doi:10.11887/j.cn.202205018 http://journal.nudt.edu.cn

高速互连网络中基于网卡的归约计算硬件卸载机制*

常俊胜,熊泽宇,徐金波
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:聚合通信广泛应用于高性能计算的研究和工程领域。在大规模的科学和工程计算中,聚合通信开销占据很大比例,有时甚至可达到全部消息传递开销的80%,是高性能计算系统的性能瓶颈。因此提出了一种基于网卡的归约计算硬件卸载机制,通过在网卡上嵌入归约操作逻辑部件,实现了数据在传输过程中的计算,减轻了CPU的负担,降低了通信延迟。通过FPGA平台实现了16节点的归约操作实验,并基于xNetSimPlus模拟器模拟了不同节点规模的归约操作。实验证明,卸载机制能有效减少聚合通信中归约操作的时间,所提支持归约计算的网卡卸载机制最高可以加速归约操作2.71倍。

关键词:聚合通信;归约操作;硬件卸载
中图分类号:TN95 **文献标志码**:A **文章编号**:1001-2486(2022)05-171-09

NIC-based offloading mechanism supporting reduction operation on high-speed interconnection system

CHANG Junsheng, *XIONG Zeyu*, *XU Jinbo*
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: Collective communication is widely used in the field of high-performance computing research and engineering. In large-scale scientific and engineering computing, collective communication overhead accounts for a large proportion, sometimes even reaching 80% of the total messaging overhead. It is the performance bottleneck of the high-performance computing system. A NIC-based offloading mechanism supporting reduction operation was proposed. By embedding reduction operation logic components on NIC, the calculation of data during transmission was implemented, and the burden on the CPU and the communication delay were reduced. A 16-node protocol operation experiment was realized through the FPGA (field programmable gate array) platform, and the protocol operation in different node size was simulated based on the xNetSimPlus simulator. Experiments show that the method can effectively reduce the time of protocol operation in collective communication, and the proposed NIC offloading mechanism that supports reduction operation hardware offload can accelerate all-reduce operations by up to 2.71 times.

Keywords: collective communication; all-reduce operation; hardware offload

计算机技术的快速发展,越发突显高性能计算在各学科领域研究中的重要性,高性能计算机技术是计算机技术乃至整个信息技术发展的制高点。目前,高性能计算在情报处理、原子核建模、天气预报、天体物理、遥感图像处理和生物系统等科学和工程计算中都发挥着举足轻重的作用。但随着大数据和人工智能等在各领域的应用与发展,迫切需要高性能计算机提高性能,提供更强大的计算能力。为满足需求,高性能计算机通过增加处理器数量和高速并行工作模式相配合,峰值计算速度和持续速度得到不断提升。2022年6月发布的TOP500高性能计算机排行榜中,排名第一的"前沿"(Frontier)高性能计算机系统的峰值性能已经达到E级,即计算性能可达到每秒百万万亿次浮点计算[1]。

1 目的

高性能计算(high performance computing, HPC)的快速发展,使得系统规模越来越大,在提高性能的同时,也带来一些问题。大规模的并行应用程序中,节点间的通信受网络链路影响,存在一定的通信延迟,而且每次通信,通信消息都要频繁进出内存,这将带来一定的启动和接收延迟,且这些延迟会随着系统规模的增大而成倍增加[2]。从应用的角度来看,在科学和工程计算中,高性能计算机的性能主要体现在应用程序的执行时间上。高性能计算中,许多并行程序的执行都是计算和通信交替进行,在计算阶段,各个进程在处理

* 收稿日期:2020-09-01
基金项目:国家自然科学基金资助项目(61201336,41301490)
作者简介:常俊胜(1979—),男,河南唐河人,副研究员,博士,硕士生导师,E-mail:cjs7908@163.com

节点上独立运行,通信阶段进程则在互连网络中执行同步和数据交换;在通信阶段,处理节点实际上是处于等待状态,不进行任何计算操作,这极大地降低了高性能计算机的效率,并行程序难以获得理想的运算速度和并行效率[3]。

消息传递并行编程模型[4](message passing interface,MPI),是目前解决并行应用主要的编程模型,在高性能计算机系统中被广泛采用。聚合通信是MPI中的重要组成部分,在并行系统中,聚合通信通常是应用程序中一组进程间的互相协作,通过进程间的消息传递来实现任务控制、数据交换和数据计算。有数据统计表明,在大规模的科学和工程计算中,聚合通信开销占据很大比例,有时甚至可达到全部消息传递开销的80%[5]。与此同时,聚合通信可以高效管理消息缓冲区,避免缓存占用大及其管理开销过大的问题。因此,聚合通信的使用及其优化对提升高性能计算机性能,提高并行应用程序的执行效率具有重要意义。

在高性能计算机中,通信系统的异构性会导致系统内不同层级的通信性能存在一定程度的差异,这给聚合通信的优化带来了更加艰难的挑战[6]。目前,针对具体应用程序中的同步、广播和归约等聚合通信操作,有很多研究在算法领域或硬件上提供了较为有效的优化方案。但受制于基础环境和特定聚合通信操作的内在因素影响,不存在某一种优化方案在各种场景都能取得良好的优化效果。因此,在大规模的聚合通信场景下,面对高性能计算机内存在不同层级的通信性能变化,找到合理且有针对性的优化方案变得尤为重要。

本文主要针对聚合通信中的归约操作进行优化,提出了一种软硬件结合的支持归约计算网卡卸载机制,通过在网卡(network interface card,NIC)上嵌入归约操作逻辑部件,包括触发逻辑部件和浮点计算单元。方法中将原来由CPU执行的计算卸载到NIC上执行,实现了数据在传输过程中的计算,减轻了CPU的负担,降低了通信延迟。

2 现状

聚合通信的实现方式可以根据硬件的支持程度划分为三类:基于软件方式、基于硬件方式和软硬件结合的方式。最理想的情况是纯硬件方式实现,但硬件实现的复杂度高,经济投入巨大,因此本文不做详细讨论。

2.1 软件实现

软件聚合通信是在点对点通信的基础上结合算法实现的,是当前应用较为广泛的一种方式。软件算法实现具有良好的普适性,可以在不改变原语的情况下,在各平台实现。

最早的软件聚合通信算法于1988年被提出,是基于传播算法来实现的同步操作[7],至今仍被大量使用。此后,Watts和Geijn[8]提出一种广播算法,该算法将进程要发送的消息分成若干小消息,然后将这些小消息发散给不同的进程,接着各进程使用组收集操作使得每个进程都得到所有的小消息,以此完成某一进程向其他进程广播消息的目的。由于此种算法会频繁使用各进程的缓冲区,在互连网络中容易形成拥塞。Rabenseifner[9]在广播算法的基础上提出了对组归约操作的新算法,算法中将组归约操作分成两步,步骤一中做散发操作,步骤二中进行一次组收集操作。该算法的目的是将集中到根进程的计算分散到其他子进程中完成,从而改变根进程是组归约操作中的瓶颈问题。Thaur等[10]对MPICH库的实现进行优化,大大提高了同步、广播、归约和组归约等的操作性能,为软件聚合通信操作的使用与优化提供了一种重要方法。

通过软件实现聚合通信方法较简单,但由于是基于点对点通信实现,所以通信消息会频繁进出系统内存,使通信过程中存在延迟,并且当进行归约操作时,计算与通信操作不能同时进行,当消息在网络中传递时,处理器将处于空闲状态,使得效率低下。

2.2 软硬件联合实现

通过软硬件结合实现聚合通信操作的方式包含了软件与硬件分别实现的优点,使得高性能计算机在执行聚合通信操作时可以兼顾编程简便、延迟降低以及计算与通信重叠,大大提高聚合通信的性能。

最早的软硬件结合方法由Buntinas等[11]提出,在Myrinet上实现了基于网卡卸载的同步、广播和归约等聚合通信操作,与软件聚合通信相比,性能有很大的提升。受此启发,Moody等[12]在QsNet上提出基于网卡卸载的归约算法,该方法在网卡端集成了浮点计算单元,将本该由CPU进行的计算卸载到网卡端执行,从而提高计算与通信的重叠率以及归约性能。实验证明,卸载到网卡端执行的归约操作,整数归约性能提高了121%,浮点数归约性能提高了40%。之后,IBM

公司也在其 Blue Gene 系列系统上提出软硬件结合的聚合通信方法,该方法通过直接存储器访问(direct memory access, DMA)和浮点计算单元等部件实现卸载操作,并在通信不满足专用网络拓扑时设置有专用的聚合通信网络。经验证,Blue Gene 系列系统上聚合通信操作性能相比 MPI 软件实现得到很大提升[13-14]。CM-5 系统同样采用了专用的聚合通信网络,专用网络为两套二叉树,其中一套实现归约操作,一套实现广播操作。Portals 网络上,通过在网卡端集成 Portals 单元和 DMA 部件来实现聚合通信操作的卸载执行[15]。Portals 单元中包含与处理器计算速度相当的浮点计算单元,大大降低系统在执行归约操作时的各种时间延迟。实验证明,在多节点树形算法的情况下,利用此种方法实现的归约操作性能比软件聚合通信提高了 1.8 倍左右。

此外,一些公司也通过交换机来实现聚合通信操作[16],如 Voltaire 公司的 FCA(fabric channel accelerator)和 Nvidia 公司的交换机中实现 SHARP 协议。Voltaire 公司产品通过在 MPI 库中集成 FCA 技术[17],使得聚合通信操作在执行时能够获取物理拓扑网络的信息,并根据拓扑网络构建聚合通信树。在聚合通信树中,一旦父节点得知计算结果,利用 FCA 就可以通过交换机进行广播操作,将结果通知其他节点。FCA 技术还可以将聚合通信消息与交换机中的其他消息隔离,从而消除竞争。SHARP 是 Mellanox 公司推出的将聚合通信操作卸载到网络中执行的技术,该技术实现了消息在传输过程中就进行计算,使得交换机成为协处理器,降低处理器的负担,加速归约计算速度,提高系统性能[18-19]。SHARP 同时适用于深度学习、大数据计算等领域,并有良好的应用性能。

通过软硬件结合实现聚合通信操作的方法,能够大大提高系统在进行聚合通信操作时的性能,降低通信延迟,提高计算与通信的重叠率[20]。但通过网卡或交换机实现硬件卸载,势必增加硬件设计的复杂度,从而导致经济投入增大。

3　方法

高性能计算领域的不断发展为软硬件开发提供了更多便利,为提出更有效的聚合通信优化方案提供了基础。针对“天河”超级计算机系统的需求,本文提出了一种软硬件结合的支持归约计算网卡卸载机制,将原来由 CPU 执行的计算卸载到 NIC 上执行,实现了数据在传输过程中的计算,降低了 CPU 的负担以及通信延迟。

“天河”高速互连网络在聚合通信优化方面的设计思想类似于 Portals 网络,都是在网卡端加入特殊的硬件来卸载处理机端的事务,不同的是“天河高速互连网络”的网卡端加入的硬件更加简单,这样,在无须大幅增加硬件复杂性和经济成本的情况下仍能使聚合通信的操作性能有了很大的提升。相比 Portals 网络,本文提出的方法经济成本低,使用硬件程序产生的软件额外开销小。

3.1　聚合通信网卡卸载机制硬件设计

硬件实现结构示意图如图 1 所示,主要包含请求发送模块、请求接收与处理模块、请求消息接收队列和算术逻辑单元(arithmetic logic unit, ALU)。

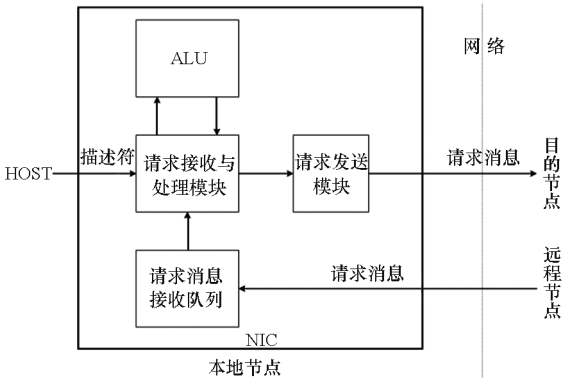


图 1　聚合通信硬件实现结构示意图

Fig. 1　Schematic diagram of the implementation of collective communication hardware

1)请求发送模块:请求发送模块用于解析本节点提交的描述符序列,然后产生相应的请求消息发送到目的节点。针对聚合通信描述符,模块内部实现了触发执行的机制。描述符格式如图 2 所示,描述符中设置有 CP 和 CPCnt 标志位;CP 位域表示该序列中各描述符形成的报文到达目的节点时,将使对应虚端口的全局计数器 CP_Counter 加 1;CPCnt 表示触发描述符执行需接收的 CP 报文数量的阈值,该值与 CP_counter 进行比较,当满足 CP_counter ≥ CPCnt 时,触发描述符序列执行。同时,根据描述符中的 SwpFlag 位域决定是否在本模块中进行数据交换。

2)请求消息接收队列:请求消息接收队列用于接收并存储来自其他节点的请求消息。消息是否写入接收队列,受主机提交描述符中的 SwpMp 标志控制。NIC 硬件中设置有一定深度的缓冲区,采用动态分配多队列方式管理;请求消息写入缓冲区,即认为消息已到达目的节点。在执行归约操作时,缓冲区可缓存最大归约分支度个归约请求消息。

0	63 60	59	58	57	56	55	54	53	52	51 42	41 36	35 33	32	31 26	25 20	19 0
	Head	Type	S/M	Lint	Rint	Inst	Err	Fen	SwpMP	CollRegion	CpCnt	SwpFlag	NoCnct	SrcVP	DestVP	Dest ID/ SrcID
1	ReduceData Num + ErrVec + BranchDegree + Rtype + Rbit															
2	ReduceData0															
3	Location0															
4	ReduceData1															
5	Location1															
6	ReduceData2															
7	Location2															
8	ReduceData3															
9	Location3															
A	ReduceData4															
B	Location4															
C	ReduceData5															
D	Location5															
E	ReduceData6															
F	Location6															

图2 聚合通信描述符格式

Fig. 2 Collective communication descriptor format

3)请求接收与处理模块:请求接收与处理模块基于本地描述符和接收到的请求消息进行计算操作,生成新的请求消息,并由请求发送模块通过网络发送到目的节点。请求接收与处理模块检查请求消息接收队列是否收到其他节点发来的请求消息,如果有,则对消息进行解析,并根据解析结果和描述符中相关数据发送到 ALU 部件中进行计算操作。ALU 部件完成相应的计算操作后,将计算结果返回给请求接收与处理模块。对于归约操作,新的请求消息通常由描述符和接收到的请求消息中的数据通过一定的计算操作来产生。

4)ALU:由于 NIC 本身不具备进行浮点运算的能力,因此,本方法在 NIC 中嵌入 ALU 模块。该模块可实现对 32 位和 64 位数据进行归约计算。计算类型包括浮点加法、浮点求最大值或最小值且附加位置信息,有符号和无符号的整数加法、整数求最大值或最小值且附加位置信息,逻辑与、逻辑或、逻辑异或和位与、位或、位异或。

3.2 软硬件结合的归约操作实现

执行一次归约操作,首先由并行程序的一次 MPI 归约调用开始,进入 MPI 调用后,软件构造归约通信域,将参与线程与物理节点的映射关系发送给所有参与归约的节点。

基于软件线程号和通信域,构建归约操作相关的描述符,包括用于计算的归约描述符和用于通知的广播描述符。由软件按指定算法计算出归约操作中的叶节点、中间节点和根节点。由于并行应用程序使用线程标号进行归约操作,因此描述符以线程号作为源节点号和目的节点号。

基于各节点信息,完成线程号与节点号的映射,生成最终的归约描述符和广播描述符。生成的归约描述符中,需要包含源节点号、目的节点号、参与归约的数据个数、错误向量、归约分支度、归约类型、归约数据位宽、归约数据和位置信息等归约信息,硬件在解析描述符后,按规定执行归约操作。生成的广播描述符需指定源节点号、目的节点号,描述符本身无须准备数据,而是提交后在 NIC 中等待触发执行,并用归约计算结果替换对应的数据位域,最终完成广播操作。

用于归约操作的描述符生成后,软件向硬件提交描述符。所有 NIC 硬件收到归约描述符后,除叶节点对应 NIC 外,其他 NIC 在收到描述符后等待触发执行。叶节点 NIC 收到描述符后直接形成归约报文发往父节点;父节点 NIC 收到子节点发来的归约报文,并不会提交给节点,而是将归约报文存储于 NIC 中的请求消息接收队列中。

然后请求接收与处理模块对请求消息进行解析,并按约定进行处理。首先将子节点发来的归约类报文送入 ALU 中进行计算,并在计算结束后触发父节点提交到 NIC 中的归约描述符,所提描述符与之前的计算结果再次进行归约计算,并在计算完毕后形成归约报文按算法向上发送,具体处理过程如图 3 所示。以此类推,直到根节点。根节点 NIC 中的归约计算与上述父节点中类似,不同的是根节点提交的归约描述符生成的归约报文的目的节点为自身,且分支度为 1,因此该归约报文不会再进行归约计算,而是触发执行根节点提交的广播描述符,将归约计算结果通知给所有参与此次归约操作的节点。

归约描述符与广播描述符都执行完成后,按照描述符中的规定,所有 NIC 硬件向各自对应节点中指定内存地址写入完成值,完成值为计算结果。当软件收到硬件返回的归约结果后,则表示一次完整的归约操作结束。

如图 3 所示,当请求接收与处理模块检测到 buffer 中有报文,则将报文顺序读出,并对读出的报文进行计数,计数由 Rcnt1 开始。同时,将 Rcnt1 报文中的 Branch、Rtype 和 Data 等信息存入中间结果寄存器中;读出 Rcnt2 报文后,先后比较该报文与中间结果寄存器中的 Branch 和 Rtype 位域,如果一致,则将二者中的数据一起送入 ALU 中进行计算;ALU 返回的计算结果直接更新到中间结果寄存器中;此时,比较 Rcnt 和 Branch 的值,如果不等,则继续读出报文进行计算,计算过程与前述一致;如果二者的值相等,则检查请求发送模块接收的第一个描述符,判断描述符是否为归约描述符(Rflag),继而比较描述符与中间结果寄存器中的 Rtype,当描述符与 Rtype 一致时,将描述符与中间结果寄存器中的数据送入 ALU 中进行计算,并将计算结果再次更新到中间结果寄存器;当归约计算结束时,将中间结果寄存器中的数据替换到前述描述符中;如果请求发送模块接收的第一个描述符不是归约描述符,那么直接用中间结果寄存器中的数据替换此描述符中的数据位域。

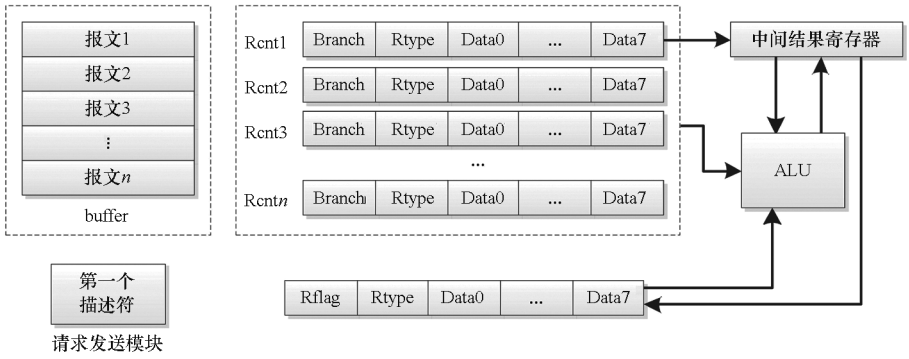


图 3　NIC 中归约操作流程示意图

Fig. 3　Schematic diagram of protocol operation in NIC

具体的操作流程如下:

1)请求接收与处理模块检测 buffer 中的报文,当有报文则按顺序读出执行。

2)对读出的报文进行计数,由 Rcnt1 开始,并将 Rcnt1 报文中的 Branch、Rtype 和 data 等位域存入中间结果寄存器中。

3)检测 Rcnt2 报文与中间结果寄存器中报文(Rcnt1 报文)的 Branch 和 Rtype 位域是否一致,如果一致则将二者报文中的数据送入 ALU 中进行计算。

4)ALU 中计算结束后将计算结果更新到中间结果寄存器。

5)检查 Rcnt 是否等于 Branch,如果不等于,则继续读出报文进行计算;如果相等,则检查本节点提交的第一个描述符是否为归约类型。

6)重复步骤 3)、步骤 4),直到步骤 5)满足条件。

7)当请求发送模块接收的描述符 Rflag 位域有效,检测当前 Rtype 是否与之前一致,如果一致则将中间结果寄存器中的数据和描述符中的数据送入 ALU 中进行计算,并将最终结果更新到中间结果寄存器中。

8)使用中间结果寄存器中的数据替换本地提交的描述符中的数据位域。

综上所述,"天河"高速互连网络基于触发实现归约操作的优化有以下优势:

1)硬件实现简单,没有大幅增加硬件复杂性和经济成本:由于"天河"高速互连网络只在网卡端加入了触发逻辑,而且触发逻辑只需实现简单的功能,从而"天河"高速互连网络的网卡较一般

的网卡并没有增加太多的硬件,因此经济成本较之前的互连网络并没有太大的增加。

2)网卡端进行数据自动复制和自动触发发送,加快消息的传递速率,大大减少了节点端消息的停留时间,使得聚合通信对规模的敏感性降低,使得快速、大规模的归约操作成为可能。

3)网卡端自动处理数据而不需要处理机的参与,减小了系统噪声对归约操作的影响,增加了归约操作的性能和可靠性。

4)在进行聚合通信时网卡端进行消息处理的同时处理机端可以进行其他的计算工作,增加了计算与通信的重叠率。

4 实验

基于本文提出的支持归约计算的网卡卸载机制,在 FPGA 平台搭建了 16 节点规模的测试环境;基于 xNetSimPlus 模拟器[21]模拟实现了 32、64、128、256 节点的测试。在软件层面,将实现集成到“天河”超级计算系统的通信库 MPICH_GLEX 中,软硬件接口的整体结构如图 4 所示,MPICH_GLEX 运行在用户层,调用用户层函数 Libglex 和内核层设备驱动接口 gdev 来进行网卡端软硬件资源的使用。具体实验步骤如下:

1)配置节点规模、拓扑结构等参数。

2)root 节点与 leaf 节点进行同步,root 节点给 leaf 节点分配 leaf id,交换内存地址等信息。

3)root 节点进入循环,提交归约描述符与集合描述符,其中归约描述符设置 coll_counter = num_of_leafs,接受节点发来的 mp 报文后触发,向自身发送 mp 报文;集合描述符 coll_counter = 1,受自身归约 mp 报文触发,得出归约计算结果并校验。

4)leaf 节点进入循环,提交归约描述符,向 root 节点立即发送 mp 报文;等待接收 root 节点发来的集合报文。

5)统计结果并记录。

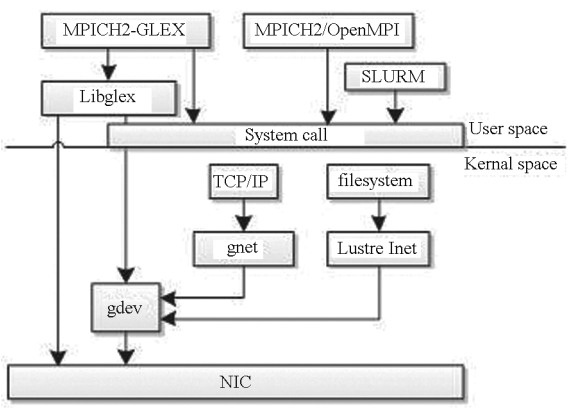


图 4 软硬件接口整体结构

Fig. 4 Overall structure of software and hardware interface

为了评价改进后聚合通信操作的性能,本文将使用 k-ary n-tree 胖树拓扑来测试有无硬件卸载机制对聚合通信性能的影响。胖树拓扑是一种典型的多层次树形拓扑,在高性能计算机及数据中心中广泛使用,节点之间的通路自叶向根逐渐变宽,适应了通信带宽自叶向根逐渐变大的实际要求,实现了网络带宽的平滑扩展,胖树拓朴结构如图 5 所示。模拟时,默认每个节点上只运行一个进程,测试的节点规模选定为 16、32、64、128 和 256。

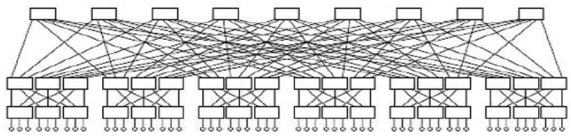


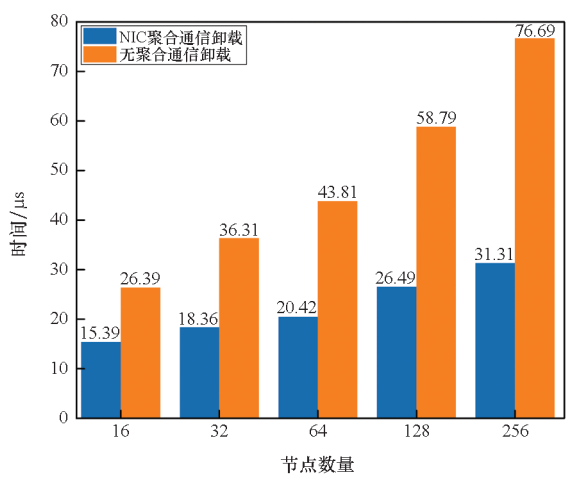
图 5 胖树拓扑结构

Fig. 5 Fat tree topology

要准确评价聚合通信的性能,必须考虑整个聚合通信过程对并行应用的影响,在此基础上选取合适的评价指标。聚合通信由一组节点共同参与完成,其中某个节点进行聚合通信的时间并不能反映聚合通信的性能。Nupairoj 和 Ni[22]提出了以聚合通信操作的完成时间为评价指标来评测聚合通信的性能,聚合通信操作完成时间定义:让所有进程在 t_0 时刻调用聚合通信操作,从 t_0 开始直到所有节点都完成该操作为止的这段时间被定义为聚合通信操作的完成时间 t_c。

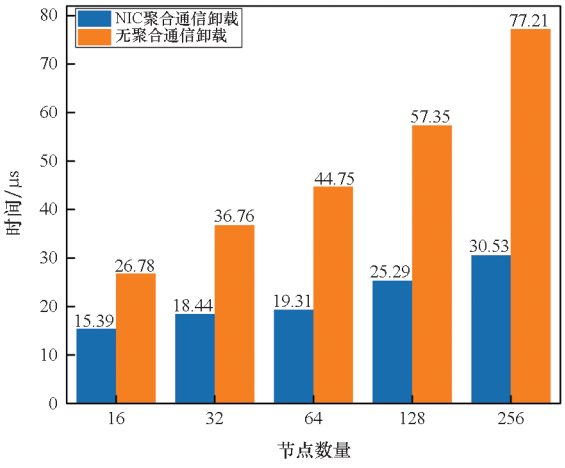
5 结果

图 6 给出了消息大小为 8 Byte,节点数量为 16、32、64、128、256 时不同数据类型 Reduce 操作



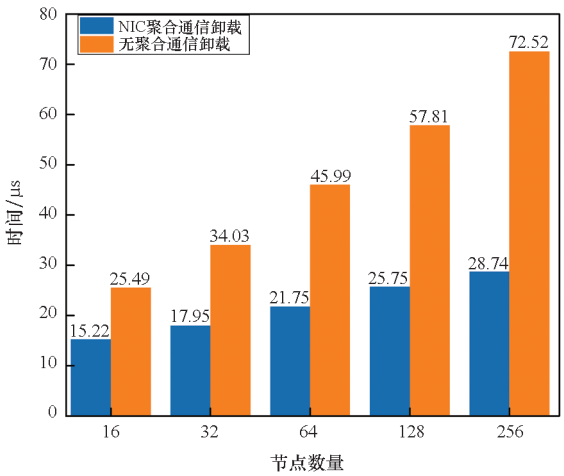
(a) 双精度浮点型

(a) Double float



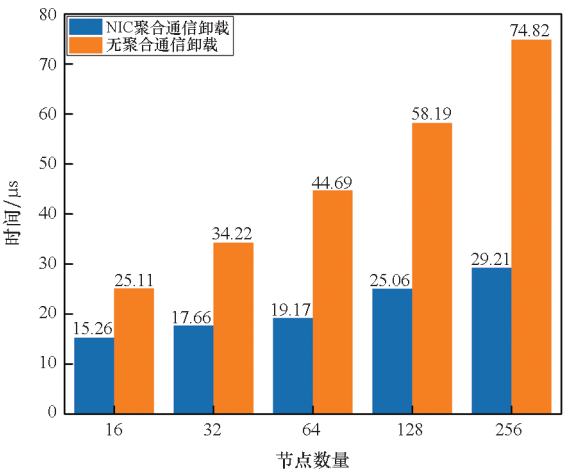
（b）浮点型

（b）Float



（c）64 位整型

（c）Int64



（d）64 位无符号整型

（d）Unit64

图 6　不同节点规模下,不同数据类型归约操作完成时间结果对比

Fig. 6　Comparison of the completion time results of the different data type reduction operations on different node size

的完成时间对比图。不同消息大小下归约操作的完成时间如图 7 所示。

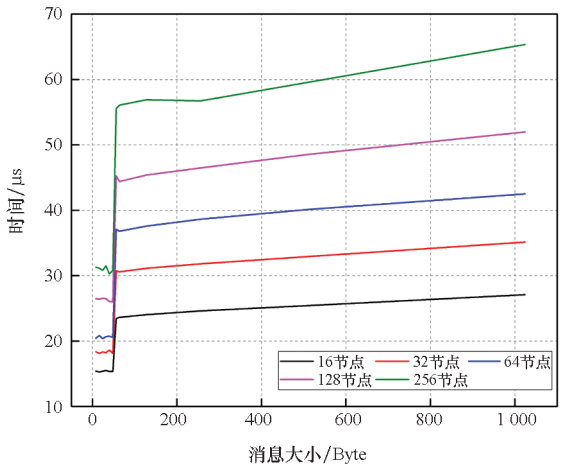


图 7　不同消息大小下,归约操作的完成时间

Fig. 7　Completion time of All-reduce operation under different message sizes

通过调整实验中消息大小,得到了表 1 至表 5中的实验结果。横向表头为操作数类型;“无”表示无聚合通信卸载,“有”表示 NIC 聚合通信卸载;纵向表头为节点数量;数据单位为 μs。

表 1　消息大小为 16 Byte 时不同节点规模通信时间

Tab. 1　Communication time of different node scales when the message size is 16 Byte

节点规模	双精度浮点	
	无	有
16	26.174	15.286
32	35.974	18.140
64	43.860	20.884
128	58.900	26.424
256	70.110	31.120

表 2　消息大小为 24 Byte 时不同节点规模通信时间

Tab. 2　Communication time of different node scales when the message size is 24 Byte

节点规模	双精度浮点	
	无	有
16	29.252	15.364
32	39.922	18.326
64	48.288	20.366
128	61.604	26.532
256	77.608	30.838

表3 消息大小为32 Byte时不同节点规模通信时间

Tab. 3 Communication time of different node scales when the message size is 32 Byte

节点规模	双精度浮点	
	无	有
16	29.336	15.502
32	39.982	18.226
64	48.032	20.686
128	63.270	26.462
256	81.324	31.492

表4 消息大小为40 Byte时不同节点规模通信时间

Tab. 4 Communication time of different node scales when the message size is 40 Byte

节点规模	双精度浮点	
	无	有
16	29.450	15.324
32	39.616	18.586
64	48.138	20.742
128	63.374	26.076
256	82.074	30.290

表5 消息大小为48 Byte时不同节点规模通信时间

Tab. 5 Communication time of different node scales when the message size is 48 Byte

节点规模	双精度浮点	
	无	有
16	29.274	15.330
32	40.210	18.132
64	48.062	20.568
128	62.502	26.014
256	78.934	30.774

从图6可以看出,本文提出的支持归约计算的网卡卸载机制优势明显。从图7可以看出,在节点规模增大以及传输数据量变大时,卸载方案优势依然稳定,但当消息大小由48 Byte增大至56 Byte时,通信时间存在较大波动,这是由于聚合通信描述符在设计时最大只能携带48 Byte的Reducedata,当消息大小大于48 Byte时,传输一条消息时会生成更多的描述符用于数据传输,导致系统整体性能下降。

对比不同节点规模Reduce操作的实验结果,可以看出在256节点规模下,相比于无聚合通信卸载,NIC卸载方法最高能获得271%的加速效果,而利用Portals网卡进行卸载的同规模Reduce操作最高能获得180%的加速效果[23]。

6 结论

基于硬件卸载的聚合通信实现机制是提升高性能网络通信性能的重要途径。本文提出了一种基于网卡的归约计算硬件卸载机制,通过在网卡上嵌入归约操作逻辑部件,在数据传输过程中完成计算功能,减轻了CPU的负担,降低了通信延迟。实验表明,本文所提出的基于NIC的硬件卸载机制能有效减少聚合通信中归约操作的时间,提升通信性能。

后续的研究中,将考虑优化现有的归约实现机制,通过保证规约计算的顺序性,实现多次规约计算具有相同的结果。此外,考虑融合多核架构和网络功能,把更大规模数据通信相关的计算功能卸载在网络上进行实现,提升系统性能。

参考文献(References)

[1] MEUER H, STROHMAIER E, DONGARRA J, et al. TOP500 supercomputer sites[EB/OL]. [2022-08-25]. http://www.top500.org.

[2] JAROS J, OHLIDAL M, DVORAK V. An evolutionary approach to collective communication scheduling[C]// Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, 2007: 2037-2044.

[3] SANCHO J C, BARKER K J, KERBYSON D J, et al. Quantifying the potential benefit of overlapping communication and computation in large-scale scientific applications[C]// Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, 2006.

[4] Open MPI. A high performance message passing library[EB/OL]. [2020-08-25]. https://www.open-mpi.org/.

[5] PETRINI F, KERBYSON D J, PAKIN S. The case of the missing supercomputer performance: achieving optimal performance on the 8 192 processors of ASCI Q[C]// Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, 2003.

[6] HUANG H, CHOW E. Overlapping communications with other communications and its application to distributed dense matrix computations[C]//Proceedings of IEEE International Parallel and Distributed Processing Symposium, 2019: 501-510.

[7] HENSGEN D, FINKEL R, MANBER U. Two algorithms for barrier synchronization[J]. International Journal of Parallel Programming, 1988, 17(1): 1-17.

[8] WATTS J, VAN DE GEIJN R. A pipelined broadcast for multidimensional meshes[J]. Parallel Processing Letters, 1995, 5(2): 281-292.

[9] RABENSEIFNER R. Optimization of collective reduction

operations[C]//Proceedings of International Conference on Computational Science, 2004: 1 - 9.

[10] THAKUR R, RABENSEIFNER R, GROPP W. Optimization of collective communication operations in MPICH[J]. The International Journal of High Performance Computing Applications, 2005, 19(1): 49 - 66.

[11] BUNTINAS D, PANDA D K, SADAYAPPAN P. Performance benefits of NIC-based barrier on myrinet/GM[C]// Proceedings of the 15th International Parallel & Distributed Processing Symposium, 2001.

[12] MOODY A, FERNANDEZ J, PETRINI F, et al. Scalable NIC-based reduction on large-scale clusters[C]//Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, 2003.

[13] ALMÁSI G, DÓZSA G, ERWAY C C, et al. Efficient implementation of allreduce on BlueGene/L collective network[C]// Proceedings of Recent Advances in Parallel Virtual Machine and Message Passing Interface, 2005: 57 - 66.

[14] MAMIDALA A R, FARAJ D, KUMAR S, et al. Optimizing MPI collectives using efficient intra-node communication techniques over the blue gene/P supercomputer [C]// Proceedings of IEEE International Symposium on Parallel and Distributed Processing Workshops and PHD Forum, 2011: 771 - 780.

[15] UNDERWOOD K D, COFFMAN J, LARSEN R, et al. Enabling flexible collective communication offload with triggered operations[C]//Proceedings of IEEE 19th Annual Symposium on High Performance Interconnects, 2011: 35 - 42.

[16] DI GIROLAMO S, JOLIVET P, UNDERWOOD K D, et al. Exploiting offload-enabled network interfaces [J]. IEEE Micro, 2016, 36(4): 6 - 17.

[17] ZHANG L F, CHEN L. Testing and performance analysis of FCA algorithm accelerating IBM platform MPI [C]// Proceedings of the 4th International Conference on Intelligent Structure and Vibration Control, 2014: 429 - 433.

[18] GRAHAM R L, BUREDDY D, LUI P, et al. Scalable hierarchical aggregation protocol (SHArP): a hardware architecture for efficient data reduction[C]//Proceedings of First International Workshop on Communication Optimizations in HPC, 2016: 1 - 10.

[19] GRAHAM R L, LEVI L, BURREDY D, et al. Scalable hierarchical aggregation and reduction protocol (SHARP)TM streaming-aggregation hardware design and evaluation[C]// Proceedings of International Conference on High Performance Computing, 2020: 41 - 59.

[20] BAYATPOUR M, SARKAUSKAS N, SUBRAMONI H, et al. BluesMPI: efficient MPI non-blocking alltoall offloading designs on modern bluefield smart NICs[C]//Proceedings of High Performance Computing. Cham: Springer, 2021: 18 - 37.

[21] ANON. OMNeT + + simulation manual[EB/OL]. [2020 - 08 - 25]. https://doc.omnetpp.org/omnetpp/manual/.

[22] NUPAIROJ N, NI L M. Performance metrics and measurement techniques of collective communication services[J]// Proceedings of Communication and Architectural Support for Network-Based Parallel Computer, 2005: 212 - 226.

[23] UNDERWOOD K D, COFFMAN J, LARSEN R, et al. Enabling flexible collective communication offload with triggered operations [C]//Proceedings of the 19th Annual IEEE Symposium on High Performance Interconnects, 2011: 35 - 42.