ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2017/11(02)-0252-10
doi: 10.3778/j.issn.1673-9418.1512057

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

MPI集合通信性能可扩展性研究与分析*

罗红兵+,张晓霞
北京应用物理与计算数学研究所 高性能计算中心,北京 100094

Analysis of Scalability for MPI Collective Communication*

LUO Hongbing+, ZHANG Xiaoxia
High Performance Computing Center, Institute of Applied Physics and Computational Mathematics, Beijing 100094, China
+ Corresponding author: E-mail: hbluo@iapcm.ac.cn

LUO Hongbing, ZHANG Xiaoxia. Analysis of scalability for MPI collective communication. Journal of Frontiers of Computer Science and Technology, 2017, 11(2): 252-261.

Abstract: The performance of collective communications impacts the efficiency of large scale parallel numerical computing application. Since the theoretic analysis and evaluation on every type of collective communications is still insufficient for the programmer, the communication modules of many applications are designed and used unreasonably. Using Intel IMB benchmark, this paper analyzes and evaluates the typical MPI (message passing interface) collective communication on BXJ supercomputer (a parallel machine platform), and gives the theoretic results based on the current model and algorithm. The results show that the performance of different MPI collective communications is different. There is an obvious gap between the actual value of some collective communication and theoretic value. The results reflect that there are still many researches, such as accurate theoretic performance model for some collective communication, optimization of collective communication, and optimization of application communication module, etc.
Key words: collective communication; communication performance; scalability

摘 要: 集合通信性能是影响并行程序并行效率的重要因素之一,但对于大规模并行计算机上不同类别集合通信的评测和理论分析仍较为缺乏,许多应用程序的通信模块设计和使用不合理。基于某国产并行机平台,

* The National High Technology Research and Development Program of China under Grant No. 2014AA01A302 (国家高技术研究发展计划(863计划)).
Received 2015-12, Accepted 2016-02.
CNKI网络优先出版: 2016-02-03, http://www.cnki.net/kcms/detail/11.5602.TP.20160203.1126.010.html

利用IMB测试程序,对各典型MPI(message passing interface)集合通信性能进行了分析,并基于现有通信模型和算法进行理论拟合。结果显示:不同类别的MPI集合通信操作的性能差异很大,并且许多集合通信的性能在超大规模下与理论差距很大。一方面反映出现有理论和模型的不足;另一方面也体现出,无论是集合通信的优化,还是基于集合通信的特征进行应用程序的通信模块设计,仍然大有可为。

关键词:集合通信;通信性能;可扩展性

文献标志码:A **中图分类号:**TP311

1 引言

随着大规模并行计算机处理器核数的不断增加,如何开发与并行计算机处理器核数相匹配的并行程序,从而支撑大规模乃至超大规模数值模拟,成为一个备受瞩目的重要问题。实际上,在许多成百上千处理器的高性能计算机上,相当数量的工程应用并行软件,难以获得理想的绝对计算速度和并行效率,许多并行程序甚至难以获得正向的加速效果。其中,影响应用程序计算性能发挥的主要瓶颈之一是消息传递通信,许多应用程序消息传递通信模块不合理,导致出现全局通信频率过高,或导致通信堵塞。

要优化并行程序的集合通信性能,首先需要深刻理解集合通信开销,尤其是通信性能的可扩展性。集合通讯的性能建立在基本的通讯模型基础上,其中被广泛使用的通信性能模型是LogP模型[1],该模型是一个针对分布式存储的多处理器模型,处理器间采用点对点通信。LogGP(latency,overhead,gap,and processor)模型[2]在LogP模型的基础上增加了一个参数G,该参数可以描述在传递长消息时获得的带宽。考虑到集合通信中,随着消息块大小的不同,通讯延迟对集合通信性能的影响是不同的,对MPI(message passing interface)集合通信的实现有各种形式的优化[3-8],如最为经典的是结合二叉树模型实现bcast和reduce等操作。国内也有一些针对集合通信性能评估和优化方面的研究,包括针对Alltoall通信的测试和优化[9-12],对集合通信模型的评估[13]。

尽管对于集合通信开销的评估、优化和理论研究较多,但对于大规模并行计算机上较为全面的集合通信性能分析研究并不多见。不同类别的集合通信性能究竟呈现什么特征?在实际大规模并行机上集合通信的性能是否与相关理论吻合,是否具备了高可扩展性?集合通信理论建模与实测性能间的差距究竟有多大?这些都需要人们对MPI集合通信性能进行全面的定量分析,以期发现大规模并行计算机通信系统运行中的问题,并为大规模并行程序通讯模式的设计提供参考。

2 集合通讯实现算法分析

集合通讯的性能与参与的进程数、消息块的大小、系统通讯延迟和带宽相关,同时依赖于集合通讯的实现算法。以下分别对典型的集合通讯性能进行理论分析,其中p为进程数,k为消息的大小,α为通讯延迟,β为通信带宽的倒数,γ为执行Reduce操作时每字节的计算开销。

2.1 Bcast和Reduce实现算法

实现Bcast最著名的算法是最小跨距算法(minimum-spanning tree algorithm,MST),其基本思想是把参与通讯的节点等分为两个无交集的集合,首先从不包含root节点的子集中选取一个目的节点,执行root至该目的节点的一次消息发送;然后分别以root和该目的节点作为root节点,在这两个子集合中递归执行上面的步骤,直至所有节点都收到消息。显然,MST算法将Bcast和Reduce通信开销从p降到了lb(p)量级,对于大规模的集合通信,其意义是显而易见的。

在不考虑网络冲突的前提下,MST算法的开销如下:

$$T_{\mathrm{MSTBcast}}(p,n)=\lceil \mathrm{lb}(p)\rceil(\alpha+n\beta) \tag{1}$$

Reduce与Bcast的实现类似,采用MST算法实现时的开销如下:

$$T_{\mathrm{MSTReduce}}(p,n)=\lceil \mathrm{lb}(p)\rceil(\alpha+n\beta+n\gamma) \tag{2}$$

2.2 Scatter和Gather实现算法

Scatter和Gather的实现也可以采用MST算法,其中Gather是Scatter的逆过程,Scatter和Gather的开

销如下：

$$T_{\text{MSTScatter}}(p,n)=\sum_{k=1}^{\lceil \text{lb}(p)\rceil}(\alpha+2^{-k}n\beta)=\lceil \text{lb}(p)\rceil\alpha+\frac{p-1}{p}n\beta \quad (3)$$

$$T_{\text{MSTGather}}(p,n)=\sum_{k=1}^{\lceil \text{lb}(p)\rceil}(\alpha+2^{-k}n\beta)=\lceil \text{lb}(p)\rceil\alpha+\frac{p-1}{p}n\beta \quad (4)$$

2.3 Allgather实现算法

Allgather实现算法包括早期的环(ring)方法，其开销随进程数p的增加呈线性增长；Allgather实现算法优化的主要思路是减少通讯步数，将时间开销降低为与进程数p呈对数关系，以减少通信延迟的影响。主要的实现算法有递归加倍(recursive doubling)算法和Bruck算法，这些算法的开销如下：

$$T_{\text{ring}}=(p-1)\alpha+\frac{p-1}{p}n\beta \quad (5)$$

$$T_{\text{rec\_dbl}}=\text{lb}(p)\alpha+\frac{p-1}{p}n\beta \quad (6)$$

$$T_{\text{bruck}}=\lceil \text{lb}(p)\rceil\alpha+\frac{p-1}{p}n\beta \quad (7)$$

2.4 Allreduce实现算法

与Allgather的实现类似，Allreduce可以用递归加倍(recursive doubling)算法和rabenseifner算法实现，区别在于每个通讯步Allreduce还涉及本地的规约操作，相应的开销如下：

$$T_{\text{rec\_dbl}}=\text{lb}(p)\alpha+n\,\text{lb}(p)\beta+n\,\text{lb}(p)\gamma \quad (8)$$

$$T_{\text{rabenseifner}}=2\,\text{lb}(p)\alpha+2\frac{p-1}{p}n\beta+\frac{p-1}{p}n\gamma \quad (9)$$

2.5 Alltoall实现算法

对于短消息(≤256 B)，Bruck算法以额外的通讯来换取p的对数级的执行步骤，是当前较好的Alltoall实现算法；对于中等长度的消息(256 B至32 KB之间)，irecv-isend算法较好；对长消息，通常采用成对交换(pairwise-exchange)算法，其开销如下：

$$T_{\text{bruck}}=\lceil \text{lb}(p)\rceil\alpha+\left(\frac{n}{2}\text{lb}(p)+\frac{n}{p}\left(p-2^{\lfloor \text{lb}(p)\rfloor}\right)\right)\beta \quad (10)$$

$$T_{\text{long}}=(p-1)\alpha+n\beta \quad (11)$$

3 测试平台及分析方法

3.1 测试平台

测试平台选择某国产并行机(简称BXJ)，该系统的每个计算节点包含2颗英特尔微处理器，每颗微处理器包含6个计算核心；互联系统采用自主设计的高阶路由芯片NRC和高速网络接口芯片NIC，实现光电混合的二层胖树结构高阶路由网络互连，基本参数如表1。

Table 1 Basic parameters for communication system of BXJ parallel computer

表1 BXJ并行机通信系统基本参数

结构	类别	参数值
设备层	GLEX通讯点对点最小延迟/μs	1.58
	GLEX通讯单向最大带宽/(MB/s)	6 343
	GLEX通讯双向最大带宽/(MB/s)	9 710
软件层	MPI通讯延迟/μs	2.37
	MPI单向通讯带宽/(MB/s)	6 343
	MPI双向通讯带宽/(MB/s)	9 236

由于该测试平台处于生产性运行状态，测试时没有机会占用全系统，文中相关测试的最大并行规模为8 192个MPI进程。

3.2 测试和分析方法

对于MPI各种集合通信性能的理论值，依据上文的相应算法进行计算。通讯延迟α和通信带宽β依据系统基本通讯的实际测试值设为固定值，为简便起见，不考虑这些参数随通讯距离、消息块大小的变化；γ参数依据微处理器的主频、SIMD位数等估算，不考虑操作的访存开销。

对于MPI集合通信的性能测试，选用Intel IMB测试程序，评测的集合通讯操作主要包括Reduce、Gather、Alltoall、Bcast、Scatter、Allgather和Allreduce。测试结果取多次测试的均值。

4 基本通讯参数测试与数据分析

首先，利用IMB中PingPong程序和Sendrecv程序测试MPI通讯延迟和传输带宽情况。

表2结果显示：(1)BXJ的最小延迟2.72 μs与系统提供的参数2.37 μs基本相当，但PingPong的数据量超过一个数据包64 Byte(packet)时，延迟显著上升。显然在BXJ处于生产性运行时，测试程序明显与其他程序存在通讯通道的竞争。(2)单进程MPI通讯带宽随消息块尺寸的增加而不断增加，但单MPI进程很难达到最大通讯带宽，即便传输MB量级的消息。

Table 2 Results of PingPong

表2 PingPong测试结果

数据量/Byte	重复次数	延迟/μs	带宽/(MB/s)
0	1 000	2.71	0
16	1 000	2.73	5.58
128	1 000	7.93	15.39
1 024	1 000	8.60	113.53
4 096	1 000	9.56	408.81
8 192	1 000	10.95	713.37
16 384	1 000	14.16	1 103.77
32 768	1 000	20.46	1 527.63
65 536	640	19.63	3 183.32
262 144	160	47.38	5 276.02
1 048 576	40	188.56	5 303.29
4 194 304	10	801.46	4 990.92

表3和表4分别是每个计算节点执行8个进程和12个进程时，Sendrecv程序运行8 192个进程时的延迟和带宽情况。

Table 3 Results of Sendrecv (8 processes per node)

表3 Sendrecv测试结果(单节点8进程)

数据量/Byte	重复次数	最小延迟/μs	最大延迟/μs	平均延迟/μs	带宽/(MB/s)
0	1 000	3.06	5.83	3.25	0
128	1 000	11.86	12.81	12.11	19.06
4 096	1 000	17.96	24.24	20.43	322.26
8 192	1 000	23.37	32.42	25.29	481.99
16 384	1 000	37.56	54.87	40.72	569.50
32 768	1 000	70.60	99.62	76.67	627.40
65 536	640	139.65	190.39	149.32	656.54

Table 4 Results of Sendrecv (12 processes per node)

表4 Sendrecv测试结果(单节点12进程)

数据量/Byte	重复次数	最小延迟/μs	最大延迟/μs	平均延迟/μs	带宽/(MB/s)
0	1 000	3.12	3.59	3.19	0
128	1 000	14.72	20.17	15.24	12.11
4 096	1 000	24.08	35.65	25.99	219.13
8 192	1 000	32.07	45.96	34.16	339.96
16 384	1 000	53.09	102.83	56.90	303.90
32 768	1 000	101.92	143.50	109.37	435.53

表3和表4测试结果显示：(1)上述两种执行方式下Sendrecv的平均最小延迟基本都在3.20 μs左右，考虑到Sendrecv包含send和receive操作各一次，8 192个进程执行时的延迟基本与BXJ的通讯参数相符。(2)每个计算节点执行8个进程时，每个进程可以获得的带宽为656.54 MB/s，即单计算节点可获得的通讯带宽为8×656.54 MB/s=5 252.32 MB/s；每个计算节点执行12个进程时，每个进程可以获得的带宽为450.74 MB/s，即该节点可获得的通讯带宽为8×450.74 MB/s=5 408.88 MB/s。(3)以α-β模型估算，Sendrecv操作传输64 KB数据包的延迟在141.86 μs至280.52 μs之间(12进程/单节点)，分别针对send和receive完成可以重叠及完全没有重叠，与实测数据基本吻合；Sendrecv操作传输64 KB数据包的延迟在98.40 μs至193.60 μs之间(8进程/单节点)，与实测数据也基本吻合。

5 典型全局通信性能的测试与评估

5.1 Bcast性能测试与分析

依据当前Bcast实现算法，其开销与MPI进程数、数据量、MPI通讯延迟和带宽有关，其中与MPI进程数呈对数关系，与传输的数据量、MPI通讯延迟和通讯带宽则分别呈正比。图1、图2分别是16个MPI进程至8 192个MPI进程下，对128 B和16 KB的消息执行Bcast操作的通讯延迟情况，即固定消息块大小，测试随MPI进程数增长时的Bcast通讯延迟变化情况。其中实测值有两组，分别是每个计算节点启动8个MPI进程和12个MPI进程的结果。对于理论值，基于MST算法进行了估算，其中的系统通讯最小延迟α取2.37 μs，β根据PingPong时的通信带宽3 183 MB/s换算。

图1的测试数据显示：(1)对于128 B消息，Bcast通讯延迟的实测值与MST算法的理论值吻合得较好，无论是每个计算节点启动8个MPI进程，还是启动12个MPI进程，Bcast通讯延迟的实测值与MST算法的理论值相差不大。(2)由于计算节点启动12个MPI进程时，通讯带宽竞争比启动8个MPI进程时更大，相应地Bcast通讯延迟就略大一些。(3)由于实际系统不是独占的系统，实际运行时的通讯延迟和带宽有一定的波动，实测值也存在一定的波动。

图2显示：(1)对于16 KB的消息，Bcast通讯延迟

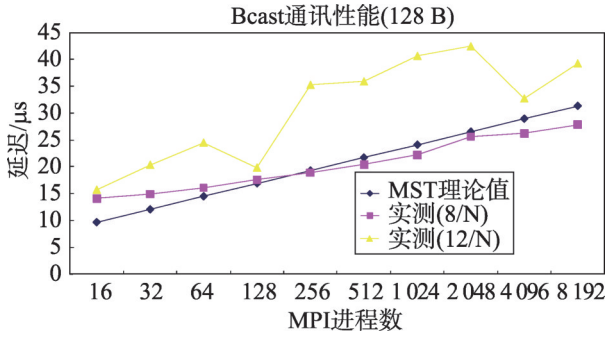


Fig.1 Latency of Bcast for different MPI processes (128 B message)

图1 Bcast开销与MPI进程数的关系(128 B消息块)

的实测值的变化趋势在MPI进程数不大于4 096时与理论值的趋势基本一致,实测值与MST算法的理论值相差较大,这主要是源自该通讯规模下理论值的通讯带宽与MPI实际通讯带宽偏差较大。(2)当MPI进程数达到8 192时,Bcast通讯延迟发生了突变,达到了4 096进程时的2倍以上,原因很难解释。

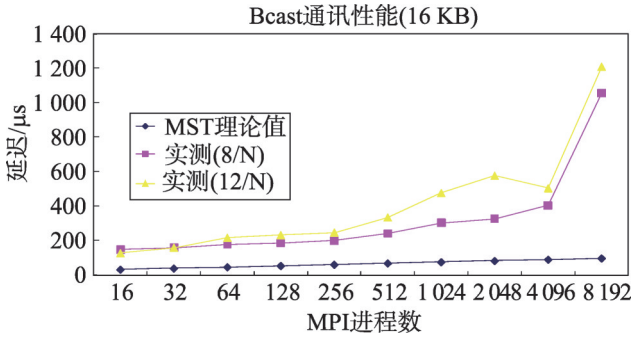


Fig.2 Latency of Bcast for different MPI processes (16 KB message)

图2 Bcast开销与MPI进程数的关系(16 KB消息块)

5.2 Reduce性能测试与分析

Reduce开销除了与参与的MPI进程数、传输的数据量、MPI通讯延迟和通讯带宽有关,还与Reduce运算的性能有关。由于运算的性能远高于通讯性能,起主导性的仍是通讯延迟和通讯带宽。图3、图4分别是16个MPI进程至8 192个MPI进程下,对128 B和16 KB的消息执行Reduce操作的通讯延迟情况。Reduce理论值同样基于MST算法估算,延迟α和β取值同Bcast。对于γ,以主频2.2 GHz计算,并假定每个时钟周期可以执行256位计算。

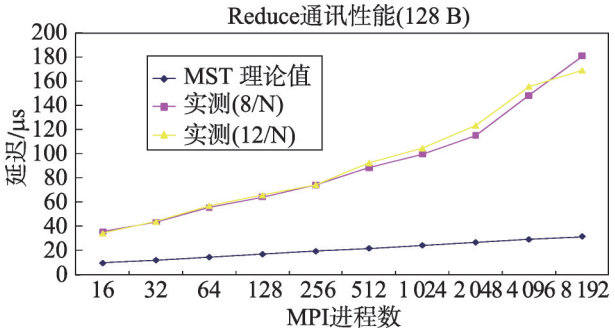


Fig.3 Latency of Reduce for different MPI processes (128 B message)

图3 Reduce开销与MPI进程数的关系(128 B消息块)

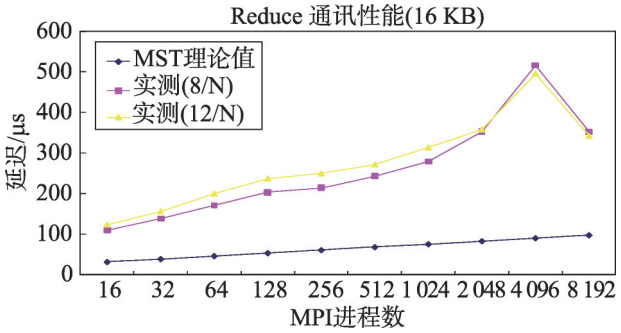


Fig.4 Latency of Reduce for different MPI processes (16 KB message)

图4 Reduce开销与MPI进程数的关系(16 KB消息块)

图3、图4的测试数据显示:(1)Reduce的实测性能与理论性能有较大差距,其直接原因是实际操作所能获得的系统通讯带宽低于理论估计。(2)计算节点启动8个MPI进程和12个MPI进程时,Reduce的延迟差别不大,表明制约Reduce操作性能的瓶颈主要不是节点内进程对通讯带宽的竞争。(3)Reduce操作在4 096进程时发生了性能抖动,反映出通讯系统性能波动对Reduce性能的影响。

5.3 Gather性能

Gather实测值和理论值见图5、图6。实测值分别是16个MPI进程至8 192个MPI进程下,对128 B和18 KB的消息执行Gather操作的通讯延迟情况,包含每个计算节点启动8个MPI进程和12个MPI进程的结果。理论值基于2.2节式(4)计算,系统通讯最小延迟α和β取值同Bcast。

图5、图6的测试数据显示:(1)Gather通讯延迟

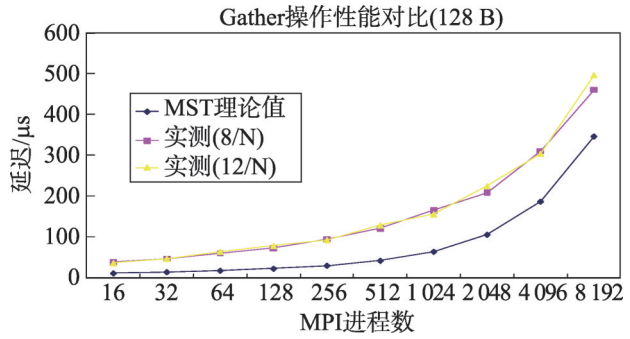


Fig.5 Latency of Gather for different MPI processes (128 B message)

图5 Gathert开销与MPI进程数的关系(128 B消息块)

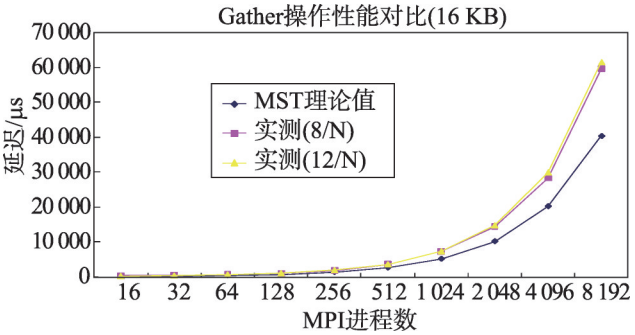


Fig.6 Latency of Gather for different MPI processes (16 KB message)

图6 Gathert开销与MPI进程数的关系(16 KB消息块)

的实测值与MST算法的理论值吻合得较好，无论是计算节点启动8个MPI进程，还是启动12个MPI进程，Gather延迟的实测值与理论值相差不大。(2)总体上看，计算节点启动的进程数与Gather通信延迟的关系不大，计算节点启动12个MPI进程时的Gather通讯延迟略大一些。

5.4 Scatter性能

图7、图8分别是16个MPI进程至8 192个MPI进程下，对128 B和16 KB的消息执行Scatter操作的通讯延迟情况。其中实测值有两组，分别是每个计算节点启动8个MPI进程和12个MPI进程的结果。Scatter操作的理论值基于2.2节的式(3)估算，α和β取值同Bcast。

图7、图8的测试数据显示：(1)当消息块大小为128 B时，Scatter通讯延迟的实测值与MST算法的理论值吻合得较好，无论计算节点启动8个MPI进程，还是启动12个MPI进程，Scatter通讯延迟的实测值与理论值相差不大。(2)对于16 KB的消息，Scatter通讯延迟的实测值在进程数小于等于512时与理论值基本一致，当MPI进程数超过1 024之后，则明显大于理论值。(3)计算节点启动的MPI进程数对Scatter操作的延迟情况影响不大，计算节点启动12个MPI进程时，通讯延迟略大一些。

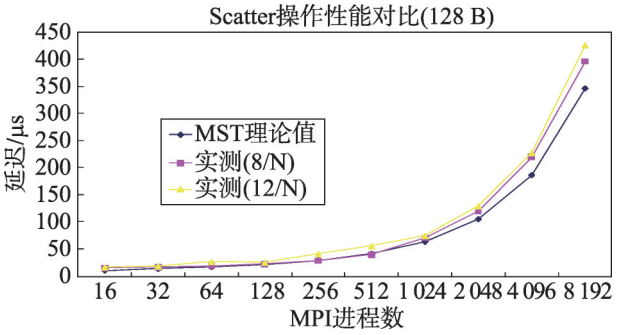


Fig.7 Latency of Scatter for different MPI processes (128 B message)

图7 Scatter开销与MPI进程数的关系(128 B消息块)

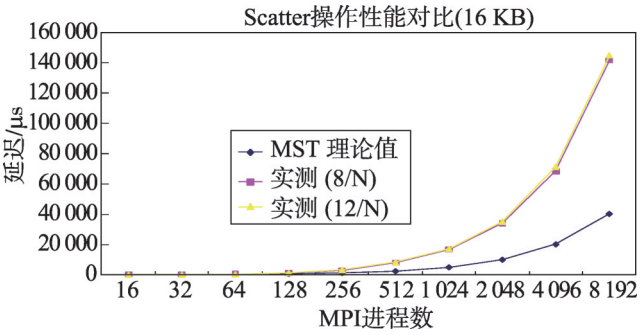


Fig.8 Latency of Scatter for different MPI processes (16 KB message)

图8 Scatter开销与MPI进程数的关系(16 KB消息块)

5.5 Allgather性能

Allgather与Gather类似，区别是所有进程同时收集数据，也相当于任一进程先调一次Gather，然后再对收集的数据进行一次Bcast。对Allgather操作的性能进行理论估算时，采用2.3节的式(5)和式(7)(本文仅考虑进程数为2的幂的情况)。对于α和β，考虑到Allgather操作过程中，各进程同时在做大量的发送和接收操作，对通讯带宽的竞争明显，理论分析时的传输带宽以Sendrecv时实测值为基准，12进程/单

节点时传输带宽为450 MB/s,8进程/单节点时传输带宽为569 MB/s。图9~图12是测试结果。

图9~图12的结果显示:(1)消息块为16 KB时,Allgather性能变化趋势与ring算法和bruck算法的理论值基本一致,但随着进程数的增加,Allgather的实测值与理论值的差距逐步增大。(2)消息块为128 B时,Allgather的实测值在进程数为2 048以内时与理论值很接近,大于bruck算法的理论值,小于ring算法的理论值,反映出bruck算法对于短消息的优势;但是,当进程数为4 096和8 192时,Allgather的实测值急剧增加,原因很难准确估计。考虑到每个计算节点启动8个MPI进程或12个MPI进程都出现这一情况,可以将这一情况视作必然出现或很容易出现。

5.6 Allreduce性能

Allreduce与Reduce类似,区别是所有进程同时做规约,也相当于任一进程先调一次Reduce,然后再对收集的数据进行一次Bcast。对Allreduce操作的性能进行理论估算时,采用2.4节的式(8)和式(9)。对于α和β,取值与Allgather理论性能估算类似一致。对于γ,与Reduce操作的理论估算一样,以主频2.2 GHz计算,并假定每个时钟周期可以执行256位计算。图13~图16是128 B和16 KB的消息块Allreduce操作的通讯延迟随MPI进程数增长的变化情况,及与理论值的对比。

图13~图16的结果显示:(1)消息块为16 KB时,无论是每个计算节点启动8个MPI进程,还是每个计算节点启动12个MPI进程,Allreduce性能变化趋势与理论值基本一致,但当进程数为4 096或8 192时,Allreduce实测值有时会急剧增加,与理论值的差距增大。(2)消息块为128 B时,Allreduce的实测值与理

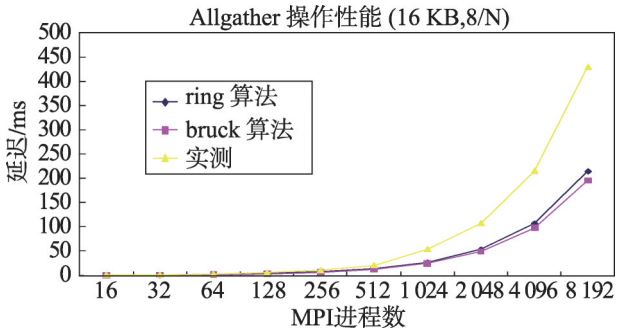


Fig.9 Latency of Allgather for different MPI processes (16 KB message, 8 processes per node)

图9 Allgather开销与MPI进程数的关系(16 KB消息块,单节点运行8个MPI进程)

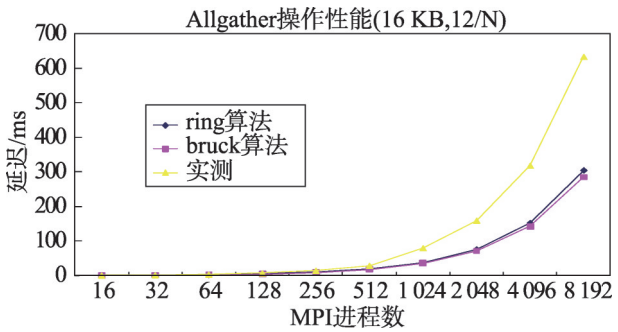


Fig.10 Latency of Allgather for different MPI processes (16 KB message, 12 processes per node)

图10 Allgather开销与MPI进程数的关系(16 KB消息块,单节点运行12个MPI进程)

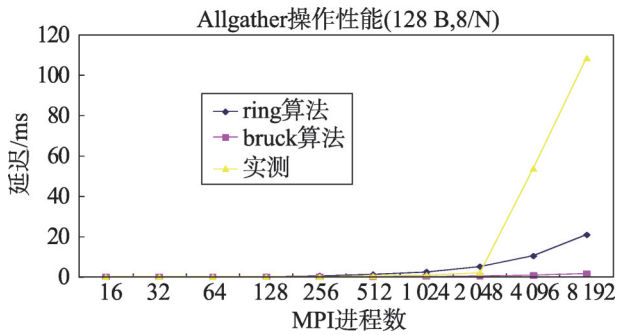


Fig.11 Latency of Allgather for different MPI processes (128 B message, 8 processes per node)

图11 Allgather开销与MPI进程数的关系(128 B消息块,单节点运行8个MPI进程)

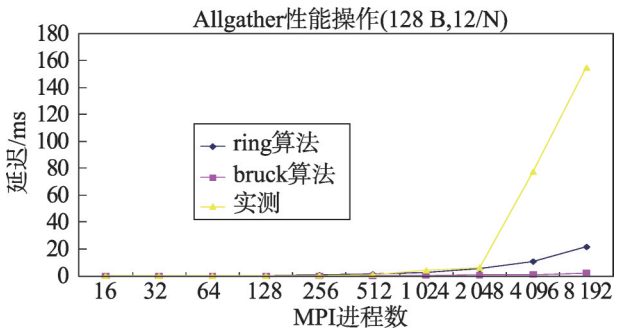


Fig.12 Latency of Allgather for different MPI processes (128 B message, 12 processes per node)

图12 Allgathert开销与MPI进程数的关系(128 B消息块,单节点运行12个MPI进程)

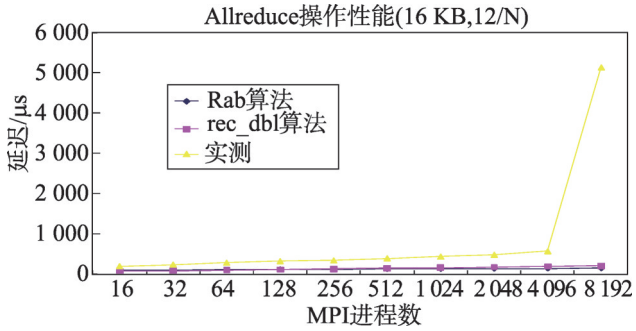


Fig.13 Latency of Allreduce for different MPI processes (16 KB message, 12 processes per node)

图13 Allreduce开销与MPI进程数的关系（16 KB消息块，单节点运行12个MPI进程）

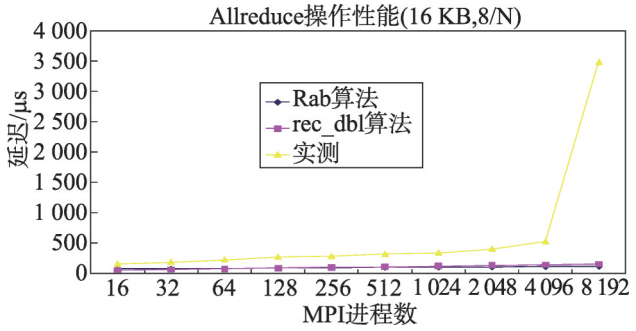


Fig.14 Latency of Allreduce for different MPI processes (16 KB message, 8 processes per node)

图14 Allreduce开销与MPI进程数的关系（16 KB消息块，单节点运行8个MPI进程）

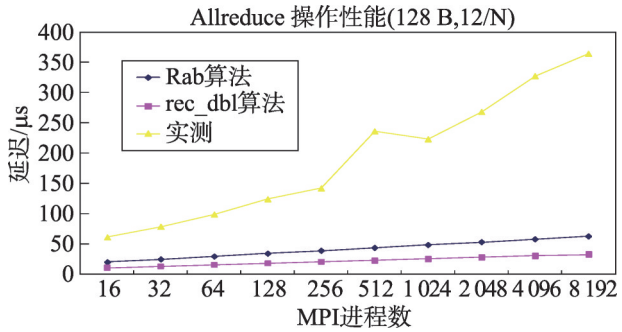


Fig.15 Latency of Allreduce for different MPI processes (128 B message, 12 processes per node)

图15 Allreduce开销与MPI进程数的关系（128 B消息块，单节点运行12个MPI进程）

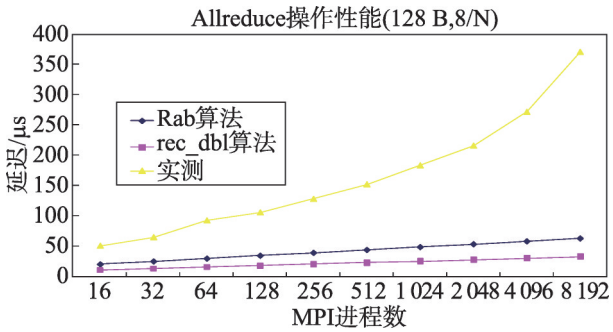


Fig.16 Latency of Allreduce for different MPI processes (128 B message, 8 processes per node)

图16 Allreduce开销与MPI进程数的关系（128 B消息块，单节点运行8个MPI进程）

论值很接近，尽管实测值随进程数的增加，与理论值的差距略微增加。

5.7 Alltoall性能

Alltoall需要所有的进程参与，且每个进程发送和接收消息块尺寸乘以进程数的消息量。对Alltoall操作的性能进行理论估算时，采用2.5节的式(10)和式(11)，传输带宽以Sendrecv时实测值为基准，12进程/单节点时传输带宽为450 MB/s，8进程/单节点时传输带宽为569 MB/s。图17~图20分别是128 B和16 KB的消息块Alltoall操作的通讯延迟随MPI进程数增长的变化情况，及与理论值的对比。

图17~图20的结果显示：(1)对于16 KB的消息块，Alltoall性能在MPI进程数小于2 048时，变化趋势与理论值基本一致，但当进程数为4 096或8 192时，Alltoall实测值有时会急剧增加，尤其是8 192个MPI进程。(2)消息块为128 B时，Alltoall的实测值与理论值很接近，在每个计算节点启动12个MPI进程的测试中，测试值在long算法和bruck算法的理论值之间，表明系统对于该规模的消息块实际采用的肯定不是long算法，而是bruck算法。(3)Alltoall实际性能的波动很大，明显比其他类别集合通讯的性能波动大。

5.8 典型全局通信性能对比分析

为便于更好地理解Reduce、Gather、Alltoall、Bcast、Scatter、Allgather和Allreduce等全局通信的性能，以下固定全局通信过程中的消息长度，对比并评估上述全局通信的延迟情况。表5是消息长度为16 KB时使用8 192个MPI进程执行上述全局通讯操作在BXJ上的执行延迟情况。表5中数据显示：(1)不同

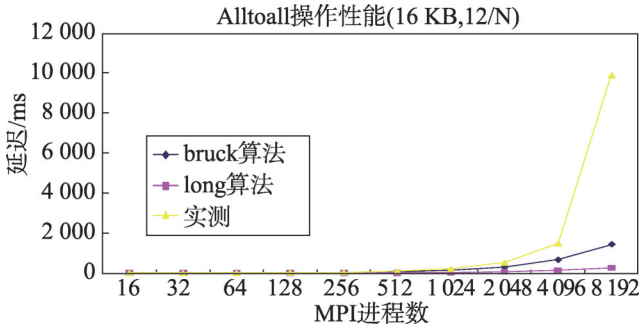


Fig.17 Latency of Alltoall for different MPI processes (16 KB message, 12 processes per node)

图17 Alltoall开销与MPI进程数的关系（16 KB消息块，单节点运行12个MPI进程）

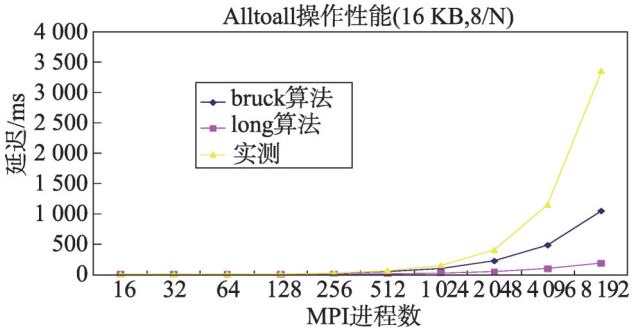


Fig.18 Latency of Alltoall for different MPI processes (16 KB message, 8 processes per node)

图18 Alltoall开销与MPI进程数的关系（16 KB消息块，单节点运行8个MPI进程）

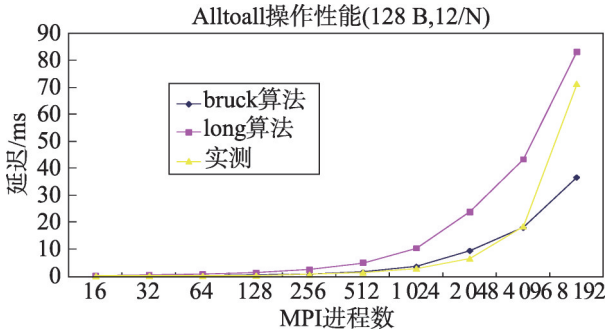


Fig.19 Latency of Alltoall for different MPI processes (128 B message, 12 processes per node)

图19 Alltoall开销与MPI进程数的关系（128 B消息块，单节点运行12个MPI进程）

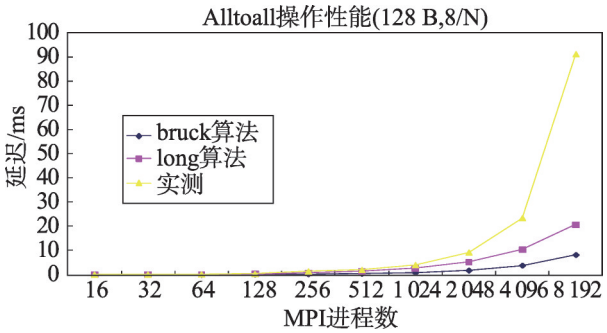


Fig.20 Latency of Alltoall for different MPI processes (128 B message, 8 processes per node)

图20 Alltoall开销与MPI进程数的关系（128 B消息块，单节点运行8个MPI进程）

Table 5 Latency of collective communications

表5 典型全局通讯操作延迟情况

序号	类别	延迟/μs	
		12进程/节点	8进程/节点
1	Reduce	342	352
2	Bcast	1 208	1 056
3	Gather	61 466	59 600
4	Scatter	144 919	141 772
5	Allgather	632 637	429 510
6	Allreduce	5 126	3 487
7	Alltoall	9 886 531	3 383 299

类别的MPI全局通讯操作具有不同的延迟情况，从数百微秒到数百毫秒，直至数秒；(2)在Reduce、Gather、Bcast、Scatter这些单边集合通讯中，Reduce操作延迟最小；(3)对于Alltoall、Allgather和Allreduce操作，Allreduce延迟最小。

6 小结

以上测试和分析，反映出大规模并行计算机上集合通信性能的如下特点：(1)不同类别的MPI集合通信操作具有差别极大的延迟情况，尤其在大规模情况下，许多全局通信操作的延迟往往以超线性的速度增加。(2)消息块的大小对于MPI集合通信的性能影响很大，即便是性能较好的Bcast操作的延迟，在大消息块的情况下也会呈超线性增长。

除了以上在并行程序通信模式设计时需要予以重视的集合通信性能特征之外，还有以下问题：(1)现有的模型对Reduce等单边集合通信操作的理论建模基本准确，但大规模情况下仍需进一步完善。(2)对Alltoall等多对多集合通信性能的理论建模是值得深入研究的问题，尤其在超大规模情况下，理论值与实际值差距很大，且缺乏理论模型方面的解释。

References:

[1] Culler D E, Karp R M, Patterson D, et al. LogP: a practical model of parallel computation[J]. Communications of the ACM, 1996, 39(11): 78-85.

[2] Alexanddrov A, Ionescu M F, Schauser K E, et al. LogGP: incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation [C]//Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures, Santa Barbara, USA, Jun 24-26, 1995. New York: ACM, 1995: 95-105.

[3] Chan E, Heimlich M, Purkayastha A, et al. Collective communication: theory, practice, and experience[J]. Concurrency and Computation: Practice & Experience, 2007, 19(13): 1749-1783.

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

[5] Kielmann T, Hofman R F H, Bal H E, et al. MAGPIE: MPI's collective communication operations for clustered wide area systems[C]//Proceedings of the 7th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Atlanta, USA, May 4-6, 1999. New York: ACM, 1999: 131-140.

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

[7] Chan E W, Heimlich M F, Purkayastha A, et al. On optimizing collective communication[C]//Proceedings of the 2004 IEEE International Conference on Cluster Computing, San Diego, USA, Sep 20-23, 2004. Washington: IEEE Computer Society, 2004: 145-155.

[8] Kumar R, Mamidala A R, Panda D K. Scaling alltoall collective on multicore systems[C]//Proceedings the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, USA, Apr 14-18, 2008. Piscataway, USA: IEEE, 2008: 1-8.

[9] Rao Li, Zhang Yunquan, Li Yuchneg. Performance test and analysis of alltoall collective communication on domestic hundred trillion cluster system[J]. Computer Science, 2010, 37(8): 186-188.

[10] Li Qiang, Sun Ninghui, Huo Zhigang, et al. Optimizing MPI alltoall communications in multicore cluster[J]. Journal of Computer Research and Development, 2013, 50(8): 1744-1754.

[11] Zhang Panyong, Meng Dan, Huo Zhigang. Research of collectives optimization on modern multicore clusters[J]. Chinese Journal of Computers, 2010, 33(2): 317-324.

[12] Chen Jing, Zhang Yunquan, Zhang Linbo, et al. A new MPI allgather algorithm and implement[J]. Chinese Journal of Computers, 2006, 29(5): 808-814.

[13] Liu Yang, Cao Jianwen, Li Yucheng. Testing and analyzing of collective communication models[J]. Computer Engineering and Applications, 2006, 42(6): 30-33.

附中文参考文献:

[9] 饶立, 张云泉, 李玉成. 国产百万亿次机群系统Alltoall性能测试与分析[J]. 计算机科学, 2010, 37(8): 186-188.

[10] 李强, 孙凝晖, 霍志刚, 等. MPI Alltoall通信在多核机群中的优化[J]. 计算机研究与发展, 2013, 50(8): 1744-1754.

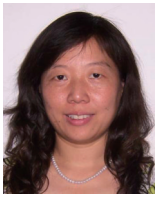
[11] 张攀勇, 孟丹, 霍志刚. 多核环境下高效集合通信关键技术研究[J]. 计算机学报, 2010, 33(2): 317-324.

[12] 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优化[J]. 计算机学报, 2006, 29(5): 808-814.

[13] 刘洋, 曹建文, 李玉成. 聚合通信模型的测试与分析[J]. 计算机工程与应用, 2006, 42(6): 30-33.

LUO Hongbing was born in 1968. He received the M.S. degree in computer software from Huazhong University of Science and Technology in 1992. Now he is a professor at Institute of Applied Physics and Computational Mathematics. His research interests include parallel computing and performance optimization, etc.

罗红兵(1968—),男,江西永新人,1992年于华中科技大学获得硕士学位,现为北京应用物理与计算数学研究所研究员,主要研究领域为高性能计算,性能优化等。发表学术论文20余篇,承担国家863计划等项目。



ZHANG Xiaoxia was born in 1973. She received the M.S. degree in computer system from National University of Defense Technology in 1998. Now she is a senior engineer at Institute of Applied Physics and Computational Mathematics. Her research interests include parallel computing and performance evaluation, etc.

张晓霞(1973—),女,河南焦作人,1998年于国防科学技术大学获得硕士学位,现为北京应用物理与计算数学研究所高级工程师,主要研究领域为并行计算,性能评测等。