doi:10.11887/j.issn.1001-2486.25030003 http://journal.nudt.edu.cn

面向集合通信硬件卸载的维序触发机制和数据缓存方法

徐金波*,董德尊,李宝峰,张 伟,邢建英,张 鹏
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:为了对"天河"网络中基于网卡的集合通信硬件卸载功能进行进一步优化,以支持更多类型的集合通信算法以及更大的消息尺寸,研究了面向集合通信硬件卸载的维序触发机制和数据缓存方法。提出面向多任务并发的维序触发机制,既满足了期望的集合通信语义,又确保了浮点计算操作结果的可复现性;提出基于哈希表和脉冲信用流控的网络数据动态缓存方法,以缓解有限的硬件缓存资源和多任务并发的大量网络数据缓存需求之间的矛盾问题。实验结果表明,与基于软件方式的集合通信操作相比,该方法可以支持多种典型集合通信操作的多种算法的硬件卸载,且性能提升效果显著,同时,硬件实现代价较低,尤其是在缓存资源方面具有较高的利用率。

关键词:集合通信;硬件卸载;触发机制;数据缓存
中图分类号:TP302.2 **文献标志码**:A **文章编号**:1001-2486(2025)06-013-11

论文拓展

Order-preserving triggering mechanism and data buffering method for collective communication hardware offloading

*XU Jinbo**, *DONG Dezun*, *LI Baofeng*, *ZHANG Wei*, *XING Jianying*, *ZHANG Peng*
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: To further optimize the hardware offloading of collective communication based on the network interface card in the "Tianhe" network, and to support more types of collective communication algorithms and larger message sizes, the order-preserving triggering mechanism and data buffering method for collective communication hardware offloading was investigated. An order-preserving triggering mechanism for concurrent multitasking was proposed, which meets the desired semantics of collective communication and ensures the reproducibility of floating-point computation results. A dynamic network data buffering method based on Hash tables and pulsed credit flow control was proposed to alleviate the contradiction between limited hardware buffering resources and the high demand for buffering a large amount of network data from concurrent multitasking. Experimental results show that compared with software-based collective communication operations, this method can support the hardware offloading of various algorithms for several typical collective communication operations, with significant performance improvement. Meanwhile, the hardware implementation cost is low, especially with high utilization of buffering resources.

Keywords: collective communication; hardware offloading; triggering mechanism; data buffering

集合通信广泛运用在并行计算和分布式深度学习中,其通过将多个计算节点的多个进程组织起来协同完成一系列通信操作以达到数据交换、任务控制和全局计算的目的。每台主机计算或训练好的参数通过集合通信进行全局更新和同步,随着数据量和集群规模增大,集合通信将成为性能瓶颈。

传统的消息传递接口(message passing interface, MPI)实现依赖于中央处理器(central processing unit, CPU)进行通信调度,导致计算与通信的竞争,限制了整体性能[1]。小消息集合通信(如深度学习中的梯度同步)和大数据通信(如科学计算中的矩阵传输)对通信模式的需求不同。小消息集合通信更关注延迟优化,而大数据通信则更注重带宽利用率[2-4]。为了提升集合通信性能,研究者们从多个方面进行研究。智能网

收稿日期:2025-03-04
基金项目:国防科技重点实验室基金资助项目(2022-KJWPDL-11);自主创新科学基金资助项目(22-ZZCX-002)
*第一作者:徐金波(1981—),男,山东高唐人,副研究员,博士,E-mail:xujinbo@nudt.edu.cn
引用格式:徐金波, 董德尊, 李宝峰, 等. 面向集合通信硬件卸载的维序触发机制和数据缓存方法[J]. 国防科技大学学报, 2025, 47(6): 13-23.
Citation: XU J B, DONG D Z, LI B F, et al. Order-preserving triggering mechanism and data buffering method for collective communication hardware offloading[J]. Journal of National University of Defense Technology, 2025, 47(6): 13-23.

络设备,如 NVIDIA BlueField 数据处理器(data processing unit, DPU)通过将通信操作卸载到专用硬件,显著减少了 CPU 的负担。例如,Sarkauskas[5] 和 Graham 等[6] 的研究表明,BlueField DPU 能够高效支持大规模消息的非阻塞集合操作(如 Allgather 和 Broadcast),并实现计算与通信的重叠。网络内计算技术通过在交换机或网络接口卡(network interface card, NIC)中嵌入计算逻辑,直接在网络层完成集合操作。例如,Ramesh 等[7] 提出了一种基于网络内计算的 MPI 归约操作优化框架,显著减少了通信延迟;Wang 等[8] 设计的 Roar 路由器微架构进一步支持了网络内 Allreduce 操作,提升了分布式人工智能(artificial intelligence, AI)训练的通信效率。拓扑感知的集合通信算法能够根据网络结构动态调整通信路径,从而优化性能。例如,Liang 等[9] 提出的自适应 MPI_Bcast 算法在"天河"互联网络中实现了高效的广播操作;Liu 等[2] 开发的 TH-Allreduce 算法则针对"天河"系统优化了小数据 Allreduce 操作。多主机通道适配器(host channel adapter, HCA)技术通过利用多个网络接口并行传输数据,提升了集合通信的带宽利用率。Tran 等[4] 的研究表明,多 HCA 感知的集合通信在分布式深度学习任务中显著减少了通信时间。融合集合操作将多个通信操作合并为单个操作,减少了通信开销。Haghi 等[10] 提出的 SmartFuse 框架通过可重构智能交换机加速了融合集合操作,适用于高性能计算应用。分布式 AI 训练(如深度学习模型)对集合通信的带宽需求极高。Rashidi 等[11] 提出的 Themis 调度策略通过动态调整集合操作的执行顺序,优化了带宽利用率。现场可编程门阵列(field programmable gate array, FPGA)因其低延迟和高并行性,在分布式 AI 的集合通信中展现出巨大潜力。He 等[12] 开发的阿里巴巴集合通信库(Alibaba collective communication library, ACCL+)引擎通过 FPGA 实现了高效的集合通信,显著提升了分布式应用的性能。未来的集合通信优化需要更紧密的硬件-软件协同设计,以充分发挥 SmartNICs、DPU 和 FPGA 等硬件的潜力。不同高性能计算(high performance computing, HPC)系统(如"天河"、Intel PIUMA)的硬件架构差异对集合通信优化提出了挑战,需开发更具通用性的优化框架。

本研究小组所在的国防科技大学"天河"系列超级计算机研制团队也在其自主研制的"天河"互联网络接口硬件中集成了集合通信卸载机制[13-17]。

基于网络硬件卸载的集合通信优化取得了较好的性能提升效果,但仍面临如下几个关键问题:

1)多任务并发时,多个集合通信树的多个节点、多个数据通道之间的数据流控制问题:执行集合操作硬件卸载的网卡或交换机需要从数据输入接口接收来自属于不同任务、不同节点的多组数据,这些数据通常需要按照一定的顺序进行处理,才能够满足集合通信操作的语义。因此,需要设计一种数据流控制机制,以确保所有数据能够按照期望的顺序被处理。

2)有限的硬件资源,尤其是数据缓存资源的合理调度问题:随着高性能计算和 AI 计算的技术发展,在执行分布式处理时,节点之间的数据通信量急剧增加。因此,集合通信硬件卸载的数据缓存压力也越来越大,需要对网卡和交换机上有限的缓存资源进行科学管理,以尽可能地提高存储空间利用效率。

3)归约(Reduce)/全归约(AllReduce)等涉及浮点计算的集合通信操作在硬件卸载时的结果可复现问题:对于浮点数而言,任意小的连续区间内都有无限的浮点数,则必然存在无限不能精确表示的数值。从数学角度说,任何浮点数标准都面临用有限的集合去映射无限的集合的问题,都一定会存在精度问题。浮点运算系统中舍入误差的存在导致了浮点运算不满足交换律的特性,当计算顺序不同时,由于产生的舍入误差不同所以计算结果也不相同。现代计算机的多级并行结构和动态资源调度加剧了计算的不确定性,这进一步提高了不可复现现象的出现频率。为了避免由计算顺序变化导致的结果不复现问题,集合通信硬件卸载同样需要对数据到达和执行的顺序进行控制。

针对上述关键问题,本文提出了面向多任务并发的维序触发机制、基于哈希(Hash)表和脉冲信用流控的网络数据动态缓存方法,并在面向"天河"网络的 FPGA 验证平台中对所提出的方法进行了实验测试。

1 面向集合通信的 NIC 硬件卸载

1.1 总体结构

图 1 给出了所提出的面向集合通信的 NIC 硬件卸载总体结构。

NIC 和主机之间通过外围组件快速互联(peripheral component interconnect express, PCIE)

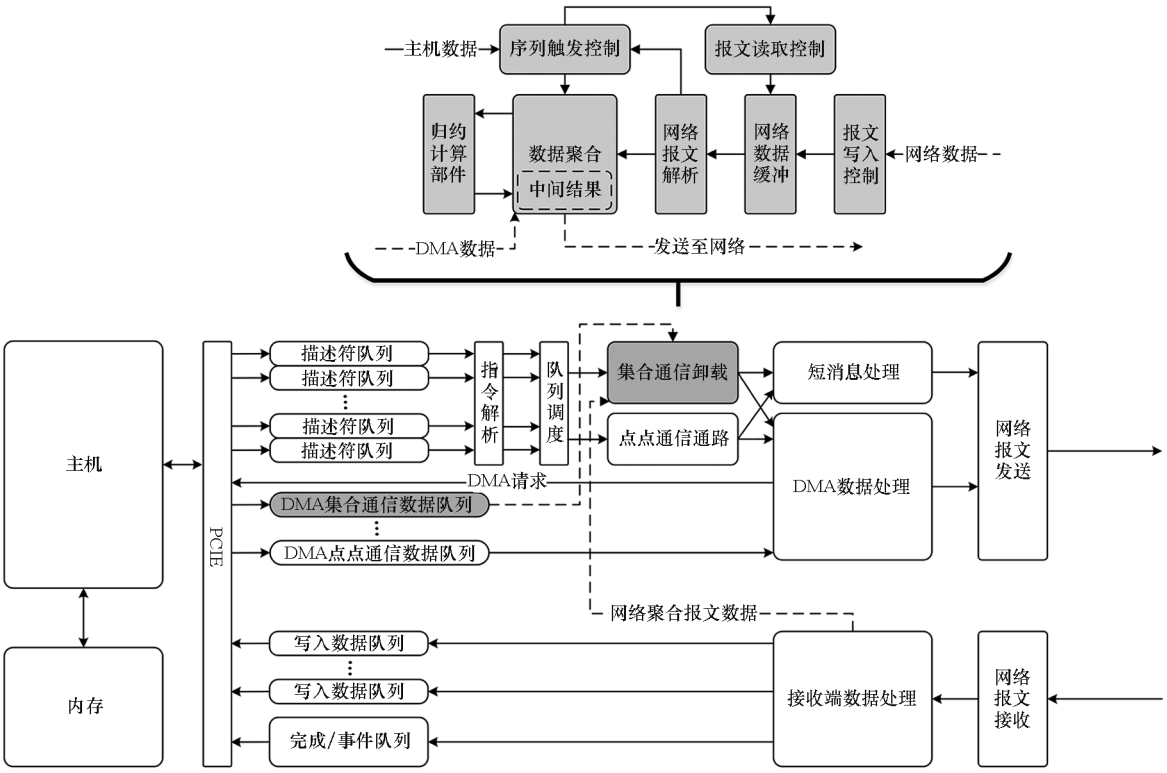


图 1　面向集合通信的 NIC 硬件卸载总体结构

Fig. 1　Overall architecture of NIC hardware offloading for collective communication

接口进行数据交互。主机根据作业需求产生描述符序列,通过 PCIE 接口发送到 NIC 的描述符队列(descriptor queue,DQ)。描述符是主机和 NIC 之间的软硬件交互指令,包括小报文(mini packet, MP)和远程直接内存访问(remote direct memory access, RDMA)两大类。MP 描述符通常在描述符 Payload 中直接搭载需要传输的数据用于小消息传输;RDMA 描述符则包含需要传输的数据的内存地址信息,并基于这些信息启动内存数据访问来完成大消息传输。“天河”互联网络使用虚拟端口(virtual port,VP)机制支持硬件资源的虚拟化来灵活实现用户级通信,它为每个进程提供了独占使用通信硬件的编程视角。当多个进程并发执行时,来自不同进程的通信操作相互独立,彼此不会产生干扰。每个 VP 是一组内存映射寄存器和一组相关的内存数据结构的组合。每个 VP 通过各自的 DQ 进行描述符的交互。

描述符进入 DQ 后,指令解析部件对 DQ 顶部的描述符进行解析,根据相关信息判断每个 VP 的顶部描述符是点点通信描述符还是集合通信描述符。之后,队列调度部件根据各 DQ 的队列空满情况以及后续数据通路的空闲情况调度出一个描述符送入点点通信通路或集合通信卸载部件。

对于点点通信,MP 描述符进入短消息处理部件进行后续处理,RDMA 描述符进入 DMA 数据处理部件,通过 DMA 数据通路从主机内存读取数据后封装为网络报文发送到其他节点。

对于集合通信卸载部件,本工作在“天河”网络当前卸载机制的基础上进行优化,加入了维序触发逻辑以及基于 Hash 表和脉冲信用流控的数据缓存结构。当前主机的集合通信描述符序列到达集合通信卸载部件后,进入序列触发控制模块,等待来自网络其他节点的控制报文。网络报文到达当前节点后,如果为非集合通信操作相关的报文,则进行处理后通过 PCIE 接口写回主机;如果为集合通信操作报文,则由报文写入控制模块根据报文内容构造存储查询键值,采用 Hash 方式存储在集合通信报文数据缓冲区中。报文读取控制模块在读取缓冲区中的数据时,依照节点 ID、进程 VP 号以及报文序列号等信息产生期望的查询键值,从而在对应的缓冲区地址中读出相应报文数据。这种方式可以确保网络报文按照特定的顺序读出,以满足集合通信语义。从缓冲区中读出的报文被解析后,进入序列触发控制模块。网络报文中包含触发控制信息,而当前主机的描述符中包含预设的触发条件,当触发控制信息满足预设的触发条件时,当前描述符进入数据聚合部件

启动执行。根据集合操作类型,数据聚合部件执行相应的具体操作,如复制、替换、计算等。如果该集合操作不是通过 MP 短消息直接搭载数据,则由 DMA 通路从内存读取数据后执行数据操作。当涉及计算操作时,数据聚合部件将操作数流水发送到归约计算部件,中间结果保存在中间结果缓冲区中。处理完成的数据封装为网络报文后发送到其他节点。

为了支持大消息的归约操作,需要对节点间的流量进行控制,以避免因拥塞导致的不同任务之间的流量干扰。通过限制未完成归约的数据总数量来调节节点间的流量。

1.2　面向多任务并发的维序触发机制

为了增强“天河”网络的多任务并发集合通信卸载功能,同时支持更多集合通信算法,提出了基于 JobID + NodeID + VP + SeqID 的维序触发机制。其中 JobID 用于标识不同的集合通信任务;NodeID 用于标识报文的目的节点;VP 用于标识当前任务所使用的软硬件交互虚拟端口;SeqID 用于标识当前报文在当前集合通信任务的执行序列中的顺序。每个报文在报文头中包含了这些信息,JobID 为 8 bit,用于支持最多 256 个并发任务;NodeID 为 24 bit,用于支持数千万节点规模的 HPC 系统;VP 为 11 bit,支持最多 2 048 个虚拟端口;SeqID 为 5 bit,支持最多 32 级操作步骤。

图 2 给出了集合通信相关的描述符格式定义。其中 C 表示该描述符为集合通信操作描述符;CH 表示该描述符为某个集合通信操作描述符序列的首个描述符;CT 为该序列的最后一个描述符;CP 表示该描述符通过网络报文封装发送到目的节点后,将触发目的节点的集合通信操作;CPCount 为当前描述符所处的操作步骤序列号,用于与从网络上接收到的报文的 SeqID 进行比对;SwapLocalData 和 SwapLocalVP 表示当前描述符所搭载的数据和 VP 号将会被执行替换操作;SwapRemoteData 和 SwapRemoteAddr 表示该描述符通过网络报文封装发送到目的节点后,将会替换目的节点当前描述符搭载的数据和地址;ReduceFlag 表示该描述符将执行归约类操作;CollEndFlag 表示该描述符通过网络报文封装发送到目的节点后,其所属的集合操作任务完成,通过向目的节点的完成队列写入完成事件来结束该任务;DestID 为报文的目的节点;SourceVP 为当前任务在源节点所使用的 VP,DestVP 为当前任务目的节点所使用的 VP。描述符的 Payload 部分将搭载实际的操作数(小消息)或操作数在内存中的地址和长度信息(大消息)。

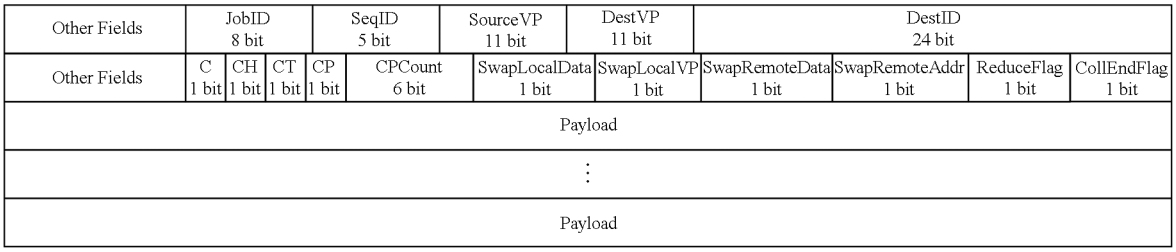


图 2　集合通信相关的描述符格式定义

Fig. 2　Definition of descriptor format for collective communication

图 3 给出了面向多任务并发的维序触发机制示意图。主机数据通过 PCIE 接口进入 NIC 后,根据描述符的 VP 号进入对应的虚拟端口缓冲队列。在描述符解析阶段,对其中的 C、CH、CT 等字段进行解析,对于 C 为 1 的描述符,送入集合通信卸载部件。

为了支持多任务并发的功能,配置多个集合通信卸载部件,所有来自主机的描述符通过负载均衡调度方式共享这些卸载部件。另外,同一卸载部件也可以支持多个任务的分时并发。属于同一序列的描述符需要送入同一卸载部件,即 CH 为 1 的描述符和 CT 为 1 的描述符之间的所有描述符进入同一部件。

对于进入当前集合通信卸载部件的当前描述符序列,通过对描述符进行依次解析,提取出其中的 JobID、VP、CPCount 等信息,送入报文维序读取控制部件,生成用来访问网络数据缓冲区的地址。通过这种方式,只有能够与当前描述符匹配的网络报文才能够从缓冲区中读出,从而确保了网络报文按照期望的顺序被执行,即使网络报文到达当前节点的顺序是不确定的。网络报文缓冲区的组织方式以及读写地址的生成方式将在下一小节阐述。每个描述符所需要的网络报文的数量可能为 1 个或多个,由描述符中的 CPCount 决定。

当指定数量的网络报文被读出后,触发当前

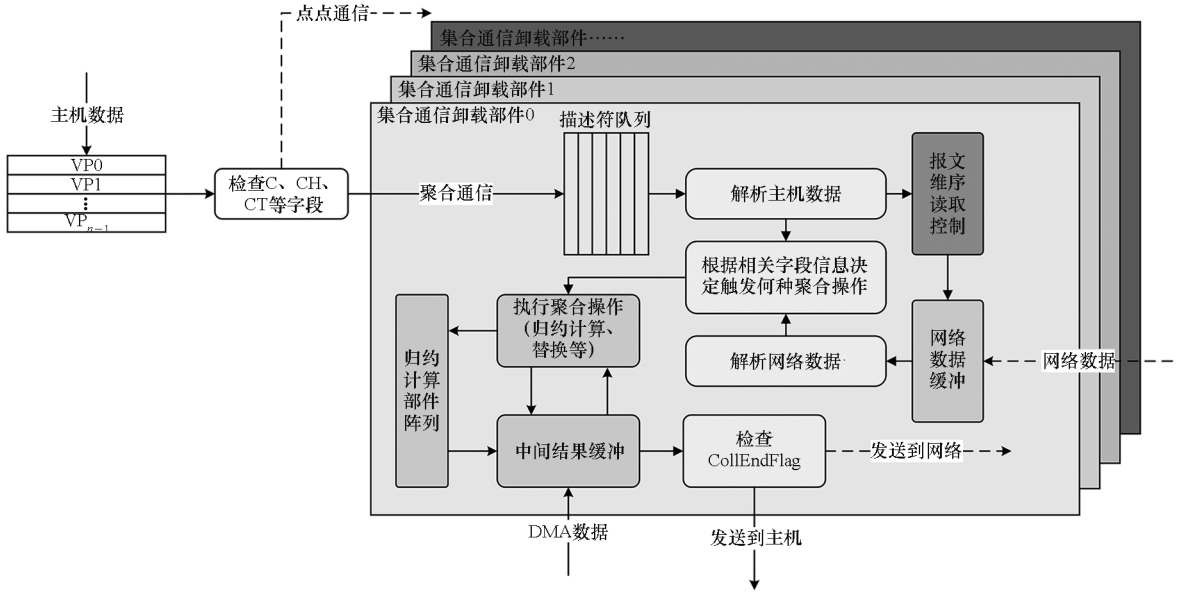


图 3　面向多任务并发的维序触发机制示意图

Fig. 3　Schematic diagram of order-preserving triggering mechanism for concurrent multitasking

描述符的执行。通过描述符和网络报文中的字段信息决定具体执行何种聚合操作,如替换数据、替换 VP、替换地址、进行归约计算等。

网络报文中的 CollEndFlag 字段决定执行完当前操作后,是否还需要继续通过网络发送到其他节点执行后续操作。CollEndFlag 为 1 表明当前集合操作已执行完成,将执行结果和完成事件标志写回当前节点。

所提出的硬件卸载结构通过按需配置描述符中的 CPCount、ReduceFlag 等字段,可以支持不同算法的触发条件。例如,递归减半算法通过递增 SeqID 实现多阶段归约,而二项树算法通过单次触发完成多节点聚合。实际测试表明,本设计能自适应多种算法,仅需软件层调整描述符序列即可切换算法逻辑。

1.3　基于 Hash 表和脉冲信用流控的网络数据动态缓存方法

集合通信数据的存储问题是面向多任务并发的维序触发集合通信卸载需要解决的关键点。原因如下:

1)多任务并发时,不同集合通信任务之间的存储资源划分问题。如前所述,同一卸载部件也可以支持多个任务的分时并发,而每个任务的数据缓存需求是动态变化的,因此普通的静态分配策略并不适用。另外,集合通信网络报文基本不存在时间重用性,因此 Cache 结构也不适用。

2)同一集合通信任务的不同子节点网络数据的存储资源划分问题。处于同一集合通信任务中的节点通常需要与多个节点进行数据通信,当关联的节点数量较多时,来自这些节点的网络数据在当前节点存储空间中的缓存压力也会相应增大。关联节点的数量以及每个节点所需要缓存的数据量都是不固定的。

3)维序触发机制导致来自其他节点的网络数据在缓冲区中的保存时间不确定。由于不同节点的数据到达当前节点的时刻是不确定的,而维序触发机制需要这些数据按照特定的顺序被读取,因此数据从写入到读出的时间也是不确定的。

为解决上述问题,采用基于 Hash 表的快速访问动态缓存结构,通过报文中的 JobID + NodeID + VP + SeqID 等信息构建 Hash 表的查询键值,并使用开放地址法解决 Hash 冲突。图 4 给出了基于 Hash 表的数据缓存结构示意图。当从网络端口接收到报文数据后,从报文中解析出 JobID + NodeID + VP + SeqID 信息,通过 Hash 表控制逻辑产生用于保存该报文的存储器地址,之后将该报文写入存储器,同时更新 BitMap,该 BitMap 用于标识对应当前 JobID + NodeID + VP + SeqID 的报文数据是否有效。当需要从存储器中读取特定内容时,由报文维序读取控制逻辑根据当前描述符生成满足当前集合通信操作语义需求的读取指令,指令中包含 JobID + NodeID + VP + SeqID 信息,如果对应该信息的 BitMap 中的位为有效,则通过该信息生成访问存储器的 Hash 地址,从存储器中读出对应的报文数据;如果 BitMap 中的位为无效,则挂起当前访问请求,转而执行其他不存在语义依赖关系的访存请求。

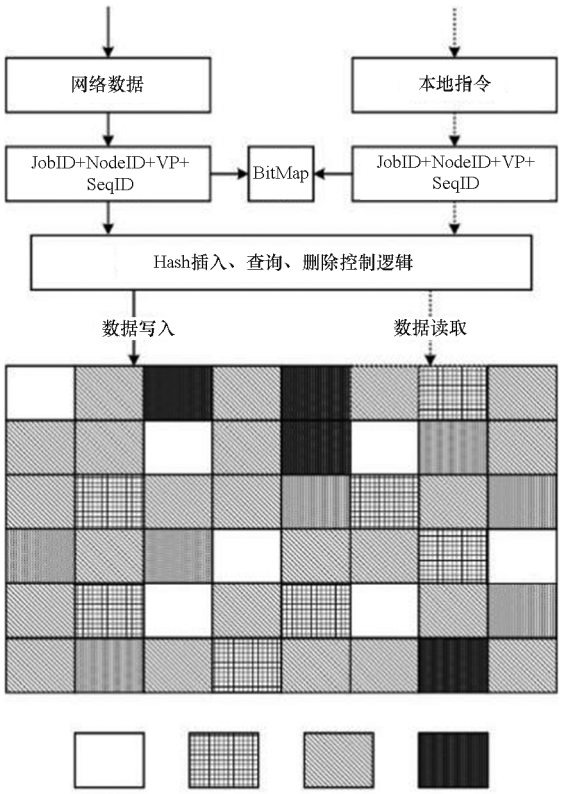


图4 基于Hash表的数据缓存结构

Fig. 4 Data buffering structure based on Hash table

为了支持大消息的归约操作,本文提出一种流量调控方法,该方法通过约束网络中未完成归约数据的数量,对节点间通信链路实施主动式背压控制,旨在从系统层面规避由拥塞导致的跨任务流量干扰问题。例如,对于指定节点规模内的总网络归约表容量 C,可以确保同时归约的数据元素不超过 C 个。对于归约大小为 m 个元素且 $m>C$ 的情况,将 m 分成多次归约脉冲 $\{P_i \mid i\in \mathbf{Z}, 0\leqslant i<\lceil m/C\rceil\}$。流量脉冲之间需要同步以确保一个脉冲完成后才开始下一个脉冲的数据注入。为了隐藏这种同步延迟,将脉冲的大小设置为 $\lceil C/k\rceil, k\in\{1,\cdots,C\}$。允许 k 个脉冲同时进行,以粗粒度流水线方式隐藏同步延迟。在执行大消息归约时,需要通过 DMA 引擎向内存发出数据请求,另外需要从网络接收归约数据。通过为每个脉冲发出单独的 DMA 请求来实现数据的分块。本文提出了一种轻量级机制,通过计数器和保留信用的能力来增强 DMA 引擎。DMA 引擎从处理器接收 DMA 请求,并尝试在脉冲控制器中分配一个计数器。如果计数器分配成功,脉冲控制器返回一个脉冲的信用。只要请求控制器有信用,它就会继续发出内存请求,无论是到本地内存系统还是网络。响应在脉冲控制器中计数,只有当计数达到指定的脉冲大小时,信用才返回给请求生成器。如果无法分配计数器,则在一段时间后重试,并最终向主机处理器发送错误。

2 实验与分析

2.1 实验环境

在本研究小组现有的32节点FPGA测试环境中,将所提出的硬件卸载结构在FPGA中进行实现并加载到该测试环境中,对本工作进行实验与分析。节点服务器通过PCIE线缆连接到FPGA,每个FPGA实现了“天河”NIC的功能。FPGA之间或FPGA与交换机之间通过四通道小型可插拔连接器(quad small form-factor pluggable, QSFP)光纤相互连接。每个节点内存容量为32 GB,FPGA使用Virtex®系列FPGA。在软件层面,基于现有的“天河”软件协议栈框架,所提出的硬件卸载方法的软件层通信接口被集成到该框架中。“天河”超级计算系统的通信库MPICH_GLEX在用户级运行,通过调用用户级函数Libglex和内核级设备驱动程序接口gdev,实现软件与网卡端硬件之间的通信以及相关资源的使用。

为了准确评估集合通信的性能,必须考虑整个集合通信过程对并行应用程序的影响,并基于此选择合适的评估指标。集合通信涉及一组节点协同工作,单个节点执行集合通信所花费的时间并不能反映集合通信的整体性能。Nupairoj和Ni[18]提出使用集合通信操作的执行时间作为评估集合通信性能的指标,近年来相关研究工作所采用的基准级测试和应用级测试均采用类似方式进行集合通信操作执行时间的统计。集合通信操作的执行时间定义如下:所有进程在时间 t_0 调用集合通信操作,从 t_0 到所有节点完成操作的时间被定义为集合通信操作的执行时间 t_c。在每种作业规模下,测试多次归约操作的迭代执行时间,然后计算平均时间。这种性能评价方式可以客观反映大规模分布式系统中的集合通信操作对全系统的性能提升效果。在许多基于MPI的并行科学计算应用中,集合通信占据大部分的通信时间消耗,有的甚至高达70%。一些科学应用程序和基准测试在全归约操作上消耗的时间在总的通信时间中的占比超过50%。

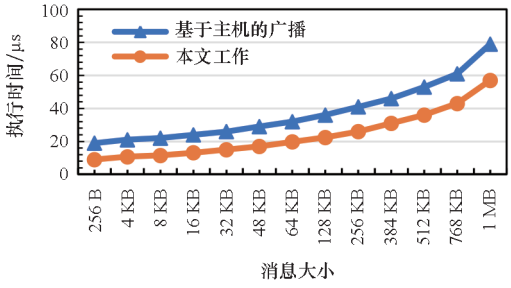
2.2 性能测试

基于上述实验环境,对所述工作进行了性能测试。在每个FPGA上基于原有的“天河”NIC实现了8套集合通信卸载部件,所有集合通信任务

在所有集合通信卸载部件上根据部件忙闲程度与缓冲区占用情况进行公平调度。每套部件上每个集合通信任务最多支持 16 路其他节点发送的网络报文。另外,在本实验中最多测试 32 路集合通信任务并发执行。每套集合通信卸载部件使用 512 KB RAM 对网络报文进行缓存。

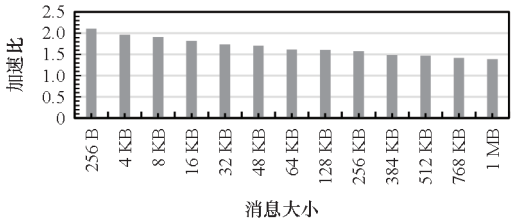
分别测试了 1、8、32 路集合通信任务并发时,不同消息大小(256 B、4 KB、8 KB、16 KB、32 KB、48 KB、64 KB、128 KB、256 KB、384 KB、512 KB、768 KB 和 1 MB)的 RDMA 广播、64 位双精度浮点加法归约和 64 位双精度浮点加法全归约操作的执行时间,每路集合通信任务在每个节点上启动 1 个进程。其中广播操作采用双树算法,归约操作采用向量减半 + 距离加倍 + 二项树算法,全归约操作采用 Rabenseifner 算法。选取上述算法作为实验算法的原因为这些算法是当前主流的集合通信算法中性能较优的算法(以全归约为例,Rabenseifner 算法与另一种主流算法——Ring 算法相比,具有执行步骤更少的优势)。另外,如前所述,所提出的硬件卸载机制能够自适应多种算法,仅需软件层调整描述符序列即可切换算法逻辑。因此,选取这些算法进行性能对比测试具有更好的代表性。

图 5 给出了本工作和基于主机的 RDMA 广播操作在不同任务并发度和不同消息大小情况下的性能表现,分别给出了 1、8、32 路广播任务的不同消息大小的执行时间,以及本工作与基于主机执行广播操作的加速比。



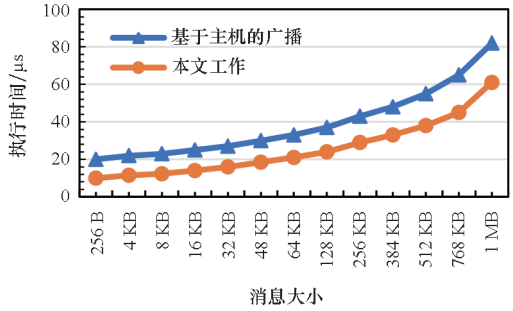
(a) 1 路广播操作执行时间对比

(a) Comparison of execution time for 1 way broadcast operation



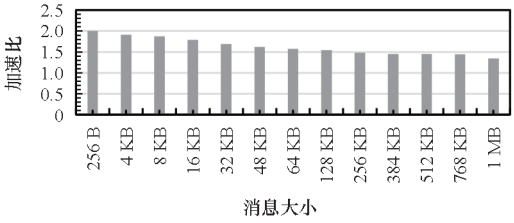
(b) 1 路广播操作加速比

(b) Speedup ratio of 1 way broadcast operation



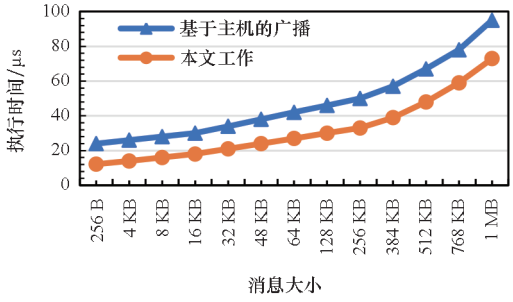
(c) 8 路广播操作执行时间对比

(c) Comparison of execution time for 8 way broadcast operation



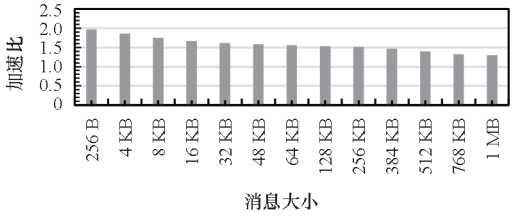
(d) 8 路广播操作加速比

(d) Speedup ratio of 8 way broadcast operation



(e) 32 路广播操作执行时间对比

(e) Comparison of execution time for 32 way broadcast operation



(f) 32 路广播操作加速比

(f) Speedup ratio of 32 way broadcast operation

图 5　本工作和基于主机的广播操作性能对比

Fig. 5　Performance comparison of broadcast operation between this work and host-based approaches

首先分析本工作与基于主机的广播方式相比的软件和硬件处理开销。通常,点对点通信的延迟可以分为 CPU、I/O(例如 PCIE)和网络三个部分,分别表示为 L_{CPU}、$L_{I/O}$ 和 $L_{network}$。对于基于主机的广播,当中间节点 NIC 从父节点接收到 RDMA 广播数据时,数据将首先通过 PCIE 存储在内存中。CPU 需要轮询从父节点数据传输的完成,然

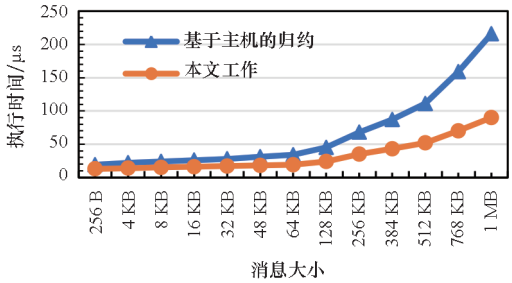
后使用它构建描述符以进行广播操作。之后，由 CPU 预设的描述符将被 NIC 获取并执行，以将广播数据发送到所有子节点。因此，父节点到子节点的 RDMA 广播操作延迟为 MP 控制描述符的构建时间 L_{CPU}、传播时间 $2\times L_{\mathrm{I/O}}+L_{\mathrm{network}}$、RDMA 描述符的构建时间 L_{CPU}、RDMA 数据传播时间 $2\times L_{\mathrm{I/O}}+L_{\mathrm{network}}+T_{\mathrm{MsgSize}}$（其中，$T_{\mathrm{MsgSize}}$ 为流水化传输数据时与消息尺寸线性相关的传输启动时间）、CPU 轮询时间 L_{Poll} 的和。对于本文所提出的硬件卸载广播方式，中间节点 NIC 一旦从其父节点接收完广播数据，就可以直接由 NIC 控制数据传输流程，将数据传输到其子节点，从而消除了 L_{Poll} 开销。因此，硬件卸载广播相对于基于主机的广播方式的相邻节点间执行时间加速比可估算为：

$$1+\frac{L_{\mathrm{Poll}}}{2\times(L_{\mathrm{CPU}}+2\times L_{\mathrm{I/O}}+L_{\mathrm{network}})+T_{\mathrm{MsgSize}}} \tag{1}$$

从图 5 可以看出，本工作相对于基于主机的广播操作，在执行时间上均有一定的性能提升。随着消息尺寸的增加，加速比呈逐步下降的趋势，该结果与上述公式的分析相符。这主要是由于本工作在执行广播操作的硬件卸载时，卸载部件主要通过 MP 描述符序列和控制报文对广播操作数据传输过程进行启动和调度控制，而不会对 RDMA 数据传输过程本身进行加速。尽管如此，这种启动和调度控制的网卡卸载机制仍取得了一定的性能收益：对于 1 路 256 B 的广播操作，可以实现 2.11 的加速比；对于 32 路 1 MB 的广播操作，也可以实现 1.3 的加速比。由于本工作实现了 8 套集合通信引擎，且每套引擎均可以支持多路集合通信操作，因此对于 1、8、32 路并发集合通信任务均实现了一定的性能提升。

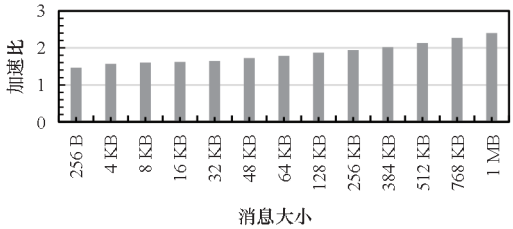
广播操作不需要对集合通信数据进行计算，而归约操作则需要对通信数据执行计算操作。为了测试本工作的集合通信卸载部件对于归约操作的加速性能，对 1、8、32 路并发的基于向量减半 + 距离加倍 + 二项树算法的归约操作进行了性能测试，与广播操作选择同样的消息尺寸。图 6 给出了本工作和基于主机的归约操作在不同任务并发度和不同消息大小情况下的性能表现，分别给出了 1、8、32 路归约任务的不同消息大小的执行时间，以及本工作与基于主机执行归约操作的加速比。

下面分析本工作与基于主机的归约方式相比的软件和硬件处理开销。对于基于主机的归约，当父节点 NIC 从子节点接收到待归约的数据时，



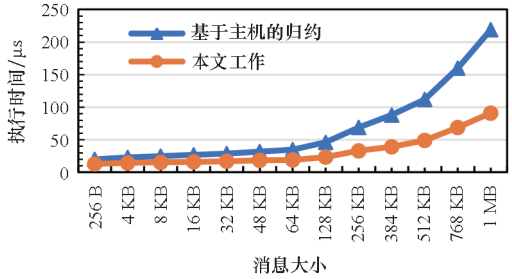
（a）1 路归约操作执行时间对比

（a）Comparison of execution time for 1 way reduce operation



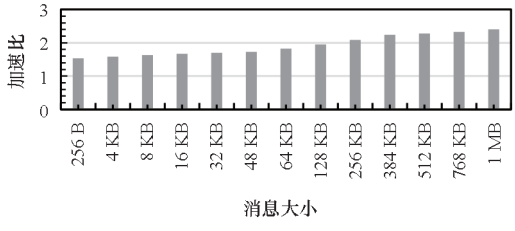
（b）1 路归约操作加速比

（b）Speedup ratio of 1 way reduce operation



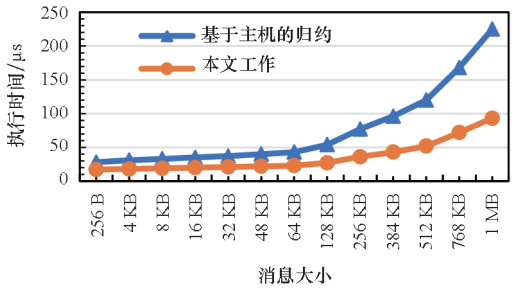
（c）8 路归约操作执行时间对比

（c）Comparison of execution time for 8 way reduce operation



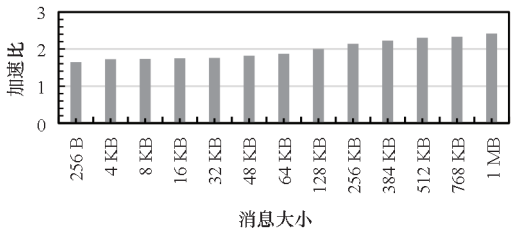
（d）8 路归约操作加速比

（d）Speedup ratio of 8 way reduce operation



（e）32 路归约操作执行时间对比

（e）Comparison of execution time for 32 way reduce operation



(f) 32 路归约操作加速比

(f) Speedup ratio of 32 way reduce operation

图 6　本工作和基于主机的归约操作性能对比

Fig. 6　Performance comparison of reduce operation between this work and host-based approaches

数据将首先通过 PCIE 传输到 CPU,在 CPU 执行计算。之后,计算完成的中间结果再次通过 PCIE 传输到 NIC,并通过网络发送到下一级节点。因此,基于主机执行归约操作的相邻节点间延迟由描述符的构建时间 L_{CPU}、传播时间 $2\times L_{\mathrm{I/O}}+L_{\mathrm{network}}+T_{\mathrm{soft\_startup}}$(其中,$T_{\mathrm{soft_startup}}$为流水化传输数据时与消息尺寸线性相关的从内存到网络的传输启动时间)、CPU 归约计算时间 T_{Compute} 组成。对于所提出的硬件卸载广播方式,父节点 NIC 不再将归约数据传输到 CPU,而是由 NIC 直接执行归约计算并再次发送到下一级节点,因此该方式的相邻节点间延迟为传播时间 $L_{\mathrm{network}}+T_{\mathrm{nic\_startup}}$(其中,$T_{\mathrm{nic_startup}}$为流水化传输数据时与消息尺寸线性相关的从 NIC 到网络端口的传输启动时间)。由于数据不需要再从内存获取,而是直接在 NIC 中处理,即 $T_{\mathrm{nic_startup}}$不包含从内存获取数据的时间,因此 $T_{\mathrm{nic\_startup}}<T_{\mathrm{soft\_startup}}$,且随着消息尺寸的增大,$T_{\mathrm{soft_startup}}$与 $T_{\mathrm{nic\_startup}}$ 的比值逐步增大。另外,NIC 上的流水化归约计算时间仅为几个硬件时钟周期,可忽略不计。因此,硬件卸载归约相对于基于主机的归约方式的相邻节点间执行时间加速比可估算为:

$$\frac{L_{\mathrm{CPU}}+2\times L_{\mathrm{I/O}}+L_{\mathrm{network}}+T_{\mathrm{soft\_startup}}+T_{\mathrm{Compute}}}{L_{\mathrm{network}}+T_{\mathrm{nic\_startup}}} \quad (2)$$

从图 6 可以看出,随着消息尺寸的增加,本工作的归约操作卸载方法相对于基于主机的归约加速比逐步增大,这是因为本工作的硬件卸载机制不仅对归约数据处理流程进行了控制,而且参与了归约数据的计算过程,这也与上述公式的分析相符。同时本工作的并发卸载机制和高效缓存结构也使得在 8 路和 32 路并发归约实验中实现了较大的性能加速,3 组实验实现了 1.46 ~ 2.42 的加速比。

全归约操作相对于归约操作,需要将归约计算完成的结果分发到各节点。实验测试了全归约操作的性能对比情况,基于主机的全归约采用 Rabenseifner 算法,本工作的硬件卸载方法通过“天河”软件协议栈结合网卡硬件加速部件实现对等的操作。实验结果表明,全归约操作的性能结果与归约操作呈现类似的加速比趋势,加速比数据略高于相应归约操作的加速比,这是由于相对于基于主机的方式,本工作的硬件卸载全归约操作不仅对归约计算阶段进行加速,同时也对归约结果分发阶段进行卸载,归约结果在计算完成后直接由网卡转发到网络。

2.3　Hash 缓冲区使用效率测试

为了评估 Hash 缓冲区的使用效率,测试了 1、8、32 路集合通信任务并发时,对不同消息大小的 64 位双精度浮点加法全归约操作进行硬件卸载时,根节点的 8 套集合通信卸载部件的 Hash 缓冲区中所有被使用的存储地址占存储总容量的平均百分比,其中缓冲区保留 10% 用于 Hash 冲突处理。测试结果如表 1 所示。结果显示,当所有子节点的单次操作的数据总和较小时,Hash 缓冲区保持较小的占用率;当数据总和逐渐增大时,缓冲区占用率逐渐增大;当数据总和明显超过缓冲区总容量时,Hash 缓冲区仍能够基本满足网络报文的缓存需求。这主要得益于所提出的集合通信卸载部件的流水化执行方式、Hash 结构的动态缓存方式,以及脉冲信用流控机制既能确保所期望

表 1　单路 Hash 缓冲区平均使用效率测试结果

Tab. 1　Average utilization efficiency test results for single-path Hash buffer

%

消息大小	1 路并发	8 路并发	32 路并发
256 B	0.01	0.09	0.39
4 KB	0.12	1.13	4.31
8 KB	0.49	2.82	10.10
16 KB	1.61	5.79	19.97
32 KB	1.85	6.37	23.52
48 KB	2.27	10.70	36.20
64 KB	2.60	19.10	49.17
128 KB	4.71	29.16	72.36
256 KB	7.16	45.11	81.47
384 KB	10.23	70.15	89.55
512 KB	13.79	85.92	93.69
768 KB	17.36	98.20	101.24
1 MB	23.69	100.92	103.47

的网络数据在到达缓冲区后尽可能早地读出执行,也能够动态利用缓冲区资源以提高缓存利用率,同时在缓冲区资源紧张时通过信用反压方式降低网络报文接收压力。可以看出,所提出的结构与方法对于大消息的集合通信操作具有更为显著的效率优化意义。

2.4 硬件资源使用情况

基于32节点"天河"互联网络FPGA测试环境,对本工作的硬件卸载结构的硬件资源使用情况进行了评估,主要统计了图1中集合通信卸载部件的资源使用情况,具体如表2所示。资源评估结果显示,该卸载架构在实现其高效通信功能的同时,保持了较好的硬件资源效率。由LUT的使用情况可见,组合逻辑(LUT as Logic)和分布式存储器(LUT as Memory)的分布特点符合本设计作为高效数据通路控制逻辑的预期。在块存储器资源方面,适中的Block RAM Tile使用量反映了本架构在数据缓冲和报文缓存管理上的高效性。

表2 本工作在FPGA上实现的资源使用情况

Tab. 2 Resource utilization of this work on FPGA

资源类型	数量
CLB LUTs	119 112
CLB Registers	112 392
CLB	22 544
LUT as Logic	117 824
LUT as Memory	1 288
LUT Flip Flop Pairs	58 496
Block RAM Tile	168

3 结论

本文研究了面向集合通信NIC硬件卸载的维序触发机制和数据缓存方法。所提出的面向多任务并发的维序触发机制,在硬件卸载结构中基于报文内置ID信息将到达当前节点的网络报文按照期望的顺序进行触发,确保了多个集合通信树的多个节点以及多个数据通道之间的多组数据块能够以确定的顺序被执行,既满足了期望的集合通信语义,又确保了浮点计算操作结果的可复现性。所提出的基于Hash表和脉冲信用流控的网络数据动态缓存方法,缓解了有限的硬件缓存资源和多任务并发的大量网络数据缓存需求之间的矛盾问题,与常规的任务间固定分区缓存的方式相比,缓存效率得到显著提高。在面向"天河"网络的FPGA验证平台中对所提出的方法进行了实验测试,实验结果表明,与基于软件方式的集合通信操作相比,所提方法可以支持广播、同步、归约、全归约等多种典型操作的多种算法的硬件卸载,且性能提升效果显著,同时,硬件实现代价较低,尤其是在缓存资源方面具有较高的利用率。

参考文献(References)

[1] HAGHI P, MARSHALL R, CHEN P H, et al. A survey of potential MPI complex collectives: large-scale mining and analysis of HPC applications[EB/OL]. (2023-05-31)[2025-01-12]. https://arxiv.org/abs/2305.19946.

[2] LIU P, PENG J T, LIU J, et al. TH-Allreduce: optimizing small data Allreduce operation on Tianhe system[C]// Proceedings of the IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), 2024.

[3] LIU P, PENG J T, LIU J, et al. GLEX_Allreduce: optimization for medium and small message of Allreduce on Tianhe system[C]//Proceedings of the IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS), 2024.

[4] TRAN T, RAMESH B, MICHALOWICZ B, et al. Accelerating communication with multi-HCA aware collectives in MPI[J]. Concurrency and Computation: Practice and Experience, 2024, 36(1): e7879.

[5] SARKAUSKAS N R. Large-message nonblocking Allgather and Broadcast offload via BlueField-2 DPU[D]. Ohio: The Ohio State University, 2022.

[6] GRAHAM R, BOSILCA G, QIN Y, et al. Optimizing application performance with BlueField: accelerating large-message blocking and nonblocking collective operations[C]// ISC High Performance 2024 Research Paper Proceedings (39th International Conference), 2024.

[7] RAMESH B, KUNCHAM G K R, SURESH K K, et al. Designing in-network computing aware reduction collectives in MPI[C]//Proceedings of the IEEE Symposium on High-Performance Interconnects (HOTI), 2023.

[8] WANG R Q, DONG D Z, LEI F, et al. Roar: a router microarchitecture for in-network allreduce[C]//Proceedings of the 37th ACM International Conference on Supercomputing, 2023.

[9] LIANG C S, DAI Y, XIA J, et al. The self-adaptive and topology-aware MPI_Bcast leveraging collective offload on Tianhe express interconnect[C]//Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2024.

[10] HAGHI P, TAN C, GUO A Q, et al. SmartFuse: reconfigurable smart switches to accelerate fused collectives in HPC applications[C]//Proceedings of the 38th ACM International Conference on Supercomputing, 2024.

[11] RASHIDI S, WON W, SRINIVASAN S, et al. Themis: a network bandwidth-aware collective scheduling policy for distributed training of DL models[C]//Proceedings of the 49th Annual International Symposium on Computer Architecture, 2022.

[12] HE Z H, KOROLIJA D, ZHU Y, et al. ACCL + : an FPGA-based collective engine for distributed applications[EB/OL]. (2023 - 12 - 18)[2025 - 01 - 12]. https://arxiv.org/abs/2312.11742.

[13] YANG X J, LIAO X K, LU K, et al. The TianHe-1A supercomputer: its hardware and software[J]. Journal of Computer Science and Technology, 2011, 26(3): 344 - 351.

[14] XIE M, LU Y T, WANG K F, et al. Tianhe-1A interconnect and message-passing services[J]. IEEE Micro, 2012, 32(1): 8 - 20.

[15] LIAO X K, PANG Z B, WANG K F, et al. High performance interconnect network for Tianhe system[J]. Journal of Computer Science and Technology, 2015, 30(2): 259 - 272.

[16] 王浩, 张伟, 谢旻, 等. 基于天河互连 MPI 聚合通信归约操作卸载优化[J]. 计算机工程与科学, 2020, 42(11): 1981 - 1987.
WANG H, ZHANG W, XIE M, et al. Reduction operation offloading optimization based on Tianhe interconnect MPI collective[J]. Computer Engineering & Science, 2020, 42(11): 1981 - 1987. (in Chinese)

[17] 常俊胜, 熊泽宇, 徐金波. 高速互连网络中基于网卡的归约计算硬件卸载机制[J]. 国防科技大学学报, 2022, 44(5): 171 - 179.
CHANG J S, XIONG Z Y, XU J B. NIC-based offloading mechanism supporting reduction operation on high-speed interconnection system[J]. Journal of National University of Defense Technology, 2022, 44(5): 171 - 179. (in Chinese)

[18] NUPAIROJ N, NI L M. Performance metrics and measurement techniques of collective communication services[C]// Proceedings of International Workshop on Communication, Architecture, and Applications for Network-Based Parallel Computing, 1997.