一个 NT 平台上分布式对象数据库服务器系统*

于 戈[1], 王国仁[1], 金泰勇[2], 牧之内显文[2]

[1](东北大学 计算机科学与工程系,辽宁 沈阳 110004);
[2](九州大学 智能系统工学科,福冈 812,日本)
E-mail: yuge@mail.neu.edu.cn; akifumi@is.kyushu-u.ac.jp
http://www.neu.edu.cn

摘要: FISH 系统是一个用于支持先进应用(如 GIS,EC,CIMS)的新一代分布式对象数据库系统.该系统采用了许多新颖技术,如 DSVM(distributed shared virtual memory)、持久堆、页式对象、透明锁、紧凑提交等.重点介绍了该系统的总体结构和设计思想,特别是 FISH 系统在 Windows NT 上实现所涉及的底层技术,包括内存映射、共享内存、远程过程调用、多线索连接、页面故障处理等.基于 OO7 的性能测试表明,FISH 系统在 NT 机群环境下取得了与在分布式 UNIX 环境下同样高的分布执行效率.

关 键 词: 面向对象数据库;服务器;事务管理;DSVM(distributed shared virtual memory);OO7
中图法分类号: TP311, TP133 文献标识码: A

随着计算机性能价格比的不断提高,由拥有高性能 CPU、大容量内存、大容量硬盘的工作站和高速通信网络组成的计算机机群系统 NOW[1],已经成为未来计算环境的一种新趋势.建立基于 NOW 环境的高性能系统的研究与开发一直是国际上的热门课题.例如,美国威斯康星大学开发的 Shore 系统[2]、美国赖斯大学开发的 TradeMarks 系统[3].这种系统的特点是支持数据库、事务管理和控制逻辑在多个节点上的分布处理,可充分利用每台计算机拥有的存储资源和计算资源.并且,具有高速通信网络和并行体系结构,提供高性能的并行处理.这种系统的体系结构更加灵活,能够适应分布式的管理和应用环境,易于扩充和整合已有计算资源,节省用户的投资.FISH 系统是我们联合开发的一个支持先进应用的分布式面向对象数据库系统.FISH 系统提供 ODMG 2.0 对象模型、C++绑定的 OML 操作语言及 OQL 查询语言.通过与 Internet 环境下多媒体应用(如静止图像、动画图像、声音、空间信息的存储和检索)的无缝集成,支持对多媒体数据的高效处理.

当前,分布式面向对象数据库的系统有很多,如 Exodus,Orion,O_2,ObjectStore 和 GemStone[4]等.它们的体系结构可以分成 3 类,即对象服务器结构、页面服务器结构和文件服务器结构.

· 对象服务器:特点是在客户和服务器之间数据的传输单位是对象.服务器有页面缓冲区和对象缓冲区,而客户只有对象缓冲区.对象服务器能够直接处理对象,执行对象的方法.O_2 和 Orion 采用了这种结构.该方法的优点是,可以方便地实现对象级控制,如对象锁.而主要缺点有:(1) 当客户访问的对象不在本地对象缓冲区时,对每个对象的访问都将引起一次远程通信,加大网络传输代价;(2) 对象服务器对其页面缓冲区中的对象,在被客户访问之前,需要转换成对象缓冲区的格式.这种格式的转换,会造成性能下降;(3) 一个对象在网络上可能有多个副本,为了解决它们之间的一致性问题,O_2 采用了将所有更新的对象立即传回给服务器的策略.Orion 采用了预先消除多余的副本的方法.这样做都有可能造成不必要的网络开销,降低了系统的性能.

* 收稿日期: 2000-07-11; 修改日期: 2001-01-31
基金项目: 国家自然科学基金资助项目(69803004);国家教育部跨世纪优秀人才基金资助项目;国家教育部博士点基金资助项目;国家教育部高等学校优秀青年教师基金资助项目

作者简介: 于戈(1962－),男,辽宁大连人,教授,博士生导师,主要研究领域为数据库系统理论与技术;王国仁(1966－),男,湖北崇阳人,博士,教授,主要研究领域为面向对象数据库,并行数据库,查询处理;金泰勇(1971－),男,辽宁沈阳人,博士,主要研究领域为事务管理,DSVM;牧之内显文(1944－),男,日本名古屋人,教授,主要研究领域为数据库系统,多媒体技术.

· 页面服务器:服务器方只有页面缓冲区,客户方可以有页面缓冲区和对象缓冲区.在客户和服务器之间的传输单位是页面.Exodus 和 ObjectStore 采用的是这种结构.该结构的优点是,由于服务器将页面直接传送给客户处理,使服务器的额外处理开销最小化,并且服务器结构简单.但主要缺点有:(1) 服务器不能执行方法,如果要完成在一个大的集合中查找的工作,将把整个集合传送给客户,而不是仅传送由服务器查找后得出的一个小的结果集合;(2) 对象级控制难以实现.

· 文件服务器:文件服务器结构利用远程文件服务(如 NFS),直接读写数据库的页面.GemStone 使用的是这种结构.文件服务器结构的优点在于访问速度较快,因为远程文件服务是由操作系统直接提供的,比通过上层网络通信接口访问服务器的效率要高.但是,文件服务器具有页面服务器的缺点,并且难以实现并发控制.

为了综合对象服务器与页面服务器的优点,既支持对象级控制,又具有高效的页面级处理,FISH 系统采用了页面服务器结构(在大多数情况下,页面服务器的性能高于对象服务器[4]),数据传输单位是页面.同时,在服务器中保持有对象的描述信息,支持对象级控制和处理(如对象锁、对象方法调用).因此,我们称之为页式对象(paged-object)服务器.其实现方法是,利用数据映射机制,将页面以及所包含的对象从磁盘直接映射到服务器和客户工作区中,即直接建立客户缓冲区、服务器缓冲区和磁盘三者之间的映射,同时将被访问页面中的对象信息登记在服务器上.对象在 3 个存储空间中的格式完全相同,避免额外的转换开销.其实现主要是基于分布式共享虚拟内存(DSVM)技术,通过 3 种映射机制来实现的.

(1) 同一场地上内存到磁盘的映射,称为磁盘映射(disk mapping);

(2) 同一场地上客户内存到服务器内存的映射,称为共享内存映射(shared memory mapping);

(3) 不同场地间内存的映射,称为分布式共享虚拟内存映射(distributed shared memory mapping).

与大多数数据库系统一样,FISH 系统的第 1 个版本是于 1994 年在 UNIX 平台上开发的[5].WINDOWS NT 作为一种新的操作系统,具有许多新的特点和价格方面的优势,在应用中越来越普及.但是,由于与 UNIX 系统的不兼容性,如何开发基于 NT 平台的先进数据库系统成为一个新的课题.在 WINDOWS NT 平台上实现 FISH 系统的主要难点有:

- 磁盘映射管理:通过内存和磁盘间的数据映射,支持持久性对象管理.
- 共享内存管理:利用相同地址空间,支持本地不同进程间的数据共享.

Pagefault 处理:利用例外处理器(exception handler),支持透明的页面级和对象级控制,如封锁、回叫处理(callback)等.

- RPC 通信管理:利用 RPC 机制,支持不同进程间的通信.
- 多线程调度:利用多线程机制,支持客户和服务器程序的并发执行.

因此,本文主要介绍在 NT 平台上开发 FISH 系统的主要设计思想以及所涉及的主要底层实现技术.本文第 1 节介绍 FISH 系统的结构和对底层系统支持的要求.第 2 节给出具体的解决方案和采用的主要技术.第 3 节给出性能测试和评价.最后总结全文.

1 系统结构

FISH 系统的体系结构为对等式的(peer-to-peer)多客户/多服务器体系结构,如图 1 所示.每个场地都有一个主服务器守护进程和多个客户进程.服务器进程负责数据库管理和事务管理,客户进程负责执行连接有 FISH 动态库的应用程序.对象模型和 OML 语言功能由该动态库提供.

客户进程和服务器进程都用堆(heap)来存储数据.堆分为持久性和挥发性两种:保存到文件中的堆称为持久堆,仅建立在内存中的堆称为挥发堆.只要堆有足够大,堆中可以任意存放各种对象,不受对象的大小和结构的限制(堆的大小由操作系统的虚拟内存的大小决定,在 32 位操作系统上最大可为 4GB).每个堆被划分成固定长度的页面.从底层来看,页面是数据库的存储、访问和控制单位.系统为每个被打开的堆建立有相应的线程,用于响应客户对堆中数据的访问请求,并完成对堆中数据的管理和控制.FISH 将数据库建立在持久堆上,实现了对象的持久化.任何堆都可由系统中的所有客户和服务器访问和共享.一个堆的创建其原始场地称为“主场地”.其他的由这个堆的副本形成的场地称为“镜像场地”.对于更新操作,采用回叫(callback)协议,通过远程锁请求来

实现.即,申请修改者向所有拥有者节点(在该节点上至少有一个客户持有锁)发出回叫请求.当拥有者节点接受请求后释放封锁.如果拥有者修改了页面,还需将修改内容传输给申请场地.

客户和服务器进程之间以及不同场地的服务器进程之间,使用远程过程调用(RPC)传递请求,进而在客户堆和服务器堆线程之间建立 socket 连接,由 socket 完成传递访问请求、加锁请求、加锁应答等控制信息,完成事务提交或废弃.

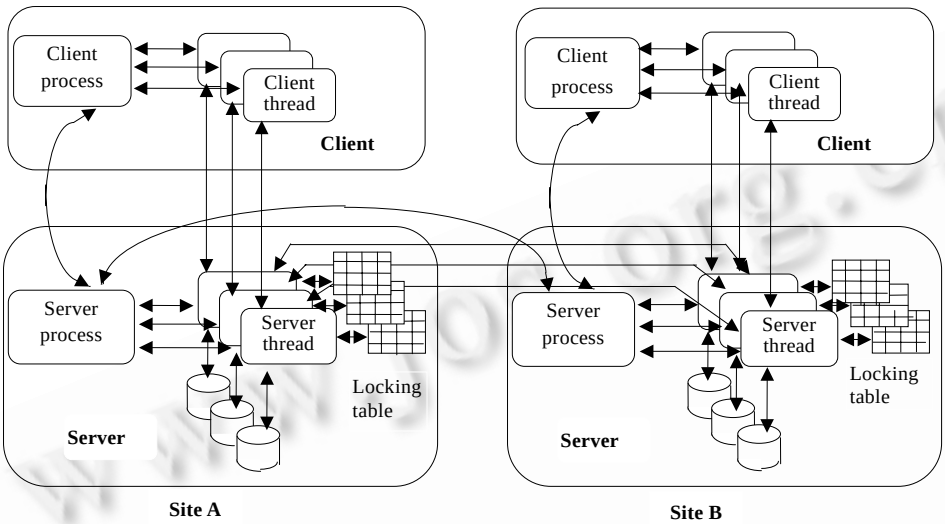

Fig.1 Architecture of FISH system

图 1 FISH 系统体系结构图

2 主要实现技术

2.1 持久对象管理

以往的面向对象数据库系统是把数据库建立在低速的磁盘上,在内存中建立对象缓冲区和页面缓冲区,经过缓冲区进行文件读写操作.这样做所存在的问题主要有:

(1) 页面缓冲区和磁盘之间格式相同,但对象缓冲区中的数据格式与磁盘中的数据格式不同.在读写数据库时,服务器需要在页面缓冲区和对象缓冲区之间进行格式转换.

(2) 服务器在运行期间要对页面缓冲区进行管理,负责在页面缓冲区中查找页、请求页、淘汰页,完成缓冲区与磁盘交换等工作.

(3) 当在页面缓冲区的一个页面上无法容纳一个大的对象时,对象将被分片,存放在多个页面上.对于因某些需要存放在一个连续的地址空间的对象(如图像,视频),程序在处理时需要将存放在页面缓冲区不同页面中的数据转换到一个连续的地址空间.

FISH 系统中采用的页式对象方法解决了以上问题.使对象在内存和在磁盘上的格式是完全相同的,不需要另外建立缓冲区和进行格式转换.由操作系统负责页面在内存和外存之间的交换,其磁盘操作比数据库系统更有效.通过内存映射的方法,还可以将一个大的对象存储在一个连续的地址空间中.在实现上,FISH 系统的客户和服务器都在自己的虚拟内存中为持久堆建立内存映射对象.在建立持久堆的内存映射时,可以共享读写的方式,这使得同一场地上的所有客户和服务器进程共享同一个内存映射对象,而不占用额外的磁盘空间.

在 NT 环境下,我们使用系统提供的内存映射对象.NT 系统每个进程拥有 4GB 的虚拟存储空间.用户可在虚拟地址空间中的用户区建立内存映射对象,将虚拟内存中的内容与磁盘上的文件联系起来,像使用内存一样直接读写文件.

2.2 共享内存管理

共享内存通过多个进程共享一段内存的方法,达到相互通信的目的.在 FISH 系统中,共享内存用于建立挥发性堆,实现不同客户进程和服务器之间的数据共享.但是,NT 系统不直接提供建立共享内存的过程调用.我们通过建立内存映射对象,然后在不同进程建立视图,映射同一内存映射对象的方法,实现共享内存.

在 NT 中,由于每个进程都有不同的虚拟地址空间,一个进程中的内存映射对象的句柄和起始地址在其他进程都变得无效,因此不能用直接传递内存映射对象句柄或起始地址的方法建立共享内存.解决办法为,通过获得双方进程句柄,将在一个进程建立的内存映射对象句柄复制到另一个进程的地址空间,再将复制后的内存映射对象句柄传给目的进程.目的进程使用这个句柄,将句柄指向的内存映射对象映射到自身的地址空间.这样,不同的两个进程共享了同一个内存映射对象.

2.3 透明锁和异常处理

FISH 支持透明锁,即在客户程序进行数据读写操作时,用户不需要使用加锁、解锁命令,由 FISH 系统的内存保护机制自动加锁.在 NT 系统中,每个进程的虚拟内存都分为许多页面,页面的属性可以设为禁止读写、只读或可读写.当读写操作与页面的属性发生冲突时,将产生一个访问冲突的异常事件.FISH 通过捕捉这一异常事件来实现透明锁.客户进程每建立一个堆,就在堆对象表中增加一条记录,记录堆对象的指针、堆起始地址和堆大小.堆中每页的属性设为禁止读写.当客户程序对堆中的对象进行读写操作时,将与该页面所设置的不许读写的保护属性发生冲突,产生一个访问冲突异常事件.我们通过在客户方的主程序中定义一个异常事件处理程序,捕捉整个客户程序中的所有访问冲突异常事件,以实现自动加锁.

对异常事件处理的过程如下:

(1) 客户程序对堆中的对象进行读写操作,与页面的保护属性发生冲突,引起访问冲突异常事件,触发 SEH 定义的异常事件处理函数.

(2) 异常事件处理函数从冲突信息中得出产生冲突的内存地址和冲突类型(读冲突或写冲突);根据内存地址查找堆对象表,找到冲突的地址所在堆对象的指针,计算冲突地址在堆中的页号和偏移地址;客户堆对象向服务器中对应堆的线程请求加相应的读锁或写锁.

(3) 服务器收到请求后,检查是否允许客户对该页面进行加锁.如能加锁,则服务器发送授权信息给客户;否则,将客户请求挂起,使客户保持等待直到许可为止.

(4) 客户得到对页面的授权,修改页面的保护属性为可读或可写;处理函数返回,客户程序重新执行引起异常事件的语句.

2.4 远程过程调用和通信管理

FISH 系统为了完成分布事务处理,需要进行从本地客户到本地服务器或本地服务器到远程服务器的远程过程调用(RPC).FISH 系统中定义了建立内存对象、删除内存对象、事务开始、事务提交、事务撤消等几个标准远程过程.另一方面,FISH 系统中使用有连接的 socket 来完成通信功能.socket 的端口号是随机生成的,一方先要 RPC 将自身的 socket 端口号通知另一方,从而建立起 socket 连接.

但是,NT 3.51 本身不提供 RPC 功能.我们采用 SUN 公司的 RPC 软件包,在 NT 系统增加端口映射服务.即,由 RPC 服务器上的守护进程产生一个 socket,并将任意本机端口装载在 socket 上,通过端口映射进程登记自身,等待客户端的远程过程调用请求.

FISH 客户应用程序首先通过主机名找到 RPC 服务器,进而与远程的端口映射进程联系,找到服务器的端口号,用这一端口号给服务器发送数据,等待应答.服务器根据请求的不同执行相应的程序,将结果返回客户.

2.5 多线索调度

线程是进程的执行单元,比进程开销更小,执行效率更高.在同一进程中,可创建多个线程,共用一个虚拟地址空间,通过并发执行,提高系统工作效率.以往的数据库服务器通常对每个客户仅提供一个线程处理.FISH 系统的服务器设计成一个多线程的进程.一个服务器进程不仅可以同时与多个客户进程进行通信,对同一个客户

可建立多线程连接,并行处理同一个客户的不同请求.当客户请求对服务器上的堆进行访问时,服务器就建立一个线程,与客户通信.访问结束时,终止线程.这样,当一个客户访问服务器上的多个堆时,就存在着多线索的连接.

由于在服务器进程中建立的多个线程共享同一个地址空间,它们可以直接存取全局变量,如果不在程序中加以保护,将造成数据的不一致性,因此,系统有必要为线程提供同步机制.我们使用 NT 提供的临界区的技术来同步线程,保证线程对共享资源的独占访问.临界区在 FISH 服务器进程启动时建立,一次只能有一个线程占用临界区.当一个线程需要独占地访问共享资源时,线程在没有其他线程占用临界区的情况下进入临界区,然后进行读写操作.如果其他线程同时也要读写,该线程就将被阻塞,直到占用临界区中的线程完成操作,从临界区退出.为了增加系统的并行性,减少线索同步造成的开销,堆的封锁表这一需要被线程频繁访问的信息,没有作为全部变量集中保存,而是作为局部变量存放在每个线程中.

3 性能测试与评价

3.1 OO7测试的设计

OO1 测试标准是第 1 个面向对象数据库的测试标准,但是,由于 OO1 存在有对数据库的性能描述不全面的缺陷.美国威斯康星大学提出了改进的 OO7 测试标准,设计了更复杂的测试数据库以及更全面且典型的遍历、更新和查询操作[6].

OO7 定义了一个描述部件之间复杂装配关系的工程数据库作为测试数据库.一个 OO7 测试数据库由 1~N 个“模块”组成,每个“模块”指向一个高度为 7 层的“装配树”和一个“手册”大对象,装配树的非叶子节点称为“复杂装配”节点,叶子节点称为“基本装配”节点,每个“复杂装配”节点指向 3 个子节点.每个“基本装配”节点指向 3 个“组合部件”,每个“组合部件”包含 M 个“原子部件”.每个“原子部件”与其他“原子部件”之间建立一定的联系.OO7 标准通过规定不同的“模块”个数和“手册”的大小,定义小、中、大 3 种规模的 OO7 测试数据库.对每种规模的数据库,OO7 又通过规定不同的“原子部件”连接度,定义了 3,6,9 这 3 种不同复杂度的数据库.

OO7 的测试方法分为冷测试和热测试两种.冷测试(cold test)是指,当数据库缓冲区和磁盘缓冲区中不存在任何数据库数据时的测试,热测试(hot test)与之相反.为了在不关机的情况下进行冷测试,我们每次先关闭 FISH 服务器,然后执行一个磁盘缓冲区内容清空程序,保证数据库缓冲区和磁盘缓冲区中不含任何数据库的数据.为了进行热测试,则重复运行两遍同一测试程序,取第 2 次作为热测试.对每一项冷、热测试,我们还要对本地和远程两种情况分别进行测试.本地测试是指客户和数据库在同一场地上,而远程测试是指客户和数据库在两个不同的场地,即需执行分布式操作.在测试中,每一项测试都定义为一个事务.这个事务开始时进行计时,得到了事务开始的时间,到事务结束时得到结束的时间,两者相减得到的是执行测试的程序的运行时间(精确到毫秒).同一种测试完成 4 次,取平均值作为最后的测试结果.

测试使用的硬件环境为由 10MB 以太网连接的两台 Pentium PC 组成的计算机机群系统,处理器为 Intel 133MHZ,具有 32MB 内存和 1.2GB 硬盘,磁盘交换区大小为 200MB 字节.软件环境为操作系统 WINDOWS NT 3.51,VISUAL C++ 4.0 语言.整个程序用 C++和 INADA 语言编写.

3.2 测试结果分析

我们对小、中两种规模的 OO7 测试数据库进行了测试.小型库的大小分别为 7.0 MB,9.2 MB 和 12.6MB.中型库的大小分别为 56.3 MB,85.3 MB 和 113.1MB.

为了节省篇幅,这里仅给出 T2b 的测试结果(如图 2 和图 3 所示).T2b 测试从模块开始,按深度优先算法遍历整个装配树,对于每个叶子节点的基本装配,搜索其每个组合部件.然后,对每个组合部件,深度优先遍历每个简单装配的原子部件图,并对每个原子部件作更新操作.

由于小型库的数据量小于 10MB,测试机器的内存为 32MB,除去被操作系统守护进程所占用的一部分内存空间外,留给测试进程可用的内存还剩 20MB 左右.在热测试时,FISH 的内存映射管理已将所有的数据装入缓存中,使得在作测试操作时几乎没有 I/O 开销,因而结果中的本地热执行要比本地冷执行快,并且,远地的热执行结

果与本地热执行的结果几乎相同.但是,远地的冷执行结果约为本地的冷执行的 2 倍,这是因为在远地除了网络传送时间以外,还需加上本地的局部磁盘缓存(local caching)时间.

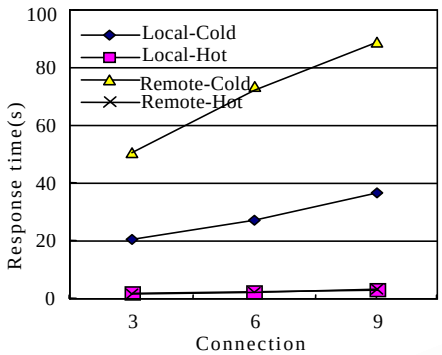

Fig.2 T2b results on small database

图 2 T2b 小型库测试结果

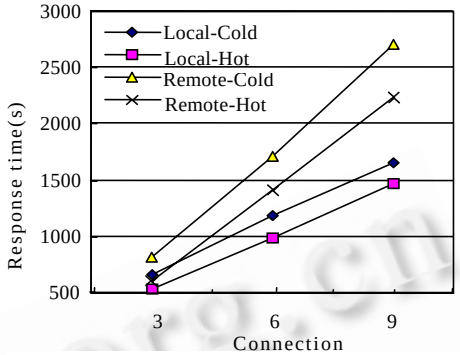

Fig.3 T2b results on medium database

图 3 T2b 中型库测试结果

中型库的测试结果与小型库有所不同.由于中型库内存比较大,在 56MB~113MB 之间,超过了内存 32MB 的容量,缓冲区不可能像处理小型库那样,把整个库调入缓冲区,因此,缓冲区的作用已经不大,I/O 操作不可避免,在本地热执行的结果与冷执行的结果其差别不像小型库测试那样大.但是,在远地热执行的结果与本地的冷执行结果近似,这是因为局部磁盘缓存(local caching)起了作用,省去了远地的网络通信代价.

此外,由于 NT 采用的是“局部 FIFO”算法,进程所占用的内存空间在一定时间内保持不变,在内存中停留最久的页面将最先被换出到硬盘上,因此,当进行热测试所需要的缓冲区不能满足需要时,“热点”数据在缓冲区的命中率较低,频繁地进行换入和换出操作,影响了系统的性能.由于我们的系统是依赖操作系统进行缓冲区调度的,因此,系统热执行性能的提高决定于新的操作系统在这方面的改进.

通过与 FISH 系统在相同硬件配置下的 Solaris 平台上的测试结果比较,性能大致相同.小型库的冷执行、中型库的冷执行和热执行的效率略高于 Solaris 结果[7].

4 结束语

本文讨论了 FISH 系统的系统结构,阐述了 FISH 系统在 Windows/NT 系统上的若干实现关键技术,并给出了基于 OO7 的性能测试结果.在 FISH 系统上,我们开发了支持 ODMG 2.0 标准的对象查询语言 OQL 编译器,并且以 FISH 系统为研究平台,开展了一些研究和应用,如并行路径表达式算法、基于 CORBA 信息集成系统的元数据仓等.我们正在进行的工作包括并行 OQL 编译器、XML 数据管理、空间数据管理等.我们期待 FISH 系统在 Internet 社会的多媒体应用领域中,发挥出更突出的作用.

References:

[1] Anderson, T., Culler, D., Patterson, D. A case for networks of workstations (NOW). Technical Report, Berkeley: EECS Department, University of California, 1996.

[2] Carey, M., DeWitt, D., Franklin, M., *et al*. Shoring up persistent applications. In: Proceedings of the SIGMOD Conference on Data Management. Minneapolis. 1994. 383~394.

[3] Amza, C., Cox, A., *et al*. TreadMarks: shared memory computing on networks of workstations. IEEE Computer, 1996,29(2):18~28.

[4] Dewitt, D., Futtersack, P. A study of three alternative workstation-server architectures for object oriented database systems. In: Proceedings of the 16th VLDB Conference. 1990. 107~121.

[5] Bai, G., Akifumi, M. Implementation and performance evaluation of a distributed paged-object storage server. IEICE Transactions on Information and Systems, 1995,E78-D(11):1439~1448.

[6] Carey, M., DeWitt, D., *et al*. The OO7 benchmark. In: Proceedings of the SIGMOD Conference on Data Management. 1993. 12~21.

[7] Wang, Xin-hui, Ye, Feng. Implementation and performance analysis of OO7 benchmark on an OODBMS. Computer Science, 1999,26(4):26~30 (in Chinese).

附中文参考文献:

[7] 王欣晖,叶峰.OO7 测试标准及其在 OODBMS 上的实现与分析.计算机科学,1999,26(4):26~30.

A Distributed Object Database Server System on NT Platform*

YU Ge[1], WANG Guo-ren[1], JIN Tai-yong[2], MAKINOUCHI Akifumi[2]

[1](*Department of Computer Science and Engineering, Northeastern University, Shenyang* 110004, *China*);

[2](*Department of Intelligent System, Kyushu University, Fukuoka* 812, *Japan*)

E-mail: yuge@mail.neu.edu.cn; akifumi@is.kyushu-u.ac.jp

http://www.neu.edu.cn

Abstract: FISH system is a new generation distributed object database system to support advanced applications like GIS, EC, CIMS. In this paper, the design idea and the architecture of FISH system that adopts many novel techniques such as DSVM (distributed shared virtual memory), Persistent heap, Paged-Object, Transparent locking, Compact commit, and focuses on the implementation issues on Windows NT platform, including memory mapping management, shared memory management, RPC, multi-threads scheduling, page-fault handling are presented. The benchmarking testing based on OO7 is made and the performance analysis is presented. The results show that the FISH on Windows NT cluster is as effective as that on Unix.

Key words: object-oriented database; server; transaction management; DSVM (distributed shared virtual memory); OO7

* Received July 11, 2000; accepted January 31, 2001

Supported by the National Natural Science Foundation of China under Grant No.69803004; the Cross Century Excellent Young Teacher Foundation of the Ministry of Education of China; the National Research Foundation for the Doctoral Program of Higher Education of China; the Award Program for Outstanding Young Teachers in Higher Education Institution of the Ministry of Education of China