多核环境下高效集合通信关键技术研究

张攀勇[1),2),3)] 孟 丹[1),2)] 霍志刚[1),2)]

[1)](中国科学院计算技术研究所国家智能计算机研究开发中心 北京 100190)

[2)](中国科学院计算机系统结构重点实验室 北京 100190)

[3)](中国科学院研究生院 北京 100049)

摘 要 随着高性能计算需求的日益增长,多核处理器在高性能计算中间得到了广泛的普及.为了保证高性能计算机系统的效率,需要保持计算和通信的平衡性,多核的广泛使用对通信系统的效率提出了更高的要求.集合通信作为通信系统中的重要组成部分,研究多核环境下的高效集合通信具有十分重要的意义.文中首先研究了多核对集合通信性能的影响,并根据多核处理器共享 Cache 以及内存竞争的特点,提出了层次化算法、限制并发、NUMA 感知的优化方法和 Cache 友好的优化算法,并分别在 MPI_Barrier、MPI_Bcast 和 MPI_Alltoall 中进行了验证.实验结果表明优化方法能够有效地利用多核结构特点,降低竞争带来的影响,提高了多核环境下集合通信的性能和可扩展性.

关键词 高性能计算;多核机群;集合通信优化;NUMA;MPI

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2010.00317

Research of Collectives Optimization on Modern Multicore Clusters

ZHANG Pan-Yong[1),2),3)] MENG Dan[1),2)] HUO Zhi-Gang[1),2)]

[1)](*National Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2)](*Key Laboratory of Computer System and Architecture, Chinese Academy of Sciences, Beijing* 100190)

[3)](*Graduate University of Chinese Academy of Sciences, Beijing* 100049)

Abstract With the rapid increase in HPC computing requirement, the multicore is widely deployed in HPC systems. To keep the efficiency of application in large scale systems, it is very important keep up the balance of communication to computation, thus multicore brings more requirement for communication systems. As collective communication is an important part in communication systems and is critical for the whole systems, thus it is important to research on the impacts of multicore environment on collective performance. This paper first analyzes how multicore impacts on collective communication. It is found out that multicore SMP clusters brings two conflict impacts, it not only has faster intra-socket communication path which can speed up the performance of collective communications, but also it brings memory/cache contention which might degrade the communication performance. Based on these aspects, this paper proposes multicore-aware collecitves optimization techniques, which includes: hierarchy-aware algorithms, limited-concurrency, NUMA-aware algorithms and cache-friendly optimization. These optimization methods are implemented in MPI_Barrier, MPI_Bcast and MPI_Alltoall. Experiments show that the proposed algorithms increase the performance and scalability of collective communication.

Keywords HPC; multicore clusters; collectives optimization; NUMA; MPI

收稿日期:2009-07-15;最终修改稿收到日期:2009-09-16.本课题得到国家"八六三"高技术研究发展计划项目"曙光 5000A 高效能计算机"(2006AA01A102)资助.**张攀勇**,男,1981 年生,博士研究生,主要研究方向为高效机群通信系统、高性能计算.E-mail:zhangpanyong@ncic.ac.cn.**孟 丹**,男,1965 年生,研究员,博士生导师,主要研究领域为高性能计算、分布式系统.**霍志刚**,男,1978 年生,博士,副研究员,主要研究方向为机群容错、高性能计算.

1 引 言

由于微处理器工艺、散热等诸多限制,处理器的发展已经由原有的提高主频的方式变为芯片内部集成更多处理单元的方式."多核时代"的摩尔定律预测了芯片内部集成核数成指数增长,在不久的将来一个芯片内部就能够集成上百的处理器核[1].多核处理器对系统性能带来了两种可能的影响:一方面,芯片内部集成更多的处理核,为了提高核之间的数据交换速度,一般要在芯片内实现大容量的共享Cache,这种共享Cache能够带来片内数据交换的性能提升.另一方面,多核处理器共享使用芯片上的数据通路和共享Cache,很容易出现竞争带来性能的下降.

高性能计算对计算能力的需求永无止境,这使得多核处理器在高性能计算机中得到了广泛使用.在2009年11月的Top500中,绝大部分系统采用了多核处理器,仅有4套系统使用单核处理器.另一方面,并行应用为了保证计算效率,需要高性能系统保持计算和通信性能的平衡性.集合通信(collective communication)作为通信系统的重要组成部分,为多个进程参与的组通信原语,被并行应用广泛用于一组进程之间的数据操作和同步操作.当并行应用程序的规模越来越大时,所使用的处理器的规模也越来越大,集合通信组内部进程之间的通信量相应增大,且需要相互协作完成通信语义,因此集合通信往往成为系统的性能瓶颈,需要优化集合通信以提高整体系统性能.例如对Cray T3E 900系统上运行的并行应用分析[2]表明,一些应用的集合通信时间占到了总通信时间的80%.另一方面,多核导致了单个节点上能够运行更多的进程[3],集合通信的节点内部分在通信中变得更加重要,而多核引入的片内数据通路和共享Cache在访问模式和速度上与共享内存相比存在差异,尽管现有的集合通信算法可以平滑地移植到多核机群中,但是传统SMP机群中的集合通信算法只针对通信网络和共享内存通信的特点进行优化,并没有考虑到多核的结构特点,如果直接使用这些算法会带来集合通信性能和可扩展性的问题.由于集合通信的性能对并行应用来说非常关键,因此研究多核环境对集合通信的影响,并针对多核环境的特点进行相应的集合通信算法优化成为了一个研究热点.

本文的主要贡献有两点:(1)详细分析了多核环境对集合通信的影响,发现多核对通信性能带来了两种冲突的效果:多核中共享Cache的快速数据通路能够提高集合通信的性能;共享Cache和内存的竞争降低了集合通信的性能,这对未来多核环境下集合通信优化具有指导意义.(2)针对多核的特点,提出了层次化算法、限制并发、NUMA感知和Cache友好的优化方法.这些优化方法能够提高共享Cache利用率,降低内存竞争带来的影响,从而提高了集合通信的性能.

本文第2节简要描述本文的研究背景和相关工作;第3节评价多核对集合通信的影响;第4节介绍本文提出的多核感知集合通信算法;第5节介绍实验方法和实验结果;第6节对全文进行总结和展望.

2 研究背景和相关工作

本节将介绍本文的研究背景和相关工作,包括多核处理器和集合通信.

2.1 多核处理器

多核处理器指在一个处理器芯片内部集成多个核心.为了提高片内核之间数据交换的速度,减少内存操作,现有多核芯片在片内集成了大容量的共享Cache,如Intel早期的四核Xeon集成了2MB的共享L2 Cache,Nehalem集成了8MB的共享L3 Cache,AMD的四核系统Barcelona集成了2MB L3 Cache.由于Cache和内存的存取模式以及性能差异,导致了多核处理器系统相比传统的SMP系统,由传统的两层通信模型:节点内(Intra-Node)通过共享内存通信、节点间(Inter-Node)通过高性能网络通信,变为三层通信模型[3]:片内(Intra-Socket)通过共享Cache通信、片间(Inter-Socket)通过共享内存通信、节点间(Inter-Node)通过高性能网络通信.各个通信层次之间在性能和访问方式上有明显的差异.因此,通信系统需要针对多核环境下的三层通信进行相应的优化.

另一方面,由于CPU和Memory性能提升具有不平衡性,"内存墙"(memory wall)问题一直存在.而多核处理器由于在芯片内部集成更多的核,导致"内存墙"问题更加严重.为了提高芯片访存性能,非一致访存结构NUMA将变成必然.新的多核处理器如Intel Nehalem处理器和AMD的Opteron处理器内部均集成内存控制器.芯片之间采用NUMA结构进程互联,构成cc-NUMA系统.

2.2 集合通信

MPI 的集合通信为一组进程构成一个通信域(communicator)，在进程之间进行的通信操作，用于数据分发和同步操作. 集合通信可以分为两种类型：(1)一到多，如 MPI_Gather，MPI_Scatter，MPI_Reduce，MPI_Bcast；(2)多到多，如 MPI_Allgather，MPI_Allreduce，MPI_Alltoall. 由于集合通信随着并行程序规模的扩大，所使用的进程数目越来越多，进程之间需要更多的同步和数据交互操作. 如果集合通信算法不具有较好的性能和可扩展性，大规模情况下会降低整体通信的性能，从而降低并行应用的效率. 因此对集合通信的性能和可扩展性的研究一直是一个非常重要的研究方向.

传统的集合通信实现都是基于 MPI 的点到点消息传递接口实现，通过进程之间的消息传递，最终完成集合通信的语义. 为了提高 SMP 机群环境下集合通信的性能，一般采用了基于共享内存的集合通信优化算法，节点内利用了共享内存进行数据交换操作，节点间通过高性能网络进行数据交换. 这种方式利用了共享内存方式通信相比通信网络通信的高性能，同时减少了通信网络上的消息，带来了性能的提升. 相关的研究有：文献[4]研究了节点内共享内存方式对多种集合通信的性能影响，关注层次方式带来的性能提升；文献[5]提出了一系列基于共享内存的通信原语，并基于这些通信原语实现了 MPI_Bcast 和 MPI_Scatter，结果表明共享内存方式提高了通信性能.

而多核环境下的集合通信性能优化，已经引起了学术界的关注，国内外很多研究者对此问题进行了研究. 文献[6]中针对 Intel Clovertown 多核系统中共享 L2 Cache 的特点，提出了两层的通信算法来优化 MPI_Bcast 和 MPI_Allgather，同时使用 NUMA 特点，优化共享内存的 MPI_Allreduce，并使用通信调度的方式充分利用 HT 总线的双向带宽，以优化 MPI_Alltoall 性能. 该研究重点关注了 Intel Clovertown 系统的硬件特点，而不是多核系统普遍的特点，以及多核环境对集合通信算法的影响. 文献[7]中利用多核和 ConnectX IB 网卡在多进程并发时性能高的特点使用多个首进程来优化 MPI_Alltoall 的性能. 文献[8]则通过共享内存来优化多核环境下的同步操作，研究了小消息时树形通信算法的度的选择，以及大消息时并发访问度的问题. 该研究关注多核环境下集合通信算法实现的问题. 文献[9]研究了多核环境下的内存层次模型，提出了 mLogn(P)模型，并通过该模型指导优化 MPI_Bcast 的性能，该研究重点关注了多核的内存层次性. 本文重点分析多核对集合通信的影响，同时针对这些特征提出多核环境下集合通信优化的关键技术.

3 多核对集合通信的影响

3.1 快速数据通路

多核处理器设计中为了提高多核内部核心之间的数据交换效率，在多核内部集成了共享 Cache. 核心之间通过共享 Cache 能够快速地进行数据传输，对于 AMD Barcelona 8347HE 1.9GHz 四核处理器使用 lmbench 测试发现，随机访问 L3 Cache 的开销为 26.45ns，而对应的访问主存的开销为 109.93ns. 因此为了提高通信效率，需要尽可能地利用共享 Cache 进行核之间的数据传输. 核间通过共享 Cache 高速传输的性能直接体现到 MPI 的点到点通信性能上面，使用 IMB 测试程序测试点到点通信性能发现，传输 8Bytes 消息，片内传输延迟为 0.58μs，片间传输延迟为 0.74μs. 这 0.16μs 来自于片内通信和片间通信的性能差异，因此为了提高通信系统性能，需要尽可能利用片内通信进行数据传递.

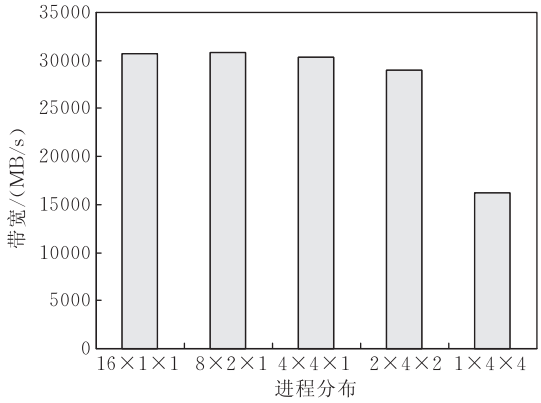
另一方面，SMP 系统内部的数据传输最后表征为 Cache-to-Cache 数据操作，因此 Cache-to-Cache 的传输性能对小数据传输性能非常关键. Cache-to-Cache 的性能决定于传输总线的性能，由于多核的系统总线集成于芯片内部，芯片内部系统总线的速度要远大于芯片之间系统总线速度，因此多核内部 Cache-to-Cache 传输速度会有很大的提升. 本文使用 c2cbench 测试了多核内部和芯片之间的 Cache-to-Cache 操作的性能开销，测试数据为 128KB，分块大小 8KB，测试结果发现，片内核之间传输时间为 2.32μs，而芯片间传输时间为 5.13μs. 这验证了芯片内部和芯片之间的 Cache-to-Cache 操作的性能差异.

3.2 内存和 Cache 的竞争

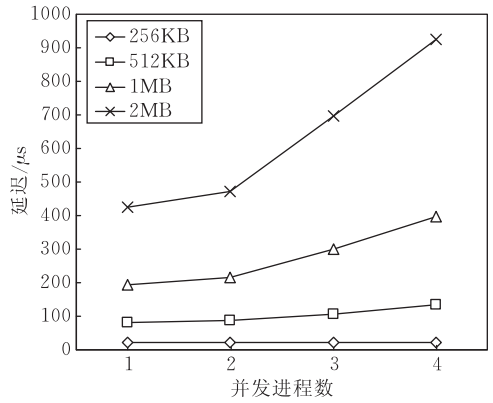
由于处理器的发展速度远快于内存的发展速度，“内存墙”的问题一直存在. 多核处理器在一个芯片内部集成多个核，使得多个核共享内存带宽. 如果没有很好的控制，共享带来的内存竞争(memory contention)会极大地降低内存访问的性能.

为了检验内存竞争带来的访存性能的降低，本文使用 STREAM 测试了片内分布不同进程数的模

式下的内存带宽.将 16 个进程分布到不同的节点 CPU 芯片和核上,使用 N×S×C 来表示不同的配置,C 越大表示一个芯片内部分布了越多的进程,因此也更有可能导致内存竞争.从图 1(a)中的测试结果可以看出,如果将进程分布到不同的芯片上,由于每个进程能够独立地使用内存带宽,不存在内存竞争,因此 16×1×1, 8×2×1, 4×4×1 这 3 种情况下内存带宽基本相同.而当一个芯片内部集中了 2 个进程共享内存带宽时(2×4×2 的配置),就带来了 1188MB/s 的访存性能下降.当芯片内部集中了 4 个进程时(1×4×4 的配置),由于多个进程并发访问带来了严重的内存竞争,访存性降低了13979MB/s.该结果证实了内存竞争导致了性能严重下降.



(a) 不同配置下STREAM性能



(b) Cache竞争效果

图 1 竞争对性能的影响

另一方面,多核处理器内部的共享 Cache 尽管能够提高核心之间的数据传输速度,但是由于工艺的限制,共享 Cache 一般容量有限.多个核共享操作共享 Cache,很容易由于冲突缺失(conflict miss)和容量缺失(capacity miss)导致 Cache 缺失率增大,从而带来性能的下降.本文采用测试程序测试 Cache 竞争带来的性能下降,测试模式为多个进程并发读写不同的内存块,记录操作所耗费的时间.分别测试了一个芯片内部存在 1～4 个进程的情况,读写内存块大小为 256KB、512KB 和 1MB、2MB 的情况.从图 1(b)的测试结果中可以看出,如果程序访问集小于 CPU 核私有的 L2 Cache 大小时(图中 256KB 的情况),由于各个核访问互相不冲突,因此操作时间基本相同.当进程的访问集大于 L2 Cache 大小,使用到共享 L3 Cache 时(512KB,1MB 的情况),芯片内部存在更多的进程会带来更加严重的 Cache 冲突,因此操作时间会随着进程数的增加而变大.当进程访问集大于 L3 Cache 时(2MB 的情况),由于 Cache 容量限制带来的容量缺失,导致了 Cache 抖动,使得访问性能严重下降,由 2 个进程操作时间 427.11μs 增长到 3 个进程时的 696.99μs.这也就验证了共享 Cache 竞争带来的性能下降.

3.3 多核对集合通信性能的影响

由于共享内存对通信系统带来了两种冲突的性能影响,而集合通信为多个进程参与通信的过程,更容易在多核环境下产生竞争.另一方面,由于现有的操作系统在多核 SMP 系统上面使用和在 SMP 系统中相同的方式支持共享内存,因此可以将 SMP 机群上的基于共享内存的集合通信优化算法平滑移植到多核 SMP 系统中.但是由于现有集合通信优化算法并不区分多核环境下的片内通信和片间通信,可能会导致集合通信的性能瓶颈.因此有必要研究多核环境对于集合通信到底带来了什么样的影响,这对于后续集合通信的优化具有重要的指导意义.

本文使用 MPI_Alltoall 来验证多核对集合通信的影响.使用 $S\times C$ 来表示进程分布到 S 个芯片和 C 个核上的映射方式.C 越大表示芯片内部存在更多的处理核.测试结果如图 2 所示,从结果中可以看出,在小消息的时候,由于利用到多核内部的快速数据通路,将更多的进程集中到芯片内部,可以利用高速的片内通信,因此,1×4 的情况在小消息的时候延迟比 4×1 的情况要低.当消息变大的时候,由于多核共享 Cache 和内存竞争带来了严重的性能下降,最终带来了集合通信的性能下降,而将进程分布到不同的芯片上,能够利用独立的 Cache 和内存带宽,性能反而比多个进程集中在一个芯片的情况好,因此图中 8MB 的 1×4 情况的延迟远高于 4×1 的情况的延迟.

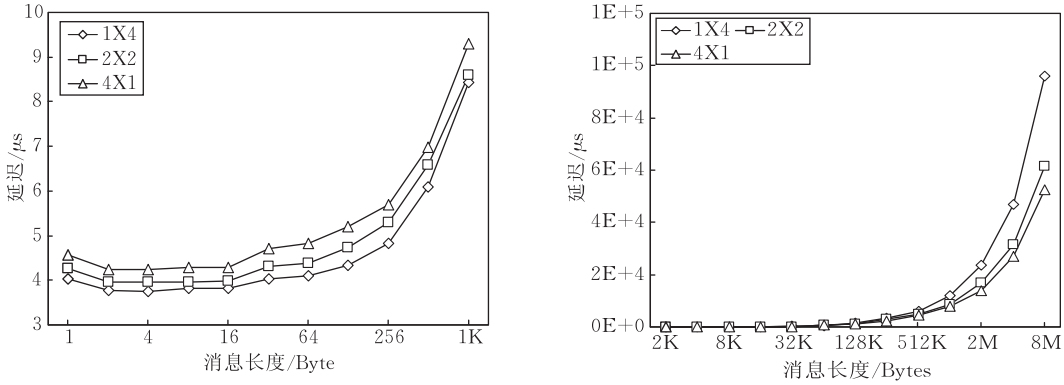


图 2　Cache 竞争效果

综上所述，为了在多核环境下实现高效集合通信，对于小消息要尽可能地利用快速的片内通信；而对于大消息，则需要尽可能地降低 Cache 和内存的竞争带来的性能下降.

4　多核感知的集合通信算法

为了提高集合通信在多核环境下的可扩展性和性能，本文针对多核的特点提出了多层次的集合通信算法、限制并发、NUMA 感知和 Cache 友好的优化技术. 分别在 MPI_Barrier、MPI_Bcast 和 MPI_Alltoall 中进行实现，下面将分别描述多核感知的集合通信算法.

4.1　MPI_Barrier

MPI_Barrier 用于多个进程之间的同步操作，由于进行 Barrier 操作时，所有其他程序均需阻塞直到 Barrier 操作完成，现有的基于共享内存的 MPI_Barrier 算法采用两层通信的方式实现：(1) 节点间同步阶段：用于节点级首进程之间的同步操作. 相应的优化有 OSU 大学[4] 采用 RDMA 操作直接实现 Barrier 的数据交互，提高了节点间的性能. (2) 节点内同步阶段：采用了基于共享内存的 gather/scatter 双队列算法[10] 进行实现.

现有基于共享内存的双队列算法中，由于 gather 和 scatter 阶段使用了两个不同的队列，因此 Cache 缺失率提高会带来 Barrier 操作的性能下降. 本文使用的算法中考虑到 gather 和 scatter 两个阶段的顺序性，使用同一个队列来进行 gather 和 scatter 操作，减少了队列占用空间，提高了共享 Cache 利用率.

为了使 MPI_Barrier 能够感知到多核的片内结构，本算法将传统的片内、片间两层算法进行扩展，实现了多层次的集合通信. MPI_Barrier 算法实现仍然分为 gather 和 scatter 两个阶段，同时将 gather/scatter 的操作分为 3 层：片内、节点内和芯片间. 而算法根据进程所在层次和处理的任务不同，将进程分为 3 种：(1) 节点级首进程. 其首先负责芯片级主进程的 gather 操作，如果 gather 操作完成表示本芯片内部 gather 操作完成，然后首进程之间进行 gather 和 scatter 操作以完成它们之间的同步操作，在首进程之间的同步完成之后，通知芯片级主进程，开始 scatter 操作. (2) 芯片级主进程. 其负责芯片内部的子进程都完成 gather 操作，然后向节点级首进程通知 gather 操作完成，等待节点级首进程分发 scatter 操作，在接收到首进程发出的 scatter 操作之后，向子进程分发 scatter 操作. (3) 子进程. 其负责向自己对应的芯片级主进程通知本进程完成 gather 操作，同时等待芯片级主进程通知的 scatter 操作，在子进程获取到 scatter 操作通知，则表示本次 Barrier 操作完成. 多层的集合通信算法的层次较好地符合了多核的三层结构，相比现有算法，能够将更多的通信操作集中到片内通信上完成，因此能够提高 MPI_Barrier 的通信效率.

4.2　MPI_Bcast

MPI_Bcast 用于将主进程中的数据广播到所有进程上. MPI_Bcast 的算法实现中，在小消息时为了降低通信延迟，直接采用 binomial tree 算法进行实现；针对大消息，为了提高通信的并发性，节点间的进程之间采用了 scatter-allgather 算法实现，节点内进程之间采用共享内存的方式，首进程首先将数据拷贝到共享内存，然后其它进程直接从共享内存并发拷贝数据. 现有算法的大消息通信存在两个问题：一方面，所有进程同时从共享内存中拷贝数据，这种并发拷贝会带来严重的内存竞争，从而大大降低了通信系统的性能. 另一方面，由于现有算法的共享内存分配一般是由首进程分配，对于 NUMA 系统来说，共享内存位于靠近首进程的位置，这样多个进程

访问该块内存只能利用 NUMA 系统的一个内存控制器的带宽,而不能够利用系统中的所有内存带宽,因此限制了 MPI_Bcast 的性能.

本文使用了限制并发、NUMA 感知的集合通信优化方法来优化 MPI_Bcast 的大消息性能. 首先在共享内存分配的时候,对内存分段,每一段使用 numactl库控制该段内存位于靠近相应进程的内存中. 当首进程完成向共享内存的数据拷贝之后,各个进程在从共享内存中拷贝数据时,限制每一个共享内存段同时并发访问的进程数的个数,具体的算法如图 3 所示.

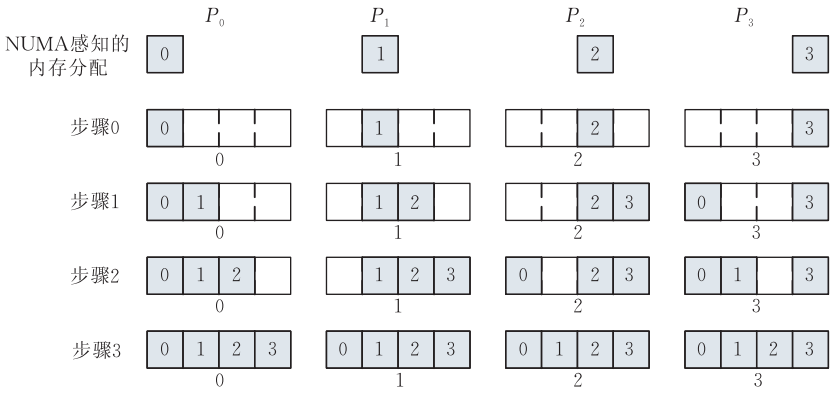


图 3 限制并发、NUMA 感知 MPI_Bcast 示意图

在本示意图中,每一步骤中限制并发访问共享内存段的并发度为 1,步骤 0 时 P0 进程组访问 P0 临近的共享内存块 0;步骤 1 时访问下一块共享内存块 1;步骤 2 时访问共享内存块 2,步骤 3 时访问共享内存块 3. 其它进程类推,在这 4 步完成之后,每一个进程都获取到全部 Bcast 的数据. 在限制并发度为 k 的时候,P0 被看作是 k 个进程的组合,此时同时访问一个共享内存块的进程为 k. 由于本算法限制了并发访问同一块内存块的并发度为 k,而不是现有算法的节点内进程数 n,因此能够极大地降低内存竞争. 另一方面由于共享内存分配到 NUMA 系统的不同内存中去,可以利用 NUMA 系统中的所有内存控制器的带宽,提高了 Bcast 的性能. 本优化算法只改变了节点内对共享内存的访问顺序,节点间仍然使用原有算法中首进程组之间使用的 scatter-allgather 算法进行数据交互.

4.3 MPI_Alltoall

MPI_Alltoall 算法用于进程组的所有进程两两进行数据交互. 由于单个算法不能够满足 MPI_Alltoall的性能需求,因此现有实现中,对于小消息采用了 Bruck 算法,对于中等消息采用了 Irecv-Isend 的算法,对于大消息的时候采用 pair-wise exchange 算法[11].

在 3.3 节中可以看到,由于 Cache 和 Memory 的竞争会带来中等消息和大消息情况下 MPI_Alltoall 性能的严重下降. 为了降低这种性能损失,本文提出了 Cache 感知的优化算法,利用 Data-tiling 技术[12],将 MPI_Alltoall 中的每一次通信的收发大数据块进行切分,分为多个适合共享 Cache 大小的小数据块进行通信分发. 这种方式能够降低大数据访问由于 Cache 竞争带来的 Cache 缺失,提高共享 Cache 的利用率,减少由于 Cache 缺失带来的内存访问,因此能够提高 MPI_Alltoall 的集合通信性能.

5 实验方法和结果

本实验的实验平台为曙光 5000A 系统,每个节点为四路四核 AMD Barcelona 8347HE 1.9GHz,带有 64GB 的 DDR2 667MHz 内存,使用 Mellanox ConnectX DDR MT25418 20Gbps HCA 将各个节点连接起来,顶级交换级为 Voltari 288 口 IB 交换机,使用 OFED-1.4 版本,操作系统为 SLES10 sp2,MPI 环境为 MVAPICH-1.1,编译器为 gcc 4.1.2.

5.1 MPI_Barrier

为了测试各种优化的效果,本节对比了 3 种算法:(1)原始算法. 该算法为 MVAPICH-1.1 使用,使用双队列 scatter/gather 算法,队列中的项放在连续的内存区中. (2)数据放置优化算法. 该算法使用单队列,同时优化了单队列中每一个单元的内存放置策略,使得每一项独占一个 Cache 行. 该算法和原始算法比较可以获得 Cache 优化技术的效果. (3)多核优化的算法. 该算法不仅仅对队列的放置进行了 Cache 行的优化,同时实现了 4.1 节的多层次优化算法. 对比可以得到多层次优化技术的效

果.本实验使用的测试程序为 IMB(Intel Micro-benchmark) 3.2,测试 MPI_Barrier 的性能.

测试结果如图 4 所示,从图中可以看出,数据放置优化算法由于更好地组织了单队列结构,降低了 Cache 假相关导致的 Cache 缺失,因此能够在 16 进程的时候将 MPI_Barrier 的开销从 5.7μs 降低到 4.97μs.而多核优化算法采用了多层次通信优化,进一步地提高了 MPI_Barrier 的性能,仅为 2.28μs.相比原始算法带来了 60%的性能提升.当进程数据进一步增加的时候,节点间采用 RDMA 进行 Barrier 操作的收集和分发,节点内集合通信优化算法的效果仍然具有较好的可扩展性,在 256 个进程时多核优化算法仍然能够带来 11.03%的性能提升,这也验证了本集合通信算法优化效果的可扩展性.

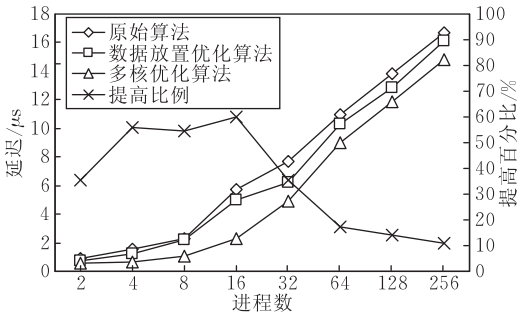


图 4　MPI_Barrier 优化效果

为了获得 MPI_Barrier 性能提高的原因,使用 Oprofile 测试了平均一次 Barrier 操作的 L2 Cache Miss 和 CPU 芯片间的 Cache-to-Cache 传输,测试结果如表 1 所示.可以看出,多核优化算法使 L2 Cache Miss 从 8.5/次降低到 6.9/次,这表明多核感知的算法提高了 Cache 利用率,另一方面,使得芯片之间的 Cache-to-Cache 操作从 6.1/次降低到 4.9/次,这表明多层次算法将 Barrier 操作聚合到芯片内部,降低了芯片之间的 Cache-to-Cache 操作,因此带来了多核感知算法的性能提升.

表 1　MPI_Barrier 优化效果 Oprofile 比较

	原始算法	多核优化算法
L2 Cache Miss	8.5	6.9
Cache-to-Cache Transfer	6.1	4.2

5.2　MPI_Bcast

为了测试 MPI_Bcast 优化技术的效果,本文使用几种 MPI_Bcast 实现进行比较:(1)原始算法.这种算法中,节点内操作小消息采用 binomial tree 算法,大消息采用了并发拷贝的方式从共享内存中间获取数据.(2)限制并发算法.该算法针对大消息的并发拷贝的并发度进行限制,并不改变共享缓冲区的分配方式,和原始算法相比,可以得到限制并发方式对性能的影响.(3)多核感知的算法.在该算法中,首先对共享内存分段,同时对每一段内存进行 NUMA 感知的放置,在访问时限制了对每一段内存并发访问的并发度.实验采用 IMB 3.2 测试程序进行测试,测试了 16 进程的情况,测试结果如图 5 所示.

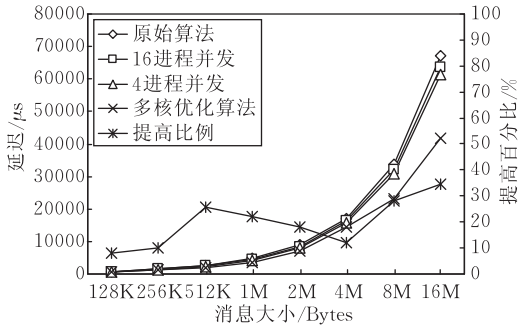


图 5　MPI_Bcast 优化效果

从结果中可以看出,将并发访问共享内存的并发度从 16 降到 4,能够将 MPI_Bcast 在 512KB 运行时间缩短 15%,在 16MB 的时候,能够缩短 3.5%,这种性能提升是由于限制了并发,降低了内存竞争导致的开销.

对于多核优化的算法,由于使用 NUMA 感知的数据访问,同时利用了 NUMA 系统中的 4 个内存控制器,提高了数据访问的并发度,同时由于限制一个内存控制器上的并发访问度为 4,大大降低了内存竞争,因此带来了很高的性能提升,从图中可以看出,在 128KB 的时候,带来了 34.6%的性能提升,在 16MB 的时候,多核优化算法带来了 34.58%的性能提升,运行时间为 41648.18μs.

5.3　MPI_Alltoall

为了测试Cache友好的优化算法对 MPI_Alltoall 的性能影响,本文对比测试了两种 MPI_Alltoall 的实现:(1)原始算法.原始算法在大消息时采用 pairwise-exchange 算法.(2)多核优化算法.本算法采用 data-tiling 算法,提高 Cache 的利用率.实验采用 IMB 3.2 进行测试,测试了 16 进程时的 MPI_Alltoall 的运行时间.

测试结果如图 6 所示,从结果中间可以看出,data-tiling 技术能够带来 MPI_Alltoall 的性能提升,在 64KB 的时候带来 4%的性能提升,在 16MB 的时候能够带来 14%的性能提升.这种性能的提升主要来自于多核优化的算法有效利用了 Cache,同

时相应地降低了对内存访问的需求，因此带来了性能的提升.

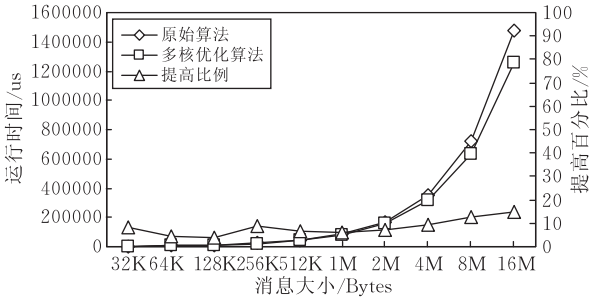


图 6 MPI_Alltoall 优化效果

6 结论和下一步工作

由于多核在高性能计算中得到了广泛应用，因此了解多核对集合通信带来了什么样的影响并针对这些特点进行优化，对多核环境下的集合通信具有重要的意义. 本文分析了多核环境对集合通信的影响，一方面片内通信的快速的数据通路能够提高小消息的通信，因此需要集合通信算法尽可能利用片内通信；另一方面，多核共享内存带宽和 Cache 导致的竞争会严重地降低集合通信的性能，因此需要集合通信算法在大消息时降低内存竞争. 本文针对多核的特点，提出了 3 种优化方法：多层次的集合通信算法、限制并发和 NUMA 感知的优化方法以及 Cache 友好的优化方法，并分别在 MPI_Barrier、MPI_Bcast、MPI_Alltoall 中进行了验证. 实验表明，这些优化方法能够较好地利用多核的特点，提高集合通信的性能. 在 16 进程规模情况下，MPI_Barrier 能够提高 60%，MPI_Bcast 能够提高 34.58%，MPI_Alltoall 能够提高 14%.

在本文的工作基础上，后续将继续研究如何优化其它的 MPI 集合通信算法，如 MPI_Allgather、MPI_Reduce 等，同时将在大规模平台上面测试多核环境下集合通信优化对并行应用性能的影响.

参 考 文 献

[1] Asanovic K, Bodik R, Catanzaro B C, Gebis J J, Husbands P, Keutzer K, Patterson D A, Plishker W L, Shalf J, Williams S W, Yelick K A. The landscape of parallel computing research: A view from berkeley. University of California at Berkeley, Berkeley: No. UCB/EECS-2006-183, 2006

[2] Rabenseifner Rolf. Automatic MPI counter profiling of all users: First results on a CRAY T3E 900-512//Proceedings of the Message Passing Interface Developers and Users Conference 1999 (MPIDC). Atlanta, USA, 1999: 35-42

[3] Chai Lei, Gao Qi, Panda Dhabaleswar K. Understanding the impact of multicore architecture in lcuster computing: A case study with Intel dual-core system//Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07). Rio de Janeiro. Brazil, 2007: 471-478

[4] Tipparaju Vinod, Nieplocha Jarek, Panda Dhabaleswar K. Fast collective operations using shared and remote memory access protocols on clusters//Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2003). Nice, France, 2003: 10

[5] Wu Meng-Shiou, Kendall Ricky A, Aluru Srinivas. Exploring collective communications on a cluster of SMPs//Proceedings of the 7th International Conference on High Performance Computing and Grid in Asia Pacific Region (HPCAsia 2004). Tokyo Area, Japan, 2004: 114-117

[6] Mamidala Amith R, Kumar Rahul, De Debraj, Panda Dhabaleswar K. MPI collectives on modern multicore clusters: Performance optimizations and communication characteristics//Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'08). Lyon, France, 2008: 130-137

[7] Kumar Rahul, Mamidala Amith R, Panda Dhabaleswar K. Scaling alltoall collective on multi-core systems//Proceedings of the IEEE International Symposium on Parallel and Distributed Processing 2008 (IPDPS 2008). Miami, USA, 2008: 1-8

[8] Graham Richard L, Shipman Galen M. MPI support for multi-core architectures: Optimized shared memory collectives//Proceedings of the Recent Advances in Parallel Virtual Machine and Message Passing Interface. Dublin, Ireland, 2008: 130-140

[9] Tu Bi-Bo, Fan Jian-Ping, Zhan Jian-Feng, et al. Accurate analytical models for message passing on multi-core clusters//Proceedings of the 2009 Parallel, Distributed and Network-Based Processing (PDP 2009). Weimar, Germany, 2009: 133-139

[10] Cheng Liqun, Carter John B. Fast barriers for scalable cc-NUMA systems//Proceedings of the International Conference on Parallel Processing 2005 (ICPP 2005). Oslo, Norway, 2005: 241-250

[11] Thakur Rajeev, Rabenseifner Rolf, Gropp William. Optimization of collective communication operations in MPICH. International Journal of High Performance Computing Applications, 2005, 10: 49-66

[12] Kadayif Ismail, Kandemir Mahmut T. Data space-oriented tiling for enhancing locality. ACM Transactions on Embedded Computing Systems, 2005, 4(2): 388-414

ZHANG Pan-Yong, born in 1981, Ph. D. candidate. His research interests focus on high performance communication.

MENG Dan, born in 1965, professor, Ph. D. supervisor. His main research interests include HPC and distributed systems.

HUO Zhi-Gang, born in 1978, Ph. D., associate professor. His main research interests include focus on tolerance in HPC.

Background

The problem discussed in this paper is belongs to optimization of high performance communication optimization. In recent years, as multicore is widespreadly deployed in HPC systems, performance optimization of collectives in multicore clusters is an active research area. Past researches do not analysis how multicore impacts the performance of collectives, and the performance of collectives is not well enough in multicore environments.

This paper investigates the impacts of multicore on collective communication and proposes several methods to improve its performance. Comparing with old algorithms, the performance has been improved.

The research in this paper belongs to the National Natural Science Foundation of China 863 project "Dawning 5000 High Performance Computer" grant No. 2006AA01A102, and the principal of the project is Ninghui Sun. The first author Zhang Panyong of this paper is a graduate student, and his supervisor is Meng Dan.

Petascale scale HPC systems have to face several challenges such as efficiency, scalability, and the Dawning 5000 project focuses on solving these challenges, provides key technology to achieve petascale and build a practical 100P system for validation. It researches on new HPP architecture, and propose new system controller, new communication systems and new communication model.

Now, the research group has achieved some productions. Several papers have been published in "Chinese Journal of Computer", "Journal of Computer Research and Development", and other Journals, and IPDPS, HotI, HiPC, and other conferences.

Collective communication is important for the efficiency in large scale HPC systems, and the research content of this paper is an important part of optimization for collective communications in HPC systems.