计算机科学1995 Vol.22№.5

Client/Server 计算与现代 DBMS 的体系结构*)

车敦仁　周立柱　　杨亚奇
(清华大学计算机系　北京100084)　(中国农业银行吉林省分行信息处)

摘　要　Presently, both the Client/Server computing and Multimedia are in the fashion, just like Object-Orientation is a few years ago. Client/Server has implications in computer network, hardware, software and mordern DBMS'(especially OODBMS') architectures, so makes some confusion. This paper first proceeds from the coarse implications of the Client/Server computing, to a comparatively deep discussion of the Client/Server architecture, and then to a more systematic survey of the mordern architectures of OODBMSs, which have potential benefits to supporting multimedia and engineering applications.

关键词　Client/Server, Software architecture, DBMS' architecture, OODBMS, Page-Server.

1. 引　言

人们过去更多强调的是计算机系统和计算机硬件系统的体系结构，随着软件系统规模和复杂度的日渐升级，人们越来越认识到好的软件体系结构对于好的软件系统的重要性。就体系结构的本意而言，它是指建立系统时的构造范型、构造风格和构造模式[3]。软件系统的体系结构亦如此，它对于软件系统的构造过程具有重要的指导作用，同时可以使软件设计师暂时抛开软件系统的功能细节来谈论或探讨软件系统的总体构架。

广义而言，软件体系结构涉及很多方面的内容[3]：(1)软件的成份及系统框架；(2)软件成份的选择、各成分之间的相互作用、软件成份的进一步复合以及指导软件复合过程的总体模式；(3)系统的功能、性能、设计以及从多种方案及选项中进行选择的决策；(4)软件体系结构更为关注的是产品及其成份，而方法论更关心建立产品的过程等等。本文不拟深入讨论软件体系结构所涉及的这诸多方面，重点就其核心内容，即系统结构及其特点进行较系统的论述。

下面先从 Client/Server 计算入手，对 Client/Server(软件/硬件)体系结构进行初步探讨，进而详细论述现代 DBMS 的 Client/Server 体系结构，尤其是对能较好地支持多媒体和工程应用的 OODBMS 的体系结构进行较深入的讨论。

2. Client/Server(客户/服务器)计算

对于今天的企业或机构来说，使用网络把各种不同的计算机连成一个整体计算环境已是很平常的事。利用网络把多台规模较小的计算机联结起来替代造价较高的大型机或巨型机已成为一种发展潮流。

Client/Server 计算技术，简称 Client/Server 计算，正是在局域网络 LAN 技术和价格低廉的工作站及 PC 机高度发展的背景下崛起的一门新技术。

Client/Server 技术具有硬件和软件两方面的含意。硬件方面的含义是指由桌面计算机、网络和服务器构成的一种网络计算环境。桌面计算机可以是由廉价的 PC 到功能强大的工作站之间的各档机器，网络主要是局域网 LAN，服务器既可是通用的计算机也可是专用服务器，如专门用作数据库服务器的机器(或数据库机)等。

在网络环境下存在两种基本的计算模型：Client/Server 和 peer-to-peer(对等)模型。有人把 Client/Server 比喻为网络计算的"独裁主义"(authoritarian)模型[1]，共享资源均在一个或多个服务器的集中控制之下，其优点是安全性好，并且网络资源可被集中和严密地控制；缺点是，凡共享资源都要通过服务器连接到网络上，而连接到各工作站的资

*)国家863计划 CIMS 主题资助。车敦仁　博士后，主要从事新一代数据库系统的开发与研究工作。

源都属本地的专用资源，不能被全网络所访问和利用。

peer-to-peer 模型与 Client/Server 模型恰恰相反，没有集中服务器的概念。peer-to-peer 可以被比喻着网络计算的"无政府主义"(anarchistic)模型[1]。其优点是系统灵活性大，网络资源利用率高；缺点是管理困难，安全性和性能较之 Client/Server 模型均有所不及。

Client/Server 计算在软件方面的主要含意是指，一个软件或应用系统的实现不是作为犹如磐石般的一个统一整块，而是被设计成包含很多成份的复合系统，这些软件成份甚至可被分布于网络中不同的机器节点上，并且依据软件成份的相对角色之不同区分为"客户"(Client)和"服务器"(Server)，客户软件能够请求服务器软件的服务[2]。

所以说在 Client/Server 体系结构下，客户和服务器均有两种含义，"客户机"和"服务器机"，或"客户软件"和"服务器软件"。若是动态地看还可区分"客户进程"与"服务器进程"。本文在下面的讨论中，将把"工作站"和"客户机"视为同一个概念。

到底什么是、什么不是软件的 Client/Server 体系结构，这曾在工业界引起过激烈的争论，这里引用文[2]中给出的几条准则：

(1)在客户与 Server 之间应有清晰的功能划分，各自均应有具体的功能角色，但从应用的观点看它们之间的交互应是无缝的。

(2)客户与服务器可以运行在不同的机器上，但这不是必须的，这样有利于应用的可伸缩性(Scalability)。

(3)一个服务器应能并发地支持多个客户，这意味着服务器对客户来说是一个共享的"资源"。

(4)对服务器的改变不应对客户产生影响。这对于系统的可伸缩性(可改换不同规模的服务器)和可用性(坏的服务器可换以好的服务器)都非常重要。

(5)对一个客户的修改不应影响到其它的客户。这意味着客户之间的独立性。

使用 Client/Server 体系结构的一个主要动因是希望利用当前工作站技术所能够提供的丰富而廉价的资源。把系统功能从服务器转移到客户端能够给用户带来下列好处[1,2]：

①允许用户按自己的爱好选择其最乐意使用的图形、界面和工具支持，从而有利于提高生产率；

②把更多的事务运行在成本较低的平台上，则可大幅度降低计算成本；

③可通过减少客户与服务器的交互和尽量避免处处都要从远程节点上获取数据的开销来改进系统的性能；

④一般可通过对系统的共享资源，如服务器机和网络等(它们是潜在的系统瓶颈)，卸载(offloading)的办法来提高系统的可伸缩性。

3. 现代 DBMS 的体系结构

无论是集中式 DBMS 还是分布式 DBMS 作为一种软件，其体系结构可以划分为两种基本类型，即 Host/Slave 结构和 Client/Server 结构。在只有字符终端和在主机上运行批程序的年代，早期的多数关系数据库都是围绕 Host/Slave(主/从)型计算模型而设计和建造的。为方便起见，也把早期 DBMS 的体系结构称为 Host/Slave 模型。从本质上说，这种体系结构让用户看到的是单一的进程，应用和数据库服务器被集为一体[10]。

Client/Server 是近年来发展起来且在数据库界很流行的一种新型的 DBMS 体系结构，其真正要点是把数据应用视作是与数据库分离且不同的进程，它们可以在地理上分布，也可以同驻一台机器上。由此看来 DBMS 的 Client/Server 体系结构恰好意味着 Host/Slave 体系结构的对应物[10]。

在具有 Client/Server 结构的 DBMS 中，数据的表现(data presentation)和数据的存取(data access)被分别作为客户与服务器的职责。客户工作站主要是把数据表现给用户，而服务器(通常是一个存储容量很大，处理能力很强的机器)则集中于数据的存储与检索。

当用户查询数据库时，客户机即把一个查询提交给服务器机。驻留在服务器机上的数据库管理系统接收查询后，将之编译并选择一种最佳的优化策略，然后再把查询结果汇集起来发送给客户机(这种体系结构也被称之为 data-shipping(数据装运)体系结构[1])。

大多数 Client/Server 结构的数据库系统通常还要提供一个应用程序接口 API，客户应用可以很方便地通过 API 完成对远程(或本地)服务器数据库的访问。客户机与提供 API 的数据库服务器之间的典型协议[11]概括为图1。

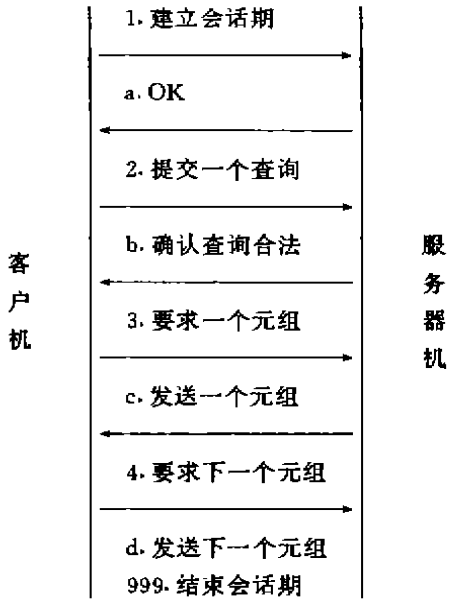

图1

(1)建立会话期：建立与服务器的会话连接，连接完成后服务器应做相应的回答；

(2)准备一个查询，将准备好的查询通过 API 提交服务器后，服务器通过编译完成查询的合法性检查，并进行优化，然后向客户确认查询有效；

(3)客户机向服务器逐一索取查询结果集合中的每个元组，有时为了提高网络传输效率，服务器实际上是成批地将查询结果传送到客户机的通讯缓冲区中。

(4)结束会话期：切断客户机与服务器之间的会话连接。

支持 Client/Server 体系结构的 RDBMS 有：Sybase 的 SQL Server、IBM 的 EE、Oracle、SQL Gupta 和 VAX Rdb，其中数 SQL Server 的 Client/Server 体系结构特点最为典型。

Sybase 的 SQL Server 包括下列三个基本成分[11,12]：(1)TransAct SQL：这是 SQL Server 所支持的 SQL 方言；(2)DB-Library：即 SQL Server 提供的 API；(3)SAF(System Administrator Facility)：供 DBA 定义数据库、用户、口令，以及备份与恢复数据库等。

SQL Server 不仅是典型的 Client/Server 体系结构，而且对 Client/Server 的基本思想进行了深化和发展。在 SQL Server 环境下，一台机器可以是另一台机器的客户，或是另一些机器的服务器，或者同时担当客户和服务器二者的角色[10]。

4. OODBS 的 Client/Server 体系结构

OODBS 把 Client/Server 体系结构又向前发展了一大步，这与其数据模型中的复杂对象处理的要求是一致的。

OODBS 的 Client/Server 体系结构可以划分为下列四种基本类型[2,4,5,7]：数据库服务器、对象服务器、页服务器和文件服务器，见图2。它们的主要区别是对系统的客户成分和服务器成分所赋予的职责的多少不同，以及客户与服务器在交互时数据传输的粒度不同。

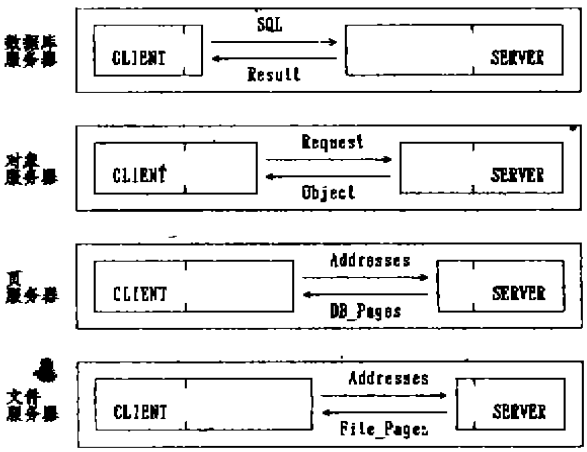

图2 OODB 的四种 Client/Server 体系结构

下面分别对这四种 OODB 体系结构及其优缺点进行论述。

4.1 数据库服务器

在数据库服务器体系结构中，几乎所有的数据库处理均在服务器方完成，只有极少的处理在客户方。客户进程只是负责把应用的数据请求发送到服务器，并从服务器接收结果，然后再传送给应用[2]。

在 DBMS 的 Client/Server 体系结构中，数据库服务器是现代 RDBMS 广泛采用的一种体系结构，主要因为 RDBMS 诞生于大型机和小型机那个时代。数据库服务器不适合 OODB 应用处理的特点，因而很少被具有 Client/Server 体系结构的 OODB 所采用[2]，其原因是，OODB 要支持的应用主要不是商务处理，而是诸如 CAD/CAM，OIS 和 AI 等所谓的非标准数据库应用。这些非标准应用的特点是要对

复杂对象进行较复杂的处理，常常要求长事务支持。若采用数据库服务器体系结构的话，势必会导致服务器机器负载过重，从而无法满足非标准应用的性能需求。鉴于目前工作站常常具有大量的剩余处理能力和虚存空间可资利用，因而非标准应用非常希望把更多的工作放在工作站上来完成，一来可为非标准应用提供更快的响应速度，二来可减轻服务器上的过载负荷。对象服务器，特别是页服务器便是在这种情景驱动下的产物。

数据库服务器体系结构的优缺点均是显而易见的：需要进行网络数据传输的量少，服务器成为系统瓶颈，不能满足应用的快速响应要求。

4.2 对象服务器

对象服务器型的 Client/Server 体系结构把 DBMS 的处理功能基本上均衡地分布在客户与服务器之间[2]。服务器进程与客户进程进行交互的是逻辑数据单元，即单个的对象[7]。服务器进程的典型职责是[2,4]：保证数据完整性、进行查询的优化、执行存储过程、完成存储管理。因而需要管理锁、实施对磁盘的修改、维护和搜索索引结构，还要负责物理日志和恢复。而由客户进程来协调各事务，完成部分模式管理工作，并把持久对象表现到程序设计语言环境中。

在对象服务器体系结构中，工作站和服务器各自都维护有自己的对象缓冲区，当客户需要访问某一个对象时，先在自己的缓冲区中进行搜索，若未找到，再向服务器发出对象请求；若对象亦不在服务器的缓冲区中，服务器便从磁盘中检索该对象并回送给发出请求的客户；若对象（逻辑上）虽在服务器的缓冲区中，但已被其它客户加了相冲突的锁，服务器还需要在本次对象请求与持有冲突锁的客户之间进行协调，只有当协调成功后才能把该对象发送给申请者并授予对该对象的相应的读/写权限。回调式封锁（callback locking[7]）是很适合 Client/Server 体系结构的一种并发控制技术，它是在出现锁冲突时，由服务器向每个持有冲突锁的客户发出对象回调请求，各客户根据本地事务的实际使用情况做出相应的回答。本文在第5节还将对回调技术进行更详细的介绍。

对象服务器体系结构最本质的好处是可以在服务器与客户之间只传送消息和结果对象，少传送操作数对象。特别是可先在服务器上对大范围的对象进行筛选后，只把选中的对象传送到客户机上。假如筛选的命中率为1%，就可节省99%的通讯负荷。对象服务器的另一个好处是对方法执行地点的选择具有很高的灵活性。由于服务器和客户所看到的均是数据的逻辑单元（即 object），因而对象方法在任何一处都能执行，这样不仅便于选择计算成本较低的一方执行方法，也有利于平衡工作站和服务器之间的负载，因而对象服务器体系结构可以获得很好的响应时间，较大的系统吞吐率，和较低的计算成本[2]。

对象服务器也有下列一些严重缺点：

1）在最坏的情形，工作站上的每次对象引用都将导致一次远程过程调用（RPC）。若借助有效的簇聚策略（Clustering），通过成组对象传输则可在一定程度上缓解这一问题。

2）服务器的设计极为复杂。

3）当一个方法应用于一组对象，而这组对象同时分布在工作站 cache、服务器 cache 和磁盘上时，情况是很糟的。

4）还有另外一些小的问题：（1）不利于页面级的封锁；（2）对象将被多次拷贝；（3）当应用仅需一个大对象的一部分时，传输的却是整个大对象，这是极不经济的。

5）最后一个问题是，与当前计算机技术发展的趋势不合拍。目前具有10个 MIPS 的工作站正在普及，至少90%的计算能力都将在工作站上来完成，因此进行从工作站到服务器的负载平衡几乎没有意义。

4.3 页服务器

在页服务器体系结构中，工作站与服务器之间进行数据交换的单元是页，在此种结构中，服务器没有对象的概念，只能够处理页面，也无法执行任何施用到对象上的方法，大部分的数据库处理是在客户一端，只有极少的工作在服务器方完成。

页服务器的主要职责是：（1）对磁盘页面进行存储和检索（也包括对磁盘资源的分配）；（2）管理内存页面缓冲区；（3）提供并发控制和恢复服务[4]。

页服务器体系结构中的处理方法与对象服务器体系结构的处理方法是相似的，所不同的是前者的数据传输、并发控制和复制管理均以页面为粒度，后者均以对象为粒度。

页服务器体系结构的主要优点如下：1）OODBMS 的绝大部分复杂管理功能均在工作站上进行，从而使服务器上的负载减小到最低限度，高效的簇聚机制可使传输到工作站上的页面得到更充分的利用，2）由于在页服务器体系结构中，单个服务器可以支持更多的工作站，一般情况下单个服务器就

够用了，因而可免去了实现多服务器时的复杂性。3)避免了上述对象服务器体系结构中遇到的所有问题。

页服务器体系结构之主要缺点如下：1)方法只得在工作站上执行。2)对象级的封锁较难实现。3)非两段锁的B树封锁亦较难实现。4)恢复要同时涉及到客户和服务器两方，但它总比对象服务器体系结构中的多服务器分布数据库的恢复的实现来得简单。5)系统的性能对簇聚机制的有效性依赖较大。

4.4 文件服务器

文件服务器体系结构是对页服务器的进一步简化。工作站使用远程文件服务，如NFS[4]，直接读写数据库页面，服务器的基本功能同页服务器体系结构中的服务器一样，主要进行并发控制和恢复管理。

该结构基本具有页面体系结构中的全部优缺点。另外两个额外的缺点是：

1)NFS的Write操作较慢(因无写缓冲)；

2)工作站直接存取数据库页面，使LOCK请求与页面请求不能同步(要分离进行，造成多余的开销)。

4.5 小结

在OODBS的四种Client/Server体系结构中，从数据库服务器到文件服务器，依次把愈来愈多的数据库处理从服务器转移到了工作站，这与目前工作站的处理能力及存储容量的迅猛提高这种发展势头是一致的。在数据库服务器和对象服务器这两种体系结构中，工作站与服务器间进行数据交换的单位均为对象，而在页服务器和文件服务器体系结构中数据交换单位均为页面。在OODBS体系结构谱系中，数据库服务器与文件服务器分别处在两极，被目前的OODB产品或原型系统所采用的较少。RDB(进行了OO扩充)采用数据库服务器的较多，这与RDB的历史有关；Objectivity采用的是文件服务器。早期的OODB多数采用了对象服务器体系结构，如Itasca(及ORION)、Versant和O2的第1版(是一个原型系统)均属于对象服务器体系结构[7]。较新的OODB产品系统更多地采用了页服务器体系结构，如EXODUS、GemStone、O2(V2以后)、ObjectStore、SHORE和MIDS/BUAA[12,13]等均采用了页服务器体系结构[7]，ONTOS则允许程序员在对象服务器与页服务器之间选择。页服务器之所以占取了主导地位，是因为此种结构与当前网络计算环境的发展趋势比较相符，以及前面已经述及的此种结构的种种好处所具有的魅力。

不仅于此，页服务器体系结构在受到较新的OODBS青睐的同时，还演变出了一系列不同的变体[7]，本文在第6节介绍三种重要的变体。

5. 提高OODBMS性能的三个关键技术

改进OODBMS系统性能的两个基本措施是对通讯和服务器磁盘存取进行极小化。页服务器和对象服务器所共同采用的典型作法是允许客户机利用本地存贮空间支持数据项的缓冲，为了更加充分地利用客户机存储资源，甚至提出了跨事务缓冲(Intertransaction Caching[7])的想法，跨事务缓冲需要专门的缓冲一致性维护协议，以确保客户看到的是一个协调的数据库视图。

在Client/Server环境下维护数据缓冲区的一致性，离不开并发控制和复制管理(Replica management)。并发控制普遍采用封锁策略。在RDb/VMS的自适应封锁算法中采用了"衰减锁"(lock deescalation)[8]技术，这与在主存数据库环境中独立提出的衰减锁思想[9]几乎是完全相同的概念。自适应封锁算法(或衰减锁技术)在多节点数据共享环境下有着非常好的性能优势[7]。而在复制管理中，回调[6,7]技术是性能优势很明显的一种复制管理方法。

下面对跨事务缓冲、衰减锁和回调技术的思想精华再做更详细一点的介绍。

(1)跨事务缓冲：支持跨事务缓冲的目的是为了更充分地利用客户机内存，提高对应用的数据请求的响应速度。跨事务缓冲使得在客户机上维护的缓冲区的内容可跨越多个事务的执行。在相继执行的事务流中，处在后面的事务可以直接利用在其前面执行的事务已经获得的数据库子集，而不必通过网络再向服务器申请所需的全部数据项。这样不仅减轻了通讯负载，且能更快地响应客户的数据请求。跨事务缓冲也相应地要求较复杂的缓冲一致性维护协议。

(2)自适应封锁(或衰减锁)：其基本思想是指锁粒度不是一成不变的(比如固定为页面级封锁或对象级封锁)，而是可以根据需要逐渐缩小。在不存在并发控制冲突的前题下，按最大的粒度(如文件)获得锁以便与(分布在别处的)锁管理器的交互最少；不过为了避免过度的资源(数据)竞争，如果随后有冲突发生的话，则允许让已获得的锁再衰减为更细的粒度(如页面或对象)。

(3)回调技术：并发控制(封锁)即是要强制多个并发事务的可串行化，复制管理是对因跨事务缓冲

而导致的相同数据项在不同的客户缓冲区中的多个副本之间的一致性的管理。客户缓冲区的一致性管理不仅要求并发控制还要求对复制进行管理，并发控制的手段是封锁，复制管理的一种有效方法是回调。此二者结合起来形成的所谓"回调式封锁"策略已被证明是维护客户缓冲区一致性的一种非常有效的策略。

回调式封锁算法包括两个方面：回调读 CB_Read 和回调写 CB_Write。回调读的过程如下：(假如采用页级锁粒度)当客户需要访问的页面不在其缓冲区时，便向服务器发出对该页面的请求。若其它客户均未持有对该页面的写锁，服务器便向请求者返回该页的一份拷贝；否则服务器就需等待所有这些锁都被释放后才能将所申请的页面发往申请该页的客户。

回调写的过程如下：为了修改某一个页面，客户必须首先从服务器获得对该页的写锁，当服务器收到了对某一页的写锁请求，而该页又不是被封锁在服务器上，服务器便向除申请者以外所有在其缓冲区中持有该页副本的节点发出一个回调请求。各客户便将这样的回调请求当作是对指定页面的独占锁的请求。如果因为与本地活动事务的锁有冲突使得回调请求不能立即得到许可，客户便向服务器应答："该页当前正被使用"。当回调请求获得了对该页的独占访问许可后，这一页便被清除出当前客户的缓冲区，同时向服务器发送回执。一旦收到了所有的回调回执，服务器便在本次申请者的名下记录下对该页的写锁，并通告申请者其写锁请求已得到许可。随后的其它客户事务对该页的读/写请求将被阻塞在服务器上，直至由持有该页写锁的事务将写锁释放为止。在事务结束的时候，客户先要确保所有被修改过的页均在服务器上有相应的副本。然后才能将写锁释放，并继续维护缓冲区中尚存的所有页面副本(隐含着对这些页面的读许可)。

6. OODBS 页服务器体系结构的三种变体

在概念级，一个数据库由一集对象组成；在物理级，一个数据库可被划分为一集有固定长度的页面。页面是在磁盘和主存之间数据传输的最小粒度。对象是一个逻辑单位，其大小与物理页的尺寸无关。较大的对象一个就能跨越多个页面，较小的对象可以多个共居一页。普通对象的尺寸通常都不超过一页，只有特殊对象(如 BLOB)需要跨越多个页面。为了讨论方便起见，下面不妨假定只涉及普通对象，即多个对象共居一个页面。

OODBMS 在许多方面都存在数据粒度的选择问题，正是这些不同的粒度选择构成了 OODBMS 不同的 Client/Server 体系结构。下面首先介绍 OODBS 的三种重要的系统功能所涉及到的数据粒度，这三种粒度选择与前面介绍的三种关键技术均有密切关系，而且是彼此正交的。

(1)客户/服务器间数据传输的粒度——粗线条地决定了客户/服务器体系结构的种类。OODB 通常选择的传输粒度为对象或页面，从而形成了所谓的对象服务器和页服务器。传输粒度的选择也极大地影响 DBMS 功能在客户与服务器进程之间的分布。

(2)并发控制(封锁粒度)——为支持多客户(即多用户)对数据的访问，DBMS 需要保证事务可串行化。大粒度封锁(极端情形为把整个库锁起来——退化为单用户系统)的特点是低开销、高冲突。

(3)复制管理(回调)——跨事务缓冲允许同一份数据在多个客户缓冲区中都持有数据的一个副本，缓冲一致性的维护要求对复制的管理与并发控制相配合。复制管理同样有一个粒度的选择问题。

页服务器体系结构由于其显著的性能优势，不仅是被当前的 OODB 原型和产品采用最多的一种 Client/Server 体系结构，也是被研究最深入的一种 Client/Server 结构。在前面介绍的三维粒度选择中，把第一维(即传输粒度)固定选为页面，根据第二、三维的不同组合就可形成页服务器体系结构的一系列变体。下面介绍三种有重要意义的变体。

为了叙述上的方便，下面把页服务器简写为 PS，而用 PS-xy 表示 PS 的变体，其中 x 代表并发控制的粒度，y 代表复制管理的粒度。x 与 y 均可取"O"或"A"，分别表示静态对象粒度和自适应(动态决定)粒度。下面讨论的三种 PS 变体即 PS-OO、PS-OA 和 PS-AA 均是从两个方面对基本 PS 进行扩充的结果：1)页粒度(大)传输的通讯效率较高，但伪冲突(对同一页上不同对象的访问)发生的概率亦较大。因而需要做的第一个扩充便是在维持页粒度的高通讯效率的同时，允许并发控制和复制管理采用比页面更小的粒度，即对象粒度；2)第二方面的扩充即自适应粒度调整，其目的是试图回避细粒度的缓冲一致性维护之潜在的高通讯成本。在数据(资源)争抢不太厉害的情况下，细粒度的封锁与回调必然会增加维护缓冲区一致性所必须的通讯次数，因而希望系统能够根据不同的负载，动态地选择与当前数据竞争的激烈程度相适配的封锁及复制管理的粒度。

6.1 对象封锁/对象回调(PS-OO)

PS-OO 方案试图把页面传输具有的通讯优势与对象的封锁和回调所允许的高并发度结合起来。在 PS-OO 中,若对象已在缓冲区中,则客户可不受服务器的干预直接去读这些对象。若所要求的对象不在本地缓冲区中,则由客户决定对象所驻的页面,并向服务器请求这个页面。服务器可在必要的时候从磁盘读取该页面并要保证其它客户均未对这个对象施加写锁。在把这个页面发送到请求该页面的客户之前,服务器还要把该页中已经被其它客户加了写锁的对象打上"不可用"标记,然后才将该页发送到请求该页的客户机上。

当客户希望修改某个对象时,便向服务器请求该对象的写锁。服务器随后所做的处理就如同基本对象服务器的处理一样,即向持有该对象副本的各远程客户发出对象级的回调请求,只不过在 PS-OO 中,客户对回调请求采取的响应不是把要调回的对象全部清除出缓冲区,而只是打上"不可用"标记。

6.2 对象封锁/自适应回调(PS-OA)

PS-OA 与 PS-OO 的差别仅在于在对回调的响应上采取了更为灵活的做法。在 PS-OO 中有可能会发生这样的情形:客户 A 读了页面 P,P 便保留在 A 的缓冲区中且不再被 A 使用。此时若有另一个客户 B 希望修改页面 P 中的多个对象,A 就不得不对服务器因 B 而发出的多次回调请求进行响应。PS-OA 正是为了避免 PS-OO 在这种情况下因多次通信造成的低效性。

在 PS-OA 中,当服务器收到了对给定页面上某个对象的写锁请求后,便向持有该页的副本的所有其它客户发出回调请求。收到回调请求的客户便要检查该页中是否有任何对象正被本地某个事务封锁着。若是,回调响应与 PS-OO 中的相同,即只允许把特定的对象回调回去。否则,若该页中已没有任何对象正被本地事务所使用(加锁),便把整个页面从缓冲区中清除,这时的回调响应与基本 PS 中的回调响应类似。

6.3 自适应封锁/自适应回调(PS-AA)

PS-AA 在封锁和回调上都试图采用自适应方式。在没有数据冲突的情况下,PS-AA 与基本 PS 是一样的,而在发生冲突时只把对发生了冲突的数据的操作衰减为更细的粒度。当引起衰减的数据冲突消退后,PS-AA 还具有再次恢复大粒度操作的能力。

在 PS-AA 中,客户要在对象和页面两种粒度上记载本地读/写操作,而服务器仍同在 PS-OA 中一样,只在页的粒度上跟踪被各客户缓冲的数据副本。

当客户希望读一个不在其缓冲区中(或标记为"不可用")的对象时,便向服务器发出读申请。服务器收到读对象申请后,即检查(页和/或对象级的)冲突情况:

(1)无冲突——其它客户均未对该页加页级的写锁,也未对该对象加对象级的写锁,PS-AA 便与 PS-OO 及 PS-OA 的处理方式一样,即把整个页面发给申请者(把该页中已被其它客户加了写锁的对象打上"不可用"标记)。

(2)对象级冲突——若其它客户已对该对象加了写锁,本次读对象请求即被阻塞直至其它客户将写锁释放为止。

(3)页级冲突——若本次读对象请求与另一个客户已经持有的页级写锁发生冲突,此时需要持有该页写锁的客户衰减其页级锁粒度。为此,它就需要重新从服务器为它在持有该页写锁期间做过修改的(该页上的)所有对象,获得它们的对象级写锁。在完成锁粒度从页面到对象级的衰减后,服务器再检查是否还存在对象级冲突,并按前面的(1)或(2)去处理。

当客户对它希望修改的对象不具备写权限(即未获得对该对象或其所在页面的写锁)时,便对服务器发出写请求。服务器起初对写对象请求的处理就如在 PS-OA 中的一样——若无对象级的写冲突,服务器便向所有持有该页缓冲副本的客户发出回调请求。服务器收到的回调回执只可能是下列两种情况之一:(1)各客户当前都不使用该页面,便把对该页的写权限授予申请者;(2)否则,若至少有一个客户正在使用该页面,就只能把仅对该页中特定对象的写权限授予申请者。

7. 总 结

本文首先从 Client/Server 计算入手,站在软件体系结构的高度,综述了 DBMS 体系结构的发展,对现代 DBMS(特别是 OODBMS)的体系结构进行了系统综述。现把相关文献中所做的定性分析及定量模拟试验得出的一些重要结论再做一次简要总结:

·软件技术与硬件技术的发展是密切相关的,DBMS 的 Client/Server 体系结构正是工作站和网络计算环境飞速发展的结果;

·在(OO)DBMS 的四种基本体系结构(数据库服务器,对象服务器、页服务器和文件服务器)中(它们之间的主要区别在于对客户方和服务器方所分配

的DBMS功能的多少的不同)。(1)数据库服务器主要是RDB实现面向对象扩充时所采用的一种自然选择(渊源于RDB的历史背景);(2)对象服务器因语义问题简单,服务器并发控制的实现相对(页服务器)较容易,因而在最早的一批OODB原型实现中选用对象服务器的较多。对象服务器的两个主要缺陷是数据传输的效率低,且不能利用超级工作站的剩余处理能力和存储容量;(3)页服务器由于不关心语义问题,网络数据传输效率高;又因为大量的数据库处理在客户方完成(充分利用了工作站的高性能),因而服务器的吞吐率较高。现代的OODB产品多数采用了页服务器体系结构。但当对象簇聚(Clustering)程度较低且客户缓冲区不能开得足够大时,页服务器的性能会表现得不如对象服务器(这正是对象服务器的长处)。PS对于顺序浏览查询的性能是较高的;(4)文件服务器利用NFS读数据库页面的性能是非常好的,但NFS的写操作之慢也是出了名的,较大的客户缓冲区对于提高文件服务器的修改查询的性能会有一定的帮助。

在这四种基本服务器类型中,文件服务器处在一个极端,它把绝大部分的工作交由客户方完成,因而从理论上讲,文件服务器体系结构应提供最大的吞吐率。但一般认为文件服务器的综合优势不及对象服务器和PS,对象服务器又不及PS。成组对象服务器(Grouped-Object Server)[7]可认为是针对对象服务器数据传输性能欠佳的一种补偿(文[4]中提及的bundling机制其实也就是想提供成组对象服务),目前还未见到对成组对象服务器更深入的研究文献。

一般认为PS明显地优于对象服务器[5,7],因而在现今的OODB产品中采用PS结构或类似PS结构者居多。而在PS的多个变体中,具有自适应能力的PS-AA被证明能够提供最大的性能,它比纯对象服务器、纯PS以及PS的其它一些变体均有明显的性能优势[7]。

文[4]是较早论述OODB体系结构的一篇文献,其中详细论述并通过实验模拟,分析比较了三种基本的OODB体系结构,即对象服务器、页服务器和文件服务器。文[5]则侧重于从DBMS功能在客户方和服务器方之间的分配之不同,研究了三种不同的DBMS体系结构:CS,PU和EWS,并且没有把DBMS限定为OODBMS,所做的模拟实验是基于RDM的。文[5]中的这三种结构可以大致映射为文[2]中所讨论的三种结构:数据库服务器、对象服务器和页服务器。文[2]是站在Client/Server计算模型的角度定性地论述了数据库服务器、对象服务器和页服务器这三种结构。比较系统且全面地论述对象服务器、页服务器特别是页服务器的三个变体(PS-OO、PS-OA和PS-AA)的当推文献[7],文[7]还就不同的系统模型和负载模型对对象服务器、页服务器及页服务器的三种变体的性能进行了模拟,模拟结果(尽管无法穷举各种可能)表明PS-AA是最好的。

参考文献

[1]M. J. Franklin, et al., Local Disk Caching for Client-Server Database Systems, In Proc. of VLDB'93

[2]Mary E. S. Loomis, OODBMS' Client/Server Architecture, JOOP Vol. 4, No. 2, 1992

[3]Bruce Anderson, et al., Software Architecture: The Next Step for Object Technology (PANEL), OOPSLA'93

[4]D. DeWitt, et al., A Study of Three Alternative Workstation-Server Architectures for Object-Oriented Database System, In Proc. of the 16th VLDB Conf., 1990

[5]N. Roussopoulos, et al., Modern Client-Server DBMS Architectures, SIGMOD RECORD. Vol. 20, No. 3, 1991

[6]C. Lambetal, The ObjectStore Database System, CACM, Vol. 34, No. 10, 1991

[7]M. J. Carey, et al., Fine-Grained Sharing in a Page Server OODBMS, SIGMOD RECORD, Vol 23, No. 2, 1994

[8]A. Joshi, Adaptive Locking Strategies in a Multi-Node Data Sharing System, Same to [6]

[9]T. Lehman, et al., A Concurvency Control Algorithm for Memory-Resident Database Systems, 3rd Int'l. FODO Conf. Paris, France, 1989

[10]Daniel Worden, SYBASE Developer's Guide, SAMS PUBLISHING, 1994

[11]S. Khoshafian, et al, ed. INTELLIGENT OFFICES, John Wiley & Sons, Inc. 1992

[12]Dunren Che, Zhongfan Mai, The Overvall Design of MIDS/BUAA: A Multimedia Intelligent Database System, ICYCS'93. 1993

[13]车敦仁,麦中凡,MIDS/BUAA存储管理子系统,软件学报,1995年No. 4

[14]车敦仁,周立柱,王令赤,面向对象数据库系统的体系结构,软件学报,1995(待发表)