多核机群下MPI程序优化技术的研究

王 洁[1,2] 袁璐洁[1,2] 曾 宇[3]
(中国科学院计算技术研究所 北京 100190)[1] (中国科学院研究生院 北京 100049)[2]
(北京市计算中心 北京 100005)[2]

摘 要 多核处理器的新特性使多核机群的存储层次更加复杂,同时也给MPI程序带来了新的优化空间。国内外学者提出了许多多核机群下MPI程序的优化方法和技术。测试了3个不同多核机群的通信性能,并分别在Intel与AMD多核机群下实验评估了几种具有普遍意义的优化技术:混合MPI/OpenMP、优化MPI运行时参数以及优化MPI进程摆放,同时对实验结果和优化性能进行了分析。

关键词 多核机群,存储层次化,MPI程序优化,混合MPI/OpenMP,MPI运行时参数,MPI进程摆放

Research of MPI Programs Optimization Technology on Multi-core Clusters

WANG Jie[1,2] ZHONG Lu-jie[1,2] ZENG Yu[3]
(Institute of Computing Technology, Chinese Academy of Science, Beijing 100190, China)[1]
(Graduate University of Chinese Academy of Sciences, Beijing 100049, China)[2] (Beijing Computer Center, Beijing 100005, China)[3]

Abstract The new features of multi-core make the memory hierarchy of multi-core clusters more complex, and also add the optimization space for MPI programs. We tested the communication performance of three different multi-core clusters, and evaluated some general optimization technologies, such as hybrid MPI/OpenMP, tuning MPI runtime parameters and optimization of MPI process placement in Intel and AMD multi-core cluster. The experiments result and optimization performances were also analyzed.

Keywords Multi-core cluster, Memory hierarchy, MPI programs optimization, Hybrid MPI/OpenMP, MPI runtime parameters, MPI process placement

1 引言

随着多核技术以及现代网络技术的发展,越来越多的机群采用多核处理器作为核心部件,基于多核技术的机群已经成为高性能计算领域的主流平台[1]。基于2010年发布的世界Top500的超级计算机排名,约88%的超级计算机采用了Intel和AMD的多核芯片,并且约84.8%的采用了机群结构[2]。多核的出现将机群带入了多核时代,多核处理器带来了不同于单核处理器、SMP的革命性结构变化,多核机群节点存储和通信层次越来越复杂。消息传递接口MPI[3](Message Passing Interface)是机群下最常用的并行编程模型,被广泛应用于分布式以及共享内存系统,是目前并行编程事实上的标准[1]。随着多核技术更加广泛地应用于机群,多核环境下MPI程序的性能优化成为了研究的热点。虽然算法的数据局部性、负载均衡等因素是影响MPI并行应用性能的因素,但其与具体的特定应用特性有关,直接将现有的MPI程序移植到多核机群平台上,应用的性能和可扩展性并没有得到多大的改进[4]。

本文分析了3种不同架构下的多核机群存储层次,实验分析了3种同构多核机群的通信性能,并研究了多核环境下并行应用中主要的编程规范MPI的优化空间,分析了混合MPI/OpenMP、优化集合通讯、优化运行时参数以及进程拓扑等优化技术。实验证明这些方法对多核机群下MPI程序优化的有效性,并具有一定的普遍适用性和研究意义。

2 多核机群通信性能评估

目前主流的处理器厂商都提供多核处理器,它们在cache的层次组织和内存访问模式上有所差异,但都表现出类似的性能特征。几种常见的多核处理器cache的设计为:Intel的双核和四核Xeon处理器中,同一个chip上的核有各自独立的相对较小的L1 cache,同时通过总线方式共享相对较大的L2缓存。AMD的双核和四核Opteron处理器中,同一个chip上的核有自己独立的、缓存数据互斥的、相对较大的L1和L2 cache,并通过超传输接口技术交换传输数据,同时共享相对较小的L3 cache。IBM的Power4和Power5的chip内双核共享L2 cache,而Power6的chip内双核有自己独立的L2 cache,同时共享L3 cache。Sun Niagra的chip内八核处理器共享L2 cache。得益于多核节点的cache层次化特征,多核处理器核间数据存取具有高带宽、低延迟的特点,多核节点chip内核间的通讯性能可以达到节点间通讯性能的数倍以

到稿日期:2010-11-21 返修日期:2011-04-10 本文受奥地利蒂罗尔州未来基金会基金(P7030-015-024)资助。
王 洁(1977-),女,博士,讲师,主要研究方向为并行计算、机器学习、数据挖掘等,E-mail:wangjie@ncic.ac.cn;袁璐洁(1978-),女,博士生,讲师,主要研究方向为编译技术、并行计算等;曾 宇(1973-),男,博士,高工,主要研究方向为高性能计算机技术、云计算等。

上。

为了评估多核机群的通讯性能，选择了3个不同的实验平台，配置分别为：Cluster 1为一个4节点的同构InfiniBand机群，每个节点有4颗Intel 2.7GHz Xeon四核处理器，即每个节点16核，同一个chip内的双核共享8M的L2 cache，节点间的通过Infiniband交换机互联。Cluster2为一个4节点的同构nfiniBand机群，每个节点有8颗AMD Opteron 8356四核2.4GHz处理器，即每个节点32核，同一个chip内的核有独立的512k L2 cache，同时共享2M的L3 cache，节点间通过5Gb/s 5μs延迟的Infiniband交换机互联。Cluster 3为一个10节点的同构Infiniband机群，每个节点有4颗AMD 2.4 GHz Opteron 880双核处理器，即每个节点8核，同一个chip内的双核各自有独立的1M L2 cache，同时共享2M的L3 cache，节点间通过5Gb/s 5μs延迟的Infiniband交换机互联。

使用Intel MPI基准(IMB)中ping-pong基准测试3个机群中缓存结构对通信性能的影响，图1—图3显示了3种多核机群chip内、chip间以及节点间的通信带宽。

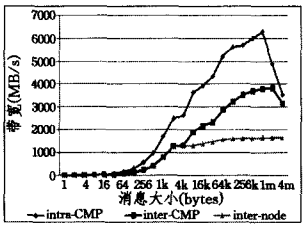

图1 四核Intel机群通信带宽

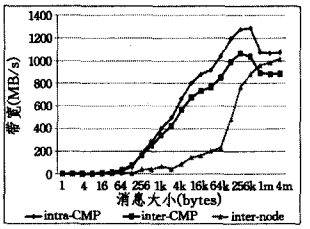

图2 四核AMD机群通信带宽

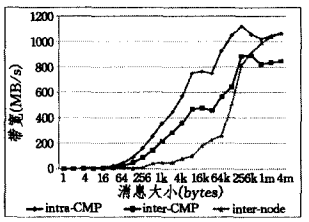

图3 双核AMD机群通信带宽

图1—图3显示，无论是Intel或AMD多核机群，同一个chip内的核间通信性能要优于不同chip间和节点间通信性能，特别是对于小消息和中型消息。这是由于chip上最外层的cache通常采用多核共享的方式(Intel的双核或四核处理器以及AMD四核处理器)或各核的最外层cache间在片内有专用的高速通信通道(AMD的双核处理器)，因此如果消息能fit到cache中，chip内核上的通信只包括两次cache的操作。与AMD四核处理器相比，Intel的四核处理器的chip内通信具有更明显的性能提升，这是由于Intel同一个chip上的双核共享L2 cache，而AMD的多核具有独立的二级cache、共享的三级cache。

从以上对于通信性能的实验可以得出，优化多核下MPI程序要充分考虑多核处理器存储层次化的特点，提高节点内多级cache的利用率，尽可能使通信缓冲区fit到cache中以减少内存拷贝的次数。

3 多核机群下MPI程序优化技术的研究

3.1 混合MPI/OpenMP优化技术

近年来，高性能计算领域的大多数系统都采用了混合的硬件设计，即节点级采用共享内存的多核CPU，分布式内存的节点间通过网络进行互联。因此混合MPI与OpenMP被认为是一种自然的编程模型，即在节点内采用OpenMP进行线程级的并行以及在节点间采用MPI进行进程级并行[5]。与MPI相比，在多核下采用混合MPI/OpenMP编程模型有可能带来一定的性能提升，这是因为MPI进程间的通讯开销较高，使用线程和共享内存的OpenMP模型更加有效。

分别在Intel和AMD的多核机群下使用NAS的多zone基准来比较MPI与混合MPI/OpenMP两种编程模型的性能。Intel多核机群由四核2.5GHz的Intel Xeon CPU L5420节点组成，同一个chip上的两个核共享12M的二级cache。AMD多核机群由四核2.4GHz的AMD Opteron 8356节点组成。AMD双核机群由双核2.4GHz的AMD Opteron 880节点组成。图4—图6显示了3种多核机群下混合编程模型的性能对比。

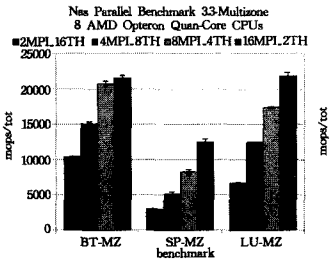

图4 AMD四核机群上混合编程模型性能对比

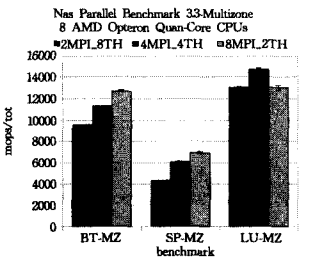

图5 AMD双核机群上混合编程模型性能对比

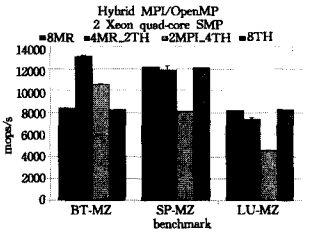

图6 Intel四核机群上混合编程模型性能对比

如图4—图6所示，对于不同的应用和问题规模，即使是在特定的系统架构下，最佳的混合编程模型配置也可能会有所不同。混合编程模型的性能在某些应用中优于MPI，例如Intel四核机群上的BT-MZ基准。chip内的双核共享L2 cache，Intel四核机群上测试的NAS基准在4MPI-2TH下性能都优于2MPI-4TH。但对于一些应用，MPI的性能依然比混合编程模型要好。

从以上实验分析可知，对于特定的多核机群结构和具体问题，混合MPI/OpenMP性能可能优于MPI，但发现其最优的编程模型并不容易。最优的MPI进程数和每节点OpenMP的线程数依赖于多个因素，例如：问题的规模、输入的数据。同时最优的编程模型需要建立在明确系统结构的基础上，包括：chip内、chip间、节点间的通信带宽和延迟，以及节点内和节点间的硬件拓扑结构。

3.2 多核机群下MPI运行时参数的优化

MPI应用的参数调优近年来被学术界广泛研究。目前主流的MPI库实现(Open MPI、MPICH等)都提供了可调的运行时参数机制，允许用户根据特定的应用需求、硬件以及操作系统来调优运行时的参数以提升MPI应用的性能。例如

可以根据通讯消息的大小来修改点到点通讯采用的协议，即修改 MPI 库中由立即通讯协议(Eager)转为集中通讯协议(Rendezvous)的阈值参数。可调的运行时参数对多核机群下的 MPI 应用的性能有着重要的影响。

文献[6]中设计并开发了工具 OTPO 来优化 Open MPI 的运行时参数。OTPO 对 Open MPI 通信模式以及性能度量的可调参数的大量组合进行系统测试，以选择给定平台下指定基准的最优参数。但 OTPO 执行一次完整的测试只服务于一个基准程序。文献[7]中建立了基于机器学习的运行时参数优化模型，但所做工作仅限于单个的多核节点，并未对方法的可扩展性进行讨论。我们将此模型进行扩展，研究了在多核机群下运行时参数的调优方法，并且可以将其运用到兼有点到点通信以及集合通信的 MPI 应用的性能优化中。

我们设计实现了一种通用的多核机群下 MPI 运行时的参数优化模型，它能自动为给定软、硬件结构的多核机群下的 MPI 程序预测接近最优的运行时参数组合。所提出的预测模型基于机器学习中的人工神经网络方法，通过对预测模型的离线训练和在线学习，能自动为未知的 MPI 应用预测接近最优的运行时参数。要预测的 MPI 程序由对源码运行一次得到的动态特征和通讯器大小等静态特征来共同描述。具体方法包括 3 个阶段：要调优的运行时参数选择、模型训练与运用已训练模型进行参数优化。图 7 描述了主要的模型训练与参数优化步骤。标准的机器学习技术——人工神经网络被用来构建优化模型，NAS 并行基准套件 2.4 的 IS 基准和 LU 基准被用来评估构造模型优化 MPI 运行时参数的准确度。

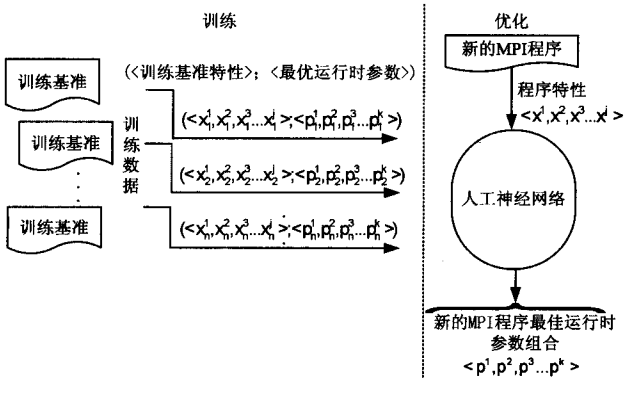

图 7　模型训练与参数优化

在一个 10 节点的同构 Infiniband 机群上测试了方法的有效性。实验平台中，每个节点有 4 颗 AMD 2.4GHz Opteron 880 双核处理器(共 40 个处理器，80 个核)。同一个 Socket 内的双核各自有独立的 1M 二级缓存和 2G RAM，节点间通过 5Gb/s 5μs 延迟的 Infiniband 交换机互联。操作系统为 64 位 CentOS(内核为 2.6.18)，MPI 库为版本 1.2.6 的 OpenMPI。Matlab 的神经网络工具箱被用来训练基于人工神经网络的优化模型。

图 8 和图 9 分别对比了 NAS 基准中的 IS 基准和 LU 基准，在不同问题规模下，通过调整运行时参数可以获得的实际最大加速比与用本文模型预测得到的优化的运行时参数带来的加速比。对于 IS 基准，通过调优运行时参数可以获得的最大加速比为 20%(Class A)，对于取值只有 0 和 1 两种可能的 mpi_paffinity_alone 参数，神经网络预测的准确度较高，约为 75%，对于其他数值型参数，神经网络预测的准确度相对较低，例如对于问题规模较大的 Class C，实际最佳运行时参数组合为{1024,32768,8192,262144,1}，使用 ANN 预测结果为{512,32768,1024,262144,1}。图 9 显示了用基于 ANN(人工神经网络)方法预测得到的优化运行时参数为 IS 基准带来的加速比平均达到实际最大加速比的 95%以上。

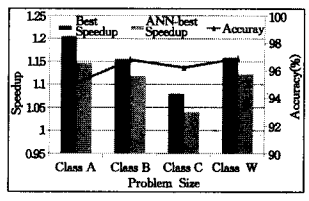

图 8　ANN 为 IS 基准预测参数产生的加速比与预测产生加速比的准确度

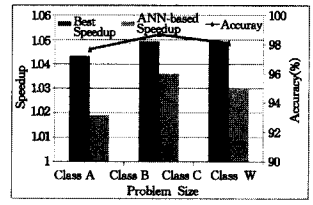

图 9　ANN 为 LU 基准预测参数产生的加速比与预测产生加速比的准确度

此研究的后续工作一方面将通过增大训练数据集，以及改进 ANN 训练数据前处理与后处理方式来进一步提高神经网络的预测准确度。另一方面将进一步抽取更丰富的 MPI 程序特征以提高预测参数的准确率，例如在计算密集的应用中，将进程与核绑定可能会降低性能，因此可以考虑抽取程序中计算与通讯比例特征。

3.3　多核机群下 MPI 进程摆放的优化

多数 MPI 应用中，线性的进程编号并不能很好地反映实际工作中进程间的通信模式，有时根据问题需要，要将进程映射到二维、三维网格甚至更加复杂的图结构上。MPI 标准本身提供了一些机制使用户可以创建和管理进程的拓扑，例如笛卡尔拓扑和图拓扑。但这些拓扑仅在逻辑上实现与问题领域特点相匹配，与底层的硬件设备无关，因此是虚拟的进程拓扑。

考虑到多核机群通信性能层次化的特点，结合多核机群特点，合理摆放 MPI 进程能充分发挥多核的优势。例如，一些进程间通信比其他进程频繁，则应该将它们重新分组并放在同一个节点内，以增加节点内的通信机会并减少节点间通信。因此在多核机群下，根据并行应用本身的通信模式和多核机群的通信性能来摆放 MPI 进程对于优化 MPI 应用具有重要意义。

David Solt 提出了一种基于 profile 的进程摆放优化策略[8]，即通过获取目标多核机群的通信性能 profile，然后根据应用的 Instrument 运行获得应用的通信 profile。在已知应用的通信模式和目标机群 core 间的通信性能后，可以用向量 $BW_{xy}=[bw_0,bw_1,\cdots,bw_n]$ 表示 core 间的通信带宽，MPI 进程间通信率可以用向量 $M_{ab}=[b_0,b_1,\cdots,b_n]$ 表示，全局的开销可以用公式 $P=\Sigma M_{P(x),P(y)}\times BW_{x,y}$ 表示。则最优的进程映射问题可以转化为计算图间的最小编辑距离等方法来求解。Mercier 设计了一种根据目标机群体系结构特点，用图论的方法来摆放 MPI 进程的方法[9]。多核机群的通信性能用一个完全加权无向图来表示，其中，权重表示机群内 core 间的通信带宽。另一个完全加权无向图用来表示进程间的通讯模式，其中权重表示进程间交换的数据量。两个图生成后，用工具软件解决通讯模式图嵌入通信带宽图的 NP 问题，得到

进程的优化摆放结果。

图 10 对比 Round-Robin 与优化进程摆放后，NAS 基准中 BT、CG 等基准(class C)的节点内通信率的对比，优化后的进程摆放策略能有效提高节点内的通信率。虽然通过合理的进程摆放能提高 MPI 程序节点内的通信率，但进一步提升 MPI 程序的性能还需要进一步提高 cache 的利用率。

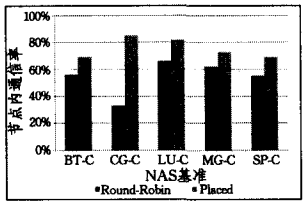

图 10　两种不同进程摆放策略下 NAS 基准全局节点内通信率比

结束语　多核机群的存储层次化使得多核机群具有与传统的 SMP 平台显著不同的特征，因此，未经优化直接运行在多核机群上的 MPI 并行程序无法得到最优的性能。本文在对比了多核机群性能特征的基础上，实验分析了几种可行的、具有普遍意义的 MPI 程序优化技术。实验证明，这几种方法对多核机群下 MPI 程序的性能提升具有一定的有效性和通用性。后续工作将继续研究多核机群下 MPI 程序可能的优化空间，例如多核机群下 MPI 集合通信的优化等。

参 考 文 献

[1] Chai L, Lai P, Jin H W, et al. Designing an efficient kernel-level and user-level hybrid approach for mpi intra-node communication on multi-core systems[C]// ICPP '08: Proceedings of the 2008 37th International Conference on Parallel Processing. Washington, DC, USA, IEEE Computer Society, 2008

[2] TOP500 Team[R]. TOP500 Report for June 2009. http://www.top500.org

[3] MPI: A Message-Passing Interface Standard[S]. http://www.mpiforum.org/docs/mpi-11-html/mpi-report.html

[4] Tu Bi-bo, Zou Ming, Zhan Jian-feng, et al. Multi-core Aware Optimization for MPI Collectives, poster[C]// The 2008 IEEE International Conference on Cluster Computing (Cluster 2008). Tsukuba, Japan, 29 September-1 October, 2008

[5] Rabenseifner R, Hager G, Jost G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes[C]// Didier El Baz, et al., eds. Proceedings of the 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009). Weimar, Germany, Computer Society Press, Feb. 2009: 427-236

[6] Chaarawi M, Squyres J M, Gabriel E, et al. A tool for optimizing runtime parameters of open mpi[C]// Proceedings of the 15th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Berlin, Heidelberg: Springer-Verlag, 2008: 210-217

[7] Pellegrini S, Wang Jie, Ahringer T, et al. Optimizing MPI Runtime Parameter Settings by Using Machine Learning[C]// Proceedings of the 16th Euro PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Espoo, Finland, 2009

[8] Solt D. A profile based approach for topology aware MPI rank placement[OL]. http://www.tlc2.uh.edu/hpcc07/Schedule/speakers/hpcc_hp-mpi_solt.ppt, 2007

[9] Mercier G, Clet-Ortega J. Towards an Efficient Process Placement Policy for MPI Applications in Multicore Environments [C]// Proceedings of the 16th Euro PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Espoo, Finland, 2009

(上接第 266 页)

[11] Klosowski J T, Held M, Mitchell J S B, et al. Efficient collision detection using bounding volume hierarchies of k-Dops[J]. IEEE Transactions on Visualization and Computer Graphics, 1998, 4(1): 21-36

[12] Hubbard P M. Approximating polyhedra with spheres for time critical collision detection[J]. ACM Transactions on Graphics, 1996, 15(3): 179-210

[13] Gottschalk S, Lin M C, Manocha D. OBB Tree: a hierarchical structure for rapid interference detection[C]// The Proceedings of SIGGRAPH. 1996: 171-180

[14] Ramírez E, Navarro H, Carmona R, et al. Optimizing Collision Detection Based on OBB Trees Generated with Genetic Algorithm[C]// IV Iberoamerican Symposium in Computer Graphics-SIACG. 2009: 1-7

[15] Fares C, Hamam Y. Collision detection for rigid bodies: A state of the art review[C]// GraphiCon. 2005

(上接第 272 页)

[3] 钱振兴，程义民，谢春辉，等. 基于嵌入矩阵的二值图像隐藏方法[J]. 电路与系统学报，2008，13(6)：128-131

[4] 谢建全，阳春华，谢勋，等. 一种大容量的二值图像信息隐藏算法[J]. 小型微型计算机系统，2008，29(10)：1874-1877

[5] 王国新，平西建，张涛，等. 空域 LSB 信息伪装及其隐写分析[J]. 计算机工程，2008，34(1)：173-174，189

[6] 王继军，张显全，韦月琼. 基于 LSB 的数字图像分存隐藏算法[J]. 计算机工程，2008，29(23)：6167-6169

[7] 侯整风，高汉军. 一个新的基于多重秘密共享的图像隐藏方案[J]. 计算机应用，2008，28(4)：902-905

[8] LIBAI. Areliable(k, n) image secretsharing scheme[C]// 2nd IEEE International Symposium on Dependable, Autonomic and SecureComputingIndianapolis. Indiana, USA: IEEE Press, 2006: 31-36

[9] 杨全周，蔡晓霞，陈红. 一种基于游程编码的二值图像隐藏方案[J]. 舰船电子对抗，2008，31(2)：86-88