非连续数据网络通信实现方法和性能分析

马潇潇[1),2)] 陆 钢[3)] 付斌章[3)] 安仲奇[1)] 朱泓睿[1),2)]
邵 恩[1)] 王 展[1)] 安学军[1),2)]

[1)](中国科学院计算技术研究所 北京 100190)
[2)](中国科学院大学 北京 100080)
[3)](华为技术有限公司罗素实验室 北京 100080)

摘 要 非连续数据通信是指发送端将位于不同地址的多块数据传输到接收端的多个非连续地址.这种通信模式在科学计算应用中十分常见,如求解计算、FFT 计算、流体力学模拟等应用均涉及矩阵的转置传输,多维矩阵的子矩阵传输,非结构化数据访问等非连续数据通信.所以,非连续数据的通信性能对众多科学计算应用有重要的影响.目前,有多种实现非连续数据通信的卸载或者非卸载的方法,但是迄今没有工作在同一平台对主流的非连续数据通信实现方法进行评测和分析,也没有工作对每一种实现方式适用的情况进行总结.本文首先总结了目前非连续数据通信的实现方式,然后,本文使用已有的测试集和自己设计的测试集对不同方式的非连续数据通信性能进行了详细的对比测试,细粒度地分析了在不同数据分布的情况下数据拷贝和 RDMA 通信的开销,尤其对基于 RDMA sg_list(scatter-gather list)和 UMR(User-mode Memory Registration)功能的卸载性能进行了分析,并总结了各种非连续数据通信方式的适用情况和存在的问题.最后,本文通过实验验证了分析结果的正确性,并对于该分析结果相关的技术提出了优化的方向.

关键词 非连续数据通信;远程数据直接访问卸载;SGRS;UMR;人工打包拆包
中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2020.01123

Implementation Methods and Performance Analysis of Non-Contiguous Data Communication in Network

MA Xiao-Xiao[1),2)] LU Gang[3)] FU Bin-Zhang[3)] AN Zhong-Qi[1)] ZHU Hong-Rui[1),2)]
SHAO En[1)] WANG Zhan[1)] AN Xue-Jun[1),2)]

[1)](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)
[2)](*University of Chinese Academy of Sciences, Beijing* 100080)
[3)](*Russell Lab, HUAWEI Technology Co., Ltd., Beijing* 100080)

Abstract Non-contiguous data communication means that the sender transfers multiple blocks of data at discontinuous addresses to multiple memory regions with discontinuous addresses on the receiver. This communication model is common in scientific computing applications, such as solution calculation, FFT calculation, fluid dynamics simulation, etc. These applications include data transfer operations such as the transfer of matrix transpose, the transfer of submatrix of 2D,

收稿日期:2019-09-27;在线出版日期:2020-02-16.本课题得到国家重点研发计划课题(2018YFB0204400)、十三五国家重点研发计划课题(2016YFB0200205)、国家自然科学基金青年基金项目(61702484)、中国科学院先导 B 类专项课题(XDB24050100)和华为技术有限公司合作项目(YBN2018015521)资助.**马潇潇**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为高性能网络和在网计算.E-mail:maxiaoxiao@ncic.ac.cn.**陆 钢**,博士,中国计算机学会(CCF)会员,主要研究方向为高性能网络、网络虚拟化、操作系统和分布式计算.**付斌章**,博士,中国计算机学会(CCF)会员,主要研究方向为高性能网络、网络虚拟化、拥塞控制和 SDN 等.**安仲奇**,工程师,中国计算机学会(CCF)会员,主要研究方向为高性能计算和高性能网络.**朱泓睿**,博士研究生,中国计算机学会(CCF)会员,主要研究方向为分布式深度学习通信网络、数据中心网络、拓扑及流量优化和网络模拟器研究.**邵 恩**,博士,工程师,主要研究方向为计算机系统结构、光电混合网络和云计算.**王 展**,副研究员,主要研究方向为高性能互联网络和分布式系统结构.**安学军**,教授,主要研究方向为计算机系统结构和高性能互联网络.

3D and 4D matrices, unstructured data access, and other non-contiguous data communication. Therefore, the communication performance of non-contiguous data has important influences on many scientific computing applications. Currently, there are some offloading or non-offloading methods to achieve non-contiguous data communication, but no one has measured these kinds of methods on one platform. Further, nobody has analyzed or proposed a guideline of which situation that each method is suitable for before now. We give the summary of implementation methods and performance analysis of non-contiguous data communication in this paper. The main contributions of this paper include: (1) a comprehensive summary of the implementation methods; (2) a series of detailed performance experiments using dependable micro-benchmarks and applications by different methods; (3) comparative analysis and useful conclusion for our experimental platform; (4) potential problems and research points of non-contiguous data communication in future work. Firstly, this paper summarizes the current implementation methods of non-contiguous data communication. The non-offloading method consists mainly of manual copy and some callable interfaces like MPI DDT (Message Passing Interface Derived Data Type) based on data movement in memory. The offloading method includes different implementations that make use of RDMA (Remote Direct Memory Access) technology to achieve different degree of decreasing data copy. After that, we use both existing benchmarks and self-designed benchmarks to measure the performance of non-contiguous data communication in different ways as we summarize in detail. All these experiments are completed on the same experimental platform for comparison and analysis. We also give the fine-grained analysis of the overhead of data copying and RDMA communication in the case of different data distributions. Especially, we relatively analyze the offloading performance based on RDMA sg_list (scatter gather list) and the offloading performance based on UMR (User-mode Memory Registration) functions, and conclude the applicable situations and potential problems of various methods of non-contiguous data communication. We list the data table as a guideline for non-contiguous data communication on our platform. We find that RDMA offloading methods truly have an advantage over memory copy in performance when the block size is large, but some problems still exist. The low efficiency of UMR MTT (Memory Table Translation) may cause performance degradation when block number becomes large. Finally, this paper verifies the correctness of the analysis results through micro-application experiments, and proposes the optimization direction of the technology related to the analysis results.

Keywords non-contiguous data communication; remote direct memory access offloading; sender gather receiver scatter; user-mode memory registration; manual packing/unpacking

1 引 言

非连续数据通信是指将发送端非连续地址空间的多个数据块传输到接收端的多个非连续地址空间内,这种通信模式被广泛地应用于科学计算应用中.例如,求解计算、矩阵计算和 FFT(Fast Fourier Transform)计算等应用往往会涉及大量的非连续数据通信[1].由于非连续数据通信在科学计算中使用频繁,其对科学计算应用的整体性能有着重要影响.

目前,非连续数据通信的主要实现方式可以分为两大类:非卸载方式和卸载方式.非卸载与卸载的区别在于是否减少内存数据拷贝.非卸载方式是指在未进行通信时,先使用数据拷贝的方式完成非连续数据空间与连续数据空间的转换,然后再使用连续数据通信的方式完成非连续数据通信;卸载方式是指使用 0-copy 方法,如远程数据直接访问,直接进行非连续数据通信,在不同程度上,减少非连续数据通信过程中的数据拷贝.表 1 给出了各种卸载、非卸载具体实现方式的对比,其中最右边一列表示各种实现方式中数据拷贝的程度,"一边"指发送端或者接收端,"双边"指发送端和接收端.

表 1　非连续数据通信方式

非卸载方案		人工打包拆包	双边人工 copy
		MPI DDT	MPI 内部 copy
RDMA 卸载方案	基于 sg_list	基于 SEND/RECEIVE 方式(SGRS)	双边 0-copy
		基于 READ/WRITE 方式	一边 0-copy，一边人工 copy
	基于 UMR	基于SEND/RECEIVE方式	双边 0-copy
		基于 READ/WRITE 方式	双边 0-copy

非卸载的实现方式主要包含人工拷贝和 MPI[2](Message Passing Interface) DDT[2](Derived Data Type). 人工打包拆包的核心思想是将发送端的非连续数据拷贝到连续缓冲区中供网卡读取并传输，当接收端网卡将接收的数据写入连续的内存后，接收端再将数据拷贝到不连续的地址空间中. 这种方式引入了大量的数据拷贝开销. 研究[3]表明，人工打包拆包操作占据了整个通信过程 90%以上的时间开销. 除此以外，这种方式要求开发人员显式地进行数据拷贝，大大增加了程序复杂度. MPI DDT 是另一种非卸载的实现方式，它为用户提供了一种便捷的非连续数据通信方式. 用户可以使用 MPI 通信库提供的衍生数据类型描述需要进行通信的非连续数据，从而，将通过这种方式定义的数据类型视作地址空间连续的数据进行通信. MPI DDT 虽然免除了用户的显式拷贝，但是其仅仅是将拷贝过程移至通信库层面，仍然存在较大的拷贝开销，且引入了对数据类型的存储和计算开销[4-7]. 以上研究表明，减少非连续数据通信中的数据拷贝开销对于其性能提升至关重要.

RDMA(Remote Direct Memory Access)技术提供了减少内存拷贝、直接进行非连续数据传输的可能. 部分相关的 RDMA 术语含义可参见附录 1. 利用 RDMA 技术进行非连续数据通信卸载主要基于 sg_list(scatter gather list)和 UMR(User-mode Memory Registration)两种机制，可以减少非连续数据通信中数据的搬移，提高通信的性能. 具体地，实现 sg_list 和 UMR 机制时又可以采用 RDMA WRITE 或 SEND 来实现. 文献[8]中提出了单边 0-copy 的 RDMA 卸载方案，该方案在发送端使用 sg_list 描述非连续数据分布，通过 RDMA WRITE 直接发送非连续数据实现发送端单边 0-copy，但是在接收端只能接收到连续的数据，需要将数据人工搬移到非连续的空间，或者发送端使用多次的 RDMA WRITE 操作将非连续的数据分别写到接收端非连续的空间. 文献[9]提出的 SGRS(Sender Gather Receiver Scatter)可实现发送端和接收端的双边 0-copy，即在 SEND/RECEIVE 通信方式下，收、发端都使用 sg_list 直接描述非连续数据的分布，发送端网卡根据 sg_list 将多个非连续的数据 Gather 发出，接收端网卡根据 sg_list 将接收到的数据 Scatter 到对应地址. 但是，使用这种方式时，通信两侧都需要 poll_cq 操作(poll_cq 用于查询网卡处理的请求是否完成)，会引入开销，并且这种方式对双边的数据分布要求严格. 此外，还有一种基于 UMR 的卸载方案可以实现双边 0-copy 的非连续数据通信[10-11]. 在进行通信之前，用户空间的地址映射关系被下发到网卡完成地址绑定；在数据传输时，使用 RDMA WRITE 方式或者 SEND/RECEIVE 方式完成双边 0-copy 的非连续数据传输，并且这种方式支持双边数据的灵活分布.

已有的研究中，多只对某一种 0-copy 的非连续数据通信方式与人工打包拆包方式的性能进行对比. 并未针对 sg_list、UMR、MPI DDT 和人工打包拆包进行系统的比较和分析. 本文将探讨各种卸载、非卸载方式进行非连续数据通信的优劣，通过详细的测试，分析不同实现方式适用的情况和存在的问题. 本文的第 2 节将简述非卸载方式的非连续数据通信；第 3 节简述 RDMA 卸载方式的非连续数据通信；第 4 节将介绍针对不同实现方式进行测试的方案；第 5 节给出实验结果，并对实验结果进行分析和总结；最后，在第 6 节，我们对本文进行总结，并指出进一步的工作方向.

2　非卸载的非连续数据通信

非卸载的非连续数据通信主要包含人工打包拆包和 MPI DDT 两种方式. 目前，对非卸载方式的研究主要集中在 MPI DDT 性能评测和优化这两个方面. 文献[3]中总结了常用非连续数据通信模式的微基准测试集，比较了人工打包拆包与 MPI DDT 的开销和性能. 文献[4-5,12-13]则着重对不同 MPI 实现、不同 DDT 类型进行了性能评测，并给出了部分性能指导模型. 以上研究均以性能评测为主，未对提升性能做出实质性的工作. 而研究[6-7,14]则对 MPI DDT 的软件库做了算法上的优化和编译上的优化，用以提升特定 MPI DDT 类型的性能，但是由于 MPI DDT 种类繁多，实现复杂，在目前通用的 MPI 实现中并未采用. 该章节将分别对人工打包拆包和 MPI DDT 进行介绍，说明使用非卸载方式进行非

连续数据通信存在的问题.

2.1 人工打包拆包

人工打包拆包方式是进行非连续数据通信的通用方式,如图 1 所示,即在发送端将非连续数据人工拷贝到连续的数据空间,然后将连续的数据发送到接收端,接收端在接收到连续的数据之后,再由人工拷贝到非连续的数据空间.

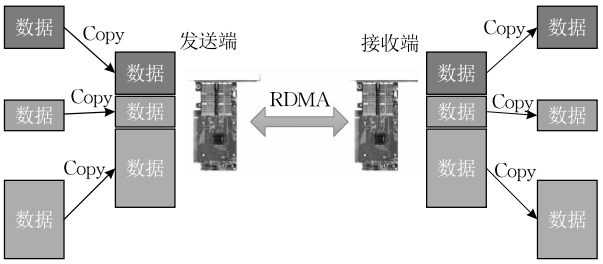


图 1 人工打包拆包非连续数据通信示意图

人工打包拆包方式的缺点是需要在通信过程中多次进行人工拷贝数据.具体而言,其一,这种方式导致程序繁琐,给编程人员带来很大的不便;其二,人工打包拆包引入多次数据拷贝的开销,影响应用的实际性能.

2.2 MPI DDT

MPI 提供了一种用户可以自定义衍生数据类型(Derived Data Type,DDT)的机制,图 2 为 OpenMPI 中 DDT 涉及到的层次图,用户使用 MPI_Type_vector、MPI_Type_indexed、MPI_Type_ struct 等函数接口传入对应的非连续数据分布描述(count,length,stride,old datatype)和要创建的自定义类型名 newtype,可以分别创建符合 vector、index、struct 等数据分布的新数据类型——newtype,在调用 MPI_Type_Commit(newtype)之后,就可以在以后的程序中,将 newtype 描述的非连续数据视为连续数据,完成自定义数据类型的数据通信.

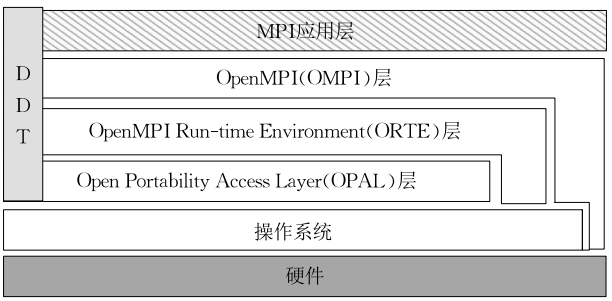


图 2 OpenMPI 中 DDT 涉及到的层次图

图 3 为 MPI DDT 中常用的类型说明:

(1) vector 类型.每个非连续数据块所包含的数据类型一致,数据长度一致,非连续数据块之间的间隔一致.

(2) index 类型.每个非连续数据块所包含的数据类型一致,数据长度可以不一致,非连续数据块之间的间隔可以不一致.

(3) struct 类型.每个非连续数据块所包含的数据类型和数据长度都可以不一致,非连续数据块之间的间隔可以不一致.

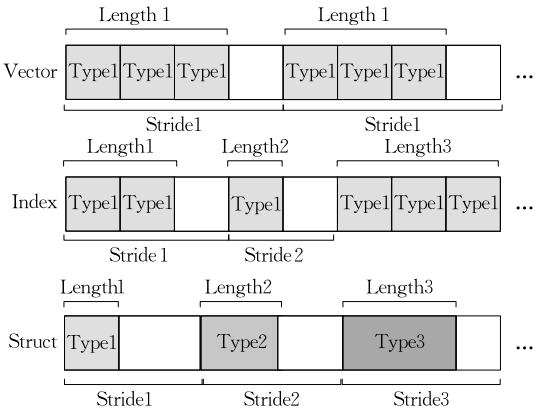


图 3 MPI DDT 典型数据分布

以上三种数据分布是 MPI DDT 中常见的三种基本类型,描述复杂度和使用灵活性逐渐增加.此外 MPI 还提供了其他的 DDT 类型和操作,如 subarray 类型用于描述子数组,MPI_Pack/Unpack 函数用于完成人工打包拆包功能等.新创建的数据类型可以直接用于点对点或者集合通信的数据传输,亦可以作为子类型来创建更加复杂的数据类型.

MPI 通信库对 DDT 处理过程如下:对于用户定义的 DDT,MPI 通信库根据用户描述的数据分布,对用户提供的数据描述进行 Normalization[6-7],Normalization 的目的是在存储数据类型描述开销和访问数据类型描述开销之间进行平衡,获得优化后的数据描述——MPI Type Tree,用于数据类型的使用.对于同一种数据分布,可以有多种 MPI DDT 描述,通过 Normalization 可以获得最少的数据类型存储开销和处理开销,提高 MPI DDT 的性能.不同的数据分布、不同的 DDT 描述和不同的 MPI 实现都会导致 MPI DDT 性能的差异[5].

总结:MPI DDT 可以为用户提供方便简洁的调用接口,但是 MPI DDT 本身在进行 DDT 创建和使用过程中会引入开销.因此,很多用户为达到更高的性能,选择不使用 MPI DDT,使用人工打包拆包方式进行非连续数据通信.此外,不同 MPI 实现在进行 DDT 处理的过程中采取的策略不同,性能也会有所差异.当非连续的数据块中包含的数据量较

大时，人工打包拆包或者 MPI 内部数据拷贝的开销巨大，进行 0-copy 传输显得十分必要.

3　RDMA 卸载的非连续数据通信

在该部分会涉及 RDMA 原语的表述，关于 RDMA 相关原语的含义，请参见文末附录 1. 目前，RDMA 技术支持的两种通信方式分别为 SEND/RECEIVE 和 READ/WRITE. 其中，SEND/RECEIVE 方式，需要接收端调用 ibv_post_recv，发送端调用 ibv_post_send，两边都需要调用 ibv_poll_cq 检查向网卡发送的 Work Request（后文称为 WR）是否完成；READ/WRITE 方式，发起操作的一端需要知道对端的地址，然后调用 ibv_post_send，向网卡下发操作为 READ/WRITE 的 WR，将数据从对端连续的空间读出或者将数据写到对端连续的空间，仅在发送端调用 ibv_poll_cq 查询操作是否完成即可.

在这两种通信方式下，可以使用 sg_list 和 UMR 描述通信中涉及的非连续数据分布，这也是目前使用 RDMA 卸载非连续数据通信的主要实现方法. 研究[8]使用 sg_list 在 RDMA WRITE 模式完成通信，但只能实现单边的数据 0-copy. 研究[9]中使用 sg_list 在 SEND/RECEIVE 模式完成通信，可以实现双边的数据 0-copy，但是面对复杂的数据分布时存在灵活性的问题. 研究[11]中使用 UMR 进行非连续数据通信，实现双边数据 0-copy，提供较高的灵活性，但该研究只与非卸载的方式进行性能对比，卸载的效果需要进一步评测分析. 本章节将对基于 sg_list 和 UMR 的卸载方式进行详细介绍.

3.1　基于 sg_list 的 RDMA 卸载

Mellanox 提供了用户模式直接访问硬件进行 RDMA 通信的 API 接口——verbs API. 在 verbs 层进行非连续数据通信时，verbs 提供了 sg_list 机制用于描述通信的多个非连续数据，sg_list 为 ibv_sge 的链表，每个 ibv_sge 含有描述数据分布的 addr（地址）、length（数据长度）和 memory key（内存键，用于识别内存访问权限），sg_list 作为 ibv_send_wr 或者 ibv_recv_wr 结构体中的一部分通过调用 ibv_post_send 或者 ibv_post_recv 被下发到网卡，网卡根据 sg_list 中的数据描述完成非连续数据通信.

图 4 为使用 sg_list 在 WRITE 通信方式下完成单边（发送端）0-copy 非连续数据通信的示意图. 发送端使用 sg_list 描述非连续的数据分布，使用 0-copy 的 RDMA WRITE 将非连续的数据写到接收端连续的数据空间，接收端再将连续的数据拷贝到非连续的空间. 图 5 为使用 sg_list 在 SEND/RECEIVE 通信方式下完成双边 0-copy 非连续数据通信（SGRS）的示意图. 收、发两端都使用 sg_list 描述对应的非连续的数据分布，由网卡直接进行 0-copy 的数据传输.

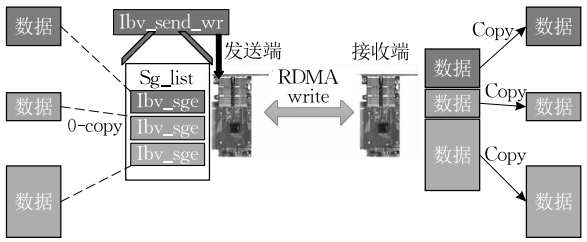


图 4　基于 sg_list 的 WRITE 方式通信示意图

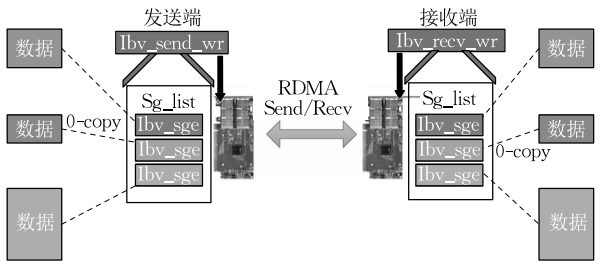


图 5　基于 sg_list 的 SEND/RECEIVE 方式通信示意图

总结：使用 sg_list 在 WRITE 方式在进行非连续数据通信时，只能实现单边 0-copy，接收端仍然需要数据拷贝；使用 sg_list 在 SEND/RECEIVE 方式下进行通信时，可以实现双边的 0-copy，但是要求发送端和接收端的 ibv_sge 数量一致，接收端的 length 值大于等于发送端的 length 值（大于的部分不会填充数据），意味着收、发端每个 ibv_sge 描述的数据长度相等才能保证通信的正确性，然而在科学计算中，很多应用收、发端数据块的数量、每个数据块的长度经常不一致，如矩阵转置、交叉数据访问等. 基于 sg_list 方式进行非连续数据通信卸载还有一个限制，每个 ibv_send_wr 或者 ibv_recv_wr 中 sg_list 所包含的 ibv_sge 数量很小，如 Mellanox 公司的 ConnectX-5 网卡支持的最大数量为 30，当非连续数据块数量超过 30 时，则需要将 sg_list 分布到多个 WR 中.

3.2　基于 UMR 的 RDMA 卸载

UMR（User-mode Memory Registration）是 Mellanox 自 ConnectX-4 起提供的功能，这一功能特性允许发送端和接收端含有数据块数量不同、每个数据块数据长度不同的数据分布（但收、发端数据总量相等），支持 SEND/RECEIVE、READ/WRITE 方式进行收、发端 0-copy 的非连续数据通信.

但是 UMR 在使用之前需要完成 Create UMR 和 Post UMR 两个步骤[10-11]，见图 6，其中 Create UMR 是指创建一个用于描述非连续地址空间的 UMR 数据结构(其中并不含有实际的地址信息，只含有 memory key)，该过程需要在 verbs 层调用 ibv_exp_create_mr，根据包含的 MR(Memory Region)的数量和允许访问的 flags 生成一个 UMR；Post UMR 是指网卡完成用户空间地址绑定的过程，用户空间地址绑定需要调用 ibv_exp_post_send 完成，用户向网卡下发的 WR 结构为 ibv_exp_send_wr，其中含有 UMR 结构体，UMR 结构体中包含众多非连续数据分布的描述，由网卡完成地址绑定.

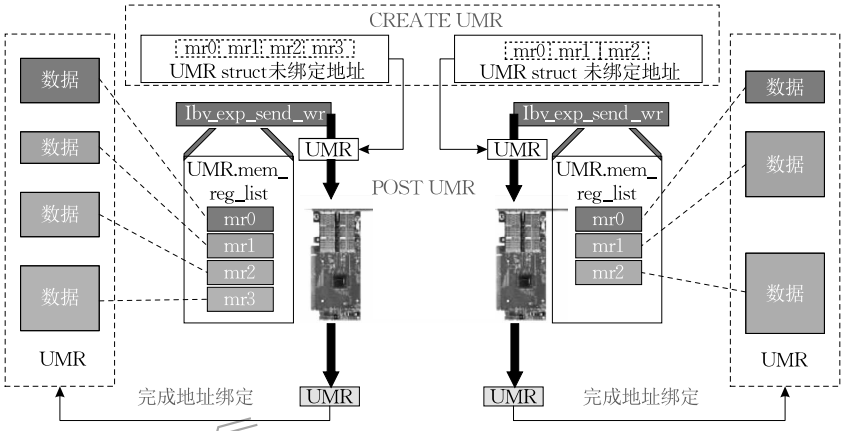


图 6 Create UMR 和 Post UMR 示意图

在完成 Create UMR 和 Post UMR 之后，用户在向网卡下发非连续数据通信的 WR 时，ibv_send_wr/ibv_recv_wr 中的 sg_list 仅需要很少项便可以描述非连续数据的分布，即把 UMR 当做一个普通连续空间的 MR 来使用. 在完成用户空间的地址绑定之后，UMR 支持图 7 所示的 SEND/RECEIVE 方式和图 8 所示的 WRITE 方式，进行非连续数据通信.

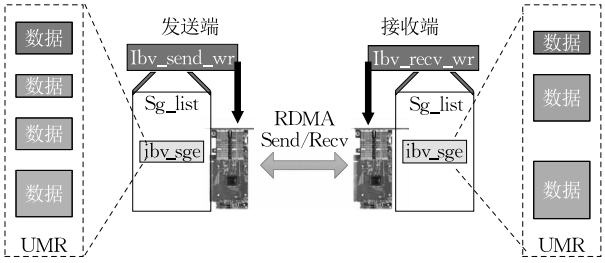


图 7 基于 UMR 的 SEND/RECEIVE 方式通信示意图

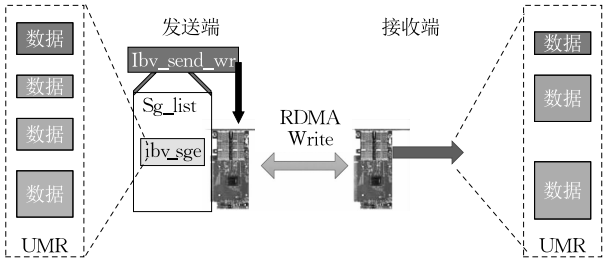


图 8 基于 UMR 的 WRITE 方式通信示意图

总结：UMR 支持 SEND/RECEIVE 方式的双边 0-copy，也支持 READ/WRITE 方式的双边 0-copy. 相比于基于 sg_list 方式，使用 UMR 有两点优势：其一，灵活性更强，UMR 允许收、发端数据块的数量不一致，每个数据块的数据长度不一致，只要收、发端描述的通信数据总量一致即可；其二，一次通信允许的非连续数据块的数量更多，Mellanox ConnectX-5 最多支持 65 536 个 MR 生成一个 UMR，一个 UMR 对应的非连续数据可以在一个 ibv_sge 中描述，这样，一个 ibv_send_wr 或者 ibv_recv_wr 最多可以完成 65 536×30 个非连续数据块的通信.

4 实验设计

上文中，对卸载和非卸载的非连续数据通信方案进行了介绍，为确定不同方案的性能差异，分析不同方案的适用情景和存在的问题，我们设计了 MPI 典型应用、微应用测试和 verbs 层测试.

4.1 非卸载方案对比

MPI DDT 为用户提供了简单方便的接口，然而在创建并使用 MPI DDT 过程中必然会引入一定的开销，不同 MPI 的实现版本也会产生影响. 下面的测试将在 2D FFT 应用[1]和典型应用的非连续数据访问模式[3-4]下，测试使用不同 MPI DDT (OpenMPI①、MVAPICH2②)与人工拷贝数据的性能.

① OpenMPI-4.0.0. https://www.open-mpi.org/doc/v4.0/
② MVAPICH2. http://mvapich.cse.ohio-state.edu/overview/

4.1.1　2D FFT

下文以两个进程对 4×4 矩阵进行 FFT 计算为例说明 2D FFT 中的非连续数据通信. 2D FFT 计算涉及到 x、y 两个维度的 FFT 计算，每个进程在完成本地数据的 x 维度 FFT 计算后，需要在进程之间交换数据，并且将数据排列成 y 维度的顺序，然后进行 y 维度的数据计算. 在计算过程中，如果使用人工打包拆包方式，则需要进行图 9 所示的非连续数据通信；如果使用 MPI DDT 方式，则需要进行图 10 所示的非连续数据通信. 在计算完成后，还要恢复成最初的数据顺序，涉及到的非连续数据通信与计算过程中的通信互为逆过程，在此不做赘述.

4.1.2　微应用测试

文献[3]中对天气模拟、量子力学、地震波传播等 7 个科学计算应用进行分析，对其中常用的非连续数据访问模式进行了总结，将其分为三类，如图 11：

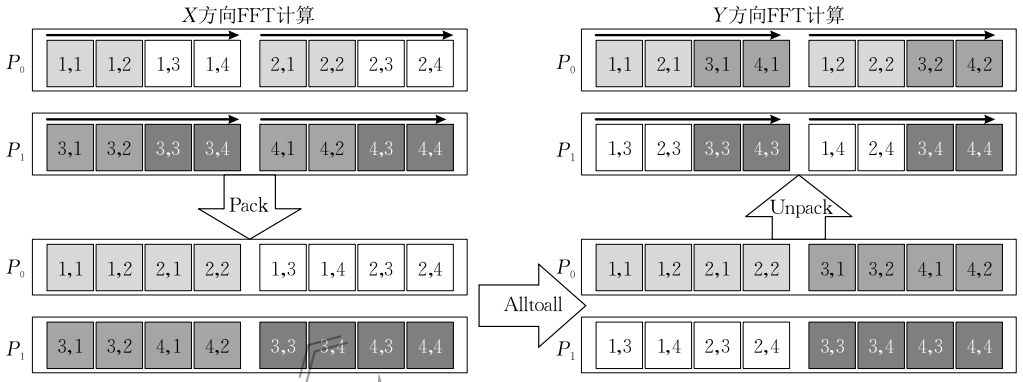


图 9　2D FFT(人工打包拆包方式)

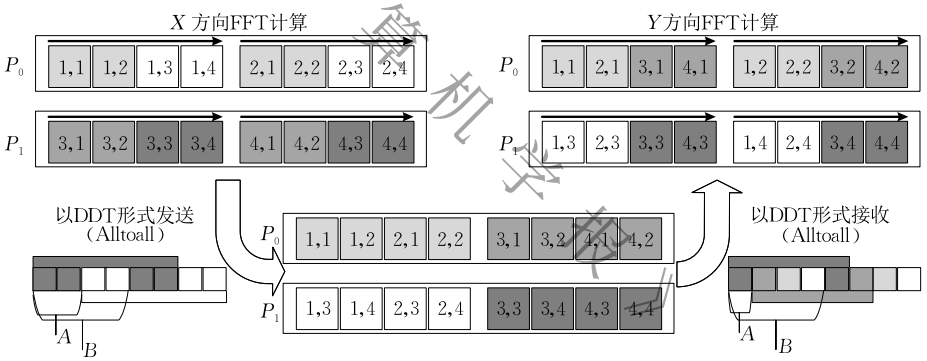
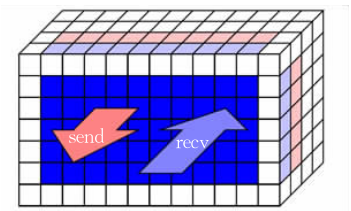
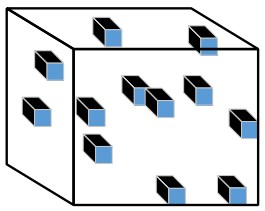


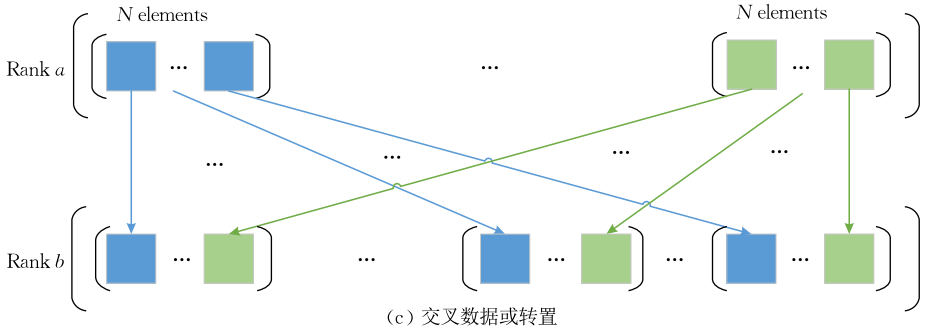
图 10　2D FFT(MPI DDT 方式)



(a) 笛卡尔数据交换



(b) 非结构化数据访问



(c) 交叉数据或转置

图 11　DDTBench 中非连续数据访问模式

(1)笛卡尔数据交换. 如 WRF(Weather Research and Forecasting)、NAS_MG(NASA Advanced Supercomputing)和 MILC(MIMD Lattice Computation);(2)非结构化数据访问. 如 LAMMPS(Molecular Dynamics Application)和 SPECFEM3D_GLOBE(Spectral-element simulation of global seismic wave propagation in 3D Earth models);(3)交叉数据或者转置. 如 FFT 和 SPECFEM3D_GLOBE. 根据典型的应用,文中总结出 Micro-Application DDT Benchmark(只有通信部分,不涉及计算),后文称 DDTBench,本文使用 DDTBench 进行测试对比 MPI DDT 与人工打包拆包性能.

4.2 RDMA 卸载方案对比

该部分直接调用 verbs API 进行 verbs 层测试,主要分析各种卸载方案的性能. 为说明卸载方式的性能特点,会以全拷贝的方式作为参考,进行对比分析. 对比的方案包括(1)双边人工打包拆包的全拷贝方式;(2)SGRS,即基于 sg_list 的 SEND/RECEIVE 方式实现通信双边 0-copy;(3)基于 UMR 的 SEND/RECEIVE 方式实现通信双边 0-copy;(4)基于 UMR 的 WRITE 方式实现通信双边 0-copy.

通信的数据分布包含两大类:(1)发送端和接收端的非连续数据块的数量相等,且对应的数据块中包含的数据量相等,下文称为对称的非连续数据分布;(2)发送端和接收端的非连续数据块数量不等,数据块中包含的数据量也不等,但是每次通信收、发的数据总量相等,下文称为非对称的非连续数据分布.

测试过程中会以数据块的数量和每个数据块中包含的数据量为变量,测试不同消息大小下各种通信方式的性能. 当数据块数量超过 30(ConnectX-5 dev_attr. max_sge)时,在不使用 UMR 的情况下,基于 sg_list 方式需要多个 WR 完成非连续数据通信,而使用 UMR 可以用一个 WR 完成非连续数据通信. 文中特意设置多数据块通信测试,对比分析 UMR 和 sg_list 的性能.

为测试 RDMA 卸载方式在实际应用中的性能,本文以 2D stencil 通信和 DDTBench 中 NAS_mg_z 为例,利用 RDMA 卸载其中的非连续数据通信,测试对比基于 sg_list 卸载、基于 UMR 卸载和人工打包拆包的性能.

5 实验结果及分析

5.1 实验测试平台

本文实验测试平台为 2 个节点,每个节点的软硬件配置如表 2 所示.

表 2 每个节点配置信息

网卡:Mellanox ConnectX-5
MPI 版本:mvapich2-2.3.1、openmpi-4.0.0
CPU:72 Intel Xeon E5-2699 v3 2.30 GHz CPUs L3 缓存:45 MB
操作系统版本:SUSE Linux Enterprise Server 11 SP3 内核版本:3.0.76-0.11 HCA 固件版本:12.20.1010 软件环境:MLNX_OFED_LINUX-4.4-2.0.7.0
RDMA 网络:RoCEv2

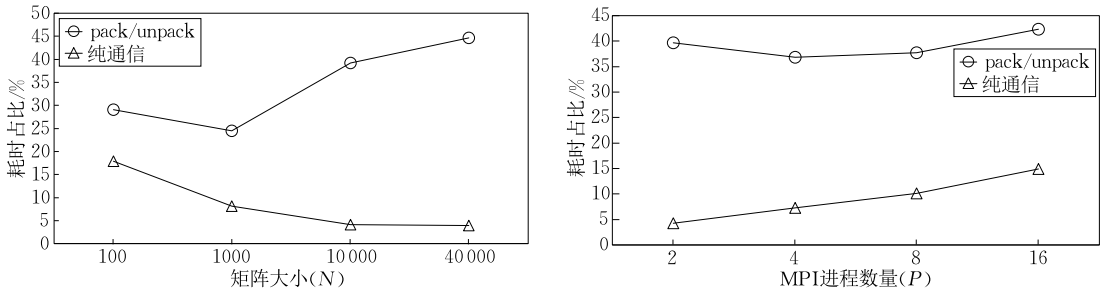
5.2 非卸载方案测试结果及分析

由于环境限制,真实测试中可以使用的 CPU 数量有限,因此在该部分测试中所有 MPI 进程均采用绑定到 CPU 核的方式,以保证测试性能数据的准确性. 该测试意在体现人工打包拆包与不同 MPI DDT 在微基准测试、应用测试中的性能差异,以及明确人工拷贝在整个通信过程中的开销.

5.2.1 2D FFT

为准确分析数据拷贝和通信的开销关系,我们测试了人工打包拆包(pack/unpack)和纯通信在整个 2D FFT 计算中的开销占比. 图 12 为 MVAPICH2 下的测试结果,耗时占比表示人工打包拆包(pack/unpack)的时间或者纯数据传输(纯通信)的时间占完成整个 2D FFT 计算时间的百分比. 图 12(a)以矩阵大小为变量,每个节点使用 1 个进程,从图中可以发现,当数据规模较小时,由于内存分布和偶然性打包拆包的占比会出现小幅波动,随着数据量进一步增大,打包拆包的开销占比逐渐增大,而纯通信的开销占比逐渐减少,二维矩阵宽度 N(矩阵为 $N\times N$ 方阵)达到 40000 时,打包拆包开销接近 45%;图 12(b)以 MPI 进程数量为变量,矩阵宽度固定为 10000 个元素,由图可见,随着进程数量增加,进行通信(Alltoall)的开销占比线性增加,打包拆包开销浮动在 40%左右,且有增加趋势.

综合图 12(a)、(b)可知,与纯通信相比,打包拆包开销是影响性能的重要方面,且受数据量影响巨大. 可以分析,当计算任务涉及到的数据量变大,需



(a) 2个进程下打包拆包与纯通信开销对比　　(b) 不同进程数量下打包拆包与纯通信开销对比

图 12　2D FFT 中打包拆包与纯通信的开销对比

要有更多的进程参与到任务中时，所有进程之间进行 alltoall 通信的开销会进一步增大，CPU 为通信进行打包拆包的开销也将进一步增加.

为比较人工打包拆包与不同 MPI 实现方式的 DDT 的性能，我们进行了如下测试，该测试使用 2 个节点，每个节点 8 个进程，二维矩阵宽度 N（矩阵为 $N\times N$ 方阵）为 40 000 个元素，每个元素包含实部和虚部两个 double 类型数据，No DDT 表示人工打包拆包数据方式，send DDT 表示仅使用发送端 MPI DDT，recv DDT 表示仅使用接收端 MPI DDT，S&R DDT 表示收、发端均使用 MPI DDT. 图 13 为 OpenMPI 和 MVAPICH2 测试结果

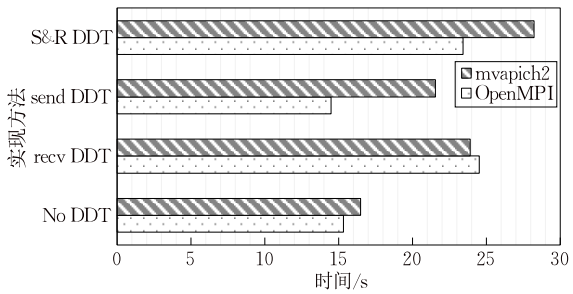


图 13　2D FFT 中 OpenMPI DDT、MVAPICH2 DDT 与人工打包拆包效果对比

从图 13 可见：(1) 不同的 MPI 库的 DDT 性能差异较大，这与不同 MPI 的 DDT 组件的实现方式不同有关；(2) 使用 MPI DDT 后对应用 (2D FFT) 的性能影响很大，我们可以看到“No DDT”即人工打包拆包的性能最好，而对于 OpenMPI 和 MVAPICH2 在接收端使用 DDT 会导致性能明显的下降 (即 S&R DDT 和 recv DDT 性能比 send DDT 更差)，这与 2D FFT 中接收端的数据分布有关，接收端需要将来自每个其他节点的数据以元素为单位放置到离散的位置，DDT 描述复杂，开销变大.

5.2.2　DDTBench

对 DDTBench 进行测试的结果和分析如下.

图 14 为两节点，每个节点 8 个进程情况下，使用 MVAPICH2，测试部分 DDTBench 中 MPI DDT 和人工打包拆包开销的实验结果. 其中，横轴单位为 KBytes，纵轴 Packing Overhead 为打包拆包开销占比，manual 表示人工打包拆包数据，mpi_pack_ddt 表示使用 MPI 接口进行数据打包拆包. 由图可见，MPI 提供的接口相比于人工打包拆包而言，占用更大的开销，尤其在 SPECFEM3D_cm (非结构化数据访问) 测试下，使用 MPI 接口的开销高达 47%.

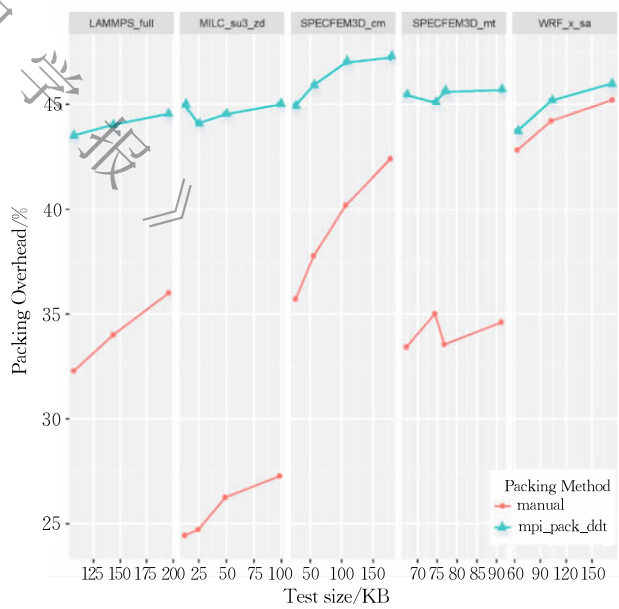


图 14　人工打包拆包和使用 MPI_Pack/Unpack 开销占比

图 15 为两节点，每个节点 8 个进程情况下，使用 MVAPICH2，测试部分 DDTBench 通信带宽的实验结果. 其中 mpi_ddt 表示直接使用 MPI DDT 接口创建新的数据类型，mpi_pack_ddt 表示使用 MPI_Pack/Unpack 接口打包拆包数据，reference 表示使用 ping-pong 方式获得的参考带宽. 从图中可见，使用 MPI 接口的带宽均不及人工打包拆包数据.

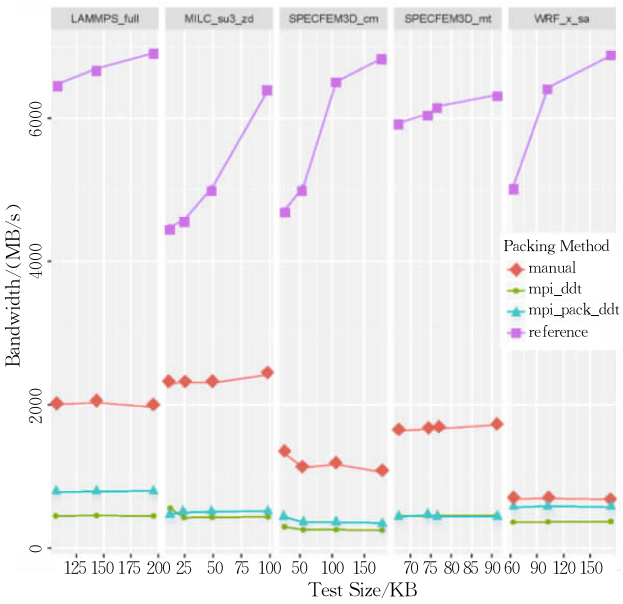


图 15 人工打包拆包和 MPI DDT 带宽

OpenMPI 的测试效果与 MVAPICH2 相似，对于 DDTBench 中的多数测试，使用 MPI DDT 的效果不及人工打包拆包的性能，只有当数据分布中存在大量的 x 维连续数据时（如 NAS_LU_x——进行 x 维度上的多个连续数据通信），如图 16，MPI DDT 会表现出略微的优势，因为此时，MPI DDT 描述简单，MPI DDT 开销很小.

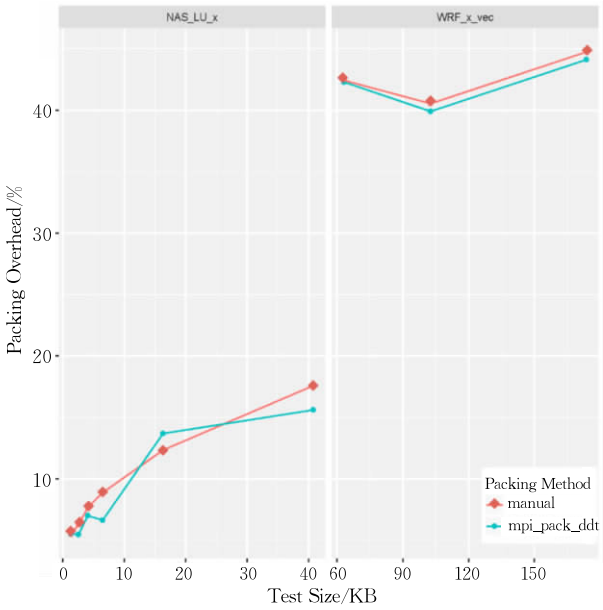


图 16 人工打包拆包和 MPI_Pack/Unpack 开销占比

在真实的应用中，为充分利用 CPU 计算资源的同时，减少通信在整个过程中的开销占比，并发进程的数量会随着数据规模的增大而线性的增加. 可以分析，当数据规模和进程规模进一步增大时，对于每个进程而言，需要处理的数据量相对稳定，这就意味着每个进程的计算量相对稳定，使用 MPI DDT 或者人工打包拆包方式进行通信的数据量也相对稳定，因此对于 MPI DDT 和人工打包拆包的性能评估并不会因为规模的变化而产生较大的影响.

5.2.3 小结

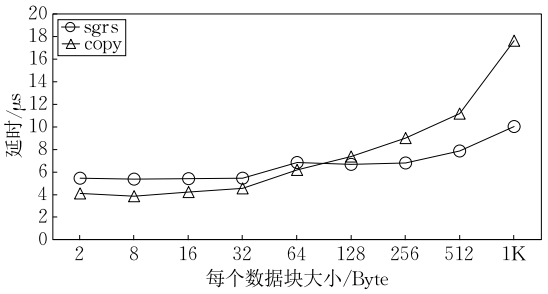
通过测试发现，人工打包拆包方式进行非连续数据通信，虽然繁琐并且会占用较大的通信开销，但是与使用简洁方便的 MPI DDT 相比，多数情况下会优于 MPI DDT. MPI DDT 在进行数据分布较为复杂的通信时，如 SPECFEM3D_cm（非结构化数据访问），表现远不及人工打包拆包. 因为，当非连续数据分布复杂时，MPI DDT 描述复杂，MPI 库对 DDT 进行存储、匹配处理的开销较大. MPI DDT 在进行数据分布较为简单的通信时，如 NAS_LU_x（大量连续数据分布），表现略优于人工打包拆包. 因为，对于简单的数据分布，DDT 描述简单，MPI 库对 DDT 存储、匹配的开销很小，经过 MPI 库处理之后的数据拷贝方法优于直接人工拷贝. 所以，MPI DDT 仅适用于非连续数据分布简单的情况. 无论人工打包拆包还是 MPI DDT，使用非卸载方法进行非连续数据通信，数据拷贝的开销巨大，导致通信性能下降，进行 0-copy 卸载很有必要.

5.3 RDMA 卸载方案测试结果及分析

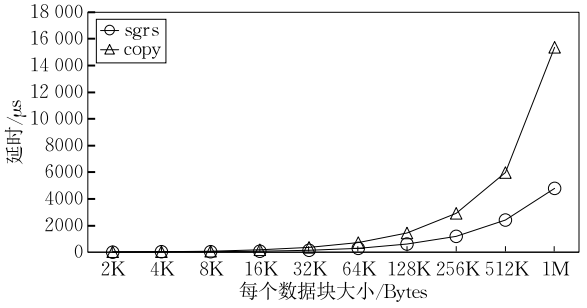
5.3.1 人工打包拆包与 SGRS 对比分析

首先，对基于 SEND/RECEIVE 方式的 sg_list（SGRS）和人工打包拆包（后文的图中均使用 copy 表示）进行测试，初步分析使用 0-copy 方式与人工打包拆包方式的性能与数据量的关系. 实验中，采用点对点测试方式，收、发端采用对称的非连续数据分布. 数据块数量为 30（ConnectX-5 dev_attr. max_sge），每个数据块之间的间隔为 45 MB（CPU LLC），每个数据块大小为 2 Bytes～1 MBytes. 图 17（a）为小消息（每个数据块大小为 2 Bytes～1 KBytes）的延时结果，图 17（b）为大消息（每个数据块大小为 2 KBytes～1 MBytes）的延时结果，图 18 为带宽结果.

从图 17（a）中发现：（1）当每个数据块大小小于 128 Bytes 时，人工打包拆包方式优于基于 sg_list 的 0-copy 方式；（2）当每个数据块大小大于 128 Bytes 时，0-copy 方式优于人工打包拆包方式. 由图 17（b）和图 18 可知，随着数据量进一步增加，0-copy 方式在延时和带宽上的优势更加明显，当每个数据块大小达到 1 M 时，人工打包拆包的延时是 0-copy 延时的 3.2 倍，带宽只有 0-copy 的 1/3，可见数据量较大时，copy 开销巨大.



(a) 小消息延时测试



(b) 大消息延时测试

图 17　延时测试(copy v. s. sgrs)

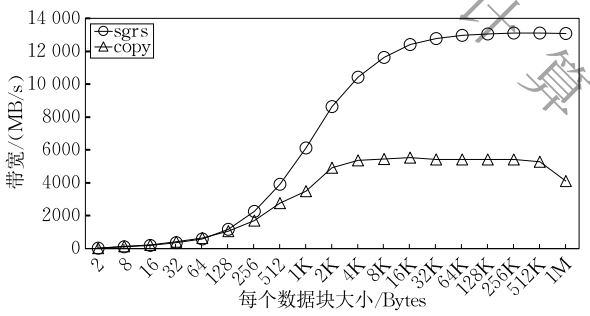
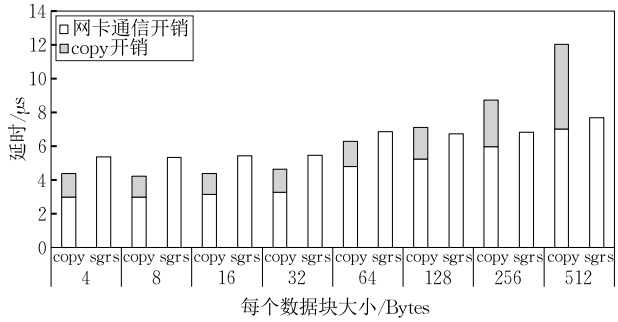


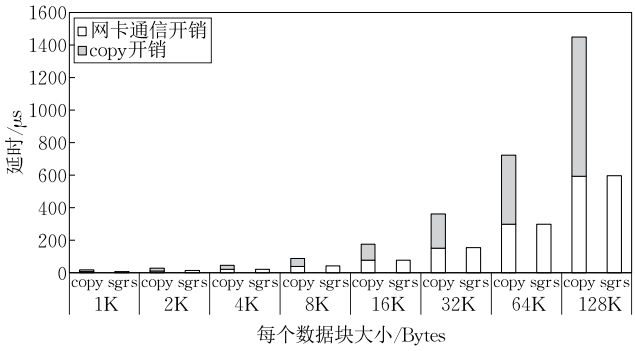
图 18　带宽测试(copy v. s. sgrs)

为进一步分析原因，我们将数据拷贝和网卡通信耗时进行了独立分析，获得图 19 的实验结果. 图 19(a)、(b)分别为数据块数量为 30 时，在不同数据块大小情况下的测试结果. 其中，copy 开销为使用人工打包拆包方式完成通信过程中数据拷贝的时间开销，网卡通信开销为网络通信的时间开销(在人工打包拆包方式中，指网卡完成一次 30×block_size 的连续数据通信的耗时；在 sgrs 中，指使用 sg_list 完成一次包含 30 个非连续数据块通信的耗时).

从图 19 可以发现：(1) copy 开销在数据块大小达到 128 Bytes 之前，增加速度缓慢，达到 256 Bytes 之后，几乎成倍增加；(2) 当数据块大小较小的时候，人工打包拆包方式的网卡通信开销和 sgrs 的耗时差大于 copy 的耗时，因此会导致小数据块情况下，人工打包拆包的方式优于网卡 0-copy 方式；(3) 当数据块大小变大时，人工打包拆包方式的网卡通信开销和 sgrs 的耗时很接近，即使用 RDMA 传输多个非连续数据块和传输等数据量的连续空间数据时间接近，而将数据拷贝到连续空间的开销很大，表现出网卡 0-copy 方式优于人工打包拆包.



(a) 小消息开销分析



(b) 大消息开销分析

图 19　开销分析(copy v. s. sgrs)

通过以上测试发现，在数据块数量一定的情况下，如果每个数据块所包含的数据量较少，使用人工打包拆包方式性能最好；如果每个数据块所包含的数据量较大，使用 RDMA 0-copy 方式性能最好.

5.3.2　UMR 与 sg_list 性能分析

在这部分测试中，将对人工打包拆包(即 copy)、sgrs(基于 sg_list 的 SEND/RECEIVE 方式)、umr_sr(基于 UMR 的 SEND/RECEIVE 方式)和 umr_w(基于 UMR 的 WRITE 方式)性能进行对比.

在进行该部分测试之前，本文先对 Create UMR 和 Post UMR 的开销进行测试，图 20 为每个 UMR 包含不同 MR 数量的情况下，Create UMR 和 Post UMR 的时间开销.

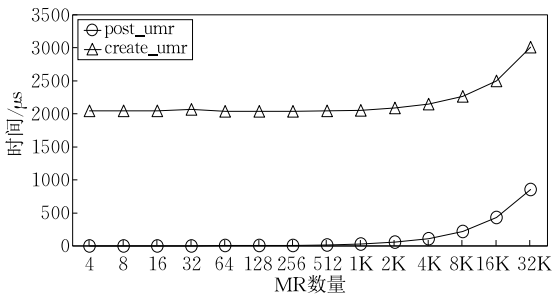


图 20　Create UMR、Post UMR 开销

从图 20 可见，Create UMR 开销巨大，Post UMR 开销较小，随着 MR 数量的增加，Create UMR 和 Post UMR 时间均有所增加，但是在 MR 数量达到 2048 之前增加缓慢，可将其视为固定开销. 在具体的应用中，如在 MPI 实现中，可以通过创建 UMR Pool 和 UMR Cache 机制掩盖 UMR 创建和地址绑定的开销[11]，本文后面的测试重点关注通信中的时间开销，剔除 UMR 创建部分的开销.

(1) 收、发端对称的非连续数据分布测试

收、发端非连续数据的数据块数量均为 16，数据块之间的间隔为 45 MB(CPU LLC)，每个数据块大小为 4 Bytes～4 KBytes，进行单向通信测试. 图 21 为延时测试结果.

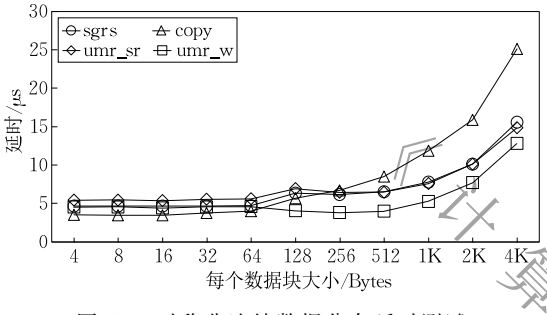


图 21 对称非连续数据分布延时测试

由图 21 中可见：

1) 当每个数据块大小小于 128 Bytes 时，人工打包拆包方式性能最优，与 5.3.1 节中测试的结论相符；

2) 在数据块数量一定的情况下，基于 SEND/RECEIVE 方式使用 sg_list 直接描述非连续数据分布(sgrs)的性能与基于 SEND/RECEIVE 方式使用 UMR 描述数据分布(umr_sr)的性能基本相同；

3) 当每个数据块大小大于 128 Bytes 时，基于 WRITE 方式使用 UMR 描述数据分布(umr_w)的性能最优. 因为，UMR WRITE 方式不需要接收端下发 WR 和 poll_cq，减少了开销.

(2) 收、发端非对称的非连续数据分布测试

当收、发端数据分布非对称时，UMR 由于灵活性的优势可以很好的支持，而 sg_list 则需要按照数据块最多的一端进行数据分布描述. 本小节测试中，发送端数据块的数量是接收端数据块数量的四倍，发送端每个数据块大小是接收端每个数据块大小的四分之一，数据块之间的间隔为 45 MB(CPU LLC)，进行单向通信，分别测试人工打包拆包(copy)、umr_sr/umr_w(基于 UMR 的 SEND/RECEIVE 方式和 WRITE 方式)和 sgrs(基于 sg_list 的 SEND/RECEIVE 方式)的延时性能. 图 22 为延时测试结果，横坐标轴为发送端每个数据块的大小.

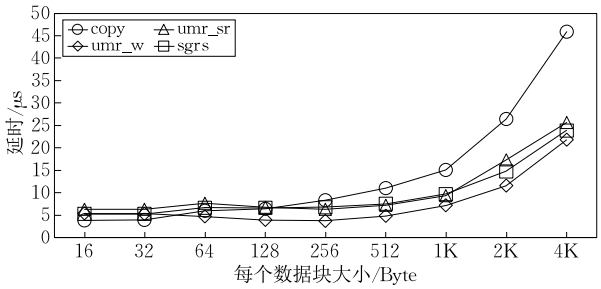
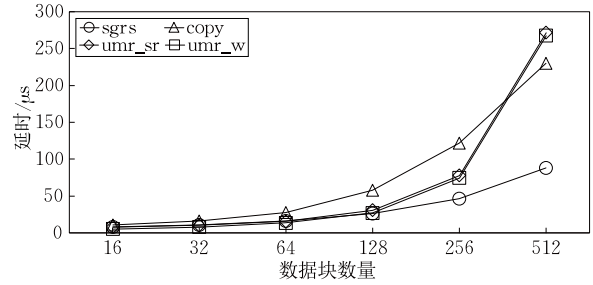


图 22 非对称非连续数据分布延时测试

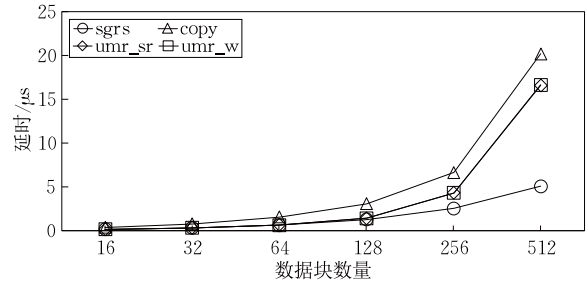
由图 22 中可见，当数据为非对称的非连续数据分布时，测试结果与对称的非连续数据分布有相似之处. 即每个数据块较小时，人工打包拆包最优，数据块变大后，UMR WRITE 方式最优. 但是 UMR WRITE 的优势只有 3 μs～5 μs，并不是很明显. 可见，当数据分布较复杂时，相对于 sg_list 直接进行数据描述，使用 UMR 在灵活性上有一定的优势，在性能方面，可以利用 WRITE 方式减少接收端的开销，但是延时上的优势并不明显.

(3) 多数据块测试

Mellanox ConnectX-5 网卡每个 ibv_send_wr 支持的最大 ibv_sge 数量为 30，而每个 UMR 支持的 MR 数量最大为 65 536. 因此，当数据块的数量很大时，使用 sg_list 方式需要下发多个 WR，而使用 UMR 只需要一个 WR 即可完成. 本小节测试固定数据块的大小，在数据块数量逐渐增大的情况下，对比了基于 SEND/RECEIVE 方式的 sg_list(sgrs)、人工打包拆包(copy)、umr_sr 和 umr_w 的延时性能. 图 23(a)、(b)分别是每个数据块大小为 1 KBytes 和



(a) 不同数据块数量的延时测试(数据块大小为1 KBytes)



(b) 不同数据块数量的延时测试(数据块大小为64 KBytes)

图 23 不同数据块数量的延时测试

64 KBytes，数据块数量为 16～512 时，copy、sgrs、umr_sr 和 umr_w 的延时测试结果.

由图 23 可知：

1）当数据块数量较小时，基于 sg_list 直接描述多个数据块，用多个 WR 完成非连续数据通信的性能与使用 UMR 描述多个数据块，只用一个 WR 完成非连续数据通信的性能接近；

2）当数据块数量较大时，基于 UMR 的性能会急剧下降，在 block_size 为 1 KBytes 时，UMR 的性能甚至要比人工打包拆包性能还要差.

所以，虽然使用 UMR 能够在一个 WR 中完成大量数据块的通信操作，但是 UMR 仅适用于数据块数量较少的情况，在数据块数量较大的情况下，UMR 的性能会急剧下降. 因为，使用 UMR 的 memory key 查询内存翻译表（Memory Translation Table，MTT[15]）时，得到的是一个新的 MTT 表，总共需要两级查询才能找到所有数据块的虚实地址映射关系，数据块较多时，MTT 表项较多，查询时间变长；使用 sg_list 直接描述时，每个 WR 中 sg_list 容纳的数据块较少，且只需要一级查询就能找到虚实地址的映射关系，查询时间短，能够及时进行数据传输.

（4）微应用测试

为说明以上测试结果以及分析的准确性，我们设置了 2D stencil 通信和 3D 矩阵面传输实验.

1）2D Stencil 通信

在该部分测试中，stencil point 数量为 5～57，stencil point 数量与数据块数量的关系为

(stencil point number+1)/2=block_num.

该测试在使用人工打包拆包（copy）、sgrs（基于 sg_list 的 SEND/RECEIVE 方式）和 umr_w（基于 UMR 的 WRITE 方式）的情况下，抽取出 2D stencil 通信时间开销对比性能. 图 24 为测试结果，时间为平均每次单向 stencil 通信的耗时.

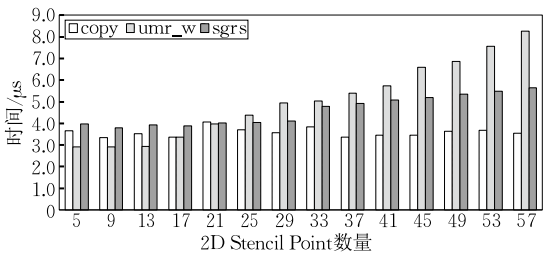


图 24　2D stencil 通信延时测试

由图 24 可知：① 当 stencil point 数量小于 21 时，umr_w 性能最优，因为此时数据块数量较少，查询 MTT 开销少，且 WRITE 方式减少了接收端开销；② 当 stencil point 数量大于 21 时，copy 方式最优，因为 stencil 的数据分布中每个 point 的数据量很少，当 point 数量变大时，数据拷贝的开销小于使用网卡 0-copy 传输多个 point 的开销（与图 19（a）中情况相似）. 使用 UMR 的性能在 point 数量变大后变差，因为查询 MTT 的开销逐渐变大，影响 UMR 的性能.

2）3D 矩阵面通信

3D 矩阵面传输，通信模式与 NAS_mg_z 一致，该实验中，矩阵面为方阵，单行元素个数为 4～128，矩阵面大小（matrix face size）与数据块数量（block_num）的关系为

matrix face size−2=block_num.

在使用人工打包拆包（copy）、sgrs 和 umr_w 的情况下，抽取出通信时间开销来对比性能. 图 25 为测试结果，时间表示平均每次单向矩阵面通信的耗时. 由图 25 可知：① 当矩阵小于 32×32 时，umr_w 性能最优，因为此时数据块数量较少，查询 MTT 开销少，且 WRITE 方式减少了接收端开销；② 当矩阵大于 32×32 时，sgrs 方式最优，因为，随着矩阵变大每个数据块的数据量变大，人工打包拆包方式拷贝开销较大，sgrs 方式优于人工拷贝方式，而 UMR 因为查询 MTT 的开销逐渐变大，导致性能变差. 所以，基于 sg_list 直接描述数据分布的 sgrs 方式性能最优.

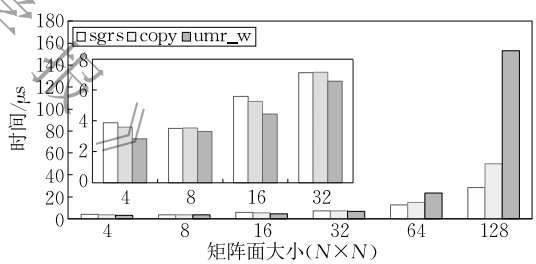


图 25　3D 矩阵面通信延时测试

5.3.3　小结

根据以上测试和实验验证，我们总结出表 3，用以说明在不同数据块大小（block_size）和数据块数量（block_num）情况下，非连续数据通信方式的最优选择，括号中的数据仅为本文测试平台经过多组数据测试总结出来的参考数据.

表 3　非连续数据通信方式的最优选择表

block_size	block_num	
	小（小于 60±30）	大（大于 60±30）
小（小于 128 B）	UMR_WRITE、人工打包拆包	人工打包拆包
大（大于 128 B）	UMR_WRITE	SGRS

经过 5.3.1 节的测试得出结论：（1）对非连续数据分布，当数据块大小（block_size）较小的时候，

人工拷贝数据开销相对于网卡多次 DMA 操作开销较小,人工打包拆包方式最优;(2)在数据块大小(block_size)到达 128 Bytes 之后,拷贝数据的开销变大,RDMA 卸载的 0-copy 方式表现出优势,随着数据量的增加,0-copy 方式的优势更加明显.该平台内存数据拷贝开销和网卡 DMA 开销的临界数据块大小约为 128 Bytes.

经过 5.3.2 节对 UMR 和 sg_list 的对比测试,得出结论:(1)UMR WRITE 方式可以在数据块数量较少时体现出优于 sg_list 的性能优势,UMR SEND/RECEIVE 方式与 sg_list SEND/RECEIVE 方式性能接近;(2)当数据块数量较大时,UMR 虽然可以满足在一个 WR 中完成大量数据块(最大 65 536×30)的通信,但是 UMR 查询 MMT 开销变大,性能会急剧下降,明显不及 sg_list 性能.此处 60±30 是在不同数据块大小下,测试不同数据块数量对不同实现方式选择的临界点的浮动范围,是该平台下多组测试的经验值.附录 2 中表 4 为部分测试数据.

6 总结与展望

本文对已有的非连续数据通信的实现方法进行了总结,对人工打包拆包、MPI DDT、基于 sg_list 的 RDMA 卸载和基于 UMR 的 RDMA 卸载进行了详细的对比测试.经过 MPI 层、verbs 层和微应用层的测试分析,我们发现了各种实现方式适用的情况和存在的问题.

(1)MPI DDT 虽然使用方便灵活,但在多数情况下不及人工打包拆包方式,尤其在数据分布复杂的情况下,MPI DDT 机制开销变大;

(2)关于 RDMA 0-copy 方式和人工打包拆包的适用情况,本文总结出表 3 供参考;

(3)使用 UMR 描述数据分布在灵活性上优于直接使用 sg_list 描述数据分布,更适合支撑 MPI DDT 的卸载,在性能方面,UMR WRITE 方式因节省接收端开销,有一定的优势,但是当数据块数量较大时,UMR 查询 MTT 的开销变大,UMR 性能变差.

目前,有相关研究对集合通信(如 Alltoall)进行 RDMA 卸载[16].在 GPU 大量使用的场景下,如何提升 GPU 之间非连续数据通信的性能也成为了一个研究的方向[17-18].使用智能网卡加速非连续数据通信也出现在最新的研究中[19].关于 RDMA 卸载非连续数据通信仍有很多工作可以进一步研究:(1)根据表 3 优化 MPI DDT 的实现机制;(2)进一步分析 MTT 的实现机制和存在的问题以优化一个 UMR 中包含多数据块时的性能;(3)减少 Create UMR 的巨大开销;(4)对集合通信提供 RDMA 卸载的方案;(5)将非连续数据通信的相关经验应用到 GPU 场景中.

致 谢 感谢华为技术有限公司北研所 2012 罗素实验室的合作和支持!

参 考 文 献

[1] Hoefler T, Gottlieb S. Parallel zero-copy algorithms for fast Fourier transform and conjugate gradient using MPI datatypes//Proceedings of the European MPI Users' Group Meeting. Stuttgart, Germany, 2010: 132-141

[2] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 3.1, 2015

[3] Schneider T, Gerstenberger R, Hoefler T. Micro-applications for communication data access patterns and MPI datatypes//Proceedings of the European MPI Users' Group Meeting. Vienna, Austria, 2012: 121-131

[4] Gropp W, Hoefler T, Thakur R, Träff J L. Performance expectations and guidelines for MPI derived datatypes//Proceedings of the European MPI Users' Group Meeting. Santorini, Greece, 2011: 150-159

[5] Xiong Q, Bangalore P V, Skjellum A, Herbordt M. MPI derived datatypes: Performance and portability issues//Proceedings of the European MPI Users' Group Meeting. Barcelona, Spain, 2018: 1-10

[6] Träff J L. Optimal MPI datatype Normalization for vector and index-block types//Proceedings of the European MPI Users' Group Meeting. Kyoto, Japan, 2014: 33-38

[7] Ganian R, Kalany M, Szeider S, Träff J L. Polynomial-time construction of optimal MPI derived datatype trees//Proceedings of the 2016 International Parallel and Distributed Processing Symposium (IPDPS). Chicago, USA, 2016: 638-647

[8] Wu J, Wyckoff P, Panda D. High performance implementation of MPI derived datatype communication over InfiniBand//Proceedings of the 2004 International Parallel and Distributed Processing Symposium (IPDPS). Santa Fe, USA, 2004: 1-14

[9] Santhanaraman G, Wu J, Panda D K. Zero-copy MPI derived datatype communication over InfiniBand//Proceedings of the Recent Advances in Parallel Virtual Machine and Message Passing Interface. Budapest, Hungary, 2004: 47-56

[10] Mellanox Technologies. RDMA Aware Networks Programming User Manual Rev 1.7. Sunnyvale, USA, 2015

[11] Li M, Subramoni H, Hamidouche K, et al. High performance MPI datatype support with user-mode memory registration: Challenges, designs, and benefits//Proceedings of the 2015 International Conference on Cluster Computing (CLUSTER). Chicago, USA, 2015: 226-235

[12] Carpen-Amarie A, Hunold S, Träff J L. On expected and observed communication performance with MPI derived datatypes//Proceedings of the European MPI Users' Group Meeting. Edinburgh, United Kingdom, 2016: 108-120

[13] Eijkhout V. Performance of MPI sends of non-contiguous data. arXiv:1809.10778, 2018

[14] Schneider T, Kjolstad F, Hoefler T. MPI datatype processing using runtime compilation//Proceedings of the European MPI Users' Group Meeting. Madrid, Spain, 2013: 19-24

[15] Mellanox Technologies. Mellanox Adapters Programmer's Reference Manual (PRM) Rev0.4. Sunnyvale, USA, MLNX-15-4845, 2016

[16] Gainaru A, Graham R L, Polyakov A, Shainer G. Using InfiniBand hardware gather-scatter capabilities to optimize MPI all-to-all//Proceedings of the European MPI Users' Group Meeting. Edinburgh, United Kingdom, 2016: 167-179

[17] Wu W, Bosilca G, Vandevaart R, et al. GPU-aware non-contiguous data movement in Open MPI//Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing. Kyoto, Japan, 2016: 231-242

[18] Chu C H, Hamidouche K, Venkatesh A, et al. Exploiting maximal overlap for non-contiguous data movement processing on modern GPU-enabled systems//Proceedings of the 2016 International Parallel and Distributed Processing Symposium (IPDPS). Chicago, USA, 2016: 983-992

[19] Di Girolamo S, Taranov K, Kurth A, et al. Network-accelerated non-contiguous memory transfers//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Denver, Colorado, 2019: 1-14

附录 1.

RDMA 原语及含义

名词介绍

QP：Queue Pair，每个 RDMA 连接包含一对 QP，QP 包含 RQ(Receive Queue)和 SQ(Send Queue)；

WR：Work Request，用户向网卡下发的请求；

WQ：Work Queue，QP 中的队列；

WQE：Work Queue Element，用户的 WR 转化成 WQ 中的 WQE 形式；

CQ：Complete Queue，用于通知 WQ 上处理完成的消息.

verbs API 接口

ibv_send_wr：发送数据的操作请求；

ibv_recv_wr：接收数据的操作请求；

ibv_post_recv：向网卡下发接收数据请求；

ibv_post_send：向网卡下发发送数据请求；

ibv_poll_cq：检查请求操作是否完成；

ibv_sge：wr 中的用于描述数据分布的数据结构；

sg_list：wr 中包含的 ibv_sge 的链表；

ibv_exp_create_mr：创建 UMR；

ibv_exp_post_send：向网卡下发高级请求

附录 2.

表 4　数据块数量测试数据

Block_size=256 Bytes latency/μs						
Type\Bock_num	16	32	64	128	256	512
sgrs	6.30	7.97	11.38	17.42	29.14	60.61
copy	6.37	7.94	11.46	17.35	31.30	63.24
umr_sr	6.43	8.40	14.38	34.87	138.31	655.12
umr_w	4.21	6.04	11.85	32.29	132.59	649.04
Block_size=1024 Bytes latency/μs						
Type\Bock_num	16	32	64	128	256	512
sgrs	8.18	11.19	16.41	26.23	46.23	88.04
copy	11.01	16.20	27.73	57.84	121.68	230.25
umr_sr	7.94	10.42	15.80	30.70	77.88	271.34
umr_w	5.45	8.14	13.56	27.11	74.55	267.59
Block_size=4096 Bytes latency/μs						
Type\Bock_num	16	32	64	128	256	512
sgrs	15.62	27.48	46.09	85.54	165.46	324.30
copy	24.07	46.46	95.85	195.58	384.35	765.06
umr_sr	15.54	27.66	48.70	97.30	280.22	1069.45
umr_w	13.11	23.69	45.45	93.05	276.96	1063.73
Block_size=16384 Bytes latency/μs						
Type\Bock_num	16	32	64	128	256	512
sgrs	46.85	85.30	164.14	322.39	640.92	1272.41
copy	94.59	195.92	380.18	758.42	1498.22	2991.63
umr_sr	48.39	88.70	170.68	363.48	1084.78	4194.10
umr_w	43.54	83.88	165.10	359.62	1081.30	4205.76

（续 表）

Block_size=65 536 Bytes latency/μs						
Type\Bock_num	16	32	64	128	256	512
sgrs	165.80	323.51	640.21	1273.03	2541.63	5073.39
copy	390.39	777.05	1548.12	3078.61	6657.21	20 205.41
umr_sr	166.79	325.93	648.61	1416.61	4304.16	16 615.55
umr_w	162.25	321.78	645.80	1415.69	4304.53	16 647.07

MA Xiao-Xiao, Ph. D. candidate. His research interests include high performance interconnection network and In-network computing.

LU Gang, Ph. D. His research interests include high performance interconnection network, network virtualization, operation system and distributed computation.

FU Bin-Zhang, Ph. D. His research interests include high performance, high reliability computer system interconnection network, network virtualization, congestion control algorithm and software defined network.

AN Zhong-Qi, engineer. His research interests include high performance computing and high performance network.

ZHU Hong-Rui, Ph. D. candidate. His research interests include distributed deep learning communication network, data center network, topology and traffic optimization, and network simulator research.

SHAO En, Ph. D., engineer. His major interests include computer architecture, optical interconnection and cloud computing.

WANG Zhan, associate professor. His main research interests include high performance interconnection network, distributed system architecture.

AN Xue-Jun, professor. His main research interests include computer system architecture and high performance interconnection network.

Background

This research is about the implementation methods and performance analysis of non-contiguous data communication in HPC (High Performance Computing) area. Non-contiguous data communication means that the sender transfers multiple blocks of data at discontinuous addresses to multiple discontinuous addresses of the receiver. This communication model is common in scientific computing applications, such as solution calculation, FFT calculation, fluid dynamics simulation, etc. These applications involve the transfer of matrix transpose, the transfer of submatrix of 2D, 3D and 4D matrices, unstructured data access and other non-contiguous data communication. Therefore, the communication performance of non-contiguous data has important influence on many scientific computing applications.

There are totally three kinds of methods to complement non-contiguous data communication. Each of these methods has its own merits and demerits, but no research has summarized the characteristics of them in detail before now. This paper summarizes the current three kinds of implementation methods of discontinuous data communication: (1) manual packing/unpacking data into continuous data space, and then process continuous data communication; (2) Message Passing Interface (MPI) Derived Data Type (DDT); (3) utilize RDMA (Remote Direct Memory Access) technology to realize 0-copy data transmission. In this paper, the performance of non-contiguous data communication in different ways is tested in detail. Especially, we analyze the offloading performance based on RDMA sg_list (scatter gather list) and UMR (User-mode Memory Registration) function, and we get the conclusion of the applicable situations and problems of various non-contiguous data communication modes. Finally, we prove the conclusion is correct by using two micro-application. In addition, Based on our experiences, we get the guideline of optimize non-contiguous data communication in MPI application and the potential problems of RDMA offloading methods.

This research is funded by the National Key Research and Development Program (No. 2018YFB0204400), the 13th Five-Year National Key Research and Development Program (No. 2016YFB0200205), the National Natural Science Foundation Youth Fund Project (No. 61702484), the Chinese Academy of Sciences Pioneer Class B Special Project (No. XDB24050100), and the Cooperation Project of HUAWEI Technologies Co., Ltd. (YBN2018015521). Our research group mainly focuses on high performance network communication technology.