的性能就不可避免地会受到影响。

应用软件的复杂性是另一个问题,所有的代码、用户接口逻辑、应用程序逻辑和数据处理逻辑全都混合在一起,这样,代码本身所具有的面向对象的可重用性就大大降低了。随着应用规模的增长,复杂度随之上升,要编出无错的程序就难上加难。代码的混合,对于升级和维护都是一个难题。同时,风格不一的用户界面也为培训和升级带来了很多问题。

解决这些问题的办法就是在客户端和服务器之间增加一个中间层,把主要的业务逻辑转移到中间层上,这样,两层的 Client/Server 结构就逐渐发展到了三层的 Client/Server 结构。

二、三层 Client/Server 结构

随着中间层的加入,以服务器为中心的 Client/Server 服务模式就变成了客户/网络(Client/Network)服务模式。三层的 Client/Server 体系结构可以表示如下:

* 第一层:表示层,前端用户接口,提供可移植的表示逻辑。
* 第二层:功能层,功能服务器,执行业务逻辑(Business Logic),包括应用程序和计算逻辑及管理数据库。
* 第三层:数据层,数据库服务器,允许用户访问数据服务,负责数据处理逻辑。

在两层的 Client/Server 结构中,用户把实际的业务逻辑或置于客户端(在表示逻辑之上),或置于后端数据库(作为包含在存储过程中的数据逻辑的一部分),而在多层体系结构中,用户把业务逻辑放在中间层部件上,中间层部件把业务逻辑从实际应用中隔离出来进行共享和控制,这样就为表示逻辑、业务逻辑和数据处理逻辑划分出了明晰的界限。

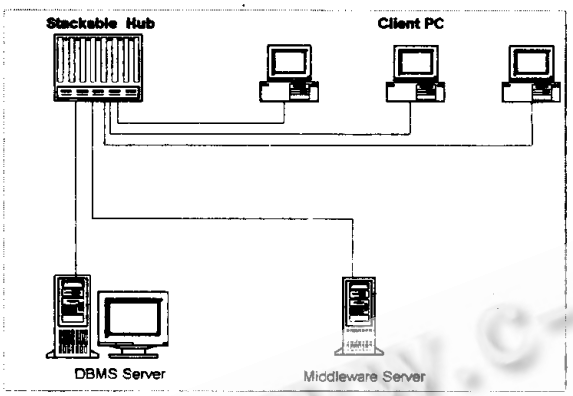
为了把客户端用户界面代码发起的请求发送到服务器或从服务器端获取数据,必须采用应用程序间(Application - to - Application)的通信机制,某一特定的通信机制可以是以下之一:

* 远地过程调用
* 对象请求转发,常用于分布式对象环境

* 面向消息的中间件,这是一种易于使用的应用程序通信技术,但没有标准化

* 直接网络编程

三层结构的一般形式可以用下图表示:



作为三层结构中的关键部件,中间层服务器可以采用不同的类型。

1.采用 TP Monitor 的三层结构

最基本的也是最老的中间层部件就是事务处理监视器即 TP Monitor,其概念可追溯到 70 年代早期的主机时代,TP Monitor 可看作是一种消息排队服务,客户机连接到 TP Monitor 而不是数据库服务器,TP Monitor 接收事务请求,进行排队并负责管理和完成事务。

TP Monitor 的出现在于它提供了加速效果,减少了 DBMS 服务器需要维持的线程数目。客户与 TP Monitor 直接连接,当 TP Monitor 接受了客户端的消息之后,客户可以释放连接并转入后面的处理。这样,两层结构中的同步会话就变成了异步会话,TP Monitor 的加入能平滑并降低数据库服务器的负荷。

TP Monitor 所提供的特性包括:在一个客户机上更新不同 DBMS 的能力;有优先级地连接数据库事务;良好的安全性等。总之与两层结构相比,带 TP Monitor 的三层模式有更好的可扩展性,效率也更高。但基于网络的 TP Monitor 也有其弱点,主要是实现 TP Monitor 的语言多是低级语言如 COBOL,其次是大多数的可视化工具不提供对 TP Monitor 的支持。

2.采用消息传递服务器(Messaging Server)的三层结构

消息传递服务器可以看作是第二代的 TP Monitor,它也有加速处理的能力。所谓消息就是一个载有关于目的地址、内容、待执行动作的自含对象,每条消息至少包含两个部分:消息头中包含优先级、地址和 ID 号;消息体中包含发送的消息,它可以是文本、图象、事务等。

与 TP Monitor 类似,消息传递服务器提供了到数据源的连接而不是到 RDBMS 的连接。客户发出的消息由它异步地按一定优先级处理,但不同于 TP Monitor 的是:消息传递服务器结构是围绕消息本身所具备的"信息"而设计的,而 TP Monitor 环境中把系统信息放在 Monitor 或应用程序服务器的处理逻辑中。在 TP Monitor 环境中,"事务"仅仅是一些包含数据的哑数据包,它们按预先定义好的现存路径传输到 TP Monitor,TP Monitor 解析并处理该事务,通常是把它的请求提交给一个服务器的应用程序,如果 TP Monitor 无法理解事务数据,那么该事务将不会提交进行处理,也就是说,TP Monitor 端要象服务器端一样对事务有足够的了解。

与此不同的是:在基于消息的结构中,消息本身包含有所需的信息,此时,消息服务器实际上是一个消息及其存储过程的容器。消息服务器对消息所作的操作都是与通信相关的,如加密进行传输等。大多数的消息被当作离散的对象进行处理,消息含有了网络服务中所需的全部信息,如网络地址,包括 IP 地址和 MAC 地址。由于消息本身包含了这些信息,基于消息系统的中间层较之 TP Monitor 就灵活得多。对于一种消息,中间层可能仅是作为两种网络服务间的一个路由节点;而对于另一种消息,中间层又可以在消息的指示下执行存储过程或业务规则。把中间层从经过它的消息的内容和行为中抽象出来,可使系统对不同的环境和网络更具可移植性,而信息的通信类型就被掩盖在消息传递服务之下。

消息传递系统具有较好的健壮性,它采用的是存储转发机制,这样消息可以在故障发生后重发。此外,消息传递系统独立于现有的有线、无线网络技术和协议,它不要求客户和服务器的固定连接。对于无线网络技术的支持,使它也适合于支持流动的或偶尔才进行连接的情况。

使用消息传递服务的应用程序体系结构的实现方式与依赖于分布式对象和对象请求转发通信相类似。

3.采用应用程序服务器(Application Server)的三层结构

一般所指的三层结构,都是指采用应用程序服务器的实现方式。在这种方式下,应用程序的业务逻辑从 PC 端转移到了一个通用、共享的主机服务器上。与前

两者客户端作为程序执行者不同的是：此时的客户端 PC 主要是用于表示服务，这与主机架构下的终端的角色是类似的。

由于应用服务器无须驱动用户的 GUI 而仅仅是一个共享的业务逻辑计算和数据搜寻引擎，因此它与主机的作用是类似的。通常服务器运行在 32 位的多任务操作系统如 Windows NT、UNIX、OS/2 之上，操作系统的硬件平台可以选择对称多处理器配置。另外还可以配置其他的并行硬件，这就使服务器在性能方面具有极好的可扩展性。

应用程序服务器的引入，带来的好处包括：

* 把重要的应用集中于服务器端，有利于加强安全控制；

* 应用程序服务器的可扩展性使整个系统具有良好的扩展性；

* 支持和安装都集中于服务器端，升级费用降低；

* 更易于实现不针对具体 DBMS 的应用程序，当转向一个具体的 DBMS 时，只需以很少的工作量改动服务器端程序而不涉及用户端编程；

* 新的支持三层应用程序服务器方式的工具，都提供了对应用程序的事后分解，这意味着代码和功能模块在建立好之后可以重新分发到新的服务器上，就带来了更大的灵活性和性能的提高；

这种结构的主要弱点在于它在技术上比两层 Client/Server 实现更为复杂。

4.采用目标 DBMS 的三层结构

这是应用程序服务器的一个变种，它以目标 DBMS 取代应用服务器作为中间层。在这种情况下，目标 DBMS 是作为加速器或"hot cache"使用的。在关系数据库管理系统中，数据通常跨越多表以正规化的形式保存并为不同的应用程序用户所访问。这种一般化的存储形式，对某一特定的应用而言在性能方面可能是不够的。目标 DBMS 就可以用于从公用存储中搜索数据，并把它组装起来，供用户有效地使用，而且只要应用程序需要，这些数据就可以永久保留。目前的 RDBMS 一般都不支持扩展的数据类型如音频、视频，这些数据类型也可保存在目标 DBMS，它能把 RDBMS 中搜索出来的数据与相应的多媒体数据关联起来。

以上所提到的均是常见的三层结构模型，TP Monitor 的提供者有 Sybase、Oracle 和 Gupta 等公司，消息传递服务器的提供者有 IBM、DEC、Sybase、Oracle 等，而提供应用程序服务器的产商则较多，如各厂家的 Web Server 实际上都属于应用程序服务器。

三、Internet/Intranet 与三层 C/S 结构

Internet/Intranet 是典型的三层 Client/Server 结构，表示层是 WWW 浏览器，客户向 URL 指定的 Web 服务器提出服务申请，Web 服务器按 HTTP 协议把文件传送给客户，而客户端由 HTML 协议负责表示逻辑。功能层是支持 CGI 接口的 Web 服务器，通常 Web 服务器由 CGI 程序连接数据库进行数据查询或处理。数据层是各种负责数据处理的 RDBMS。在 Internet/Intranet 环境下，由于客户端是通用的浏览器，也就构成了 Browser/Server 这样一种特殊的三层结构，它是一种基于 HTML 的 Web 应用体系。

基于 HTML 的 Browser/Server 结构是特殊的，三层应用程序结构中客户端被描述为"瘦客户"，但 HTML 则为提供了最"瘦"的形式：根本没有客户代码，只有表示逻辑(仅就 HTML 而言)，这样它就可以用于构造通用的客户端。每个 Web 页面所下载的 HTML 代码都是描述性的，不含自己的处理逻辑，它只含有 Web 浏览器显示所需要的信息。只有在包含表格(Form)时，才允许输入数据，并提交给基于服务器的应用程序，换言之，它构造出的 Client/Server 应用程序用户接口非常基本，是根据用户的请求动态下载的。

由于所有的应用处理和数据访问代码都驻留在服务器端，这种基于 HTML 的 Web 应用可以归为三层或后面将提到的四层、分布式对象结构中的特殊一类。早期的 Web 应用可以说是纯出版发布性质的，但技术的发展迅速使 Web 应用发展到一个新阶段，一是对可下载的 applet 动态客户端的支持，二是动态数据库连接的支持，代表了新一代 Web 应用的特征。目前在很多的 Web 应用中，其 HTML 页面都是由运行在服务器端的代码动态生成的。

基于 HTML 的 Web 应用所带来的最大的好处莫过于困扰 Client/Server 体系的软件分发问题在 Browser/Server 结构下不复存在了，因为所有处理都由服务器完成，用户界面由动态下载的 Web 页面引导，用户界面代码、应用和数据代码就没有必要安装在客户端，只要有浏览器，而没有客户端特定的应用代码也是 HTML 的主要弱点，这样客户机就象主机环境中的哑终端一样，输入而后返回输出，不能利用客户机本身所具有的计算

能力。在 Client/Server 结构中，客户端的处理至少可以提高性能，改善用户界面，HTML 的缺陷随之就有引入了 Java、JavaScript 的进行弥补。

Java applet 以及 JavaScript 的引入可以补偿 HTML 的弱点，但这两者是很不一样的。Java applet 是一段应用程序代码，它可能是经过编译的字节码，它可以动态下载到客户端。而 JavaScript 则是嵌在 HTML 代码中随之一起下载的，它是源代码。二者都可由浏览器运行，完成一定的客户端处理，如数据检查等。Java 和 JavaScript 也不存在分发和版本管理的问题，因为每次下载的都是最新的代码，也不存在安装的问题，因为代码只是在需要时才下载。

HTML、JavaScript 和 Java Applet 可以看作是处于客户端的不同层次上，JavaScript 内嵌于 HTML 代码中，对浏览器特性有较强的控制能力，但计算能力很弱，而 Java 则有较强的计算能力，同时，在 Netscape Navigator 3.0 之后就实现了 Java Applet 和 JavaScript 之间的相互通信，JavaScript 可以直接访问 Applet 中的变量、方法和函数，applet 也可以访问 JavaScript 中的对象，这两者的结合就大大地提高了客户端的处理能力。另外像 Plug－in 等技术也为客户端处理的发展注入了新的活力。

Java 还提供了其他一些重要特性，如安全性和跨平台能力。就应用程序体系结构而言，Java 使任何开发者都可以构造所需的任意层数的应用程序，具有全面的灵活性。

四、Client/Server 结构的未来

1. 多层 Client/Server 结构

象三层结构一样，多层结构也将是高度灵活的，它把用户界面代码和应用程序逻辑分离开来，对用户界面的修改或对应用程序逻辑的改变都是互相独立的，这使应用程序可以不断发展以满足新的要求。

正如其名称表示的那样，多层结构可以有三层以上。应用程序体系结构发展的下一阶段将是四层模式，它将增加新的一层模块。在三层结构中，应用程序逻辑和数据访问逻辑混合在一起，这就很难在不改变应用程序逻辑的情况下改变后端数据存储访问机制。四层结构清晰地把应用程序逻辑从数据访问逻辑中分离出来。这样当后端的数据存储机制需要改变时，不会影响到应用程序的代码。这在很多情况下是非常有用的，例如在升级一个 DBMS 时，三层结构必须要改变业务逻辑，而四层的结构则不用改变应用程序逻辑。

应用程序也可以有四层以上，许多应用就要求划分应用程序代码，把一部分放在客户端，而大部分驻留在服务器端，这种情况就是五层的结构。应用程序需要有多少层，只能取决于实际的需要。

2. 分布式对象结构

分布式对象体系是在对象间使用统一的通信机制的面向对象的实现方式，能够获得最大程度的灵活性和可扩展性。以这种方式构造的应用程序，在决定代码驻留哪端之前就可以完全实现，并且可以在应用程序安装时按需要分发代码，也可程序完成后重新分发。这种类型的结构可以带来长远的利益，它还能迅速增加对新的 DBMS 的应用，增加新的用户界面，并能升级支持各种新功能。但这种结构应用的构造需要有相当的专业知识和技巧。

下面以一个例子来说明分布式对象系统及其灵活性。这是一个管理复杂的客户帐户的应用程序，销售人员、工程人员、支持人员及其经理共享相同的客户信息，且都可以读和更新客户信息。系统可以安装在服务器端，由“瘦”客户提供用户界面。应用程序逻辑和数据访问逻辑分离，但都放在服务器上，在这样的结构下，用户可以获得授权获取和更新客户信息。

但如果销售人员外出与网络无法连接时，又该如何处理呢？“瘦”客户在没有连接服务器时是什么也干不了的。但注意到分布式对象所具有的实现任何结构的灵活性，就可以把整个应用程序安装到销售人员的 PC 机上，并使用基于 PC 的数据库保存一个服务器端的数据库的副本。当他连接到网络上时，应用程序把更新从 PC 复制到服务器上的数据库，并把服务器数据库的更新复制回来，使二者保持同步。在连接期间，应用程序自动使用服务器数据库，而在断开连接时，应用程序负责重新同步 PC 上的数据库，使之能保存最新的备份。

采用分布式对象体系结构构造的应用程序，可以很方便地适应于在 Client/Server 分布环境或在单独一台 PC 机上运行。用两层、三层或四层模式构造的应用程序，PC 端驻留的代码和服务器端的代码是事先定义好的，很难改变，而在分布式对象结构中，分界线是灵活可变的，它也是应用软件开发的理想结构。

（来稿时间：1997 年 4 月）