用户层通讯工业标准 VIA 的设计思想与工作机制*)

The Priciples and Mechanism of an Industrial Standard of User-Level Networking:VIA

刘昊飞　李朝阳　都志辉　马群生

(清华大学计算机系高性能研究所　北京100084)

Abstract　SAN(System Area Network)within cluster brings high bandwidths and low latencies to IPC (Inter-Process Communication). However, the communication performance has not been able to make full use of these hardware improvements due to many software overheads. To address this problem, Intel Corporation, Compaq Computer Corporation, and Microsoft Corporation jointly authored the VIA (Virtual Interface Architecture)specification

As an industrial standard of ULN(User-Level Networking), VI Architecture defines a set of data operations for message passing, to achieve high-bandwidth, low-latency communication between any couple nodes within cluster, as well as minimize CPU cycles. VIA gives direct access to network interface, bypasses operating system in memory region protected mode, avoid copies of data during message passing, and avoid interrupts and context switches if possible.

This article describes the principles and mechanism of the VI Architecture and THVIA.

Keywords　VI Architecture, VI, User-level Networking, SAN, Cluster

引言

VIA 的目标就是有效地提高机群内部 IPC(Inter-Process Communication,进程间通讯)的性能,IPC 的性能主要取决于软件上的发送和接收消息的开销,也包括数据在网络上的传输时间。这些软件开销来源于协议栈的层数、触发的中断数、进程的上下文切换,以及数据的反复拷贝等等。其中前三项开销与传送的消息大小是无关的,而数据拷贝和数据在网络上的飞行的时间是跟消息大小相关的。

快速的处理器可以用更少的时间来执行协议层的代码,为了提高主频,现在的处理器还使用了深度流水线、硬件智能分支预测算法,更多的片上寄存器,更大更快的 cache 等等。这些投入使得代码执行的速度有很大提高,但是同时也带来更大的不利:当代码的顺序执行被中断,当进程的上下文切换的时候,会有比原来更多的时钟周期的损失。因此,处理器时钟频率的提高并没有直接地带来所预期的结果,没有有效地减少每个消息传递的软件开销。

随着高速网络技术的引入,网络带宽从十兆提升到百兆,进而又出现了千兆级的网络,这些提高大大减少了大体积数据的传输时间。但是,通常认为网络物理层的数据率是衡量通讯性能的一个很好的标准是不科学的。事实上,通讯性能的评价必须和软件开销相关联的通讯流量和流量模型相结合起来。Martin 等[2]对机群环境下延迟,开销和带宽的关系做了更深入的讨论。

如果不知道一个系统的流量模型,我们就很难了解到软件开销和网络带宽之间相互关系的重要性,如果通讯的组成主要是大体积消息,那么带宽将起到重要的作用,从而每个字节软件开销就会相对地比较低,然而,当通讯的主要组成是网络上互相传递的众多的小消息的时候,每个字节的平均软件开销就会相对地比较高,消息的延迟就主要来自于软件开销了。

Gusella[3]的研究表明,典型的局域网流量模型呈现双峰,即多于80%的消息在200字节或更小,大约8%的消息大于8192字节。影响机群系统通讯性能的最重要的问题,就是发送和接收消息的时候,内存空间上的操作。传统的 TCP/IP 通讯协议是基于局域网和广域网环境下的,没有对 SAN 进行任何优化,并且协议栈位于操作系统之中,这样就使得消息发送和接收的关键路径上各种开销很大,实现机群系统互连的 SAN 保证了数据传输的高可靠性,使得通讯协议的流量控制、差错控制和拥塞控制等部分得到精简;用户层通讯机制避开了操作系统核心切换的巨大开销,使得用户进

*)THVIA 为国家"九八五"计划所支持的项目.

程可以直接访问网络接口 NIC。机群系统的用户层通讯协议的研究已经不是新的课题了，但是现有的用户层通讯协议尽管各有利弊，却没有统一的设计标准。计算机工业的发展需要这样的规范，在这样的背景下，Intel，Compaq，和 M $ 共同制定了 SAN 通讯协议的新规范 VIA。

目前 VI 体系已经在国际上获得了广泛的支持，Dell[4]，Compaq[5]等公司都发表了支持 VIA 的白皮书，由 LBNL（Lawrence Berkeley National Lab）所支持的 Modular VIA（M-VIA）是 VIA 标准在 Linux 操作系统下的一个实现版本，它在不同的商业网卡上软件实现了 VIA，M-VIA 1.1已经发布，M-VIA2正在开发中。Berkeley VIA Project[6]是美国加州大学 Berkeley 分校计算机系专门的 VIA 研究项目[7-8]，已经发布基于 Sun UltraSPARC Solaris 和 x86 Windows NT 的代码发行版本。清华大学计算机系相关方面的研究工作也取得了一定的成果。

VIA 主要设计原则

VIA 的设计遵循并且充实了以下原则：

·消除任何的数据相互拷贝；

·尽可能避免操作系统陷入，以避免 CPU 的上下文切换以及 cache 的抖动；

·使驱动程序不再运行于保护模式下，从而使得多个并发进程能够多路复用一个硬件资源（即网络接口）；

·最小化一个进程初始化消息传输时必须执行的指令数；

·取消在 I/O 操作初始化/完成时请求中断的强制；

·定义简单的数据操作集来发送和接收数据；

·保持整个体系的简单化使其能够软件模拟也能够硬件集成。

VIA 描述

我们先介绍几个术语：用户、用户代理 UA 与核心代理 KA。

用户是指工作于 VIA 之上的软件层，既可以是应用程序，也可以是通讯服务层。用户代理向应用程序提供用户层的编程接口，把 KA 提供给上层的功能以标准库函数的形式实现。在 M-VIA 中，UA 就是最上层的 VIPL（Virtual Interface Programming Layer），用户的应用程序通过这些标准的传输接口访问 VI，进行消息传递操作。核心代理是 VIA 规范中运行于系统核心态下的部分，它的地位类似于 VIA 的核心驱动程序。KA 提供：

·连接管理。由于 VIA 是面向连接的，VI 之间连接的建立需要 KA 的干预。由于连接管理面向所有的用户进程和创建的 VI，因此在核心态之下实现。

·VI 管理。VI 是 VIA 规范中进程通讯的接口，它由 KA 统一管理。用户通讯要申请 VI，通讯结束如果不再需要，还将销毁 VI，VI 的申请、创建和销毁，以及其信息的修改和查询都是由 KA 来完成的。

·内存注册与保护。用户通讯使用的缓冲区就是用户向 KA 申请的内存区域，KA 要提供内存区域的限制访问和虚拟地址的转换。这些都属于特权操作，需要在系统的核心态下完成。

·设备驱动程序。作为 KA 功能的一部分，设备驱动程序提供对不同的网络设备的管理和向 KA 提供统一的设备操作接口。

·网络管理。KA 根据需要，还将负责主机的网络配置、路由信息管理和网络状况的监测。

在 M-VIA 中，KA 对应 VIPK（Virtual Interface Protocol Kernel Agent）。并且，由于目前的商用以太网卡没有实现 VI-NIC 的功能，M-VIA 还在设备驱动程序之上和 VIPK 之下提供了一个 ERCL（Ethernet Ring Class Layer），其目的就是保持 VIPK 的设备无关性，向下与设备驱动程序连接，向上提供一个统一的抽象设备传输接口。M-VIA 作为 VIA 的一个实例，层次与结构如图1所示。

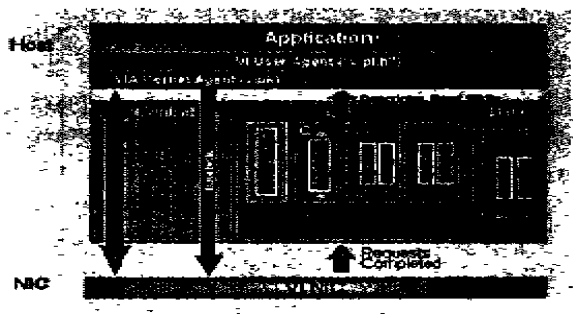

图1 VIA 的层次与结构

虚拟接口 VI

VIA 对每个进程实现了一个网络接口，这就是虚拟接口 VI。一个进程对其它进程的通讯都通过访问 VI 来完成。处于用户层的 VI 对通讯进程来说就是一个网络接口。传统的通讯协议都在操作系统中实现，由于网络接口的消息传输需要经由操作系统的管理和控制，因此通讯进程需要陷入系统核心态中才能够发送或者接收数据。在系统核心态中，所有的任务都被串行调度，因而不能占有网络接口的通讯进程只能被阻塞，直到被系统的调度程序调度才能够执行发送或者接收。虚拟接口 VI 的设计避开了每个进程陷入操作系

统核心的执行机制，用户进程需要发送或者接收数据的时候直接访问自己的虚拟接口 VI。一个进程能够拥有多个虚拟接口，多个进程的多个虚拟接口就包含了多个处于活跃状态的任务，这些任务都处于用户层，能够并发地执行。

每个 VI 拥有两个队列，一个是发送队列，一个是接收队列。两个队列都以链表的形式组成，每一个表项是一个 VI 描述符（描述符的概念我们稍后介绍）。为了把一个描述符加入到队列，用户需要建立该描述符，然后把它记入（post）到适当的工作队列的队尾。该用户还要把完成的描述符从这些描述符被记入的工作队列的队头取出。记入一个描述符包括两步，首先把被记入的描述符连接到适当的工作队列的队尾，然后把这个动作通知给网络接口。所谓通知，就是对一个映射到内存的网络接口寄存器——这个特殊用途的寄存器称为门铃（在后面作介绍）——做写入动作，来触发网络接口硬件的工作。记入一个描述符实际上就是把该描述符的所有权从产生它的进程转移到网络接口控制器上。一个拥有 VI 的进程可以记入四种类型的描述符：发送描述符、RDMA 读描述符、RDMA 写描述符和接收描述符，其中前三种放在这个 VI 的发送队列，接收描述符放在这个 VI 的接收队列。

虚拟接口是用户进程可以直接访问的网络接口，用它实现的进程间通讯，既避免了进程陷入操作系统的开销，又绕开了在操作系统的核心串行执行的耗时机制，并且并发进程的多个虚拟接口实现了对网络接口卡的硬件资源的多路复用。这也是 VIA 作为用户层通讯规范的主要的设计特点之一。

虚拟接口的队列设计就好像实际网络接口的发送和接收 FIFO——网络接口的任务就是取发送 FIFO 队头的消息包执行发送和接收网络上的消息包并放入接收 FIFO。虚拟接口实现发送和接收就是通过把 VI 描述符记入工作队列并查询记入的描述符是否完成。门铃机制是 VIA 的一个特殊的设计，它实现了从进程的虚拟接口到硬件的网络接口的连接，这种连接避免了传统的通过系统核心陷入来实现的硬件接口访问。门铃机制由网卡的硬件实现，每个 VI 的工作队列都拥有门铃。用户通过门铃，就可以使虚拟接口工作队列的描述符属于网卡硬件的控制之下，处于发送可能或者接收可能的状态。

记入方式和门铃机制的运用避免了发送和接收过程中 I/O 中断。在传统的通讯协议中，每一个数据包的发送完成和接收的开始都要触发中断，进行相应的处理，中断的产生会引起进程的切换，浪费很多的 CPU 周期。门铃机制使得 VI-NIC 能够按照一定的算法自动地处理已经记入的描述符，这种处理完全是用户层的，并且不需要触发中断。这种处理是异步的，在每个描述符完成之后需要一定的同步机制，一般使用查询策略。虽然查询可能要多次访问 VI 的工作队列，但是这些开销比起中断来还是要小得多。具体的同步问题我们在下面讨论。

完成队列是另一种标识描述符处理已经完成的方法。队列中的表项都是指向已经完成的描述符的指针。一个完成队列可以对应多个 VI 工作队列，但只能属于一个进程以避免进行传输时发生共享冲突。同一个 VI 的两个工作队列可以独立于不同的完成队列。在 VIA 规范中，完成队列是可选的组成部分。

VIA 通讯模式

VIA 实现的是面向连接的通讯，每个 VI 只能与另一个特定的 VI 相连接，才能实现通讯，消息的发送和接收都在两两互相连接的 VI 上进行。在两个相连接的 VI 上的数据传输是有序的和可靠的。完成一次传输之后需要拆除连接。由于 SAN 的数据通讯量大，而且很多消息都是小体积的，如果每次消息的传输都经过通讯协议的握手，则这些频繁的连接动作将会带来很大的开销，此时如果两个进程在建立连接的基础上再进行通讯，直到这两个进程之间不再有任何的通讯需要再拆除连接，就使得额外的连接开销减到最小，使得消息传递的平均初始化指令数大大降低。进程间相连接的 VI 使得源和目的节点之间消息传递有更低的延迟。

VIA 为用户提供了两种数据传输模式：普通的消息传递操作发送和接收（S/R）和远程直接存储器访问 RDMA 操作。其中 S/R 操作实现了源和目的之间的通用的数据传输功能。在这种传输模式下，发送方向网络接口提供本地数据的地址，接收方通过相应的接口操作提供接收缓冲区的地址，当数据达到时网络接口直接把数据放入此处。由于在 VI 之间采用面向连接的传输，因此发送和接收操作必须一一对应。如果接收和发送之间出现不匹配，则表明连接出现问题，必须对系统或 VI 连接进行重置。

在 RDMA 传输模式中包括远程读（RDMA Read）和远程写（RDMA Write）两种传输操作，对远程结点的数据进行直接访问，其中 RDMA 读目前是可选的。在 RDMA 传输时，本地和远程数据缓冲区的地址都由传输操作的请求方提供。对于 RDMA 写操作，用户提供 VI 本地的发送数据缓冲区地址和远程的接收地址，其中发送缓冲区可以由多个数据块组成，而远程的接收缓冲区只能是一段连续的地址空间。同样，在 RDMA 读操作中，用户必须提供 VI 远程的源缓冲区地址和本地的接收缓冲区地址，其中发送缓冲区地址是一段连续的地址空间，而本地的接收缓冲区地址可以由

多个数据块组成。当然，在进行 RDMA 操作之前必须保证远程的通讯缓冲区已经被设置为可 RDMA 操作的，否则可能会发生不可预知的问题。

描述符

描述符的概念是用来描述虚拟接口交给网络接口所要做的工作的。发送和接收描述符包含一个控制段和一个数可变的控制段。RDMA 读和 RDMA 写的描述符还包含一个地址段，位于控制段之后和数据段之前。

数据段包含描述符类型、描述符长度、描述符状态（初始化的时候是空，之后由网络接口卡在描述符完成的时候写入）、控制段后面的段数、队列中下一个描述符的虚拟地址和与之相关的内存句柄（后面有叙述）和立即数据（如果有）等等。地址段包含远程的内存区域的虚拟地址和相关联的内存句柄，每个数据段将描述一个内存区域，包含本地内存区域的虚拟地址，与之相关的内存句柄和该内存区域的长度（按字节记）。

描述符并不是网络接口卡要传输的数据（含有立即数的情况单讲），它是要求网卡硬件能够理解的数据格式。网卡在读入描述符以后，根据控制段信息执行相应的动作，根据数据段的信息寻址，找到发送数据的区域或者数据将要接收的区域，根据地址段的信息进行 RDMA 的读和写。

同步问题

VIA 提供了查询和阻塞两种机制实现用户进程和已完成的操作之间的同步。当描述符的处理完成之后，网络接口卡可以在该描述符的特定的域进行置位操作，包括完成位和任何类型的出错位，这个动作又把描述符的所有权回传给记入该描述符的进程。

每个 VI 的队列都可以被查询，查询操作就是检查工作队列队头的描述符是否被置为完成。如果已经完成，就调用函数把该描述符从队列头取下，并且把描述符的地址返回给调用进程；如果没有完成，就返回一个单值状态，而队列头没有任何改变。

用户也可以使用阻塞调用的方式来检查队列头描述符的完成情况。如果完成，则与上述查询策略的处理相同，否则，需要请求操作系统把该进程从活跃进程表当中摘除，直到队列头的描述符完成，然后产生一个中断，VIA 支持中断唤醒进程和回调函数处理两种不同的阻塞调用。

立即数据

VIA 允许在每个描述符中指定32位的立即数据。当发送描述符的数据准备好，网络接口卡就直接把数据送到另一端 VI 的接收描述符中。在 RDMA 写的描述符中出现的立即数据在该数据传到远端 VI 的时候消耗一个接收描述符；在 RDMA 读的描述符中出现的立即数据，那就是用来被远端的 VI 写入的，如果远端没有 VI 来完成这个操作，该处的数据域就保持不变。

立即数据传输形式的存在是很有意义的，因为立即数的发送不需要网卡再根据数据段寻址，从而不需要硬件的虚拟地址到实地址的转换和硬件启动主机系统的 DMA 发送，所有这些都将大大节省 CPU 周期。从目前的应用来说，可以用作某些同步信号的操作，也可以作为某些序号。在机群计算环境中，可能存在大量的短消息（尽管它们可能远比32个位要长）在不同的节点之间发送。如果 VIA 能够直接处理这些短消息，而不必再寻址用户数据，启动 DMA 发送的话，整体的通讯性能有很大的提高。

工作队列的顺序性和调度

VIA 在一个 VI 之内保持顺序性和数据的完整性，但是在不同的 VI 之间并不恪守该原则。同一个 VI 上被记入的描述符将按照先进先出的顺序处理，而 VIA 不特别指定不同 VI 上的描述符所完成的顺序，它们依赖于实现的调度算法。不同 VI 之间的描述符的执行并没有绝对的顺序关系，对于每个活跃的 VI，其调度可能依赖于网络接口卡实现的算法（比如轮询各个 VI 所对应的门铃，有已经按下的就启动相应的操作），描述符所关联的消息的大小和下层的传输方式等等，这些都可以有不同的实现。

内存保护

在传统的通讯协议中，通讯区域的内存分配和内存保护是由操作系统来完成的。一个进程不能够读写分配给其它进程的内存区域，同样其它进程也不能读写该进程的内存区域。否则可能出现不可预知的后果。VIA 为所有的 VI 操作提供了内存保护机制，以保证一个用户进程不能在非它拥有的内存区域进行发送和接收。核心代理 KA 实现了一种称为保护标志的机制，使用 TPT（Translation and Protection Table）—— 转换和保护表来提供内存保护。所谓保护标志就是一个与 VI 号和内存区域相关联的一个全局唯一标识符。

一个用户，在创建一个新的 VI 或者注册新的内存区域之前，必须获得内存保护标志，当一个用户请求创建一个 VI 的时候，用户必须指定一个内存标志作为请求调用的调用参数，KA 将检查并确保请求的用户的确是该保护标志的属主，如果检查失败，新的 VI 是不会被创建的。当一个用户注册一块内存区域的时候，如果 RDMA 写和 RDMA 读被使用，则用户必须指定一个与该区域属性相关的内存标志。KA 将检查并确保注册内存区域的进程确实拥有该区域和特定的保护标志；KA 还会检查被注册的区域是否标记为只读。如果请求的用户不拥有特定的保护标志，KA 将拒绝整个的注册请求；如果内存区域被操作系统标记为只读，则 KA 将拒绝使用 RDMA 写的请求。否则 KA 会

在TPT中插入适当的表项并且置一些相关的位，然后把该区域的内存句柄返回。所谓内存句柄，是一个不透明的32位的数据结构。

一个进程可以拥有多个VI，每个VI可以有相同或者不同的保护标志。每个保护标志是唯一的，不同的进程下允许共享保护标志。多个VI只能访问保护标志相同的注册内存区域。因此，并不是一个进程的所有VI可以访问该进程所拥有的所有内存区域。

虚拟地址转换

核心代理KA的另一个重要任务就是进行虚拟地址转换，即在注册内存区域的时候给网络接口卡一个从虚拟地址到物理地址的转换方法。当用户请求内存区域注册的时候，KA执行其所有权的检查，把内存页面对应到物理地址，并且探测执行虚拟地址转换的区域。KA把物理页地址加入到TPT，返回一个内存句柄。至此，用户引用该内存区域的时候能够使用虚拟地址和内存句柄的配对，而不必担心跨越页边界。

内存句柄/虚拟地址对的使用是VIA的一个独有特性，VIA这种对虚拟地址直接使用的允许，意味着编程的方便。一个硬件上支持VIA的网络接口卡(VI-NIC)负责接收内存句柄/虚拟地址对，计算物理地址以及验证用户对请求的操作具有合理的权限。VI-NIC的设计者可以自由决定内存句柄的格式和TPT的存取和获得的方式。

在M-VIA的实现中，当n个页的内存区域被注册的时候，KA在TPT分配n个连续的表项，并且指定一个内存句柄，满足[(虚拟地址>>页偏移位)+内存句柄]的计算结果恰好是序列中的第一个表项在TPT中的索引值。页偏移位是指虚拟地址中每一页的偏移量，一般使用的4096字节的页是12位的偏移量。这里的二进制位都是无符号数，>>是逻辑右移操作，虚拟地址右移之后，一方面截去了低位，另一方面在与内存句柄做减法的时候也忽略超过内存句柄位数的高位。TPT中的n个表项包含n个页的物理地址，页偏移量不计入TPT，因为它们跟虚拟地址的页偏移量是一样的。

使用内存区域注册比使用一般的页注册相比有两个附加的结果。用户不用担心跨越页边界，因而网络接口卡必须知道页的大小，这样在虚拟地址跨越另一页的时候——虚拟地址连续，但是物理页不一定就是相邻的——网络接口卡需要启动一次新的地址转换。虽然用户可以申请只注册一个页的一部分，但是实际上这个页都被注册，因此整个页都服从于被注册区域的保护属性。

VIA的工作流程

VIA规范的一次消息传递流程见图2。

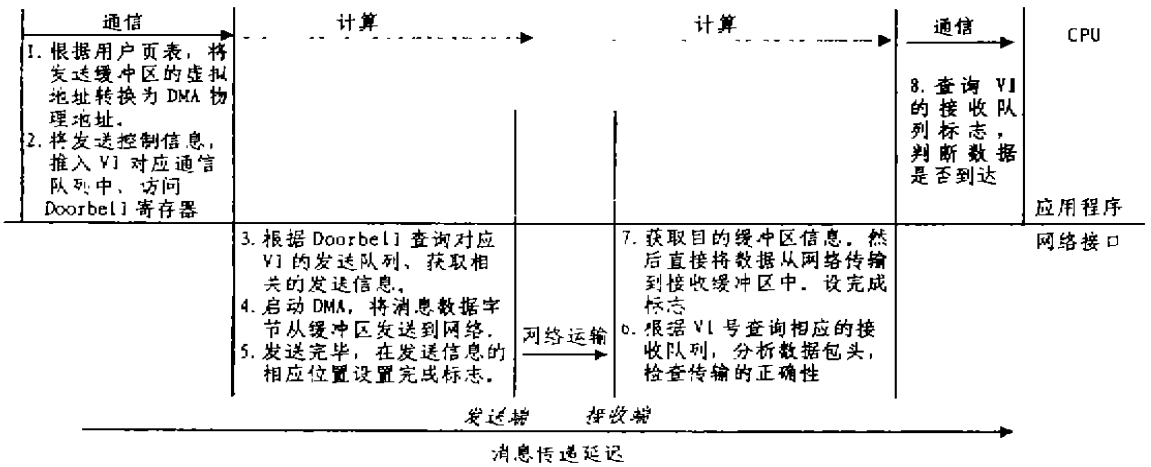

图2 VIA消息传递流程

在消息传输之前，通讯双方都要创建VI，并且使双方的VI建立连接。然后，发送方形成发送描述符，把它记入VI的发送队列，并按响门铃。待发送的数据是程序执行过程中形成的，存在于进程的存储空间，VI描述符被网络接口卡处理的时候，根据控制段形成控制信息，启动DMA发送，根据数据段寻址发送的消息，直接由网卡发送，在接收一方，接收描述符给出了接收方的控制信息，以及存放接收到的消息的内存地址，当匹配的消息到达时，网卡会根据VI号把消息分配到接收缓冲区。在这个过程中，消息从发送进程的存储区直接到达网络接口，又经过网络到达接收方的网络接口，并由它直接传送到接收进程的存储区，中间不

再有任何的数据拷贝，就这一点来说，VIA 实现了真正的零数据拷贝。

VIA 的软件模拟与硬件实现

目前的商业网络接口卡还没有实现对 VIA 的硬件支持，M-VIA 是在软件层支持 VIA 规范的，它利用 ERCL 实现了这一层的功能。硬件集成 VIA 的功能需要网络接口卡能够识别 VI 描述符格式，能够支持硬件门铃机制，能够一定程度地实现卡上的 VI 调度和管理。VIA 标准的制定之初也是为了整个体系足够简单以使其能够在硬件中实现其功能。构建于 VI-NIC 之上的 VIA 会更大幅度地提高机群通讯的性能。

国内 VIA 的研究成果

清华大学计算机系已经开始 VIA 规范和 VI-NIC 的研究，并且已经取得了初步的成果。THVIA[9]是基于 VIA 规范，借鉴 M-VIA，由清华大学计算机系，根据实际情况实现了部分功能的 VIA 系统，该系统在硬件上一定程度实现了 VI-NIC 的支持，包括根据描述符进行轮询发送，VI 门铃机制等等；在软件上实现了半用户层通讯，即发送通过用户层，而接收还经由操作系统。该系统正处于初步实现的阶段，在不久之后就会推出能够运行的版本。

总结 虚拟接口体系 VIA 实现了用户层通讯的工业规范，它向用户提供了简单的数据操作集、方便的应用编程接口；它消除了消息传输关键路径上的多次数据拷贝，实现了真正意义上的零拷贝；除了核心代理的必要的功能之外，它的所有功能都在用户层实现，避免了项操作系统核心的陷入，避免了进程上下文切换带来的 CPU 周期的开销和可能由此引起的 cache 抖动；它使进程通过虚拟接口能直接访问网卡的硬件资源，实现了并发进程对网络接口的复用；它采用了对短消息的特殊处理，这一点有更大的运用潜能；它消除了 I/O 操作触发中断的必要，有效提高了 CPU 周期的利用率和减小了通讯延迟；所以这些实现，使得 VIA 成为轻量级用户层通讯的规范和标准。

参 考 文 献

1 Virtual Interface Architecture Specification. Version 1.0. Compaq、Intel and Microsoft Corporations、Dec. 1997. http://www.viarch.org

2 Martin R P, et al. Effects of Communication Latency, Overhead and Bandwidth in a Cluster Architecture. Computer Architecture News, May 1999. 85～97

3 Gusella R. A Measurement Study of Diskless Workstation Traffic on an Ethernet. IEEE Trans. Communications, 1990, 38(9): 1557～1568

4 Virtual Interface Architecture for Clustered Systems Dell Computer Corporation white paper, September 1998. http://www.dell.com/us/en/hied/topics/vectors_1998-wpvi-a.htm

5 Virtual Interface (VI) Architecture-The New Open Standard for Distributed Messaging Within a Cluster. Compaq white papers, September 1998. http://www5.compaq.com/support/techpubs/whitepapers/ecg0980998.html

6 Berkeley VIA Project. http://www.lcs.berkeley.edu/~philiph/via/

7 Buonadonna P、Geweke A. An Implementation and Analysis of the Virtual Interface Architecture. University of California at Berkeley, Computer Science Department, Berkeley, California, May 1998

8 戈弋. 网络并行互连和通讯机制及其的研究.[清华大学博士学位论文]. 2000

（上接第63页）

单个 BP 网络相比能够达到更好的学习效果。

结论 本文提出一种子网组结构用来代替多输出的 BP 网络，并提出了一种子网组结构的学习算法。理论分析表明，子网络组使用这种学习算法与其相应的 BP 网络原型相比能够达到更好的学习效果。本文提出的思想和方法实际上表明了，在很多情况下，多输出的 BP 网络可以用一组单输出的 BP 网络代替，而且能够达到更好的学习效果。

参 考 文 献

1 刘署光、等. 前馈神经网络中的反向传播算法及改进：进展与展望. 计算机科学，1996，23(1)：70～79

2 Hornik K M、Stinchcomber M、White H Multilayer feedforward networds are universal approximators. Neural Networks，1989，2(2)：359～366