基于RDMA的RapidIO用户态通信接口实现

冀映辉[1,2] 张建东[1] 蔡 炜[1,2] 蔡惠智[1,2]
(中国科学院声学研究所 北京 100190)[1] (中国科学院研究生院物理科学学院 北京 100191)[2]

摘 要 作为一款高性能的嵌入式互联协议,RapidIO支持RDMA操作以获得高性能。目前,针对RapidIO通信接口只有以太网模拟器,这种实现机制限制了RapidIO通信性能的发挥。参考国内外基于RDMA的通信协议实现方法,并结合RapidIO互联协议的特点,提出了一套基于RDMA技术的RapidIO用户态通信接口实现方法。在此基础上,验证了通信接口的性能并对实现方案进行了多种优化。经比较,实现的RapidIO通信接口数据吞吐量是目前所有已知的RapidIO通信接口中最高的。

关键词 RapidIO,远程直接内存存取,用户态通信接口,并行信号处理系统
中图法分类号 TP393.03 **文献标识码** A

RapidIO User-level Communication Interface Realization Based on RDMA

JI Ying-hui[1,2] ZHANG Jian-dong[1] CAI Wei[1,2] CAI Hui-zhi[1,2]
(Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China)[1]
(Physical Department, Graduate University of Chinese Academy of Sciences, Beijing 100191, China)[2]

Abstract As a high performance interconnect protocol in embedded system, RapidIO promises a better performance by taking advantage of Remote Direct Memory Access operations. At present, the only accessible RapidIO communication interface is Ethernet simulator, which limits the communication performance of RapidIO. In accordance with the characte-ristics of RapidIO protocols, this paper put forward a RapidIO user-level communication interface based on RDMA by making reference to the various methods of communication protocols based on RDMA at home and broad. After that, the author validated the performance of this communication interface, and made its implementation to achieve optimization through various ways. By comparison, this RapidIO user-level communication interface is known to gain the highest throughput in all the present RapidIO communication interface.

Keywords RapidIO, Remote direct memory access, User-level communication interface, Parallel signal processing system

影响并行信号处理系统计算性能的一个重要因素是各处理器节点之间的通信性能。目前,比较流行的处理器间互联协议有HyperTransport, InfiniBand, PCI Express, Serial RapidIO[1],千兆/万兆以太网等,它们提供的通信带宽已经可以满足当前大多数计算系统的通信需求。然而,必须看到,即使有高速的硬件互联协议支持,要想给终端用户提供高速的通信接口还是非常困难的。根据文献[2-5],冗余的内存拷贝和繁重的通信开销越来越成为影响多处理器间通信性能的瓶颈,例如,目前公开的利用以太网模拟器实现的RapidIO通信接口仅发挥了RapidIO通信性能的约10%。为了解决这一问题,一些互联协议如InfiniBand, RapidIO, Myrinet, iWARP等相继开始支持Remote Direct Memory Access(简称RDMA)[6,7],在不需要CPU干预的情况下,RDMA允许一台处理器直接读写另一台处理器的内存,从而实现了很高的通信带宽和极低的通信延迟。RDMA的这一特性为在通信接口的设计中减少内存拷贝次数和降低通信开销提供了新的契机,也带来了通信接口实现机制的变革。在通信接口的设计中如何利用RDMA传输机制来最大限度地发挥硬件系统的通信性能,给并行信号处理系统的软件设计者带来了很大的挑战。

1 研究背景

近几年,国际上对基于RDMA通信接口(Communication Interface Based on RDMA,简称CIBR)的实现方法做了很多研究,学者们在各种不同的硬件和软件环境下评价CIBR的通信性能。在有虚拟内存管理的操作系统平台上,实现CIBR还有很多问题。国内外针对这方面的研究很多,笔者的研究也针对这一方面。在文献[8]中,N. J. Boden在Myrinet互联协议下实现了CIBR。在文献[9-11]中,J. Liu, V. Tippsraju, J. Wu等在InfiniBand互联协议下比较了不同平台环境下

到稿日期:2009-07-07 返修日期:2009-09-19
冀映辉(1982-),男,博士生,主要研究方向为阵列信号处理、高性能计算,E-mail:hui03090402@163.com;张建东(1979-),男,博士后,主要研究方向为阵列信号处理等;蔡 炜(1984-),男,博士生,主要研究方为为阵列信号处理等;蔡惠智(1963-),男,研究员,博士生导师,主要研究方向为阵列信号处理。

CIBR 的延迟和带宽。在文献[12-14]中，H. W. Jin，K. Magoutis，In-Su Yoon 等评价了以太网下利用 RNIC[14] 实现的 CIBR 通信性能。目前，笔者还没有发现基于 RDMA 的 RapidIO 通信接口（RapidIO Communication Interface Based on RDMA，简称 RCIBR）实现。归纳起来，在有虚拟内存管理的操作系统平台上，用户态的 CIBR 主要有以下两种实现，如图 1 所示（以 RDMA Write 为例）。

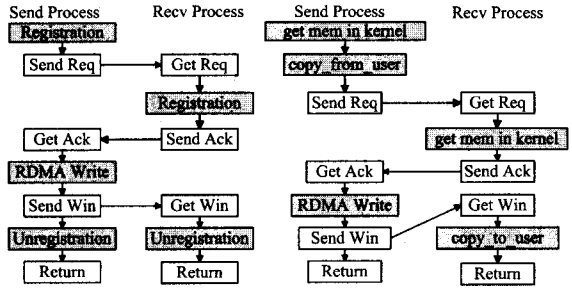

图 1　零拷贝（左）和一次拷贝（右）的 CIBR 实现

方案一　零拷贝实现 CIBR。在启动 RDMA 传输之前，发送端需要事先将存放发送数据的用户态数据区在内核中注册[9,10]，同理，接收端也需要事先将接收数据的用户态数据区在内核中注册。内存注册主要包括以下过程：1）将用户态数据区的虚拟地址映射到内核中的物理地址。2）将用户态数据区加锁，确保在 RDMA 操作的过程中，内核不会将这段数据区交换出物理内存。3）获得用户态数据区对应的物理地址信息序列（用户态连续的虚拟内存在内核中其对应的物理地址可能是分段的）。4）将启动 RDMA 传输需要的信息写入 RDMA 设备的寄存器（仅主设备需要），完成物理内存到设备内存的存储器映射。内存注册是非常耗时的工作，如何减少内存注册的开销，文献[9,13,15-17]中均进行了相关的研究，归纳起来主要有两类方式，一类是设法减少内存注册的次数，另一类是通过缓存加快内存注册的速度，它们都取得了一定的效果。但是这些方式都无一例外地对用户的行为做了特定的假设，如果用户不重复使用传输数据的内存区或者使 cache 失效，都将导致这些改进措施不能发挥作用。

方案二　一次内存拷贝实现 CIBR。RDMA 传输在连续的内核态物理内存之间进行。在启动 RDMA 传输之前，发送端和接收端都事先在内核态中分配好连续的物理内存。发送端先将要发送的数据由用户态拷贝到内核态连续的物理内存中，然后再启动 RDMA 传输。待 RDMA 传输完成后，接收端再将收到的数据由内核态拷贝到用户态的接收数据区。与方案一的内存注册开销相比，方案二主要存在以下额外开销：1）内核态连续的物理内存申请。2）用户态和内核态之间的内存拷贝。3）将启动 RDMA 传输需要的信息写入 RDMA 设备的寄存器（仅主设备需要），完成物理内存到设备内存的存储器映射。依据这一思想实现的 CIBR 仅有文献[18]。如何减少内存拷贝对 CIBR 通信性能的影响，主要有两种方法，一种是设法提高内存拷贝的速度，另一种是利用流水线方式将内存拷贝和 RDMA 操作并行。

本文用 Freescale 公司 2005 年推出的高性能双核处理器 MPC8641D 和 Tundra 公司的 TSI578（RapidIO Switch）构建了 RapidIO 通信网络。在 Linux（Linux-2.6.23 版本）操作系统平台上，对两种实现方案进行了比较，并最终选择方案二实现高性能的 RCIBR。

2　RapidIO 互联协议

2.1　串行 RapidIO 互联协议简介

1999 年，Motorola 和 Mercury 公司共同提出了当时面向下一代的系统互联技术——RapidIO。目前，RapidIO 规范由 RTA（RapidIO Trade Association）维护。RapidIO 是一款高性能、低引脚数、基于包交换的系统级互联协议。RapidIO 定义的传输模式主要有 I/O DMA（RDMA）、消息传输模式和全局内存共享。串行 RapidIO 互连架构解决了高性能嵌入式系统在可靠性和互连性方面的挑战。笔者目前使用的 RapidIO 1.3 规范支持的最高数据传输速率为 10Gbps。

2.2　RapidIO RDMA 技术

RapidIO 定义的 RDMA 传输模式和 InfiniBand 等支持的 RDMA 传输模式稍有不同，其互联双方可以对等地发起传输。RDMA 在被访问端的功能完全由硬件实现，所以被访问的器件不会有任何软件负担。RapidIO 支持的 RDMA 传输模式又可以进一步分为以下几种传输操作：1）NWIRTE 写操作，不要求接收端响应。2）NWIRTE_R 要求接收端响应的写操作。3）SWRITE 流模式写，要求数据长度必须是 8 字节的整数倍，不要求接收端响应。4）NREAD 读操作。其中，SWRITE 是最高效的传输模式；带响应的写操作或者读操作效率则较低，一般只能达到不带响应的传输效率的一半。因此，笔者选用 SWRITE 模式来实现 RCIBR。

3　RCIBR 实现

3.1　方案选择

表 1 显示了在笔者所用系统平台上测得的各种相关操作的开销（测试中假设用户态的数据区都已映射到对应的物理内存区）。根据这组测试结果，笔者选择方案二实现 RCIBR，理由如下：1）方案一每次 RDMA 传输的数据块大小不能大于 4kbytes（Linux 下的页大小），这限制了 RapidIO RDMA 的传输速度。2）方案一中内存注册和 RDMA 传输难以并行，而方案二在软件上很容易实现内存拷贝和 RDMA 传输的并行。3）方案一要求用户传送的数据在内存中必须是页对齐的，造成了内存的浪费，并限制了用户的使用范围。4）只有用户态数据块大于 256kbytes 时，内存注册开销才比内存拷贝开销明显减少。在文献[15]中，S. Liang 的研究表明，当数据块小于 100kbytes 时，内存拷贝的开销均远低于内存注册的开销，同本文的测试结果相似。

表 1　两种实现方案主要开销对比

数据大小 (kbytes)	内存注册	内核拷贝	RDMA 操作	RDMA 吞吐量 (Mbytes/s)
	时间开销(us)			
4	22	6	7	548
8	25	11	12	669
16	31	20	21	764
32	42	40	39	822
64	63	77	75	854
128	108	150	147	871
256	297	310	291	880
512	378	705	579	884
1024	740	1590	1154	887

3.2　RCIBR 发送-接收模型

RapidIO 互联协议定义了门铃消息操作（doorbell message operation），用于在两个 RapidIO 节点之间快速传递 2 字

节的消息。因此笔者选用门铃操作来发送 RCIBR 的控制消息。笔者在内核中预先用_get_free_pages 申请好接收内存缓冲区(Receive Memory Buffer,简称 RMB)和发送内存缓冲区(Send Memory Buffer,简称 SMB),并在系统启动时就完成物理内存到 RapidIO I/O 内存的映射。内核中 RMB 和 SMB 的使用由 RCIBR 驱动统一管理,应用程序要发送数据必须先向驱动申请一段 SMB,发送完成后要向驱动释放这段 SMB。接收端 RMB 的管理与此类似。图 2 显示了 RCIBR 发送-接收数据的流程。

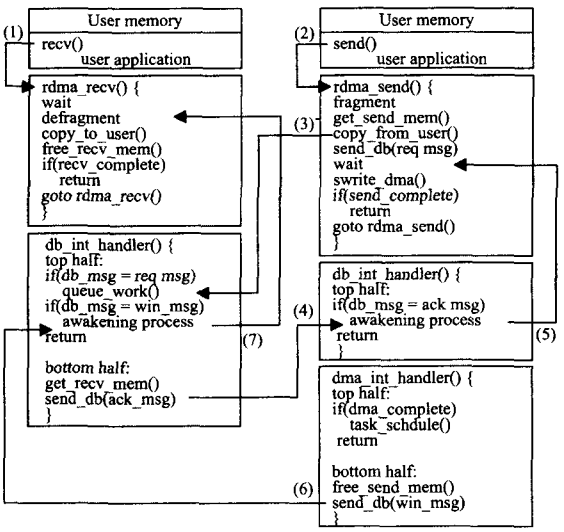

图 2 RCIBR 发送-接收数据模型

如图 2 所示,RCIBR 使用接收进程、发送进程、门铃中断和 RDMA 中断的协调处理来实现数据在用户态进程的传输,使用信号量实现必要的等待。接收端用户进程先启动并切入内核态休眠等待接收数据的到来;发送端用户进程切入内核态后,先向 RCIBR 驱动申请一段 SMB 并将用户态发送数据拷贝到申请的 SMB 中,再向接收端发送门铃消息申请一段 RMB,然后休眠等待接收端的回应消息;在接收端,门铃消息中断处理函数收到申请接收端 RMB 的门铃消息后,在门铃消息中断的下半部向驱动申请一段 RMB 并将申请到的 RMB 内存地址信息通过门铃消息发送给发送端;在发送端,门铃中断处理函数收到接收端回应的门铃消息后,取出 RMB 内存地址信息并唤醒休眠的发送进程;发送端 RDMA 传输需要的参数都已经具备,发送进程启动 RDMA 传输并立刻返回;如果用户进程数据已发送完毕则返回,否则继续进行下一次发送操作;当发送端 RDMA 操作完成后,内核执行 RDMA 中断处理函数,在 RDMA 中断处理函数的下半部,向 RCIBR 驱动释放这段 SMB 并向接收端发送 RDMA 操作完成门铃消息;接收端门铃中断处理函数收到发送端 RDMA 操作已完成门铃消息后,唤醒接收端休眠内核进程;接收端内核进程将收到的数据从 RMB 中拷贝到用户进程接收数据区,然后释放这段 RMB 给 RCIBR 驱动;如果数据已接收完毕,接收进程返回,否则继续下一次休眠等待接收数据。

本方案的设计在发送端使得 CPU 操作(包括内存拷贝操作、SMB 申请操作、RMB 申请操作、中断处理操作等)和 RDMA 操作得以并行,这种并行可以极大地提高 RCIBR 传输数据的吞吐量。发送端并行的效果如图 3 所示。

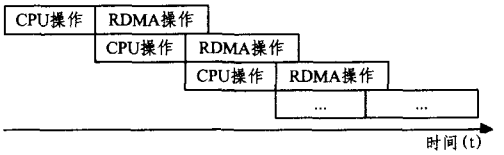

图 3 RCIBR 发送端并行效果图

3.3 RCIBR 的流量控制机制

本方案设计的流量控制机制如下。1)在发送端,当用户进程需要发送数据,向 RCIBR 驱动申请 SMB 时,已经没有空闲的 SMB 可用,则发送端内核进程阻塞直到有空闲的 SMB 可用。2)当发送端通过门铃消息向接收端 RCIBR 驱动申请 RMB 时,接收端已经没有空闲 RMB 可用,则接收端门铃中断下半部(工作队列实现)休眠等待直到有空闲的 RMB 可用,同时发送端内核进程也休眠等待接收端的门铃回应消息。3)如果启动 RDMA 传输操作时 DMA 通道忙,则 swrite_dma 函数休眠等待直到 DMA 通道空闲。

4 测试结果和改进

笔者按照这种方法实现的 RCIBR 已经可以稳定工作。实际测试表明,其在两个处理器节点之间传输数据的吞吐量最高可达到 440MB/s。经过笔者分析,本方案还有以下几方面可以改进。

(1) 参考文献[15],发现信号量唤醒需要的时间长达 31us,笔者实际测试发现信号量从执行 up 操作到唤醒休眠的进程需要 15us。RCIBR 启动一次 RDMA 传输在发送方和接收方都至少有一次休眠-唤醒操作。因此,有必要设法缩短唤醒休眠进程需要的时间。Linux 内核提供了 completion[19] 机制,completion 是一种轻量级的机制,它允许一个线程告诉另一个线程某个工作已经完成。利用这种机制和内核提供的原子操作,笔者在内核中实现了自己的轻量级信号量。笔者实现的这种信号量,其在相同的 CPU 负载下唤醒休眠的操作只需要 10us。

(2) 在发送端,在将要发送的数据由用户态拷贝到内核态之前就向接收端发送 send_db(req_msg)门铃消息申请 RMB,以进一步提高系统工作的并行度。

(3) 缩短中断处理需要的时间。对于很快就能完成的处理的中断操作,尽量放在中断的上半部处理,不使用中断的下半部。只有在门铃消息中断中申请 RMB 时需要休眠的情况下才使用工作队列实现中断的下半部。

经过这几方面的改进之后,RCIBR 的数据吞吐量最高可达到 490MB/s。改进前和改进后 RCIBR 的数据吞吐量如图 4 所示。MTU 表示 RCIBR 驱动底层每次允许传输的最大数据包大小,测试中用户进程每次传输 16Mbytes 大小的数据。

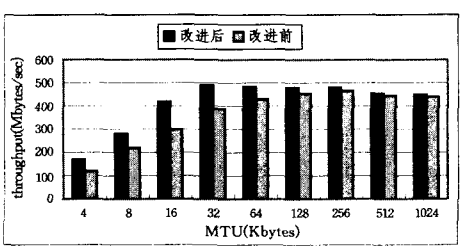

图 4 改进前和改进后不同 MTU 下 RCIBR 的数据吞吐量

如图 4 所示,改进前和改进后相比,RCIBR 的数据吞吐量

最高提高了40%(当MTU=16bytes时)。当MTU=32kbytes时,改进后RCIBR的数据吞吐量达到最高490Mbytes/sec,是方案一实现的RCIBR理想传输速度的89.4%。根据这组测试结果,笔者最终实现的RCIBR底层每次允许传输的最大数据包大小为32kbytes。

结束语 参考国内外相关文献,笔者首次在RapidIO互联协议下实现了CIBR。本方案实现的RCIBR在发送端和接收端均只有一次用户态和内核态之间的数据拷贝,而传统的通信接口在接收端至少有两次内存拷贝。通过采用将RDMA操作和内存拷贝等其它操作并行的方法,本方案实现的RCIBR表现出了良好的性能优势。参考文献[20],本方案实现的RapidIO通信接口比Linux下用以太网模拟器实现的RapidIO通信接口的最高数据吞吐量提高了390%,比文献[20]中作者用RapidIO消息模式实现的通信接口其数据吞吐量最高提高了53%。但是这种实现方案尚存在以下缺点。

(1)本方案实现的RCIBR占用CPU资源最高可达80%以上。

(2)目前,限制RCIBR数据吞吐量的瓶颈因素依然是内存拷贝的速度。

在将来,笔者考虑用双通道内存、优化内存拷贝函数等方式提高内存拷贝的速度,从而进一步提高RCIBR的数据吞吐量。

参考文献

[1] RapidIO Trade Association. RapidIO Specification 1.3 [EB/OL]. www.rapidio.org/specs/current,2001

[2] Callaghan B,Lingutla T,Chiu A. NFS over RDMA[C]// Work-in-Progress Presentation, USENIX File Access and Storage Symposium. Monterey:CA,January 2002

[3] Voruganti K,Sarkar P. An Analysis of Three Gigabit Networking Protocols for Storage Area Networks[C]// Proceeding of International Conference on Performance,Computing,and Communications. 2001:259-265

[4] Regnier G,Makineni S,Illikkal R,et al. TCP Onloading for Data Center Servers[J]. IEEE Computer,2004,37(11):48-58

[5] Romanow A,Bailey S. An Overview of RDMA over IP[C]// First International Workshop on Protocols for Fast Long-Distance Networks. 2003

[6] Recio R,Culley P,Garcia D,et al. An RDMA Protocol Specification [EB/OL]. draft-ietf-rddp-rdmap-02 (work in progress), 2004

[7] Hilland J, Culley P, Pinkerton J, et al. RDMA protocol verbs specifcation (version 1.0)[EB/OL]. RDMA Consortium,2003

[8] Boden N J,Cohen D,Felderman R E,et al. Myrinet:A gigabit per second local area network[J]. IEEE-Micro. 1995,15(1):29-36

[9] Liu J,Panda D K,Banikazemi M. Evaluating the impact of rdma on storage I/O over infiniband[C]//SAN-03 Workshop in conjunction with HPCA. 2004

[10] Tipparaju V,Santhanaraman G,Nieplocha J,et al. Host-assisted zero-copy remote memory access communication on infiniband [C]//Intel Parallel and Distributed Processing Symposium (IPDPS 04). April 2004

[11] Wu J,Wyckoff P,Panda D K. PVFS over InfiniBand:Design and performance evaluation[C]//International Conference on Parallel Processing. Oct. 2003

[12] Jin H W,Narravula S,Brown G,et al. Performance evaluation of rdma over ip:A case study with the ammasso gigabit ethernet nic[C]//Workshop on High-Performance Interconnects for Distributed Computing (at HPDC'05). July 2005

[13] Magoutis K. Memory Management Support for Multi-Programmed Remote Direct Memory Access Systems[C]// 2005 IEEE International Conference on Cluster Computing. 2005

[14] Yoon In-su, Chung Sang-hwa. Implementation and Analysis of TCP/IP Offload Engine and RDMA Transfer Mechanisms on an Embedded System[C]// Lecture Notes in Computer Science. 2005:818-830

[15] Liang S, Noronha R, Panda D K. Swapping to remote memory over infiniband:An approach using a high performance network block device[C]// IEEE International Conference on Cluster Computing (Cluster 2005). September 2005

[16] Ou L,He X,Han J. A Fast Read/Write Process to Reduce RDMA Communication Latency[C]//Proceeding of the 2006 International Workshop on Networking, Architecture, and Storage. August 2006

[17] Huang W,Gao Q,Liu J,et al. Panda. High Performance Virtual Machine Migration with RDMA over Modern Interconnects[C]// Proc. Cluster. Austin:TX. September 2007

[18] Liu J,Jiang W,Wyckoff P,et al. Design and Implementation of MPICH2 over InfiniBand with RDMA Support[C]//Proceeding of IPDPS. April 2004

[19] Corbet J, Rubini A, Kroah-Hartman G. Linux Device Drivers (Third Edition)[M]. O'RELLY,2005:114-115

[20] 梁基. 基于RapidIO的高性能通信接口的设计与实现[D]. 上海:上海大学,2008

(上接第282页)

[18] Zhou Zhi-hua, Zhang Min-ling. Multi-instance multi-label learning with application to scene classification [A]// Advances in Neural Information Processing Systems 19[C]. Cambridge,MA, 2007:1609-1616

[19] Kennedy J,Eberhart R. Particle swarm optimization [C]// Proceedings of IEEE International Conference of Neural Networks, 1995. USA:IEEE,1995:1942-1948

[20] 段其昌,张红雷. 基于搜索空间可调的自适应粒子群优化算法与仿真 [J]. 控制与决策,2008,23(10):1192-1195

[21] Sun Jun, Feng Bin, Xu Wen-bo. Particle Swarm Optimization with Particles Having Quantum Behavior[C]// IEEE Proceedings of Congress on Evolutionary Computation. USA:IEEE, 2004:326-331