·开发研究与工程应用·　　文章编号：1000-3428(2023)12-0274-08　　文献标志码：A　　中图分类号：TP393

基于“嵩山”超级计算机的UCX库分析与优化

刘　康，万　伟，刘　波，李俊宏，李　柱

（郑州大学 计算机与人工智能学院，郑州 450001）

摘　要：UCX是一个经过生产验证的优化通信框架，适用于当前的高带宽和低延迟高速网络。UCX作为“嵩山”国产高性能计算平台的通信中间件，提高了并行编程模型在InfiniBand(IB)高速互联网络上的开发效率，同时其性能也会直接影响上层应用的通信能力。基于“嵩山”超级计算平台，对平台上的UCX框架进行分析与性能测试，在此过程中归纳IB适配器通信存在的局限性以及UCX在通信传输选择中的不合理性。针对这些问题，根据“嵩山”超级计算平台的网络架构特点，在参数层面进行调优，使得UCX适配“嵩山”平台的Socket Direct架构；在代码层面修改UCX对传输的选择逻辑，使得UCX在选出共享内存传输后不再选择网卡进行传输，从而解决节点内的进程间通信抢占HCA卡资源的问题。同时，修正UCX中KNEM共享内存的带宽设置，使UCX在共享内存CMA和KNEM传输的选择上更加合理。实验结果表明，使用优化后的UCX在100个节点间进行allgather集合通信测试时，相对优化前延迟至多降低80%，节点内alltoall集合通信延迟至多降低70%，gather集合通信延迟至多降低45%。改进后的UCX通信库为“嵩山”超级计算平台上的并行编程模型和应用提供了更好的互联网络支撑，明显提升了平台的集合通信性能。

关键词：UCX框架；高性能计算；集合通信；InfiniBand协议；共享内存；消息传递接口；高速网络

开放科学(资源服务)标志码(OSID)：

中文引用格式：刘康，万伟，刘波，等. 基于“嵩山”超级计算机的UCX库分析与优化[J]. 计算机工程，2023，49(12)：274-281.

英文引用格式：LIU K，WAN W，LIU B，et al. Analysis and optimization of UCX based on "Songshan" supercomputer[J]. Computer Engineering，2023，49(12)：274-281.

Analysis and Optimization of UCX Based on "Songshan" Supercomputer

LIU Kang，WAN Wei，LIU Bo，LI Junhong，LI Zhu

（School of Computer and Artificial Intelligence，Zhengzhou University，Zhengzhou 450001，China）

【Abstract】Unified Communication X(UCX) is an optimized production proven-communication framework for modern，high-bandwidth，and low-latency networks. As the communication middleware of the "Songshan" supercomputing platform，UCX improves the development efficiency of the parallel programming model on InfiniBand(IB). In addition，the performance of UCX directly affects the communication performance of upper-layer applications. This paper presents an in-depth analysis and performance test of the UCX framework on the Songshan supercomputing platform. The limitations of IB adapter communication and the unreasonable choice of UCX communication transmission are revealed. The UCX framework is adapted to the Sockets Direct architecture of the Songshan platform by tuning the parameters at the level of the identified problems. At the code level，the selection logic of UCX for transmission is modified such that UCX no longer selects the network card for transmission after selecting the shared memory transmission. This addresses the problem of inter-process communication within the node occupying the HCA card resources. The bandwidth setting of KNEM in UCX is also modified such that UCX becomes more reasonable in selecting shared memory CMA and KNEM transfers. The experimental results indicate that the use of the optimized UCX for allgather collective communication testing between 100 nodes reduces the delay by at most 80% compared to before optimization. Further，the alltoall collective communication delay within the node is reduced by at most 70%，and the gather collective communication delay is reduced by at most 45%. The optimized UCX communication library provides better Internet support for parallel programming models and applications on the Songshan supercomputing platform，thereby improving the collective communication performance of the platform.

作者简介：刘　康(1998—)，男，硕士研究生，主研方向为高性能计算、高速网络；万　伟(通信作者)，高级工程师；刘　波，硕士研究生；李俊宏、李　柱，工程师。

收稿日期：2022-10-18　　**修回日期：**2022-12-06　　**E-mail：**liukang0404@qq.com

【Key words】UCX framework；high performance computing；collective communications；InfiniBand（IB）protocol；share memory；Message Passing Interface（MPI）；high-speed network
DOI：10. 19678/j. issn. 1000-3428. 0066016

0　概述

“嵩山”超级计算机部署于国家超级计算郑州中心，是我国国产自研的新一代超高性能计算平台，它采用32核CPU+国产加速器的异构计算架构、InfiniBand（直译为“无限带宽”技术，缩写为IB）[1-2]超高速网络以及高性能分布式并行存储系统，理论峰值算力可达100 PFlops，整机实测性能达到65 PFlops。InfiniBand是一个用于高性能计算平台的计算机网络通信标准，具有极高的吞吐量和极低的延迟，用于计算机与计算机之间的数据互连，也用作服务器与存储的互连以及存储系统之间的互连。InfiniBand搭载的Mellanox ConnectX-6网卡提供了最高200 Gb/s带宽的数据传输性能。

“嵩山”计算平台所搭载的主要通信框架为UCX（Unified Communication X）[3-4]。UCX作为底层InfiniBand网络和上层并行编程模型的通信中间件，定义了一组统一的标准化通信编程接口，以满足主流的并行编程模型，如MPI（Message Passing Interface）[5]、UPC（Unified Parallel C）[6]、PGAS（Partitioned Global Address Space）[7]等的需求，同时又可以在各种高性能平台上实现，以便在互联网络上更好地满足高性能、可移植、可伸缩等并行应用的开发需求[8-10]。

但是，由于RDMA（Remote Direct Memory Access）系统具有复杂性，因此存在很多未知的问题[11-12]，UCX作为“嵩山”RDMA系统中的通信框架，在“嵩山”特色互联网络架构上还有一定的优化空间。在存在复杂通信环境的集合通信中，通信有时会成为瓶颈而拖累了整体计算速度。UCX的通信性能直接影响了上层并行编程模型的数据传输与计算性能。因此，在“嵩山”超级计算平台上对UCX进行研究与优化具有重要的工程意义。

本文基于“嵩山”超级计算平台，以MPI为例，使用osu_benchmark测试工具[13]在不同的传输下进行多种集合通信测试，获得各种情形下的延迟与带宽数据，以发现节点设备存在的瓶颈。同时，对UCX的代码进行优化，以解决节点内通信占用网卡资源的问题。在此基础上，实现UCX在“嵩山”超级计算平台上的最优传输选择，以提升平台的集合通信能力以及整体的计算性能。

1　“嵩山”超级计算机互联网络

1.1　InfiniBand互联网络

InfiniBand是一种高速互联网络，用于连接大型集群和超级计算机，它是目前应用最广泛的高速互连网络之一，2016年在Top500互连网络中就已经达到37.5%的份额[14]。InfiniBand网络为通信提供了双边（发送-接收）和单边（RDMA）语义。InfiniBand上的通信使用队列对（QP）模型，其中，发送和接收队列分别用于发送和接收消息，工作请求被提交到这些队列中，硬件可以在队列中读取工作请求以执行通信。此外，将队列与每个QP相关联，用于通知通信完成。在通过InfiniBand进行通信时，需要注意注册硬件访问的所有内存区域。为了减少内存注册的开销，短消息（Short）可以内联到工作请求中，而较大的消息可以利用零拷贝（ZCopy）协议。这种策略意味着工作请求只获取内存缓冲区的描述信息，然后直接从缓冲区读取数据，而不需要CPU的参与。

当前InfiniBand结构实现了各种传输机制[15-16]，最常见的是RC（面向连接的可靠连接）和UD（无连接的不可靠数据报），后者只实现了双边通信语义。此外，UD一次只能传输一个MTU的数据（通常是4 KB），而RC通常能提供比UD更高的带宽和更低的延迟，代价是RC对资源有很高的要求：要完全连接N个进程；要求每个进程有$O(N^2)$个连接和$O(N)$个队列对。而UD是无连接的，因此，每个进程只需要一个UD队列对。为了减少RC的内存消耗，InfiniBand规范引入了共享接受队列以及扩展的RC传输。Mellanox后续又推出了动态连接（DC）传输服务[17]，该服务动态地创建和销毁连接，将内存消耗限制在接近UD的级别，同时提供与RC类似的内存语义。然而，DC的可扩展性设计是以性能为代价的，主要是因为存在连接事务的开销[18]。

InfiniBand的用户空间接口是Verbs API[19]，它是一个带有OFED堆栈的用户级别的库，位于内核级Verbs API之上。内核API与特定供应商的InfiniBand驱动程序和驱动程序库协作，以实现InfiniBand硬件访问。InfiniBand软件栈示意图如图1所示。

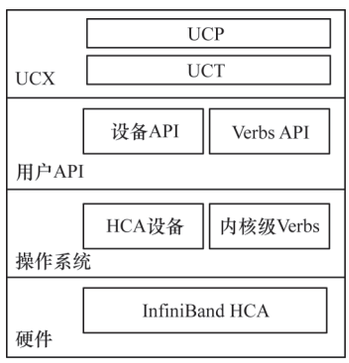

图1　InfiniBand软件栈和UCX

Fig.1　InfiniBand software stack and UCX

1.2 Socket Direct技术

随着数据量的指数级增长,使用者对服务器和计算资源提出了更高的性能要求,以便对海量数据进行实时分析。“嵩山”超级计算平台采用非均匀内存访问(NUMA)架构[20],每个处理器拥有4个CPU Die,在每个Die内集成了8个物理核心,共计32个物理核心。

在一般情况下,数据主要依靠GMI总线来进行跨Die传输,如图2(a)所示,此时Die3/Die4中的数据如果要传输至网卡,则网络流量需要使用GMI总线经过Die0/Die1然后再流入网卡。而“嵩山”平台的网络架构支持Socket Direct技术,该技术可以使节点中的每个Die(NUMA node)都可以通过其专用的PCIe接口直接连接到网络,使得网络流量无须遍历内部总线(GMI)和其他Die,如图2(b)所示。Socket Direct不仅降低了CPU的利用率、增加了网络吞吐量,还显著降低了开销与延迟,从而提高了服务器的性能。

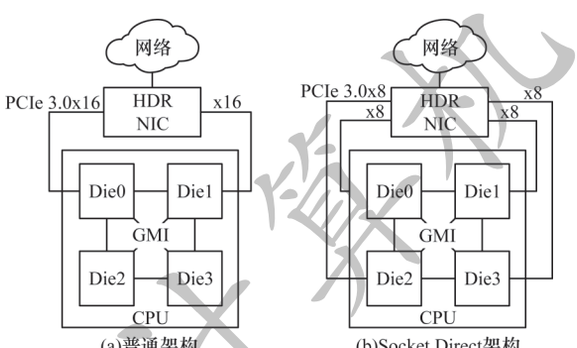

图2 跨Die传输的对比

Fig.2 Comparison of cross-Die transmission

1.3 UCX设计

随着DPU的普及以及各类DSA芯片的广泛使用[21],如何在这之上抽象出统一的内存访问语义和统一的通信方式成为一个值得研究的问题,因此,UCX应运而生。UCX可以在通信方面实现低级别的软件开销,并且提供接近原生级别的性能。UCX旨在提供一个统一的抽象通信接口,能够适配任何通信设备,并支持各种应用的需求,从而满足当前高性能、可移植且稳定可靠的并行应用开发需求,同时还能通过持续的迭代更新来适应未来的高速互联网络。

从图1可以看出UCX软件堆栈是如何放置在InfiniBand之上的,UCX由下层UCT和上层UCP这2层组成。下文将介绍UCX框架,讨论UCP和UCT这2个层之间的主要区别以及UCX内部最重要的语义。

1.3.1 UCX框架

UCX利用高速网络进行节点间通信,并利用共享内存机制进行有效的节点内通信。UCX总体采用分层结构公开一组抽象通信原语,这些原语充分利用了可用的硬件资源和负载,其中包括RDMA[22-23](InfiniBand和RoCE)、TCP、共享内存和网络原子操作。图3显示了UCX的软件栈结构。

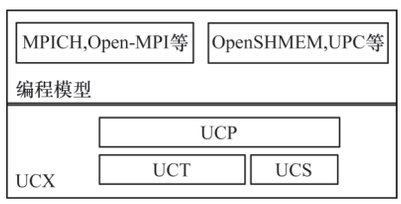

图3 UCX软件栈结构

Fig.3 UCX software stack structure

UCX通过提供高级API促进快速开发,屏蔽低层细节,同时保持高性能和可伸缩性,其框架主要由3个组件组成,即UCS(UC-Services)、UCT(UC-Transports)和UCP(UC-Protocols)。每一个组件都导出一个公共API,可以作为一个独立的库使用。底层的UCT适配各种通信设备,上层的UCP则是在UCT不同设备的基础上封装更抽象的通信接口,以方便使用。

UCT是传输层,它抽象了各种硬件架构之间的差异,并提供了一个支持通信协议实现的低级API,从单机的共享内存到常用的TCP Socket以及“嵩山”超算底层的InfiniBand协议,都有很好的支持。该层的主要目标是提供对硬件网络功能的直接有效访问,为此,UCT依赖供应商提供的低级驱动程序,如InfiniBand Verbs、Cray的uGNI等。此外,该层还提供用于通信上下文管理(基于线程和应用程序级别)以及分配和管理的构造。在通信API方面,UCT定义了立即(short)、缓冲复制、发送(BCopy)和零拷贝(ZCopy)等通信操作的接口。

UCP是协议层,通过使用UCT层公开的较低级别功能来实现上层高级编程模型(如MPI、UPC、PGAS)所使用的较高级别协议。UCP提供的功能是能够为通信选择不同的传输、消息分段、多轨通信以及初始化和完成库。目前,API具有的接口类别包括初始化、远程内存访问(RMA)通信、原子内存操作(AMO)、活动消息(Active Message)、标签匹配(Tag-Matching)和集合(Collectives)。

1.3.2 UCX语义

UCX提供的最主要语义包括通信上下文、通信原语、通信实体和连接建立。这4种语义详细叙述如下:

1)通信上下文。UCP和UCT的最主要区别在于通信上下文。UCT被设计成一个位于单个通信设备和传输层之上的通信层,而UCP可以让用户操作不同的设备和传输层。因此,UCT在设备(如InfiniBand、共享内存SM)上定义了一个内存域,用

来分配和注册进行通信的内存以及特定设备(如InfiniBand上的UD和RC)上的特定传输接口。内存域和接口都有一组它们自己的属性,这些属性来自于硬件功能。内存域属性包括内存分配限制和内存访问的凭据,接口属性包括传输机制的通信和连接能力以及协议切换的阈值。UCP将这些多个UCT内存域和接口封装在单个通信上下文中,并根据硬件属性和性能指标选择适合通信操作的接口。

2)通信原语。UCT的API定义了一组用于通信的函数,如远程单边操作、原子操作以及活动消息。因此,可以为不同的传输方法定义不同的传输协议,如对于put/get通信定义了short、BCopy和ZCopy消息传输。但是,并非每个传输层都需要实现所有功能,实际的实现情况取决于硬件功能。UCP将这些低级协议函数抽象为高级函数,即UCP提供的API用于RMA操作、远程原子内存操作和tag匹配的操作。UCP在内部选择适当的传输协议,并在必要时执行消息分段。此外,UCP为标签匹配和RMA操作提供非阻塞通信功能,可以立即重用通信资源,而UCT仅提供非阻塞的操作。

3)通信实体。Worker是UCX、UCP和UCT的核心通信实体。Worker的主要特征是有自己的进度引擎(progress engine),进度引擎会在所有打开的接口上强制执行当前的进度。在Worker要启用与另一个进程的通信时,每个进程都会创建一个端点(endpoint),并将其连接到远程进程的endpoint。UCT endpoint与特定接口(如UD、RC)绑定,即每个使用的接口对应一个UCT endpoint,而UCP endpoint拥有多个UCT endpoint。因此,在UCP中,endpoint始终连接着2个Worker。在内部,UCP负责从可用于执行通信操作的接口/UCT endpoint中选择最佳的接口/UCT endpoint。

4)连接建立。当UCP的Worker创建endpoint时,UCP层为每种类型的操作选择一个或多个接口,并且在每个接口上创建并对应一个UCT endpoint,所有的这些UCT endpoint都与父UCP endpoint相关联。如果一个接口对应无连接的传输,那么它可以立即连接到远程接口,这也就是UCP中发生的情况,即UCT endpoint通过无连接传输立即建立连接。但是,如果接口对应P2P的传输,UCP将创建一个stub endpoint。Wireup UCT endpoint始终是无连接的,通过立即发送Wireup请求然后通过P2P传输以实现所有UCT endpoint的连接。当父UCP endpoint的所有UCT endpoint都已连接时,stub endpoint即被销毁。

2 UCX在“嵩山”中的优化

2.1 参数调优

UCX可以适配多种设备、系统、架构等,因而具有繁杂的参数设置,调整各项参数可以使UCX更加适配“嵩山”平台。

“嵩山”平台的高速网络使用Socket Direct技术划分CPU为4个NUMA nodes并分别连接至4块网卡设备,实现各个Die与网卡的直连。在UCX中设置UCX_MAX_RNDV_LANES为4,为Rendezvous协议开启4端口的多轨传输,使用多块网卡同时进行传输,从而提升数据的传输效率。

“嵩山”平台CPU使用的NUMA架构,在PCIe总线传输中更适合采用宽松排序(relaxed order)的事务排序方法,即允许PCIe交换开关,将软件确认过的事务重排在其他事务之前发送,这样既提升了PCIe总线效率,又能保证程序如期执行。本文使用UCX_IB_PCI_RELAXED_ORDERING=on在UCX中开启宽松排序,使得所有使用UCX通信库的程序采用宽松排序,从而获得更高的性能。

2.2 网卡占用优化

在使用UCX进行节点内部通信时,进程间通信不仅会使用共享内存传输,还会调用网卡设备共同完成数据传输,这是由于ITIGIN[4]为UCX添加了实现,即进行进程间通信时rc会辅助共享通信,从而共同完成通信。但是在“嵩山”平台上,实测IB网卡对多进程的通信支持相较于共享内存并不友好,多进程会平分网卡带宽,导致整体性能下降。而在进行大规模节点运算时,网卡是节点间通信的主力设备,应该尽可能地保证网卡用于跨节点传输。因此,需要对UCX的传输逻辑进行修改,使其在进行节点间通信时不使用网卡。

以MPI为例,在其进行通信时,UCX会调用远程内存访问(RMA)[24]和活动消息(AM)等操作来实现快速的节点间通信。在涉及进程间通信时,UCX同样也会选择网卡来调用这些操作进行传输。

在“嵩山”超级计算平台中,MPI节点内通信使用的设备有memory和mlx5。memory会调用am、am_bw和rma_bw操作;mlx5网卡会调用am_bw和rma_bw操作。因此,mlx5网卡在节点内通信时所调用的操作完全可以被memory所取代。程序调用am_bw和rma_bw操作之前,UCT会执行ucp_wireup_add_bw_lanes函数,选出合适的传输,以此建立支持相应功能的endpoint。对此函数进行分析可以发现,函数调用ucp_wireup_select_transport,根据bitmap选出支持am_bw或rma_bw的传输,随后函数将传输放入ep的配置文件中,等待endpoint的创建,这些操作都发生在连接的准备阶段。

本文在ucp_wireup_add_bw_lanes函数中添加判断。在函数循环搜寻可用传输并将其添加到设备的dev_bitmap后,读取出Worker的上下文信息,判断此时的设备是否为共享内存传输:如果是共享内存传

输，则break跳出循环，不再搜索额外的传输；如果不是共享内存传输，则正常循环，搜寻可用传输。修改后的程序流程如图4所示，可以这样做的原因是搜索过程中的设备次序memory排在mlx5网卡之前，当选择出共享内存传输时，函数退出，便不会检索到mlx5网卡，从而在进行节点内的进程间通信时只使用共享内存通信而不是网卡传输。

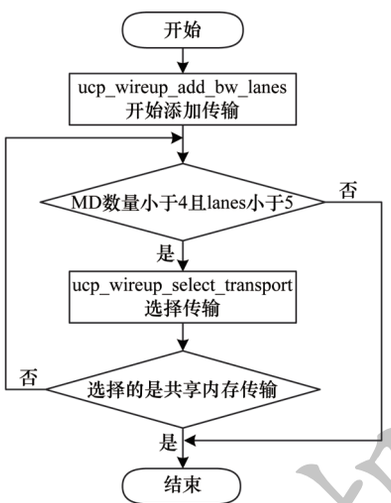

图4 优化后的程序流程

Fig.4 Optimized program procedure

使用osu_benchmark以4 MB包长测得单节点内2~32进程的alltoall集合通信延迟数据，如图5所示，其中，before是优化前同时使用网卡rc传输和共享内存传输的通信数据，after是仅使用共享内存传输的通信数据。从图5可以看出，在节点内通信时，随着PPN（Processes Per Node）的增加，2种传输方式的延迟差距愈加明显，优化后的通信延迟相较于优化前最多降低了70%。

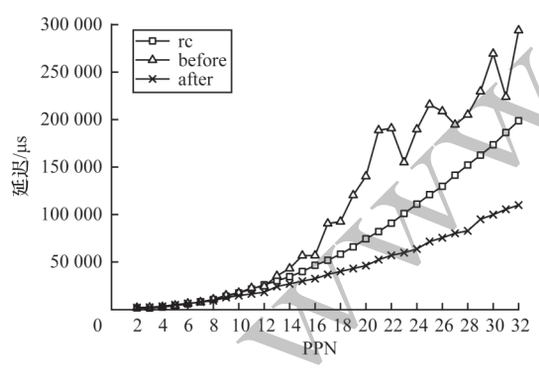

图5 节点内alltoall测试结果

Fig.5 Intra-node alltoall test results

测试不同PPN下的点对点通信带宽数据，如图6所示。从中可以明显看出，在PPN大于8时，仅使用共享内存传输的通信性能优于使用IB网卡的通信性能，此外，如果节点内的进程间通信使用了网卡传输，在绝大多数情况下，网卡通信还会对本来的进程间通信造成负面影响，降低整体的通信带宽性能。

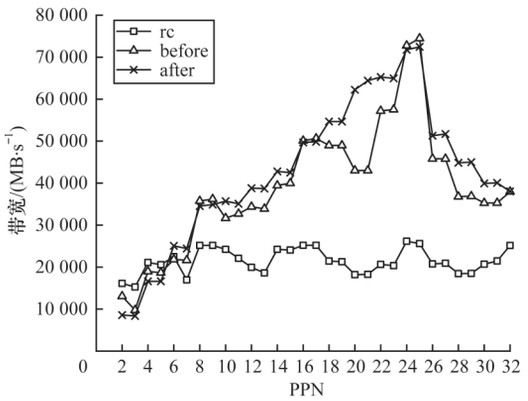

图6 节点内点对点带宽测试结果

Fig.6 Intra-node p2p bandwidth test results

2.3 共享内存通信选择优化

“嵩山”平台的MPI库中存在不同的节点内通信机制，图7主要展示了其中的2种。传统的双副本拷贝的共享内存实现，其传输数据涉及一个共享的缓冲区空间，由本地进程来交换消息，如图7(a)所示。但是，这种方式仅适用于小消息，对于较大的消息，双副本拷贝会给CPU带来额外的负担，导致缓存污染和带宽的浪费。图7(b)展示了支持内核辅助的共享内存传输实现[25]，传输实现依靠内核的援助，内核模块可以为节点内通信提供单拷贝机制，在传输较大消息时会大幅提升传输效率。

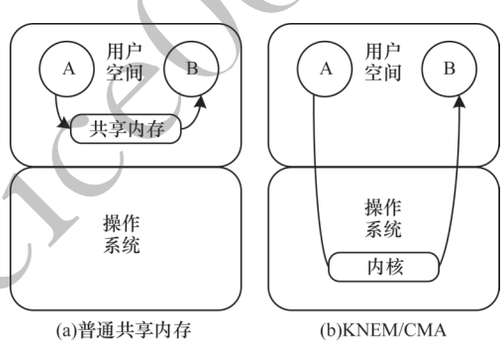

图7 2种共享内存通信机制

Fig.7 Two shared memory communication mechanisms

对于第2种通信方式，“嵩山”平台的MPI库支持CMA和KNEM[26]这2种内核模块。CMA引入了2个系统调用，分别是process_vm_readv和process_vm_writev，它们根据进程的PID和远程虚拟地址直接读写另一个进程的内存[27]。对于使用KNEM内核的通信，发送进程在KNEM驱动中声明一个发送缓冲区（不管是否连续），并将相应的标识符cookie传递给接收进程，接收进程接收到cookie并请求KNEM驱动从cookie缓冲区复制到它的本地缓冲区（连续或非连续）[26]。

本文使用osu_benchmark分别指定CMA和KNEM传输，测得2种共享内存通信的带宽与延迟如表1所示。在“嵩山”超算平台下对输出的UCX日志进行分析发现，节点在进行共享内存传输时，无论何种情况都只会选择CMA进行通信，并不会选择带

宽更高、延迟更低的KNEM。

表1　CMA与KNEM的性能参数

Table 1　Performance parameters of the CMA and KNEM

内核模块	带宽/(MB·s^{-1})	延迟/μs
CMA	9 055.98	558.67
KNEM	12 045.03	433.50

进程间在进行共享内存通信时，通过rma_bw操作来进行高速的远程内存访问。在建立连接前，UCT会根据UCX提供的一套公式来计算传输评分，选出rma_bw中评分最高的传输，添加到连接通道(lanes)中。

本文对rma_bw操作传输选择的评分机制进行分析。在UCT中，计算评分时以256 KB的消息大小为基准，调用rma_bw操作的传输的评分为时间开销的倒数，如式(1)所示：

$$s_{\mathrm{rma}}=\frac{1}{t_{\mathrm{time}}} \tag{1}$$

设m_{cost}为内存域注册开销，注册开销近似为一个线性函数，如式(2)所示：

$$m_{\mathrm{cost}}(s_{\mathrm{size}})=o_{\mathrm{md}}+g_{\mathrm{growth}}\cdot s_{\mathrm{size}} \tag{2}$$

其中：o_{md}为固定开销；g_{growth}为增长系数；s_{size}为数据大小(256 KB)。设b_l和b_r分别为本地和远程的带宽大小，因此，总开销为256 KB消息的传输时延、内存注册开销m_{cost}与传输接口间延迟l_{lr}的累加，如下：

$$t_{\mathrm{time}}=\frac{262\,144}{\min\{b_l,b_r\}}+m_{\mathrm{cost}}(262\,144)+l_{lr} \tag{3}$$

对于节点中的每个进程，其带宽b在UCX中的计算方式如式(4)所示：

$$b=b_{\mathrm{dedicated}}+\frac{b_{\mathrm{shared}}}{P_{\mathrm{PPN}}} \tag{4}$$

其中：$b_{\mathrm{dedicated}}$为专用带宽；b_{shared}为共享带宽。

对平台的UCX源代码进行分析，可以发现：在UCX 1.9.0的带宽设置下，CMA拥有11 145.00 MB的dedicated带宽，而KNEM是13 862 MB的shared带宽。根据s_{rma}的计算公式，在PPN不为1时，UCX在计算KNEM和CMA的rma_bw评分时带宽会存在巨大差异，从而导致永远不会选择KNEM传输。这是因为早期KNEM对多进程支持不如CMA，单进程时KNEM会有更高的带宽，但是存在多进程通信时，KNEM性能将不如CMA，因而将KNEM带宽值设置为shared。但是，“嵩山”超级计算平台所具有的优化KNEM对多进程的支持极好，同时支持高性能单拷贝消息传输。因此，本文将KNEM的带宽从shared改为dedicated，使KNEM获得了更合理的评分，从而在进行集合通信时共享内存方面的传输会更多地选择带宽更高、延迟更低的KNEM。

在大部分通信中，KNEM和CMA两者差异较小，但是在涉及节点内进程间gather通信时，KNEM内核相对CMA内核有较为明显的性能提升，并且随着PPN的增加提升效果愈加明显，如图8所示。

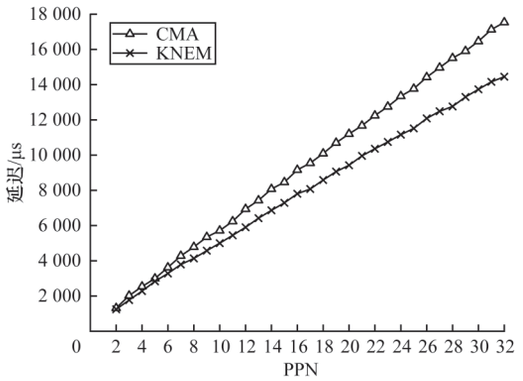

图8　节点内gather测试结果

Fig.8　Intra-node gather test results

3　数据测试与验证

3.1　实验环境

在“嵩山”超级计算机的固化节点上进行实验，单个节点配置为32核2.0 GHz CPU，网卡采用Mellanox ConnectX-6，以HDR模式(200 Gb/s)工作。操作系统为CentOS 7.6，内核版本为3.10.0-957.el7.x86_64。

在本次测试中，使用的MPI版本为Open MPI v4.0.4rc3，它在平台的共享内存通信时支持KNEM和CMA内核。使用的HPCX版本为2.7.4，UCX版本为UCX 1.9。对于点到点和集合通信测试，使用osu_benchmark v5.5来测试并记录通信性能数据。在性能测试对比数据中，before的通信底层是目前在用的由ITIGIN优化后的UCX正式版本，after采用本文优化后的UCX库。

3.2　节点内集合通信测试

首先测试优化后的UCX库在单节点内的通信表现。图9~图11展示了单节点内不同PPN下的4 MB包长的MPI集合通信测试数据，横坐标为使用核心数(总进程数)，纵坐标为平均通信延迟，每个核心绑定一个进程进行通信。从中可以看出，使用优化后UCX的MPI集合通信能力有了明显提升，alltoall的通信性能提升尤为明显，延迟最多降至优化前的30%(图9)，gather的通信延迟最多约降至优化前的55%(图10)，allreduce的通信延迟最多约降至优化前的69%(图11)。

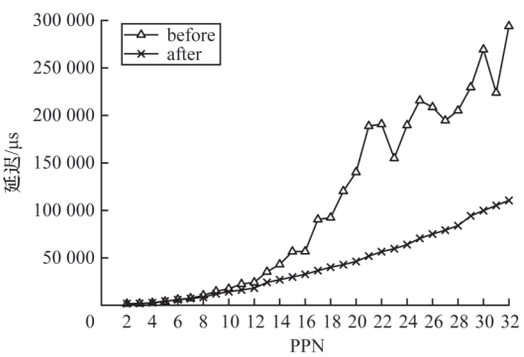

图9　优化前后节点内alltoall测试结果

Fig.9　Intra-node alltoall test results before and after optimization

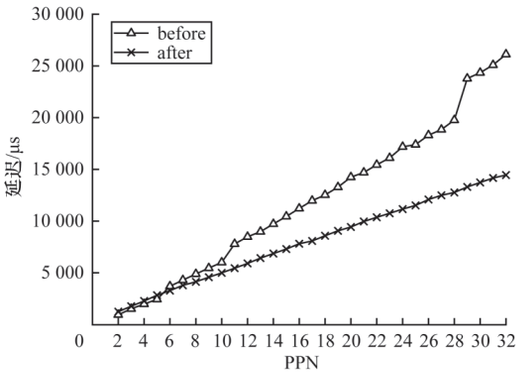

图10 优化前后节点内gather测试结果

Fig.10 Intra-node gather test results before and after optimization

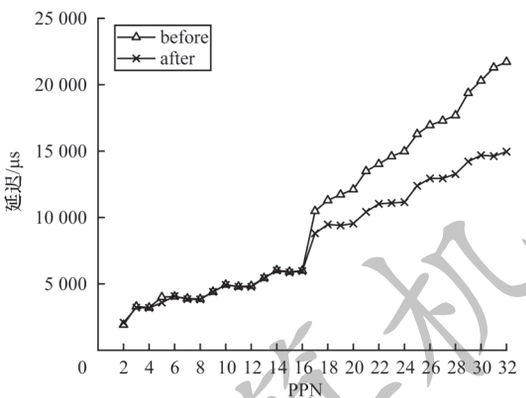

图11 优化前后节点内allreduce测试结果

Fig.11 Intra-node allreduce test results before and after optimization

3.3 节点间集合通信测试

对于节点间的集合通信，本文对2个规模（32节点和100节点）进行测试。对于32节点的规模，选取分属lka 2个交换机的32个节点，每个交换机16个节点，测试消息包长为1 MB。经过测试发现，在节点间集合通信时，其他集合通信测试效果与优化前一致，allgather通信产生了较为明显的差异，如图12所示。

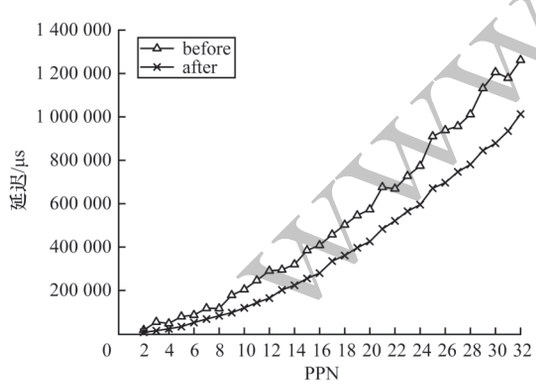

图12 32节点allgather测试结果

Fig.12 32 nodes allgather test results

本文在8箱的节点中随机选择100个节点，进行100个节点间的集合通信测试，获得节点间的2~18 PPN下1 MB包长的集合通信延迟数据。从图13可以看出，优化后的UCX在allgather集合通信中取得了极为明显的优化效果，延迟最多可降至原来的20%，并且随着进程的增多差距逐渐变大。其他的集合通信测试优化前后数据基本保持一致。

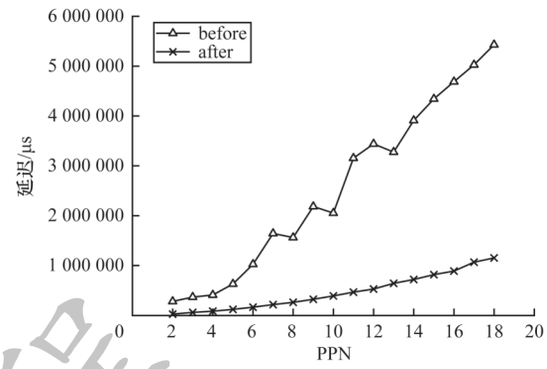

图13 100节点allgather测试结果

Fig.13 100 nodes allgather test results

4 结束语

“嵩山”超级计算平台支持多种并行编程模型，对高速互联网络进行优化有助于提升平台的整体通信性能，为平台的并行编程模型提供良好的底层通信支持。本文对“嵩山”超级计算平台上的节点进行测试，获得了节点间与节点内的通信性能数据，并且发现IB网卡在节点内多PPN通信中存在的局限性。然后，对平台的主要通信框架UCX进行分析与优化，解决了节点内进程间通信占用网卡的问题，同时改善了UCX对共享内存传输的选择机制。优化后的UCX对大PPN下的节点间allgather集合通信以及节点内的进程间集合通信性能提升效果明显。

由于RDMA具有复杂性，很多因素都可能影响RDMA系统的整体通信性能。下一步将找出其他制约节点间通信速度的因素，对算法进行改进，使得节点间的其他集合通信能力得到加强。此外，UCX根据PPN来预测带宽，依此带宽来选择传输，这种带宽计算方法还不够准确，未来将对此进行改进，从而改善UCX的传输选择评分机制。

参考文献

[1] PFISTER G F. An introduction to the InfiniBand architecture[EB/OL]. [2022-09-05]. http://www.diku.dk/hjemmesider/ansatte/vinter/cc/Infinibandchap42.pdf.

[2] 王知恒. InfiniBand网络协议层软件技术研究[D]. 杭州：浙江大学，2021.
WANG Z H. Research on InfiniBand network protocol layer software technology[D]. Hangzhou: Zhejiang University, 2021. (in Chinese)

[3] SHAMIS P, VENKATA M G, LOPEZ M G, et al. UCX: an open source framework for HPC network APIs and beyond[C]//Proceedings of 2015 IEEE Annual Symposium on High-Performance Interconnects. Washington D. C., USA: IEEE Press, 2015: 15-22.

[4] ITIGIN Y. GitHub-openucx/UCX: Unified Communication X[EB/OL]. [2022-09-05]. https://github.com/openucx/ucx/releases/tag/v1.9.0.

[5] GABRIEL E, FAGG G E, BOSILCA G, et al. Open MPI: goals,

concept, and design of a next generation MPI implementation [EB/OL]. [2022-09-05]. https://www. researchgate. net/publication/221597359_Open_MPI_Goals_Concept_and_Design_of_a_Next_Generation_MPI_Implementation.
[6] EL-GHAZAWI T, SMITH L. UPC: Unified Parallel C[C]// Proceedings of 2006 ACM/IEEE Conference on Supercomputing. Washington D. C. , USA: IEEE Press, 2006: 27-35.
[7] ALMASI G. PGAS (Partitioned Global Address Space) languages[EB/OL]. [2022-09-05]. https://link. springer. com/referenceworkentry/10. 1007/978-0-387-09766-4_210.
[8] 谢旻,张伟,周恩强,等. 面向天河互连网络的可扩展通信框架实现技术[J]. 计算机工程与科学,2020,42(10):1720-1729.
XIE M, ZHANG W, ZHOU E Q, et al. Implementation of scalable communication framework on TH-express interconnection[J]. Computer Engineering and Science, 2020, 42(10): 1720-1729. (in Chinese)
[9] CZARNUL P. Parallel programming for modern high performance computing systems[M]. [S. l.]: CRC Press, 2018.
[10] 谢旻,周恩强,董勇,等. 基于天河互连的公共通信接口UCX实现与评估[J]. 计算机应用,2019,39(S1):113-118.
XIE M, ZHOU E Q, DONG Y, et al. Implementation and evaluation of UCX communication interface on TH-express interconnection[J]. Journal of Computer Applications, 2019, 39(S1): 113-118. (in Chinese)
[11] KONG X, ZHU Y, ZHOU H, et al. Collie: finding performance anomalies in RDMA subsystems [EB/OL]. [2022-09-05]. https://www. usenix. org/system/files/nsdi22-paper-kong. pdf.
[12] CZARNUL P, PROFICZ J, DRYPCZEWSKI K. Survey of methodologies, approaches, and challenges in parallel programming using high-performance computing systems [EB/OL]. [2022-09-05]. https://www. hindawi. com/journals/sp/2020/4176794/.
[13] MVAPICH. OSU micro-benchmarks 5. 5 [EB/OL]. [2022-09-05]. https://mvapich. cse. ohio-state. edu/download/mvapich/osu-micro-benchmarks-5. 5. tar. gz.
[14] PAPADOPOULOU N, ODEN L, BALAJI P. A performance study of UCX over InfiniBand[EB/OL]. [2022-09-05]. https://www. researchgate. net/publication/318409171_A_Performance_Study_of_UCX_over_InfiniBand.
[15] LIU P N, GUITART J. Performance characterization of containerization for HPC workloads on InfiniBand clusters: an empirical study[J]. Cluster Computing, 2022, 25(2): 847-868.
[16] BAYATPOUR M, GHAZIMIRSAEED S M, XU S L, et al. Design and characterization of InfiniBand hardware tag matching in MPI[EB/OL]. [2022-09-05]. https://ieeexplore. ieee. org/document/9139619.
[17] PARK J, YEOM H. Design and implementation of software-based dynamically connected transport[EB/OL]. [2022-09-05]. https://ieeexplore. ieee. org/document/8599533.
[18] TAKAGI M, YAMAGUCHI N, GEROFI B, et al. Adaptive transport service selection for MPI with InfiniBand network[C]//Proceedings of the 3rd Workshop on Exascale MPI. New York, USA: ACM Press, 2015: 1-10.
[19] MACARTHUR P, LIU Q, RUSSELL R D, et al. An integrated tutorial on InfiniBand, Verbs, and MPI[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2894-2926.
[20] JING X J, LI H X. Construction and optimization of heterogeneous memory system based on NUMA architecture [EB/OL]. [2022-09-05]. https://ieeexplore. ieee. org/abstract/document/9778754.
[21] BURSTEIN I. Nvidia Data center Processing Unit (DPU) architecture[EB/OL]. [2022-09-05]. https://ieeexplore. ieee. org/abstract/document/9567066.
[22] MARGOLIN A, BARAK A. RDMA-based library for collective operations in MPI [EB/OL]. [2022-09-05]. https://ieeexplore. ieee. org/document/8955451.
[23] XING J, HSU K F, QIU Y, et al. Bedrock: programmable network support for secure RDMA systems [EB/OL]. [2022-09-05]. https://www. usenix. org/system/files/sec22summer_xing. pdf.
[24] HOEFLER T, DINAN J, THAKUR R, et al. Remote memory access programming in MPI-3 [J]. ACM Transactions on Parallel Computing, 2015, 2(2): 1-26.
[25] MA T, BOUTEILLER A, BOSILCA G, et al. Impact of kernel-assisted MPI communication over scientific applications: CPMD and FFTW[EB/OL]. [2022-09-05]. https://link. springer. com/chapter/10. 1007/978-3-642-24449-0_28.
[26] GOGLIN B, MOREAUD S. KNEM: a generic and scalable kernel-assisted intra-node MPI communication framework[J]. Journal of Parallel and Distributed Computing, 2013, 73(2): 176-188.
[27] VIENNE J. Benefits of cross memory attach for MPI libraries on HPC clusters[C]//Proceedings of 2014 Annual Conference on Extreme Science and Engineering Discovery Environment. New York, USA: ACM Press, 2014: 1-6.

编辑　吴云芳