面向非易失内存的MPI-IO接口优化

邓镇龙，陈志广
中山大学计算机学院，广东 广州 510006

摘要
在高性能计算环境中，MPI应用多个计算节点同时访问底层存储系统文件时，其I/O开销受到访问模式和外存设备性能的影响。针对MPI应用访问文件的特征，利用非易失内存高带宽、低时延、可字节寻址、数据可持久化等优势，提出面向非易失内存的MPI-IO接口优化方案；对文件数据建立分布式的缓存并维护持久性的元数据、对进程间数据传输策略进行优化，使应用可以有效管理、利用非易失内存设备，保持缓存数据一致有效。实验结果证明，所提系统为应用带来数十倍的读写性能提升。未来将进一步优化本方案的并行性。

关键词
非易失内存；MPI-IO；分布式数据缓存

中图分类号：TP311 文献标识码：A doi: 10.11959/j.issn.2096-0271.2021020

An optimization of MPI-IO interface for non-volatile memory

DENG Zhenlong, CHEN Zhiguang
School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510006, China

Abstract
In an HPC system where multiple computation nodes of an MPI application simultaneously access files in underlying storage systems, the I/O overhead is affected by the access mode and the properties of external storage devices. Based on the patterns of MPI applications to access files, an optimization for MPI-IO interface for persistent memories was introduced on high-bandwidth, low-latency, byte-addressable, data-persistent memories. By constructing distributed data cache, maintaining persistent metadata and leveraging optimizations on data movements among processes, applications were enabled to efficiently manage and utilize persistent memories with data consistency guaranteed, resulting in tens of times improvement on read/write bandwidth. Further optimizations on parallelism were set for future work.

Key words
non-volatile memory, MPI-IO, distributed data cache

1 引言

内存和外存的合理使用是大数据环境下计算机系统研究的重要内容。根据存储介质的特性设计计算机的存储架构、软件，可在经济的前提下得到更好的存储性能。长期以来，快速的动态随机存取存储器（dynamic random access memory，DRAM）设备与大容量的外存设备的组合使用使得计算机可以在较低的成本下实现高速的计算以及大容量的数据存储。但是，随着计算机处理器的快速迭代，外存设备的性能成为数据密集型应用的性能瓶颈，访问外存的I/O开销成为计算机应用开销的重要部分。在高性能计算（high performance computing，HPC）系统中，计算节点配备了高性能的处理器，但在访问共享文件系统中的文件时，其I/O性能同样受到底层存储设备的限制。如何减小I/O开销成为HPC系统设计的一个重要课题。

作为并行编程模型，消息传递接口（message passing interface，MPI）协议被广泛应用在HPC系统中，在科学研究与工程仿真中，常使用MPI将模型数据分布到不同的节点上进行计算仿真。MPI的I/O模块使用数据筛选（data sieving）以及聚合I/O（collective I/O）等优化手段[1-2]，将进程需要的大量小粒度I/O经过聚合形成少量的大粒度I/O，从而减少文件系统的小粒度随机访问，降低应用的I/O开销。但是，应用分布式地多次访问同一文件时，需要多次向文件系统请求数据，其I/O性能仍然限制了应用的整体性能。如果可以在计算节点中建立并维护文件的缓存，使计算节点可以从缓存中获取数据，则可进一步减小I/O开销。

新型存储介质的出现有助于实现这一设计。非易失内存（non-volatile memory，NVM）具有可按字节寻址、数据可持久性、容量大等特点，且其带宽、时延接近DRAM；Intel Optane DC PMM（persistent memory module）存储器是以3D Xpoint为介质的双列直插式存储模块（dual inline memory module，DIMM）接口非易失存储设备。非易失内存兼顾传统内存、外存的优点，将改变计算机的存储架构设计，有望被配备在HPC系统中的每一个计算节点上。使用非易失内存部署计算节点上的缓存，可在较低成本下获得优秀的缓存性能。本文研究如何优化MPI-IO接口，使应用可以在NVM设备上建立、维护并使用数据缓存，以充分发挥NVM设备的优秀性能，并降低应用的I/O开销。

为了实现对非易失内存的管理与利用、对文件数据缓存的管理与访问，本文设计并实现了面向非易失内存的MPI-IO接口优化（NVMPI-IO）。本文的工作主要包括：

- 修改MPI-IO接口，截取应用对底层共享文件系统的访问，并将其转化为对计算节点内或计算节点之间的非易失缓存的访问；
- 在非易失内存中建立并维护缓存数据，使计算节点之间的缓存一致且有效，使应用失效重启后可以快速地从非易失内存中恢复有效数据；
- 通过多种优化，降低维护、访问缓存的开销；
- 最后给出一个原型系统，并对其进行实验，实验表明，此系统可以有效地管理、应用非易失内存，并使MPI应用获得性能提升。

使用NVMPI-IO，MPI应用无须进行修改，即可通过MPI-IO中间件将非易失内存作为数据缓存，实现对文件缓存的分布

式访问，从而减少I/O开销，并减轻共享文件系统的负载；同时，在非易失内存中维护元数据，使程序在崩溃重启后可以快速恢复数据，并继续运行。

2 应用现状

2.1 MPI应用的文件访问模式

MPI是基于消息传递的并行编程模型，可使多个节点中的多个进程合作完成同一个计算任务，达到并行加速的目的。MPI被广泛地应用在科学研究与工程仿真中，常见的MPI实现包括Intel MPI、OpenMPI、MPICH等。

使用MPI的应用在访问文件时显示出以下访问特征。

- 多个进程同时访问同一文件的不同部分。研究物理模型或工程结构的MPI应用在进行计算前，首先需要准备模型文件，多个进程将同时利用模型文件上的数据进行计算，如每个进程读取多维矩阵的不同部分；进程间按需通信，并将计算结果写回文件的相应位置。
- MPI-IO使用聚合I/O与数据筛选技术，将多个进程需要的大量小粒度数据聚合成少量的大粒度数据，避免了小粒度的文件数据访问。
- MPI标准不对文件数据进行缓存。MPI应用在访问文件时可能对同一文件进行多次读写，且每次读写的位置可能不一致；同时，多个进程对文件的并行访问容易使节点内缓存失效；内存的空间有限，而工程模型的数据量可根据工程的精度呈指数型增大，将大量的文件数据缓存到内存可能影响计算效率；在允许直接输入输出（direct I/O）的文件系统（如XFS、Lustre）中，MPI建议使用直接输入输出，以避免操作系统的缓存。
- MPI应用应周期性地写出检查点（checkpoint）文件[3]。大型工程仿真项目可能需要多个节点同时长时间运行，若其中某个节点出现故障导致作业失败，仿真项目需要重新进行；为了避免过多的重复工作，MPI应用应周期性地输出检查点文件，若作业失败，则从最近的有效检查点开始继续计算。同时，检查点文件可用于仿真的可视化输出。写检查点文件时，进程需要暂停计算任务，或将文件数据复制一份，以避免数据不一致。
- MPI应用多进程对文件进行访问有显著的同步特性。数据库、文件系统等会在任何时间点接收来自多个客户端的数据请求，若有分布式的缓存，则需要随时保证数据的一致性；MPI应用的多个进程在访问同一文件时，多个进程同时访问数据，当这一阶段完成后，进程之间需同步进度后再进行下一阶段的访问。MPI应用多进程的同步避免了写后读等数据不一致的问题，可用于简化缓存的设计。

基于上述的MPI应用的数据访问特征，在计算节点上部署非易失缓存有利于MPI应用的性能提升。利用非易失内存容量大、带宽高、可按字节寻址等特点，在非易失内存上部署缓存层不占用高效的DRAM空间，并将缓慢的文件访问转变为高速的非易失内存访问，可提高MPI应用的性能，同时可减少底层文件系统的负载。

2.2 NVM设备的特性与应用

计算机使用容量更大的块设备（如固态硬盘（solid-state disk，SSD）和磁盘（hard disk drive，HDD）等）存储持久性的数据；使用速度更快的字节寻址设备（如DRAM内存等）存储程序运行产生的数据，包括进程数据、堆栈空间等。DRAM

与SSD/HDD的结合使得计算机可以在较低的成本下实现快速运行。非易失内存的性能介于两者之间，并同时具备两者性能的优点。在多线程访问下，Intel Optane DC PMM的顺序读写带宽最高可达约35 GB/s，约为DRAM的1/3；同时其随机读取的时延约为305 ns，仅为DRAM的3倍[4-5]。

非易失内存具备良好的综合性能，以及可持久性的特性，使多种应用得以重新被设计。Volos H等人[6]为了在非易失内存设备上创建和管理内存空间、保证数据的一致性，设计了轻量级的非易失内存编程模型Mnemosyne。Coburn J等人[7]设计并实现了非易失内存上的对象模型NV-Heaps。基于非易失内存可按字节寻址、容量大、数据可持久化的特点，文件系统[8-10]可被部署在非易失内存之上。Xu J等人[11]设计了DRAM与非易失内存混合的日志结构文件系统NOVA。NOVA-Fortis[12]在NOVA文件系统的基础上加入了容错措施，如快照、副本、检验和、RAID-4奇偶校验等。Path Hashing[13]、Level Hashing[14]索引结构在逻辑上采用树形结构，物理上哈希表为一个数组；通过优化哈希冲突时的插入策略以及插入失败后的重哈希策略，达到减少随机写的目的。在分布式场景中，Octopus[15]是基于远程直接数据存取（remote direct memory access，RDMA）的分布式NVM文件系统，此文件系统中包括客户端与数据服务器；其中所有文件的元数据分布在不同的数据服务器上，同一文件的元数据及其数据块分布在同一数据服务器中。与Octopus不同，Orion文件系统[16]部署了元数据服务器、客户端与数据存储。此外，NVMD[17]利用NVM的密度高、空间大、性能高等特点，结合RDMA技术，重构了MapReduce框架，可加速基于有向无环图（directed acyclic graph，DAG）的MR作业的执行。

但是，以上非易失文件系统、哈希索引等都将长期占用非易失内存设备，若其中低热度的数据长期实体化在非易失内存上，则非易失内存设备的优秀性能不能得到充分的利用。

3 系统设计

3.1 设计目的

面向非易失内存的MPI-IO接口优化被应用在HPC系统中，其修改MPI-IO模块接口以管理非易失内存的空间及访问形式。NVMPI-IO在MPI运行时初始化，并向操作系统申请NVM空间，当作业结束后释放NVM资源，使NVM资源可供其他应用使用。NVMPI-IO采用简单而有效的方法获得了以下优点。

- NVM设备的非独占使用：NVMPI-IO在运行时，只按需占用NVM的部分空间，其他应用仍可以使用该设备进行其他作业。作业结束后，NVMPI-IO立即释放NVM资源，使其可以被更高效地利用。
- 数据一致性：NVMPI-IO随MPI程序的运行而运行，每个MPI进程维护文件的部分缓存数据及相应的元数据；通过维护元数据保证缓存数据的一致性。
- 后台写回：MPI应用周期性地写出checkpoint文件，引起大量的数据写回。NVMPI-IO允许数据的后台写回，可在写回过程同时进行计算任务。
- MPI应用的快速重启：若MPI应用中某个进程失效引起整个程序的崩溃，在NVM设备上仍然存在有效的数据；使用NVMPI-IO，MPI应用重启后，可以在NVM中快速恢复数据，实现快速的重启。

● 高可移植性：MPI可以在不同计算机架构、操作系统上正常工作；NVMPI-IO继承了MPI的高可移植性。同时，NVMPI-IO不修改提供给上层应用的应用程序接口（application programming interface，API），现有的MPI应用不需要做任何修改即可在此系统上运行。

NVMPI-IO部署在HPC系统中，需对HPC系统的软硬件做相应的调整。

3.2 软硬件组织

NVMPI-IO在传统HPC集群组织中引入NVM设备。在传统HPC集群中，计算节点通过网络与共享的存储系统相连，计算节点从存储系统中获得数据，并对数据进行加工处理；同时，计算节点之间通过TCP/IP网络或RDMA技术进行通信。如**图1（a）**所示，NVMPI-IO在每个计算节点中部署NVM设备，同时保持共享存储系统的设计不变；节点间通信沿用传统HPC系统中的网络传输，若配备了RDMA网络，则可以使用RDMA直接访问其他节点的非易失内存。

NVMPI-IO为MPI应用服务，同时也利用了MPI定义的通信过程。一个典型的HPC应用包含多个MPI进程，其中每个进程使用MPI定义的接口与其他进程进行消息传递，使用MPI-IO模块对底层存储系统进行数据访问。本系统保留MPI-IO的聚合I/O与数据筛选优化，这些优化将上层应用需要的小粒度I/O整合成大粒度的I/O，避免了数据的小粒度随机访问。如**图1（b）**所示，本系统部署在MPI-IO中间件中，截取MPI-IO对文件的系统调用，并按需转发为非易失内存的数据维护、访问。NVMPI-IO在部署时，首先需要在NVM设备上挂载可以直接访问（direct access，DAX）的文件系统。在NVM设备上挂载DAX文件系统可使应用直接通过指针访问NVM上的数据。具体地，进程使用内存映射（memory map）技术将文件系统中的文件映射到应用的进程空间中，进程可以通过指针直接修改NVM，而不是修改操作系统的内核缓存（kernel cache）。此外，NVMPI-IO使用MPI定义的消息传递API与其他节点进行通信，使用POSIX API与底层文件系统进行数据的维护和访问。

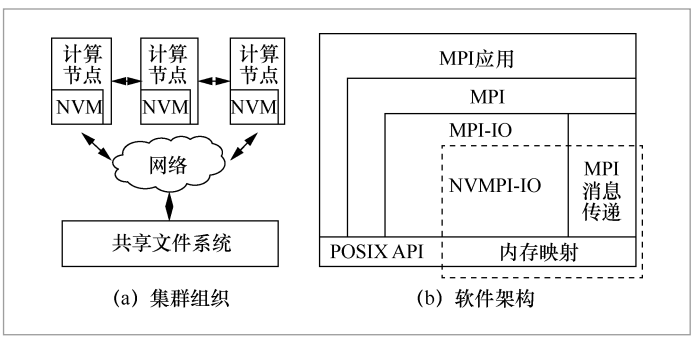


图1 NVMPI-IO的集群组织和软件架构

4 设计与实现

为了优化MPI-IO对文件的访问，并管理和利用NVM设备，本系统将文件数据缓存到NVM设备上，并允许进程通过内存映射的指针对NVM上的数据进行直接访问。NVMPI-IO将管理NVM上的文件数据及其元数据，并通过维护元数据来保证数据的正确访问。

4.1 数据布局

如**图2**所示，NVMPI-IO将NVM设备的空间在逻辑上分为两部分：数据块池和对象池。数据块池被划分为大小相同的块，用于缓存文件数据，如一个模型文件

中多维矩阵的一部分；对象池用于存储大小各异的各种数据结构，用户可以申请空间存放持久性的数据，包括用于管理数据块池及缓存文件的数据等。

本系统中，每个进程维护一个数据块池及对象池。其中，各个进程的数据块池在逻辑上是一个整体，在配备了RDMA网络的环境中，进程可以通过进程编号、数据块编号直接访问其他进程的数据块池。

在初始化时，每个进程首先向操作系统申请NVM的空间，用于创建数据块池与对象池。在对象池中，进程在该空间的起始地址创建并维护一个根对象，使对象池中的其他数据均可经过根对象直接或间接可达。根对象包含用于描述数据块池的位图和空闲队列、本进程已缓存文件的列表，其中缓存文件的列表可以索引各个文件及其缓存的元数据。元数据描述了数据缓存的位置及其状态。

应用使用NVMPI-IO打开一个文件时，相关的进程首先构建缓存的元数据：根据文件的大小，文件在逻辑上被划分为固定大小的块，各个进程互斥地成为这些数据块的管理者，即数据缓存的元数据将分布到不同的进程中。管理者将维护相应数据块的元数据。当文件中某个数据块被访问时，其数据将会被迁移到数据块池中。使用分布式的元数据及缓存数据避免了多进程间出现负载不均的情况。

4.2 数据维护与访问

NVMPI-IO将文件数据缓存到不同进程、不同节点的NVM空间中，为了使进程可以准确地访问所需的数据，本系统务必正确地管理、维护元数据信息。针对文件中的每一个块，称维护该数据块元数据的进程为管理者，称缓存该块数据的进程为所有者，称需要访问该块数据的进程为请求者。其中，管理者由该块在文件中的位置确定；对于一个数据块，一个进程可以同时承担上述多种角色。

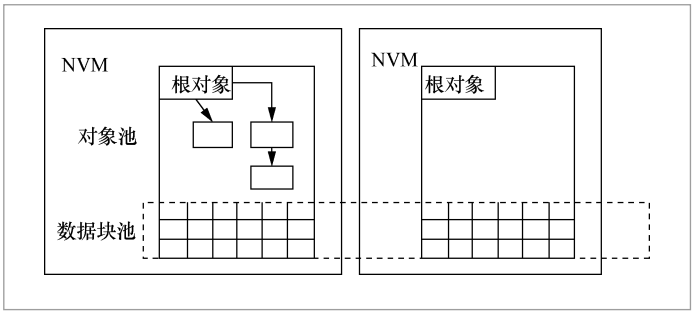


图 2 NVMPI-IO 非易失内存中的数据布局

如**图3**所示，当请求者需要访问某块数据时，其需要先查询元数据，再进行文件数据的传输。在向管理者发送元数据查询请求之前，请求者使用预更新（pre update）方案，首先准备空闲的数据块，将其编号组装到查询信息当中，一并发送至管理者。当收到管理者返回的信息后，请求者获得数据缓存的状态，向数据所有者发送信息，并按需交换数据，完成应用的需求。

管理者负责管理、维护数据块的元数据。NVMPI-IO将文件在逻辑上进行分块，通过轮询调度（round-robin）策略将数据块的管理权分配到各个进程，以缓解负载不均带来的影响；请求者可通过取模运算获得数据块的管理者编号。管理者在查询元数据的同时，需要维护元数据，使其始终有效。MPI语义规定，应用访问文件数据时上一次文件访问已经完成，这使得对元数据的维护可以延迟到下一阶段文件访问之前完成。管理者在接收到元数据查询请求后，首先使用延迟提交（late commit）方案，提交上一次文件访问引起的元数据变化，并将新的元数据记录下来，待下一阶段再实体化到NVM中；若不使用延迟提交而在元数据变化后马上更新，则会额外增加一次数据交换，带来更大的网络开销。管理者

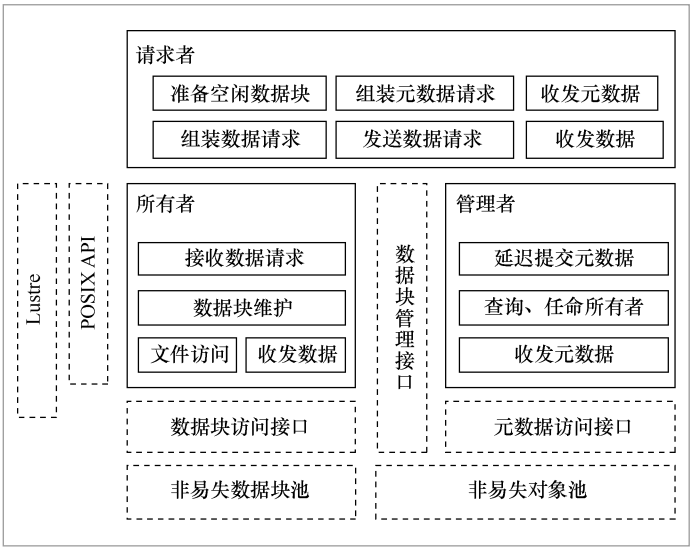


图 3 进程访问文件数据块的架构

完成延迟提交之后，查询元数据，找到数据的缓存状态，将其缓存位置及状态返回至请求者。若数据块尚未被缓存，管理者任命某个请求者为该块数据的所有者，并将缓存状态发送至该数据块的其他请求者；使用预更新方案时，允许在多个请求者同时请求一个未被缓存的数据块时，不额外引入“任命数据块的所有者”带来的网络开销。

所有者负责数据的存取。其收到请求者发送的数据请求时，该数据块未必已经被缓存在非易失缓存中。若数据块未被缓存，所有者需首先使用POSIX API从文件系统中读取数据，并将其缓存到预更新时指定的数据块中。所有者利用数据在缓存中的地址（指针），直接与请求者进行数据交换。同时，所有者需要维护一个文件数据块到缓存地址的索引，这样既可快速找到数据块的地址，又可在应用需要关闭文件时，使用POSIX API将有效的缓存写回到共享文件系统中。

NVMPI-IO使用预更新与延迟提交两种优化手段，在维持MPI-IO操作的语义不变的同时，尽可能地减少节点之间的数据传输，并有效地维护文件缓存的元数据。此外，相比使用块粒度写数据，使用数据地址直接传输数据可以有效地减小写放大，同时避免非必要的数据组装及传输。

使用分布式的非易失内存时务必维护数据的分布式一致性[18]。NVMPI-IO在节点内部使用持久性的可撤销日志（undo-log）来维护数据一致性，当进程出错重启后，进程从持久性日志中恢复原始数据。在节点之间，进程在延迟提交过程更新本地的元数据时进行通信；若存在更新元数据失败的进程，则撤销相关进程的更新操作，以维护数据的分布式一致性。

4.3 数据传输路径

在传统MPI应用的设计中，若应用需要多次访问同一个文件，则其向文件系统发出多次数据访问的请求，数据在底层的共享文件系统与计算节点之间反复迁移，引起计算节点的大量I/O开销以及底层文件系统的大量负载。NVMPI-IO将数据缓存至计算节点内的NVM设备中，将对底层文件系统的访问转化为对本地或其他节点上NVM的访问，即将中心化传输与慢速的磁盘访问转化为分布式的网络传输与低时延高带宽的NVM访问。本系统仅在首次读取数据以及关闭文件写回数据时对文件系统进行访问，即对于同一文件，最多发生两次文件系统读写，采用这一策略缩短了计算节点访问数据的时间，并减小了底层文件系统的I/O负载。

5 效果评估

5.1 实验环境

为了验证本系统设计的有效性，搭建了实验平台用于对比实验。实验平台包含4个计算节点及一个共享的存储系统，其中每

个计算节点配备Intel Xeon Gold 6230N处理器、6×32 GB DRAM、2×128 GB Intel Optane DC PMM，所有节点运行CentOS7.6系统（Linux内核版本3.10.0-957），共享的存储系统将HDD作为存储设备，挂载了Lustre-2.10文件系统。NVMPI-IO的原型系统在MPICH（v3.2.1）中实现，同时实验对照组运行MPICH（v3.2.1）。

5.2 实验设计

本文采用对比实验验证本方案的有效性。为了模拟MPI应用的文件访问模式，并最大化MPI-IO的I/O优化效果，实验组与对照组均需满足以下条件。

- 打开DIRECT I/O开关，即允许进程使用direct I/O直接从文件系统中读取数据，而不将数据缓存到操作系统的缓冲区中。
- 避免粒度较小的文件访问。MPI-IO使用聚合I/O与数据筛选的优化手段，将进程需要的小粒度数据经过节点之间的消息传递，组合成大粒度的数据，再向文件系统发起I/O请求，避免了小粒度的随机访问；实验中使用16 KB的粒度顺序地读写文件，以模拟这一访问模式。
- 对同一文件进行多次读写，比较访问非易失缓存和访问底层文件系统的性能。

本实验使用不同数量的MPI进程对同一文件进行多次顺序读写，比较MPICH与NVMPI-IO的性能差异。根据MPI应用的特点，本实验只研究系统访问文件数据的带宽。

5.3 实验效果

如**图4**所示，MPI应用访问文件时，若使用16 KB的粒度对文件重复读取，负载较轻，不会达到存储设备的性能瓶颈。随着进程数量增加，存储设备的性能得到体现，NVMPI-IO的吞吐率超过1 300 MB/s，是相同进程数量下MPICH访问文件的吞吐率的2.87倍。

如**图5**所示，对文件进行重复写入时，MPICH写入文件系统的性能较低，而NVMPI-IO的吞吐率保持在较高状态。在MPICH中，所有进程在同一节点与进程分布在4个节点时吞吐率相近，是因为在本实验的访问模式下，进程直接访问文件系统，而进程之间未发生文件数据的收发，没有引入开销；随着进程数量增加，访问文件的吞吐率提升，这说明当前并行度下未达到Lustre文件系统的性能瓶颈。在NVMPI-IO下，吞吐率保持在较高水平，是相同条件下不使用缓存的20倍以上；但是，相同数量的进程运行在4个节点上时，由于管理、访问数据缓存引入了跨节点的

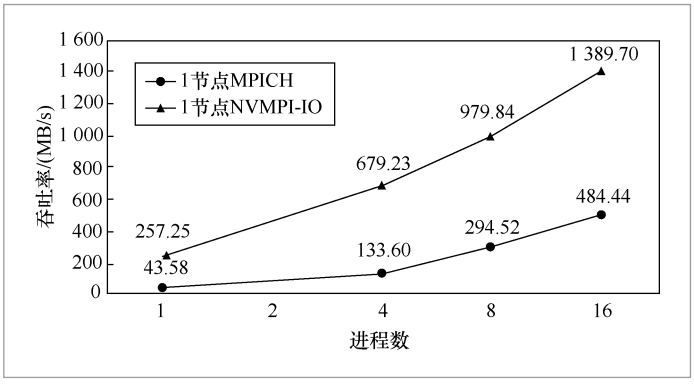


图 4 不同进程数下应用读取数据的吞吐率

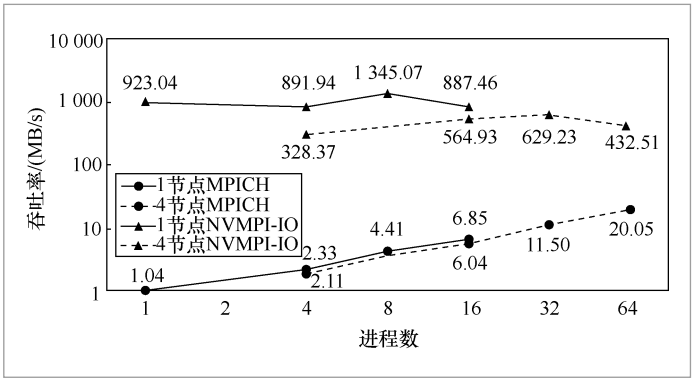


图 5 不同数量的节点、进程下，应用写入数据的吞吐率

网络开销，其性能低于进程运行在同一节点上的情况；同时，由于Intel Optane DC PMM的并行性能不理想，在并行写入数据时性能会出现明显下降[4-5]，当节点内进程数量从8增加到16后，系统的吞吐率出现下降。未来可使用更多的非易失内存DIMM，以提高系统访问非易失内存的并行性。

实验结果显示，面向非易失内存的MPI-IO接口优化尽管受到NVM设备并行性不理想、引入了节点间的网络开销等问题的影响，仍可使应用获得显著的性能提升。

6 结束语

本文针对MPI应用访问文件的模式，利用非易失内存设备的可按字节寻址、数据可持久化、读写性能优秀等特点，设计了面向非易失内存的MPI-IO接口优化方案。NVMPI-IO被透明地部署在MPI-IO中间件中，具备非独占使用NVM、高可移植性、可快速恢复程序运行状态等优点，为MPI应用提供文件的分布式缓存，在应用重复访问文件数据时，为应用带来显著的性能提升。针对NVMPI-IO现存的不足，未来可着力于解决NVM设备的并行访问问题，并尝试使用RDMA网络加速节点之间的数据传输，以降低节点间的网络开销。

参考文献:

[1] THAKUR R, GROPP W, LUSK E, et al. On implementing MPI-IO portably and with high performance[C]//The 6th Workshop on I/O in Parallel and Distributed Systems. New York: ACM Press, 1999: 23-32.

[2] THAKUR R, GROPP W, LUSK E, et al. Data sieving and collective I/O in ROMIO[C]// The 7th Symposium on the Frontiers of Massively Parallel Computation. Piscataway: IEEE Press, 1999: 182-189.

[3] SANKARAN S, SQUYRES J M, BARRETT B, et al. The LAM/MPI checkpoint/restart framework: system-initiated checkpointing[J]. The International Journal of High Performance Computing Applications, 2005, 19(4): 479-493.

[4] IZRAELEVITZ J, YANG J, ZHANG L, et al. Basic performance measurements of the Intel Optane DC persistent memory module[J]. arXiv preprint, 2019, arXiv:1903.05714.

[5] 陈游旻, 李飞, 舒继武. 大数据环境下的存储系统构建：挑战、方法和趋势[J]. 大数据, 2019, 5(4): 27-40.
CHEN Y M, LI F, SHU J W. Building storage systems in big data era: challenges, methods and trends[J]. Big Data Research, 2019, 5(4): 27-40.

[6] VOLOS H, TACK A J, SWIFT M M, et al. Mnemosyne: lightweight persistent memory[J]. ACM SIGARCH Computer Architecture News, 2011, 39(1): 91-104.

[7] COBURN J, CAULFIELD A M, AKEL A, et al. NV-Heaps: making persistent objects fast and safe with next-generation, non-volatile memories[J]. ACM SIGARCH Computer Architecture News, 2011, 39(1): 105-118.

[8] WU X J, QIU S, REDDY A L N, et al. SCMFS: a file system for storage class memory and its extensions[J]. ACM Transactions on Storage, 2013, 9(3).

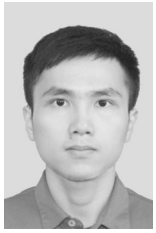
[9] CONDIT J, NIGHTINGALE E B, FROST C, et al. Better I/O through byte-addressable, persistent memory[C]// The ACM SIGOPS 22nd Symposium on Operating Systems Principles. New York: ACM Press, 2009: 133-146.

[10] CHEN Y M, SHU J W, OU J X, et al. HiNFS: a persistent memory file system with both buffering and direct-access[J]. ACM Transactions on Storage, 2018, 14(1): 1-30.

[11] XU J, SWANSON S. NOVA: a log-structured file system for hybrid volatile/

non-volatile main memories[C]//The 14th USENIX Conference on File and Storage Technologies. [S.l.:s.n.], 2016: 323-338.
[12] XU J, ZHANG L, MEMARIPOUR A, et al. NOVA-Fortis: a fault-tolerant non-volatile main memory file system[C]//The 26th Symposium on Operating Systems Principles. New York: ACM Press, 2017: 478-496.
[13] ZUO P F, HUA Y. A write-friendly and cache-optimized hashing scheme for non-volatile memory systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(5): 985-998.
[14] ZUO P F, HUA Y, WU J. Level Hashing: a high-performance and flexible-resizing persistent hashing index structure[J]. ACM Transactions on Storage, 2019, 15(2).
[15] LU Y Y, SHU J W, CHEN Y M, et al. Octopus: an RDMA-enabled distributed persistent memory file system[C]// The 2017 USENIX Annual Technical Conference. New York: ACM Press, 2017: 773-785.
[16] YANG J, IZRAELEVITZ J, SWANSON S. Orion: a distributed file system for non-volatile main memories and RDMA-capable networks[C]//The 17th USENIX Conference on File and Storage Technologies. New York: ACM Press, 2019: 221-234.
[17] RAHMAN M W, ISLAM N S, LU X, et al. NVMD: non-volatile memory assisted design for accelerating MapReduce and DAG execution frameworks on HPC systems[C]//2017 IEEE International Conference on Big Data. Piscataway: IEEE Press, 2017: 369-374.
[18] 吴昊, 陈康, 武永卫, 等. 基于RDMA和NVM的大数据系统一致性协议研究[J]. 大数据, 2019, 5(4): 89-99.
WU H, CHEN K, WU Y W, et al. Research on the consensus of big data systems based on RDMA and NVM[J]. Big Data Research, 2019, 5(4): 89-99.

作者简介



邓镇龙(1995-),男,中山大学计算机学院硕士生,主要研究方向为分布式存储。

陈志广(1984-),男,博士,中山大学计算机学院副教授,主要研究方向为大数据存储与处理、并行与分布式计算、高性能计算与超级计算机。

收稿日期: 2021-01-21
通信作者: 陈志广, zhiguang.chen@nscc-gz.cn
基金项目: 国家重点研发计划资助项目(No.2018YFB0203904);国家自然科学基金资助项目(No.61872392, No.61832020, No.U1811461);广东省自然科学基金资助项目(No.2018B030312002);广州市珠江科技新星项目(No.201906010008)
Foundation Items: The National Key Research and Development Program of China(No.2018YFB0203904), The National Natural Science Foundation of China(No.61872392, No.61832020, No.U1811461), The Natural Science Foundation of Guangdong Province (No.2018B030312002), Pearl River S & T Nova Program of Guangzhou(No.N201906010008)