国产百万亿次机群系统 Alltoall 性能测试与分析

饶 立[1] 张云泉[1,2] 李玉成[1]
(中国科学院软件研究所并行计算实验室 北京 100190)[1]
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[2]

摘 要 随着高性能计算机的应用和发展,并行应用程序所使用的处理器数越来越多,进程间的通信量也不断增多,这对应用程序的性能有很大影响。在采用一种快速傅里叶变换 HFFT 对曙光 5000A 进行性能测试时发现,MPI 集合通信函数 MPI Alltoall 的巨大通信开销是并行程序设计的瓶颈。为此,对现有主流 Alltoall 算法在曙光 5000A 和深腾 7000 上进行性能测试与分析,以期对未来的 Alltoall 算法的优化工作做出贡献。利用不同消息长度和不同进程数测试了 Alltoall 函数多种算法的性能,这些算法包括二维网格算法、三维网格算法、Bruck 算法、原始算法、成对交换算法、递归倍增算法、环算法以及 LAM/MPI 中的简单算法等。实验结果表明:消息长度较小时,在曙光 5000A 上采用原始算法和 Bruck 算法的性能较好,而在深腾 7000 上用时较少的算法是简单算法和 Bruck 算法;对于长消息,曙光 5000A 上最优的算法是环算法,深腾 7000 上成对交换性能最优。

关键词 集合通信,Alltoall,曙光 5000A,性能测试与分析

Performance Test and Analysis of Alltoall Collective Communication on Domestic Hundred Trillion Times Cluster System

RAO Li[1] ZHANG Yun-quan[1,2] LI Yu-cheng[1]
(Lab of Parallel Computing, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract As rapid development of the high performance computers, more and more cores are used and thus lead to more and more communication which debases the perfor-mance of parallel applications greatly. In the test of the performance of Dawning 5000A by a kind of Fast Fouler Transform (HFFT), we found out that the huge overhead time of MPI_Alltoall is the bottleneck of HFFT. Thus, this paper aimed to test and analyze the leading Alltoall algorithm on Dawning 5000 and Deepcomp 7000 hoping to do a favor to further collective communication optimization. In this paper, the leading Alltoall algorithms such as 2D_Mesh, 3D_Mesh, Bruck, MPICH native, Pair, recursive doubling, Ring, LAM/MPI simple were recounted and tested with different message size and core numbers. The conclusion is that for short message MPICH native and Bruck performs well on Dawning 5000A while the lower time consuming algorithms on Deepcomp 7000 are LAM/MPI simple and Bruck; when the message size is medium and large, the best choice for Dawning 5000A is Ring while the optimal algorithm on Deepcomp 7000 is Pair.

Keywords Collective communication, Alltoall, Dawning 5000A, Performance test and analysis

1 引言

随着高性能计算的发展,越来越多的应用领域如数值天气预报、核武器、石油勘测、地震数据处理、飞行器数值模拟、生物信息处理等都需要每秒数十万亿次、百万亿次乃至千万亿次浮点运算的计算机。然而高性能计算机存在的突出问题是实际获取性能低,一般的应用程序仅能发挥峰值性能的5%到20%。采用快速傅里叶变换(HFFT)算[8,9]法对曙光5000A进行性能测试,发现MPI集合通信函数MPI_Alltoall的巨大时间开销是并行程序设计的瓶颈之一。优化集合通信性能是提高并行程序效率的重要途径之一,也一直是学术界的研究热点。因此,本文通过测试和分析 Alltoall 的性能为后续集合通信优化工作做准备。本文第1节为引言;第2节介绍了集合通信算法性能测试分析的相关工作;第3节简要介绍了 Alltoall 的主流算法;第4节介绍了测试环境和测试方法;第5节是算法性能的测试结果和分析;最后是总结和未来的工作。

2 相关工作

集合通信一直是并行计算领域的研究热点,算法优化和

到稿日期:2009-09-14 返修日期:2009-12-01 本文受国家自然科学基金(No. 60303020),国家自然基金重点项目(No. 60533020),国家863(No. 2006AA01A102, No. 2006AA01A125)资助。

饶 立(1987—),男,硕士生,主要研究方向为并行计算,E-mail:renzherl@126.com;张云泉(1973—),男,研究员,博士生导师,主要研究方向为高性能计算及并行数值软件、并行计算模型、并行数据库、海量数据并行处理;李玉成(1961—),男,研究员,主要研究方向为并行软件。

性能测试分析的工作也较多。文献[1]比较了短消息下Bruck算法和MPICH原始算法(isend-irecv)的性能并指出当消息长度超过256字节时原始算法更优;文献[2]测试比较了Alltoall主流算法和自适应的Alltoall算法的性能,得出自适应Alltoall可提升算法性能;文献[3]测试分析了不同消息长度和处理器个数的各种MPI Allgather算法在曙光4000A以及深腾6800上的性能。

3 Alltoall算法简介

数据转置Alltoall是指通信器中的所有进程从其他进程收集数据,同时将自己的数据发送给其他进程,它的作用相当于将一个分布式存储的数据场在处理器间进行一次转置。本次测试选取了Alltoall如下几种实现方法:1)MPICH 4种算法[1,4]即适合短消息的Bruck算法、适合中等长度消息的原始算法、适合长消息的环算法和成对交换算法;2)LAM/MPI中的简单算法[5];3)通过调用MPI Allgather来间接实现Alltoall的递归倍增算法[1,2];4)二维三维网格等间接算法[2,6]。

原始算法中所有进程直接调用非阻送接收操作来实现Alltoall,在每一次循环 i 中,每个进程 *rank* 先调用MPI_Irecvs,然后调用MPI_Isend发送消息,最后调用MPI_Waitall。

成对交换算法需要 $p-1$ 步操作,对于第 k 步,每个进程 *rank* 和进程 $k \otimes rank$ 直接通信。

环算法和成对交换算法类似,一共需要 $p-1$ 步操作,对于步骤 k,进程从 $(rank-k)\%\ p$ 接收消息,同时发送消息到进程 $(rank+k)\%\ p$。

LAM/MPI简单算法与原始算法类似。假设 $i \to j$ 表示进程 i 发送消息给进程 j,则进程 i 的通信顺序依次是 $i \to 0$, $i \to 1, \cdots, i \to p-1$。

上述算法都需要 $p-1$ 步,每一步发送数据大小为 n,根据 $\alpha+n\beta$ 模型[2],其估算时间为:

$$T=(p-1)(\alpha+n\beta) \tag{1}$$

Bruck算法[7]需要 $\lg p$ 步完成Alltoall操作,如果进程数为2的 n 次幂,则每个进程每步需要发送和接收的数据大小为 $n/2$,因此Bruck算法估算时间为:

$$T=\lg\alpha+\frac{np}{2}\lg p\beta \tag{2}$$

递归倍增算法首先调用All-gather操作[1],将每个进程的数据都收集到其他进程,然后其他进程将自己需要的数据拷贝到接收缓存。消息长度为 np 的All-gather操作估算时间即:

$$T=\lg p\alpha+(p-1)pn\beta \tag{3}$$

二维三维网格算法适用于处理器逻辑拓扑结构为 $p=x*y$ 的二维网格和 $p=x*y*z$ 的三维网格[2],其估算时间为:

$$T_{2d}=2\times(\sqrt{p}-1)\alpha+(p-1)pn\beta \tag{4}$$

$$T_{3d}=3\times(\sqrt[3]{p}-1)\alpha+(p-1)pn\beta \tag{5}$$

4 测试环境及方法

4.1 测试环境

本文的实验平台为两台国产高性能机群系统:曙光5000A和深腾7000。表1对比介绍了两个平台的节点和处理器。

表1 曙光5000A和深腾7000节点和处理器信息

	曙光5000A	深腾7000
节点个数	1920	1240
处理器类型	Opteron 8347HE	Xeon E5450
处理器主频	1.9GHz	3GHz
核数/CPU	4	4
CPU数/节点	4	2
L1缓存	64kB	64kB
L2缓存	512kB	6144kB
L3缓存	2048kB	N/A
每个节点内存容量	64GB	32GB
每个节点计算峰	121.6 GFlops/s	96 GFlops/s

1)曙光5000A

作为当前中国最快的超级计算机,曙光5000A的设计最高峰值是233TFlops,一共由1920个节点构成,每个节点包含4路4核处理器和64G Linpack的实测值为180.6TFlops,在2008年11月的Top500排名中位于世界第10。曙光5000A采用了基于刀片服务器架构的HPP体系结构,拥有总共超过30000个CPU核心,总设计内存超过120TB。网络方面曙光5000A采用了双网模式,单向速率为20Gbps,MPI延迟为1.6us。

2)深腾7000

联想"深腾7000"是国内第一个实际性能突破每秒百万亿次的异构机群系统,成功实现了1240个2路薄节点和14个4路厚节点的协同计算,实际Linpack性能突破每秒106.5万亿次。它由计算刀片节点(1140台,双路Intel Xeon四核处理器,3GHz,32GB内存)、厚节点(38台,16路Intel Xeon四核处理器,2.93GHz,512GB内存,NUMA结构)、胖节点(2台SGI Altix4700,共384个1.67GHz Intel Itanium2双核处理器,共5TB内存)、可视化节点(12台,双路Intel Xeon四核处理器,3GHz,32GB内存,NV8800GTS显卡)等组成。

4.2 测试方法

本次测试选取了不同消息长度(512B,2kB,4kB,16kB,64kB,128kB,256kB,512kB)、不同进程数(64,128,256,512,1024)的不同算法(二维网格、三维网格、Bruck算法、原始算法、成对交换、递归倍增、环算法、简单算法),实验中曙光5000A和深腾7000每个节点都选取了8个进程运行测试程序。

为了尽可能地保证测试的准确性,算法的运行时间取50次的平均值。测试中采用MPI_Wtime()计时,一段计时方法如下:

```
t1=MPI_Wtime();
Call MPI_Alltoall function;
t2=MPI_Wtime()。
```

5 测试结果和分析

本小节从深腾7000上不同消息长度的算法性能分析、曙光5000A上不同消息长度、不同进程数的性能分析3个方面总结了Alltoall算法的测试结果。本节所有图的纵坐标都表示各Alltoall算法运行时间,单位为毫秒。

5.1 深腾7000上不同消息长度的算法性能分析

图1显示了深腾7000上64进程不同消息长度的各种算法的测试结果,可以看出递归倍增算法、间结算法(二维三维

网格算法)需要较多额外的消息传输时间,并且随着消息长度和进程数的增加传输耗时急剧增长。由式(3)—式(5)可知其原因是上述3种算法都需要较长的消息传输时间,并且随着进程数增多和消息长度变大(p 和 n 的值增大),消息传输时间增长剧烈。

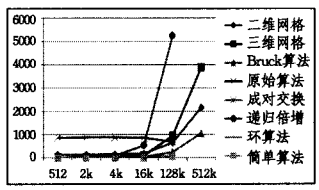

图1 深腾7000上64进程不同消息长度的测试结果

原始算法在深腾7000中的性能随着消息长度的增加变化不大并在消息长度为128k时用时减少。简单算法、成对交换以及环算法性能较好,尤其是在消息长度较大时。

从图2可以进一步看出进程数为64时,成对交换和环算法优于简单算法,这是因为尽管它们的理论估算时间一致,见式(1),但简单算法每个进程依次与其他进程通信会导致节点冲突[2]。在消息长度小于或者等于4k时环算法耗时一直较成对交换算法少,但消息变大后成对交换算法性能优于环算法。因此对于深腾7000而言,成对交换算法较环算法更适用长消息的集合通信,即成对交换算法消息长度扩展性较好。

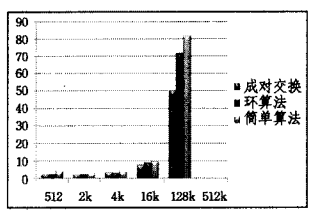

图2 深腾7000上成对交换、环算法、简单算法对比

5.2 曙光5000A上不同消息长度的算法性能分析

随着消息长度的变大,成对交换、环、原始算法表现较好,由于存在节点冲突,LAM/MPI的简单算法表现次之,消息长度扩展性较差的算法是二维网格和Bruck算法,二者都是适合于短消息的集合通信。Bruck算法和原始算法的表现同文献[1]中一致,原始算法更适合中等消息长度。上述结论可以很明显地从曙光5000A上的64进程(见图3)和128进程(见图4)的测试结果中看出。Bruck算法不适合中长消息的集合通信是因为它需要多次传输相同数据。另外,由于非常长的额外数据传输时间,三维网格和递归倍增算法在曙光5000A的用时较一般算法超长,这与在深腾7000上的性能表现一致。为了更明显地比较其他算法,后面的算法性能比较图将给出三维网格和递归倍增算法的测试结果。

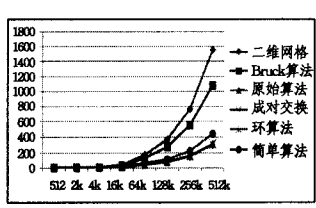

图3 曙光5000A上64进程不同消息长度测试结果

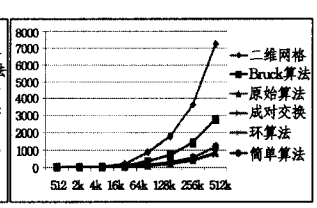

图4 曙光5000A上128进程不同消息长度测试结果

从原始算法、成对交换以及环算法的对比可以进一步看出消息长度较小时原始算法最优,随着消息长度增大,适合长消息的成对交换和环算法性能就会逐渐优于原始算法,如图5所示,消息长度大于等于4k时环算法用时比原始算法少,同时16k是成对交换算法和原始算法的拐点。从图中还可以看出曙光5000A上环算法性能一般优于成对交换算法。随着进程数增多,以上两个结论依然成立,不同的是拐点出现时消息长度更小,如图6所示,环算法的拐点的消息长度是2k而成对交换是4k。这进一步说明环算法的消息长度扩展性优于成对交换算法。

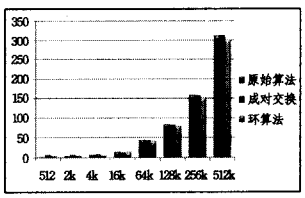

图5 曙光5000A上64进程原始算法、成对交换、环算法对比

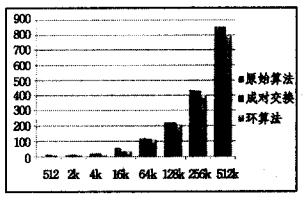

图6 曙光5000A上128进程原始算法、成对交换、环算法对比

5.3 曙光5000A上不同进程数的算法性能分析

上一小节分析了各种算法消息长度的扩展性,本节主要分析随着进程数的增加Alltoall算法的性能表现。消息长度较小时(见图7),进程扩展性较好的算法是Bruck和原始算法,同样适合短消息的二维网格算法因为需要较多的数据传输时间,所以其短消息的进程扩展性仍然欠佳;进程扩展性较差的是适合长消息的环算法和简单算法。

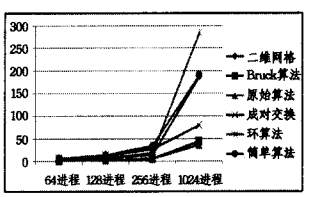

图7 曙光5000A上消息长度为512B不同进程数的测试结果

对于中等消息长度(见图8),适合短消息的二维网格算法在进程数较小时表现尚可,但随着进程数增多,额外消息传输时间增多,算法性能下降最快(图中未给出,理由同三维网格和递归倍增算法)。除二维网格算法外,中等消息进程扩展性较差的是原始算法、简单算法和Bruck算法,较好的是环算法和成对交换算法。值得注意的是,进程数较多时,中等消息的Bruck性能反而优于原始算法。

对于长消息(见图9),随着进程数的增加Bruck算法用时增加较快,进程扩展性较差,进程扩展性表现较好的是成对交换算法、环算法和原始算法,其中成对交换算法256进程的用时少于128进程的用时。

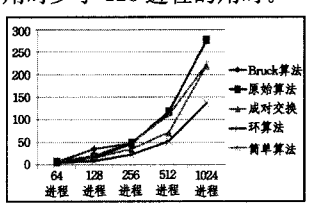

图8 曙光5000A上消息长度为4k时不同进程数的测试结果

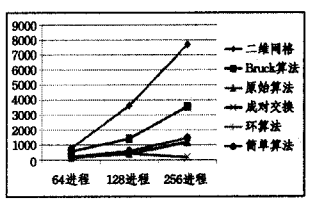

图9 曙光5000A上消息长度为256k不同进程数的测试结果

5.4 测试结果小结

由以上性能评测可以得出如下结论:1)递归倍增、二维三

(下转第207页)

据自动抽取混合模型[J]. 软件学报,2008,19(2):358-368

[36] Vassiliadis P, Bouzeghoub M, Quix C. Towards Quality-Oriented Data Warehouse Usage and Evolution[C]// Proc. of the 11th International Conference on Advanced Information Systems Engineering. Heidelberg: Springer-Verlag, 1999: 164-179

[37] Mihaila G A, Raschid L, Vidal M E. Using Quality of Data Metadata for Source Selection and Ranking[C]// Proc. of the 3rd Intl. Workshop on the Web and Databases. Heidelberg: Springer-Verlag, 2000: 93-98

[38] Mayer M, Karkaletsis V, Archer P, et al. Quality Labeling of Medical Web Content [J]. Health Informatics Journal, 2006, 12 (1): 81-87

[39] Vadrevu S, Nagarajan S, Gelgi F, et al. Automated metadata and instance extraction from news Web sites[C]// Proc. of the IEEE/WIC/ACM Intl. Conf. on Web Intelligence. Los Alamitos: IEEE Press, 2005: 38-41

[40] Yi J, Sundaresan N, Huang A W. Using Metadata to Enhance a Web Information Gathering System[C]// Proc. of the 3rd Intl. Workshop on the Web and Databases. Heidelberg: Springer-Verlag, 2000: 38-57

[41] Hess A, Kushmerick N. Learning to attach semantic metadata to Web services[C]// Proc. of the 2nd Semantic Web Conference. Heidelberg: Springer-Verlag, 2003: 258-273

[42] Brin S, Page L. The Anatomy of a Large - Scale Hypertextual Web Search Engine[J]. Computer Networks, 1998, 30(1-7): 107-117

[43] Kleinberg J M. Authoritative Sources in a Hyperlinked Environment [J]. Journal of the ACM, 1999, 46(5): 604-632

[44] Chakrabarti S, Dom B, Raghavan P, et al. Automatic resource compilation by analyzing hyperlink structure and associated text [J]. Computer Networks, 1998, 30(1-7): 65-74

[45] Adibi J, Chalupsky H, Grobelnik M, et al. KDD-2004 Workshop Report[J]. SIGKDD Explorations, 2004, 6(2): 136-139

[46] Borodin A, Roberts G O, Rosenthal J S, et al. Link Analysis Ranking: Algorithms, Theory, and Experiments [J]. ACM Transactions on Internet Technology, 2005, 5(1): 231-297

[47] Getoor L, Diehl C P. Link Mining: A Survey[J]. SIGKDD Explorations, 2005, 7(2): 3-12

(上接第 188 页)

维网格算法在消息长度较小进程数较少时表现尚可,但随着消息长度和进程数的增多用时增长剧烈;2)消息长度较小时,深腾 7000 上表现较好的是简单算法和 Bruck 算法,曙光 5000A 上表现较好的是原始算法和 Bruck 算法。值得注意的是成对交换和环算法在两个系统的短消息通信性能都不错;3)对于中长消息,深腾 7000 上表现较好的依次是成对交换、环算法和简单算法,随着消息长度进一步增加成对交换优于其他两种算法;曙光 5000A 上表现较好的依次是环算法、成对交换和原始算法,但是成对交换算法的进程扩展性优于环算法,同时中等数据长度的原始算法进程扩展性较差;4)简单算法在深腾 7000 上表现较好,但在曙光 5000A 上性能一般,但是较稳定。简单算法的用时一般情况下较环算法、成对交换算法、原始算法的多是因为存在节点冲突。

结束语 本文测试和分析了曙光 5000A 及深腾 7000 上主流 Alltoall 算法不同消息长度和不同进程数的性能。由递归倍增以及二维三维网格算法的测试结果可知,以较多的消息传递减少启动时间在曙光 5000A 以及深腾 7000 上并不能提高 Alltoll 的性能,为此集合通信操作应尽量避免额外的通信。消息长度较大时,消息的传输时间占主导地位,减少网络和节点等冲突是提高性能的关键,使两个系统的成对交换和环算法的性能都较好。

Alltoall 的优化是未来的主要工作,包括稀疏 Alltoall 和数据长度不等的 Alltoallv。调用非阻塞点对点操作的原始算法在测试中表现较好,因此非阻塞集合通信[10]也是重要的探索方向。

参 考 文 献

[1] Thakur R, Rabenseifner R, Gropp W. Optimization of Collective Communication Operations in MPICH[J]. International Journal of High Performance Computing Applications, 2005, 1(19): 49-66

[2] Faraj A, Yuan Xin. An Empirical Approach for Efficient All-to-All Personalized Communication on Ethernet Switched Clusters [Z]. ICPP, 2005: 321-328

[3] 陈靖,张云泉,张林波,等. 一种新的 MPI Allgather 算法及其在万亿次机群系统上的实现与性能分析[J]. 计算机学报,2006,29 (5):808-814

[4] MPICH-A portable implementation of MPI[OL]. http://www.mcs.anl.gov/mpi/mpich

[5] LAM/MPI Parallel Computing. http://www.lam-mpi.org/

[6] Kale L V, Kumar S, Vardarajan K. A framework for collective personalized communication[C]// Proceedings of the 17th International Parallel and Distributed Processing Symposium(IPDPS '03). 2003

[7] Bruck J, Ho C-T, Kipnis S, et al. Efficient algorithms for all-to-all communications in multiport messagepassing systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8 (11): 1143-1156

[8] Sun Jiachang. Multivariate Fourier Series over a Class of non Tensor-product Partition Domains[J]. Journal of Computational Mathematics, 2003, 12(1): 53-62

[9] 姚继锋,孙家昶. 平行十二面体区域上的快速离散傅立叶变换及其并行实现[J]. 数值计算与计算机应用,2004,25(4):303-314

[10] Hoefler T, Lumsdaine A, Rehm W. Implementation and performance analysis of non-blocking collective operations for MPI [C]// Proceedings of the 2007 ACM/IEEE conference on Supercomputing. Reno, Nevada, November 2007