用户层通信接口 VIA 的研究(一)*)

孙伟平　周敬利　余胜生

(华中科技大学计算机学院外存储国家专业实验室　武汉430074)

Study of User-Level Communication Interfaces VIA(1)

SUN Wei-Ping　ZHOU Jing-Li　YU Sheng-Sheng

(Computer College in Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Distributed applications requires fast and reliable exchange of information across a network. Traditional network architectures do not provide the performance required by these applications, due to communication overhead incurred by messages to move through different layers of the TCP/IP stack in onperating system. VIA defines mechanisms that will bypass the intervention of the operating system layers and avoid excess data copying during sending and receiving of packets. This effectively reduces latency and lowers the impact on bandwidth. The architecture of VIA and its mechanisms are discussed and the advantage and disadvantage of VIA are analyzed.

Keywords　Virtual interfaces, Queue, Memory registration, Data transfer

1. 介绍

随着物理硬件的飞速发展，网络软件越来越成为影响网络性能的重要因素。两者之间的差异主要来源于软件在处理每个消息(message)时的开销过大。在传统体系结构中，网络资源由内核来管理，操作系统(OS)将网络硬件划分成一组逻辑的通信端点，对硬件的访问在这些端点上多路复用，操作系统通过调用网络 API 来实现这个多路复用。经典的协议开销往往是实际传输迟延的几倍。另一方面，在分布式应用环境下，有时候数据需要不断在内核数据缓冲区、用户缓冲区内复制。在高性能通信环境下，发送一个简单的数据包需要十几微秒是不能接受的。一些研究者研究通过降低 TLB 的频率或 cache 的未命中率等来减少传统系统网络堆栈的开销。一些研究者采用了另一个解决方法，就是提供直接的用户级网络(user-level networking, ULN)，操作系统只用于建立数据结构和映射。这样，不需要操作系统的参与，进程就能够发送或接收数据包，从而大大减少通信开销。

Compaq, Intel 和 Microsoft 联合开发了 VI 结构(Virtual Interface Architecture)规范[1]。VI 规范定义了提供用户级数据传输的网络结构。VI 结构与传统的 OS－网络接口不同，它使得用户程序可以更直接地与网络通信，而且通过由操作系统控制的协议堆栈提供同样的保护。VI 结构解决了与网络通信的高开销相关的3个重要问题：

低带宽—网络通信软件的开销限制了网络的可用的带宽。在一些情况下，只有很低的带宽可供使用。

小消息的迟延—由于在分布式系统中进程必须同步使用网络消息，小消息的迟延会大大降低这个网络的性能。

处理开销—对消息的处理大大减少了 CPU 处理应用程序代码的时间。

VIA 已经成为一个工业标准，国外越来越多的企业开始提供支持 VIA 的软硬件产品[4~7]。作为一个为 SAN 设计的一个用户层内存映射通信协议，对其进行研究和实现是有实际意义和价值的。

2. VI 结构

VI 结构通过取消进程在网络调用的过程中采用的内核陷阱来减少传统网络协议的进程开销。事实上，不需要操作系统来发送/接收消息，客户进程就可以一个受保护的接口与网络直接进行通信。这个虚拟接口与传统的 TCP 连接中的 socket 端点相似，每个虚拟接口都是双向的，且支持点到点的数据传输。

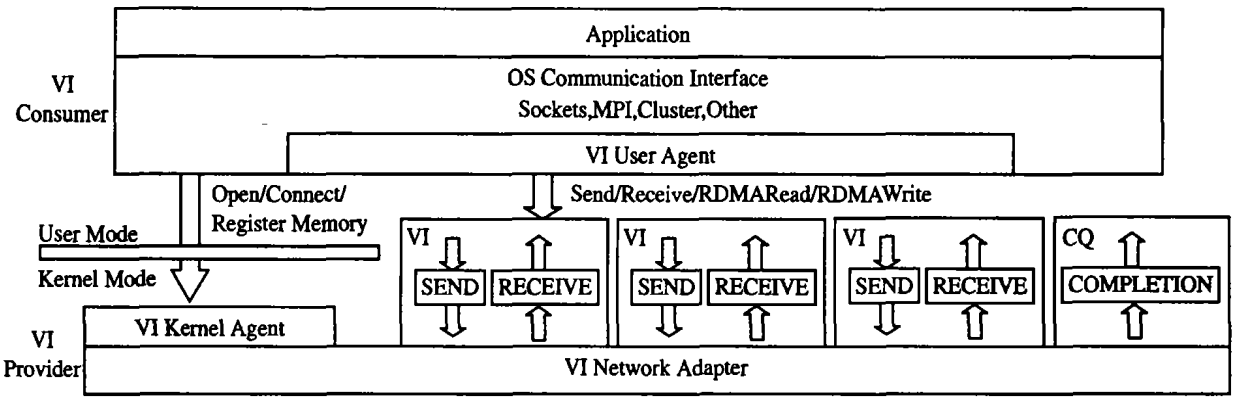

图1　VIA 体系结构

*)本课题由国防预研基金413160502及华中科技大学博士后专项基金资助。孙伟平　博士后，主要研究方向为网络存储设备与性能分析。周敬利，余胜生　教授，博士导师，主要研究方向为网络存储与存储网络，多媒体等。

VI 结构由4个基本部件组成：虚拟接口(Virtual Interfaces)，完成队列(Completion Queue)、VI 提供者(VI Provider)和 VI 使用者(VI Consumer)。其组织形式如图1所示。下面我们分别介绍这4个部件。

2.1 虚拟接口

虚拟接口是一种允许 VI 使用者直接访问 VI 提供者进行数据传输的机制，如图2所示。

一个虚拟接口包括一对工作队列：一个发送队列和一个接收队列。VI 使用者向工作队列以描述符的形式提交发送或接收数据的请求。描述符是一个包括 VI 提供者处理请求的所有信息的内存结构，包括指向数据缓冲区的指针。VI 提供者异步地处理这些提交的描述符，处理完成后用一个状态来标记这些描述符。VI 使用者从工作队列中删除已完成的描述符，并在后续的请求中使用它们。每个工作队列都有一个相对应的"门铃"(Doorbell)，用于通知 VI 网络适配器有一个新的描述符被提交到了工作队列中。Doorbell 通常直接由 VI 的网络适配器来执行，网络适配器不需要 OS 的参与而直接执行 Doorbell。完成队列允许 VI 使用者组合来自多个虚拟接口的工作队列中完成的描述符，将它们放于一个地方。

2.2 VI 提供者

VI 提供者包括一个 NIC 和一个由软件实现的内核代理(Kernel Agent)。VI NIC 用来实现虚拟接口和完成队列，并执行数据传输功能。内核代理通常是一个由 VI NIC 提供商提供的驱动器，用于维护 VI 使用者与 VI NIC 之间的虚拟接口的配置和资源管理。这些功能包括 VI 的建立与破坏，VI 的连接与断开，中断管理和/或中断进程，VI NIC 要用到的内存的管理，错误处理等。VI 使用者通过标准的操作系统机制，如系统调用，来访问内核代理。内核代理通过标准的操作系统设备管理机制与 VI NIC 进行交互。图3是一个 VI NIC 的例子。

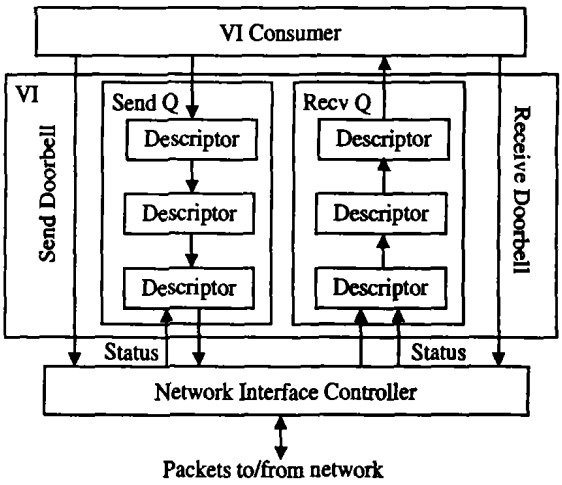

图2 虚拟接口示意图

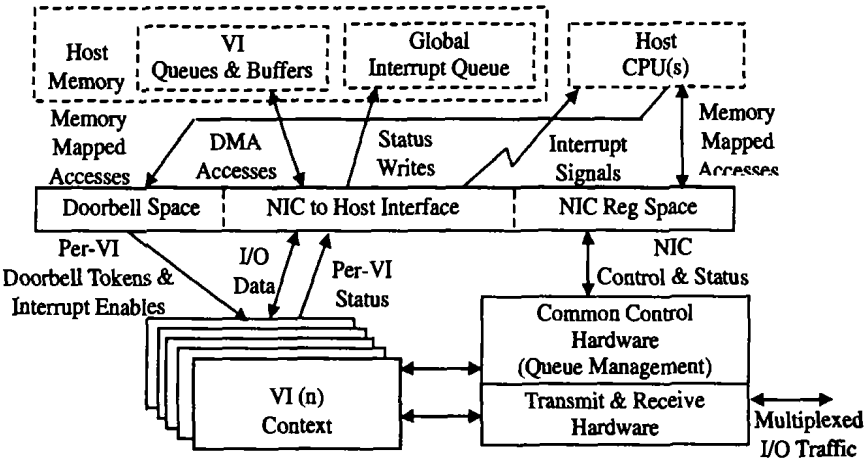

图3 一个 VI NIC 实例

2.3 VI 使用者

VI 使用者是指虚拟接口的用户，通常由一系列应用程序以及操作系统通信工具，如 Sockets 或 MPI，直接与 VI 提供者 API 通信。操作系统通过系统调用内核代理在本地创建一个 VI，并将它与远程系统的一个 VI 连接。连接一旦建立，操作系统就将应用程序的发送和接收请求提交给本地 VI，所有网络行为都不再需要操作系统内核的干预，与传统的网络协议如 TCP/IP 相比，大大降低了通信迟延。

操作系统通常需要加载一个抽象了底层通信提供者(这里指 VI 和内核代理)的详细信息的信息库，此即图1中的用户代理(User Agent)。用户代理由 VI 硬件供应商提供，与 OS 通信工具定义的一个接口匹配。

2.4 完成队列

请求已经完成的通知直接提交给一个完成队列。VI 工作队列与完成队列的关系在一个虚拟接口创建时就建立了。一旦一个工作队列与一个完成队列关连，所有的完成同步必须在完成队列上发生。VI NIC 不需要中断将通知状态加到完成队列中，VI 使用者不需要内核的转换就能同步一个完成。

3. VIA 机制

3.1 队列结构

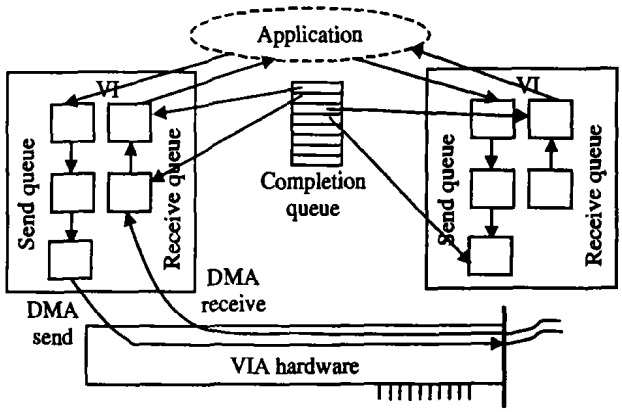

图4 VIA 中的队列

VIA 中的队列与大多数网络设备的队列结构类似。采用队列结构，而不是程序接口(如 Fast Messages 和 Active

Messages 中采用的)意味着更高的软件层能够直接管理所要求的描述符和缓冲区。另外,使用队列可以很自然地将一个操作的完成与其开始分开来。将一个描述符压入发送队列就开始了传输操作;网络接口使用描述符发送完成信号。实际的数据传输或数据接收就可以与正在执行的计算异步进行了。队列设计用于接收分散-集中型描述符,允许应用程序在不邻接的地方创建一个消息。例如,在一个滑动窗口协议中,协议将其信息放置于一个单独的缓冲区,网络接口将数据与协议信息连接起来。

VIA 中的队列如图4所示。每个应用程序可以打开多个虚拟接口(VIs),每个虚拟接口都有它自己的发送队列和接收队列。虚拟接口与一个完成队列相连,VIA 为每一个已经完成的传输或接收操作在完成队列中增加一个描述符。多个虚拟接口共享一个完成队列,通过这种方法,应用程序只需检查一个队列就可以找到与其相关的所有网络事件。

3.2 内存管理

VI 结构要求 VI 使用者向 VI 提供者注册所有的发送和接收内存缓冲区,这样可以消除内核与用户缓冲区之间的数据复制。这种复制常常在与传统网络协议堆栈相关的开销中占据了一大部分。

注册过程锁定一些内存页,允许 VI 硬件执行 DMA 操作,直接访问用户内存。锁定物理内存的缓冲区内存页后,从物理到虚拟的映射以及每个已注册的内存区域的句柄就会提供给 VI 适配器。VI 使用者获得已注册的每个内存区域的句柄,并通过虚拟地址和相关的句柄来引用所有已注册的内存。

内存注册允许 VI 使用者重复使用已注册的内存缓冲区,从而避免了重复的锁操作和转换操作。内存注册从对性能要求比较严格的数据传输路径中去掉了锁页面的开销。

大多数计算机系统要求内存页用于保存消息,被锁定,且在 NIC 能访问它们之前将其虚拟地址转化为物理地址。当数据传输完成后,内存页解锁。传统的网络传输在响应每个数据传输请求时都要执行操作。VI 结构要求 VI 使用者在提交数据传输请求之前先确定用于数据传输的内存,只有得到 VI 提供者注册的内存能用于数据传输。内存注册允许 VI 提供者在缓冲区与 VI 使用者之间直接传输数据,网络不需要在中间缓冲区之间复制数据。传统的网络传输经常在用户缓冲区与中间内核缓冲区之间复制数据。数据复制和缓冲区管理的系统开销较大,并消耗内存带宽。

由于内存注册是一个相对来说较昂贵的 VI 活动,因此通常只在每个缓冲区域的操作之初进行一次。

3.3 数据传输模型

VI 结构提供两种数据传输模型,即发送-接收模型与远程直接内存访问(RDMA)模型。发送-接收模型与传统的消息传递机制类似,包括一个接收操作,消息的接收必须指定数据放置在内存的哪个位置。RDMA 模型只涉及到发送端,不要求接收操作,源缓冲区与目标缓冲区都由发送端指定。VIA 规范定义了两种 RDMA 操作,即 RDMA 写和 RDMA 读。

远程的数据读写给进程提供了一种发送数据到另一个节点,或从其它节点获取数据的机制。这个过程并不需要对远程进程进行操作。VI 结构的发送/接收模型除了所有的发送、接收是异步的以外,与通常的在两个端点间进行数据传输的方法相同。有两个方法发现数据传输已经完成:polling,即进程不断检测已完成消息的描述符队列的头部;blocking,即,使用操作系统同步对象向用户进程发信号告知已完成消息可用。

3.4 VI 中的可靠性等级

VI 结构在 NIC 级别上支持3个可靠性等级:不可靠传输,可靠传输和可靠接收。所有的 VI NIC 都要求支持不可靠传输,而可靠发送和可靠接收则是可选的。只有具有相同的可靠性等级的虚拟接口才能进行连接。

不可靠传输—支持不可靠传输级别的虚拟接口保证一个 Send 或 RDMA Write 只向接收数据的虚拟接口发送一次,且保证错误的数据传输能被接收方检测得到。数据传输出现错误时,虚拟接口之间的连接并不断开,错误只是报告给 VI 使用者,而虚拟接口的状态并不变为错误。

可靠传输—可靠传输虚拟接口保证所有传输的数据只到达目的地一次,如果没出现错误,数据将完好无损,按照提交时的次序到达。如果数据丢失,损坏,或者次序出现错误,VI 提供者就会给 VI 使用者发送一个错误信号。只要检测到错误信号,虚拟接口的状态就会变为错误,连接断开。

可靠接收—如果数据包没有成功地到达目标主机的内存,发送错误信号。可靠接收虚拟接口与可靠传输虚拟接口的区别是:只有当数据到达目标机的内存描述符才以成功状态完成。

在这3种可靠性等级里,只要出现错误信号,那么那个端点随后的数据传输都将被丢弃,而且该端点设置相应的错误位。

总结 通过以上分析,我们知道 VIA 有以下特征:

·直接访问网络接口—提供了低迟延的通信。

·内存注册—VIA 要求每个数据传输要用到的内存必须注册。

·Zero-Copy 协议—通过内存注册,VI 提供者能直接在 VI 使用者的数据缓冲区与网络之间传输数据,而不需要在缓冲区之间复制数据,从而提升了系统的性能。

·为通信设计的协议通道—VI 结构要求为进行数据传输,一个虚拟接口必须与另一个虚拟接口连接。通过连接过程建立起来的 VI 通信通道消除了操作系统对数据传输的保护检查。

尽管 VI 结构能使数据传输路径最优化,但是它也有一些缺陷,比如 VI 的内存管理有如下缺点:

·内存注册很昂贵。VI 注册内存需要的时间比较多,特别是当内存区域较大时。这就阻止了高性能应用程序在网络访问前简单地注册或注销内存,否则会给网络通信带来很大的软件开销。注册内存在 TLB 使用,应用出现代码的复杂性上也是很昂贵的。

·缓冲区空间必须少于机器上的物理内存。许多分布式应用在处理节点间发送大量数据列,如果这些数据列充分大,那么就没有办法在计算开始前注册所有需要的空间。换言之,应用程序在程序执行的某一时刻只能使用大数据队列的一小部分,但是在程序的执行过程需要处理数据队列的许多不同部分。因而 VI 使用者面临两种选择:在计算过程中不停地注册再注销内存,或者总是使用一个缓冲区,然后将数据从这个缓冲区复制到应用程序缓冲区空间。这两种方法都会增加网络访问的迟延。

近来一些软硬件供应商宣布采用 VI 结构,如 GigaNet 在他们的 cLAN 网卡上实现了 VI 结构,另外在一些已有的硬件上通过软件实现了 VI 结构,如 ServerNet (Tandem),Myrinet (Myricom)和 Ethernet (Intel)。硬件 VI 结构可以提

供对 VI 管理、队列管理、内存映射以及标准的网络通信的支持。软件 VI 结构在 NIC 固件(Myrinet),特殊的设备驱动器(ServerNet),或在标准网络驱动器的顶端增加一个中间驱动层(Gigabit Ethernet)来实现 VI 结构的各个功能。

参 考 文 献

1 Compaq. Intel and Microsoft Corporations, Virtual Interface Architecture Specification. Version 1. 0, 1997. http://www.viarch.org

2 Dunning D,et al. The Virtual Interface Architecture. IEEE Micro, Mar.-Apr. 1998. 66～76

3 Buonadonna P, Geweke A, Culler D. An Implementation and Analysis of the Virtual Interface Architecture. In: Proc. of SuperComputing Conf. Nov. 1998

4 Giganet Inc., cLAN Performance. http://www.giganet.com/products/performance.htm

5 Dubnicki C,Bilas A,Li K,Philbin J F. Design and Implementation of Virtual Memory-Mapped Communication on Myrinet. In: Proc. of the 11th Intl. Parallel Processing Symposium, April 1997

6 Eicken T V, Basu A, Buch V, Vogels W. U-Net: A User-level Network Interface for Parallel and Distributed Computing. In: Proc. of the 15th Symposium on Operating Systems Principles, 1995. 40～53

7 National Energy Research Scientific Computing Center (NERSC), M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/research/FTG/via Study of User-Level Communication Interfaces VIA(1)

(上接第121页)

分割表是用来管理索引表块和空闲块的内存结构。分割表块把内存分成若干索引表大小的块,它的头记录着索引表开始的位置及空闲块的开始、结束位置。需要新建索引表时,从空闲块链表中取下一块放到相应的散列表里。

为了减小分割表的长度,定时检测索引链表,如果索引表数目达到了索引链允许的最大值,对于引用次数为0的索引表,把它从索引表链中删掉并链入空闲块。

需要使用一个索引时,先打开内存中的分割表,如果在内存中,即可获得该索引的索引描述符及码值描述符。不在内存,就调入。

7. 一个 SDBMS 的索引管理系统设计流程

以上几部分详细地介绍了索引管理所涉及到的数据结构和管理办法,在这些基础上,下面将介绍一下空间数据库(SDBMS)索引管理系统的设计流程:

(1)数据结构的类定义,包括数据元素的定义、索引页结构的定义以及分割表、索引表的定义等。

数据元素的表示方法多种多样,数据元素的定义正是为了规范化数据元素的表示;索引页结构的定义规定了索引结构的存储格式,本文所示的格式不仅简单明了地说明了索引页和索引页之间的联系,而且极大地增强了存储空间的利用率;分割表、索引表是为了减少磁盘 I/O 的次数提出的用于管理内存中索引信息的数据结构。

(2)与 SDBMS 其它模块的接口(如与缓冲管理器的接口、与文件管理器的接口、游标管理器的接口)设计。

处理游标管理器发出的请求,并通过与缓冲管理器、文件管理器的互操作实现对磁盘上的数据进行读写。

(3)索引算法的设计。

包括属性数据的 B^+Tree 索引算法和空间数据的 R-Tree 算法设计。这是索引管理系统的核心,算法的好坏和查询的速度、精度直接相关,可以根据需求适当地对这两种算法作些改动。

以上是建立一个索引管理系统所必备的几点要求,那么当应用层提出请求按某一码值索引某一数据文件时,操作流程又是如何呢?

(1)按照码值建立码值描述符。

(2)为需要索引的数据文件构造 B^+Tree、R-Tree,创建索引文件。

(3)把该索引描述符加到文件管理器里管理所有索引的"索引类文件"中。

(4)当需要查看或数据文件更新时,首先查看内存中的分割表头的 IndexTableSem 字段,找到散列表地址,接着由 Hash(FID+IDXNO)得到散列值,由此找到相应的索引表链。下面分两种情况:

·如果该数据文件的索引表在这个链上,则:i)得到索引信息。得到该索引的索引描述符等信息,从而按照索引描述符提供的信息,查询或修改相应的 B^+-Tree 和 R-Tree;ii)修改索引表。把索引表的状态字段设为"active",同时引用次数字段加1。

·如果该数据文件的索引表不在这个链上,则:i)查文件管理器的"索引类文件",得到索引描述符并把该索引描述符加到散列表中,以便下次引用;ii)由索引描述符信息查询或修改 B^+Tree 和 R-Tree。

结束语 空间数据库的索引管理系统是数据库管理系统中一个非常重要的层面,它的设计好坏直接影响着整个数据库系统的性能。该文从空间数据的特性出发,结合传统数据库索引方法,在研究分析的基础上,设计并实现了一套既能管理空间数据又能更好管理属性数据的空间数据索引方案,该方案对于空间数据库的研究与实现有着一定的现实意义。

参 考 文 献

1 Garcia-Molina H, Ullman J D, Widom J. Database System Implementation, Pentice Hall, 2001. 3

2 Silberschatz A, Korth H F. Database System Concept. Pentice Hall, 1999

3 Zaniolo C,et al. Advanced Database System. Pentice Hall, 1997

4 Atzeni P, Ceri S, Paraboschi S, Torlone R. Database Systems Concepts, Languages and Architectures. McGraw-Hill Book Company 1999

5 Ramakrishnan R. Database Management Systems, WCB/McGraw-Hill 1998

6 Bach M J. The Design of the UNIX Operating System. Prentice Hall, 2000. 8

7 Beckman N, Kriegel H-P, Schneider R, Seeger B. The R*-Tree: An Efficient and Robust Access Methods for Points and Rectangles. In: Proc. 1990 ACM SIGMOD conf. pp. 322～331

8 Sellis T K, Roussopoulos N, Faloutsos C. The R+-Tree: A Dynamic Index for Mutidimensional Objects. In: Proc. Intl. Conf. On Very Large Databases, 1987. 507～518

9 Mohan C, et al. ARIES: A Transact ian Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging. ACM Transactionson Database Systems, 1992, 17(1)

10 Mohan C, Levine F. ARIES/IM: An Efficient and High Concurrency Index Management Method Using Write-Ahead Logging: [IBM Research Report RJSS46]. IBM Almaden Research Center, Aug. 1989