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

陈 靖[1),2)] 张云泉[2),3)] 张林波[4),5)] 袁 伟[2),3)]

[1)](中国科学技术大学计算机科学与技术系 合肥 230026)
[2)](中国科学院软件研究所并行计算实验室 北京 100080)
[3)](中国科学院计算机科学国家重点实验室 北京 100080)
[4)](中国科学院数学与系统科学研究院 北京 100080)
[5)](中国科学院科学与工程计算国家重点实验室 北京 100080)

摘 要 给出一个新的 MPI Allgather 算法——邻居交换算法(neighbor exchange).提出的平均逻辑通信距离的概念和计算公式,可以有效地衡量通信的局部性.通过分析,发现在 4 种 MPI Allgather 算法中,邻居交换和环算法均具有最优的通信局部性.在万亿次机群深腾 6800 和曙光 4000A 上对 4 个 MPI Allgather 算法进行的性能测试和分析结果表明,邻居交换算法的长消息通信性能最优,中长消息通信性能不稳定,短消息通信性能次于递归倍增和 Bruck 算法.

关键词 MPI Allgather 算法;集合通信;性能评测;机群
中图法分类号 TP301

Implementation and Performance Analysis of a New MPI Allgather Algorithm on Terascale Linux Clusters

CHEN Jing[1),2)] ZHANG Yun-Quan[2),3)] ZHANG Lin-Bo[4),5)] YUAN Wei[2),3)]

[1)](*Department of Computer Science, University of Science and Technology of China, Hefei* 230026)
[2)](*Laboratory of Parallel Computing, Institute of Software, Chinese Academy of Sciences, Beijing* 100080)
[3)](*State Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing* 100080)
[4)](*Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing* 100080)
[5)](*State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences, Beijing* 100080)

Abstract Message Passing Interface (MPI) is one of the most important parallel programming environment. The MPI library provides point-to-point and collective communication functions, among which MPI Allgather is one of the most frequently used functions. Three kinds of algorithm are implemented for MPI Allgather in the latest versions of MPICH, i.e., the ring, the recursive doubling and the Bruck algorithms. In order to minimize the TCP traffic and congestion over Fast Ethernet, the authors propose a new MPI Allgather algorithm, namely the neighbor exchange. In the neighbor exchange algorithm, a property of pair-wise communication is incorporated and a process always exchanges data with its logical neighbor processes. A new concept, the Average Logical Communication Distance (ALCD), is proposed to measure the algorithmic communication locality. Analysis on the ALCD for the four algorithms reveals that the neighbor

收稿日期:2005-12-13;**修改稿收到日期**:2006-01-20.**本课题得到国家自然基金重点项目**(60533020)、**国家自然科学基金**(60303020)、**国家"九七三"重点基础研究发展规划项目基金**(G1999032805,2005CB321702)、**国家"八六三"高技术研究发展计划项目"高性能计算机及其核心软件"重大专项课题"高性能计算机性能测试技术及方法研究"**(2004AA104020)、**中国科学院软件研究所培育项目基金**(CXK25628)**和北京邮电大学网络与交换技术国家重点实验室开放课题**(2005-05)**资助.陈 靖,女,**1981 **年生,硕士研究生,主要研究方向为并行程序设计和性能评价.**E-mail: jingchen@mail.rdcps.ac.cn.**张云泉,男,**1973 **年生,博士,副研究员,硕士生导师,主要研究领域为大型并行数值软件、并行程序设计和性能评价、并行计算模型和非数值并行计算.张林波,男,**1962 **年生,博士,研究员,研究领域为计算数学、并行计算.袁 伟,男,**1979 **年生,硕士,工程师,主要研究方向为并行数据挖掘、并行程序设计和性能评价.

exchange and the ring algorithms have the best communication locality property among the four MPI Allgather algorithms. Numerical experiments on terascale Linux clusters DeepComp 6800 and DAWNING 4000A show that the neighbor exchange algorithm performs the best for long messages but is suboptimal for short and medium sized ones. For medium-size messages, the ring algorithm performs the best and for short messages, the recursive doubling algorithm performs the best.

Keywords　MPI Allgather algorithm; collective communication; performance evaluation; cluster

1　引　言

消息传递是目前并行计算机上广泛使用的一种程序设计模式，MPI(Message Passing Interface)则是其中最受欢迎的设计平台. 消息传递作为并行计算的基础，其通信性能对于并行应用程序性能有着重要的影响. MPI 库包括点到点通信和集合通信等消息传递函数. 集合通信(collective communication)又叫聚合通信，是指多个进程(通常大于 2)之间的通信. 如今，随着 Linux 机群越来越流行，并行机规模越来越大，已经可以容纳成千上万个处理器，相应地，并行应用程序规模也越来越大，所能使用的处理器越来越多，集合通信往往成为性能瓶颈. 当参与集合通信的进程非常多时，进程间通信量很大，并需要互相协调以完成语义，相应涉及到的处理器间的通信量也就非常大，关系错综复杂，大大影响了通信效率，从而影响应用程序的性能. 因此，改善和优化集合通信性能的意义重大，也成为非常活跃的一个研究课题.

MPI Allgather 是 MPI 库中使用频率最高的集合通信函数之一，最新的 MPICH 版本使用了三个算法实现 MPI Allgather：环、递归倍增和 Bruck 算法. 针对以太网上 TCP/IP 通信的特性，本文提出了一个新的 MPI Allgather 算法——邻居交换算法，并提出平均逻辑通信距离的概念和计算公式，以便有效地衡量通信局部性. 我们在万亿次大型机群深腾 6800 和曙光 4000A 上将邻居交换算法与环、递归倍增和 Bruck 算法进行了性能测试和分析，本文给出分析测试的结果.

本文第 2 节介绍相关算法；第 3 节详细描述邻居交换算法；第 4 节提出平均逻辑通信距离的概念，并对相关算法进行通信局部性的分析；第 5 节介绍测试环境、测试方法并分析实验结果；第 6 节是结论和未来工作.

2　相关算法

在 MPI 集合通信中，所有参与通信的进程构成一个通信域(communicator)，在有 p 个进程的通信域中，进程被依次编号为 $0,1,\cdots,p-1$. MPI Allgather 函数的功能如下：通信域内每个进程都有一份数据需要散播给其它进程，这些数据块大小相等，并称来自进程 i 的数据块为数据块 i. 经过一系列的通信步骤，最后每个参与通信的进程都收集到所有进程的数据(包括自己的数据)，且这些数据按序号在接收缓冲区中排列好.

关于 MPI Allgather 算法方面的研究很多：Benson 等在 Linux 机群上测试了 Allgather 算法的性能，并基于分发式障碍(barrier)算法提出了一个分发式 Allgather 算法[1,2]；Bruck 等提出了一个针对短消息通信的有效算法[3]；Chan 等对已有的集合算法做了一系列的测试，指出对不同长度的消息应使用不同的算法[4]；Thakur 等致力于优化集合通信函数的性能[5]. 本节介绍实现 MPI Allgather 的环、递归倍增和 Bruck 算法. 后面的讨论假设参与通信的进程数为 p.

2.1　环(Ring)算法

文献[5]给出了环算法的具体描述：所有参与通信的进程按序号形成一个环，消息在环上传递. 第 1 步，进程 i 将自己的数据块发送给进程 $((i+1)\%p)$ 并从进程 $((i-1)\%p)$ 接收数据. 从第 2 步开始，进程 i 将前一步从进程 $((i-1)\%p)$ 接收到的数据转发给进程 $((i+1)\%p)$. 该算法总共需要 $(p-1)$ 步数据传输.

2.2　递归倍增(Recursive Doubling)算法

文献[5]提到，对于递归倍增算法，当参与进程数为 2 的指数次幂时，在第 i 步 $(0<i\leq\log p)$，编号距离为 2^{i-1} 的两个进程交换数据，算法共需要 $\log p$ 步数据传输. 对于进程数为非 2 的指数次幂的情况，在各步骤中总有一些进程无法参与数据交换，须采取额外的传输步骤确保所有进程正确地接收到数据，在这种情况下，总传输步骤上界为 $2\lfloor\log p\rfloor$.

2.3　Bruck 算法

Bruck 算法[3]的提出是基于分发式障碍算法[2]的. Bruck 算法中，第 $k(0\leq k<\lfloor\log p\rfloor)$ 步，进程 i 发送 2^k 个数据块到进程 $((i-2^k)\%p)$ 并从进程

$((i+2^k)\%p)$接收数据. 如果 p 不为 2 的指数次幂，则增加一步(也就是第$\lfloor \log p \rfloor$步，为简化描述，令$s=\lfloor \log p \rfloor$)，进程 i 发送$(p-2^s)$块数据到进程$((i-2^s)\%p)$并从进程$((i+2^s)\%p)$中接收$(p-2^s)$个数据块，该算法共需要$\lceil \log p \rceil$步数据传输.

3 邻居交换(Neighbor Exchange)算法

文献[1]指出，对于 TCP/IP 协议来说，一个算法如果具有结点两两通信的性质，比如递归倍增算法，就能最小化 TCP/IP 通信量. 基于此观点，本文提出一个新的 MPI Allgather 算法——邻居交换算法. 邻居交换算法中，相邻的进程两两交换消息，即进程 A 发送数据给进程 B 同时又接收 B 发送来的数据，而不是进程 A 发送数据给进程 B 然后接收进程 C 发送来的数据.

图 1 演示了进程数为偶数时邻居交换算法的工作过程. 把进程$((i+1)\%p)$称为进程 i 的右邻居，进程$((i-1)\%p)$称为进程 i 的左邻居. 发送消息的第 1 步，进程号为偶数的进程与其右邻居交换数据，否则与左邻居交换数据. 第 2 步，进程号为偶数的进程与左邻居通信，交换它现有的两块数据(包括它本身的数据以及在第 1 步中接收到的数据). 从第 3 步开始，进程依次与左右邻居通信，交换它们在前一步中接收到的两个数据块. 一般的，在第 i 步通信完成时，每个进程都接收到$(2i-1)$块数据. 为完成功能，每个进程必须从邻居处接收$(p-1)$个数据块，因此，该算法总共需要$\frac{p}{2}$步数据传输.

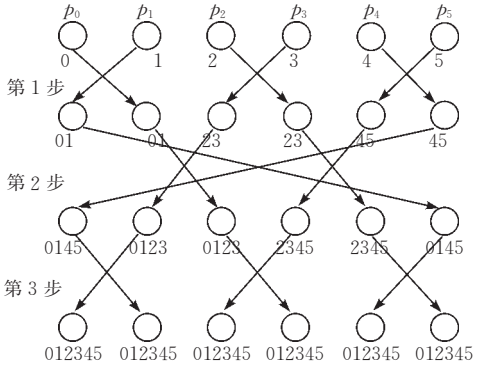

图 1 进程数为偶数的邻居交换算法

对于进程数为奇数的情况，在通信第 1 步，进程$(p-1)$将数据块传递给进程$(p-2)$，此后进程$(p-1)$不再参与消息通信，其余的$(p-1)$个进程按照上述提到的偶数个进程通信的方式交换数据，唯一不同的是需要在接收第$(p-2)$个数据块时，同时接收第$(p-1)$个数据块. 通信完成后，进程$(p-2)$将所有数据块传递给进程$(p-1)$. 这种情况共需要$\frac{p-1}{2}+2=\frac{p+3}{2}$步数据传输.

图 2 演示了进程数为奇数时邻居交换算法的工作过程.

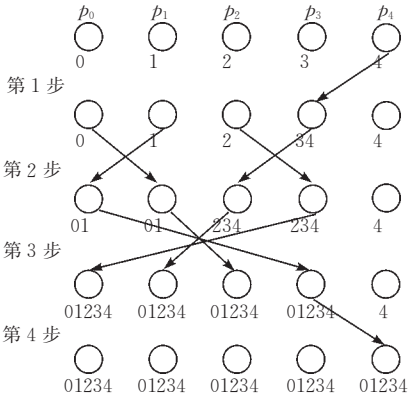

图 2 进程数为奇数的邻居交换算法

4 算法平均逻辑通信距离及通信局部性分析

目前，衡量通信局部性的参数为平均物理通信距离，以消息传递经过的网络跳数计量[6]. 通信模式平均通信距离越短，其通信局部性越好. 但是，由于实际的机器中进程与处理器之间的映射方式的不同，很难从一个算法的逻辑描述中计算出消息所经过的网络跳数，从而很难确定该算法的通信局部性. 为了便于对算法进行分析，本文提出两个新概念：逻辑通信距离和平均逻辑通信距离，并以此为基础重新定义了通信局部性.

定义消息的逻辑通信距离 d_{ij} 为消息发送进程 i 与接收进程 j 的编号之差的绝对值，即

$$d_{ij}=|i-j|.$$

用 d_{ijk} 表示算法的第 k 步消息传输中进程 i 与进程 j 的消息逻辑通信距离，则

$$d_{ijk}=\begin{cases}0, & \text{若第 } k \text{ 步进程 } i \text{ 与 } j \text{ 不通信}\\ d_{ij}, & \text{若第 } k \text{ 步进程 } i \text{ 发送消息给进程 } j\end{cases}.$$

定义 f_{ijk} 如下：

$$f_{ijk}=\begin{cases}0, & \text{若第 } k \text{ 步进程 } i \text{ 与 } j \text{ 不通信}\\ 1, & \text{若第 } k \text{ 步进程 } i \text{ 发送消息给进程 } j\end{cases}.$$

定义算法的平均逻辑通信距离 D 为算法所有消息逻辑通信距离之和除以消息总个数，即

$$D=\frac{\sum_k\sum_j\sum_i d_{ijk}}{\sum_k\sum_j\sum_i f_{ijk}},$$

其中 k 取所有的点到点消息传递步骤，i，j 取所有的进程号.

集合通信算法的平均逻辑通信距离是算法的一种基本属性，在有些情况下，实际的性能结果还要取决于进程映射及结点机的拓扑连接. 下面的分析假设处理器间的物理拓扑连接及进程映射能够保持进程间的逻辑关系，即进程号相邻的两个进程映射到的物理上的两个处理器也相邻，那么，算法的平均逻辑通信距离越小，它的通信局部性就越好.

对于 MPI Allgather 环算法，在每步发送的 p 个消息中，有$(p-1)$个消息在编号相邻的两个进程间传递，它们的逻辑通信距离为 1；剩下的一个消息由进程$(p-1)$传给进程 0，该消息的逻辑通信距离为$(p-1)$. 环算法共需要$(p-1)$步数据传输，因此算法的平均逻辑通信距离为

$$D=\frac{(1\times(p-1)+(p-1))\times(p-1)}{p\times(p-1)}=2-\frac{2}{p}.$$

对于邻居交换算法，首先讨论进程数为偶数的情况. 在奇数步中，每个消息的逻辑通信距离为 1；在偶数步中，有$(p-2)$个消息的逻辑通信距离为 1，剩下两个消息的逻辑通信距离为$(p-1)$. 当 p 为 4 的倍数时，奇、偶步骤各有 $p/4$ 步，算法的平均逻辑通信距离为

$$D=\frac{p\times(p/4)+((p-2)+2\times(p-1))\times(p/4)}{p\times(p/2)}$$
$$=\frac{2p-2}{p}=2-\frac{2}{p}.$$

当 p 不为 4 的倍数时，奇数步骤有$(p+2)/4$ 步，偶数步骤有$(p-2)/4$ 步，算法的平均逻辑通信距离为

$$D=\frac{p\times(p+2)/4+(3\times p-4)\times(p-2)/4}{p\times(p/2)}$$
$$=2-\frac{4}{p}+\frac{4}{p^2}.$$

对于邻居交换算法中进程数为奇数的情况，可看到所增加的 2 个消息均在进程$(p-1)$和进程$(p-2)$间传递，逻辑通信距离均为 1，不改变该算法的平均逻辑通信距离量级 $O(1)$.

对于递归倍增算法，当进程数为 2 的指数次幂时，每一步都发送 p 个消息，共需要 $\log p$ 步数据传输，且第 i 步$(0<i\leq\log p)$每个消息的逻辑通信距离为 2^{i-1}，算法的平均逻辑通信距离

$$D=\frac{p\times(2^0+2^1+\cdots+2^{\log p-1})}{p\times\log p}=\frac{p-1}{\log p}$$

进程数为非 2 的指数次幂的情况分析起来较复杂，但不影响量级 $O\left(\frac{p}{\log p}\right)$.

对于 Bruck 算法，通过观察可以发现，在第$k(0\leq k<\lceil\log p\rceil)$步，编号不小于 2^k 的进程发送的消息逻辑通信距离为 2^k，这样的消息有$(p-2^k)$个；编号小于 2^k 的进程发送的消息逻辑通信距离为$(p-2^k)$，这样的消息有 2^k 个. 算法的平均逻辑通信距离为

$$D=\frac{\sum_{k=0}^{\lfloor\log p\rfloor}(2^k\times(p-2^k)+(p-2^k)\times 2^k)}{p\times\lceil\log p\rceil}$$
$$=\frac{2p\times(2^{\lfloor\log p\rfloor+1}-1)-\frac{2}{3}\times(4^{\lfloor\log p\rfloor+1}-1)}{p\times\lceil\log p\rceil},$$

当 p 为 2 的指数次幂时，可简化为

$$\frac{4p^2-6p+2}{3p\times\log p}=\frac{4p-6+\frac{2}{p}}{3\log p}.$$

基于上述分析，邻居交换与环算法平均逻辑通信距离量级为 $O(1)$，具有很好的通信局部性性质；递归倍增算法与 Bruck 算法平均逻辑通信距离量级为 $O\left(\frac{p}{\log p}\right)$，通信局部性较差. 一般地，当 p 为偶数且大于 2 时，上述 4 个 MPI Allgather 算法消息的平均逻辑通信距离的数量关系为：邻居交换算法≤环算法＜递归加速算法＜Bruck 算法，其中 p 为 4 的倍数时，邻居交换算法与环算法之间取等号，否则取小于号.

表 1　4 种 MPI Allgather 算法平均逻辑通信距离

算法	平均逻辑通信距离	
环算法	$2-\frac{2}{p}$	
递归倍增算法	$\frac{p-1}{\log p}$ （注：此为 p 等于 2 的指数幂的结果，p 不为 2 的指数次幂时量级不变）	
Bruck 算法	$\frac{2p\times(2^{\lfloor\log p\rfloor+1}-1)-\frac{2}{3}\times(4^{\lfloor\log p\rfloor+1}-1)}{p\times\lceil\log p\rceil}$	
邻居交换算法	$2-\frac{2}{p}$ （p 为 4 的倍数）	$2-\frac{4}{p}+\frac{4}{p^2}$ （p 为偶数但不是 4 的倍数）

5　测试环境、方法及结果分析

为验证所提出新算法的正确性及有效性，我们在深腾 6800 上使用 QsNET 和快速以太网，在曙光 4000A 上使用快速以太网对上述 4 个 MPI Allgather 算法进行了测试. 所使用的深腾 6800 安装在中国科学院计算机网络信息中心超级计算中心，整体为 5 万亿次面向网格的超级计算机系统，包括 265 个四路结点机，每个结点配置为：4 颗 Intel Itanium 1.3GHz CPU，256KB 二级缓存，3MB 三级缓存，

8/16GB 内存，73GB SCSI 硬盘. 所使用的曙光 4000A 安装在上海超级计算中心，整体为 10 万亿次超级计算机系统，包括 640 个计算结点，每个计算结点配置为：4 颗 AMD Opetron 2.2GHz 64 位处理器，每处理器有 1MB 二级缓存，8GB 内存，内置一块 36GB 热插拔 Ultra320 SCSI 硬盘. 测试时每个结点使用一个处理器，测试使用的环、递归倍增以及 Bruck 算法实现与 MPICH-1.2.6[①] 中的算法实现一致.

文献[5]指出，对于 MPI Allgather 通信，短消息通信是指总通信数据量小于 80KB 的通信，中等长度消息总通信数据量介于 80KB 和 512KB 之间，长消息总通信数据量大于 512KB. 由于 MPI Allgather 通信的总通信数据量＝每个进程数据块大小×通信进程数，因此，我们选用单个数据块长度为 8B、8KB 以及 120KB，分别代表短消息、中等长度消息和长消息.

5.1 QsNET 的测试结果

当消息很短时，处理器准备发送或接收每个消息的时间开销(overhead)占总开销的大部分. 在这种情况下，通信步骤少的算法能够取得较好的性能. 图 3 中，递归倍增和 Bruck 算法比邻居交换及环算法所需的时间少，因为前二者需要的通信步骤为 $O(\log p)$，而后二者为 $O(p)$.

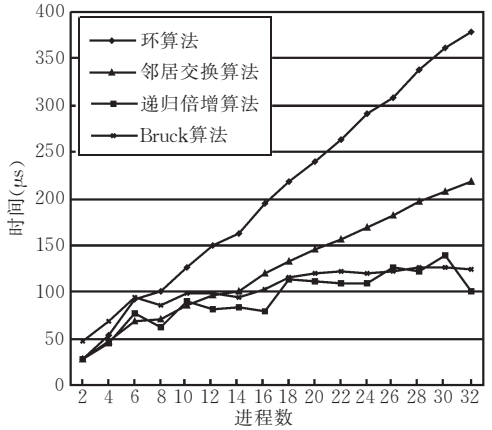

图 3 深腾 6800 QsNET 上 Allgather 算法的短消息(8B)通信性能

随着消息长度的增加，网络的延迟(指发送方开始发送消息的第一个字节到接收方接收消息的最后一个字节之间的时间间隔，与消息的长短有关)也增加了，并在总开销中占越来越大份额. 从图 4、图 5 可以看到，邻居交换算法在所有的算法中性能最好，相信这主要是由于它具有好的通信局部性所致，而在递归倍增和 Bruck 算法的许多通信步中，进行通信的进程相隔较远，带来了网络的拥塞和延迟增加. 虽然环算法也具有良好的通信局部性，但它需要的通信步骤是邻居交换算法的两倍，因此其性能也就次于邻居交换算法. 随着消息长度的增加，邻居算法与环算法之间的差距越来越小，说明在 QsNET 上，进程间两两交换消息通信模式并不比单向传递消息通信模式优越.

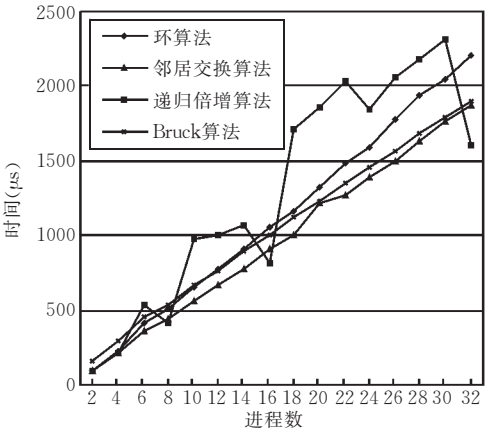

图 4 深腾 6800 QsNET 上 Allgather 算法的中等长度消息(8KB)通信性能

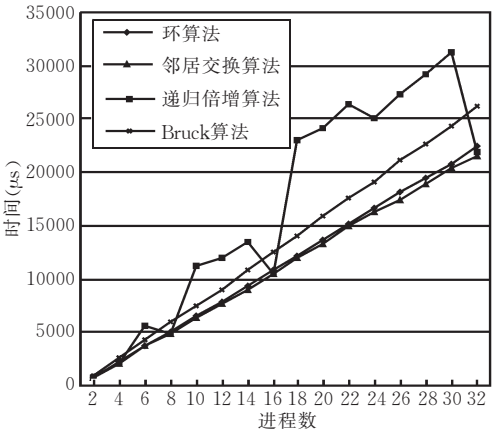

图 5 深腾 6800 QsNET 上 Allgather 算法的长消息(120KB)通信性能

5.2 快速以太网的测试结果

对于短消息通信，以太网上的结果与 QsNET 上的结果类似，都是递归倍增和 Bruck 算法的性能较好，见图 6，7.

在深腾 6800 和曙光 4000A 上测得的中等长度消息通信性能结果非常不一致. 如图 8，在深腾 6800 上邻居交换算法性能最差，而在曙光 4000A 上，如图 9，环算法与邻居交换算法的结果曲线几乎重叠在一起，二者的性能均为最优. 为什么在不同的机器上使用相同的网络和消息长度，邻居算法表现如此炯异？目前我们正在探索原因，还未找到令人满意

① http://www-unix.mcs.anl.gov/mpi/mpich/.

的答案. 对于以太网上中等长度的消息来说，环算法由于在两种机器上都表现稳定而胜出.

随着消息长度的增加，从图 10 和图 11 可以看到，邻居交换算法通信性能明显优于其它三个算法，特别是在曙光 4000A 上，邻居交换算法所需时间大概是第二快的 Bruck 算法时间的一半左右. 这是由于该算法的两个特性所致：(1)具有良好的通信局部性，有效地减少了网络拥塞；(2)使用消息交换模式，最小化 TCP 通信量，因此，在采用 TCP/IP 通信的快速以太网中，邻居交换算法获得了很好的性能.

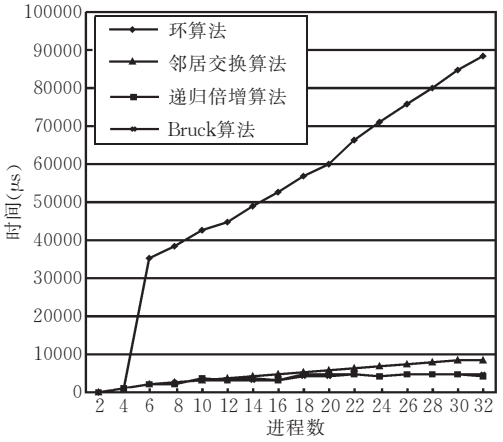

图 6　深腾 6800 以太网上 Allgather 算法的短消息(8B)通信性能

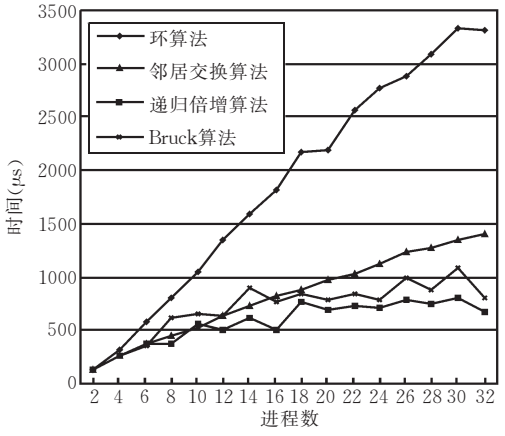

图 7　曙光 4000A 以太网上 Allgather 算法的短消息(8B)通信性能

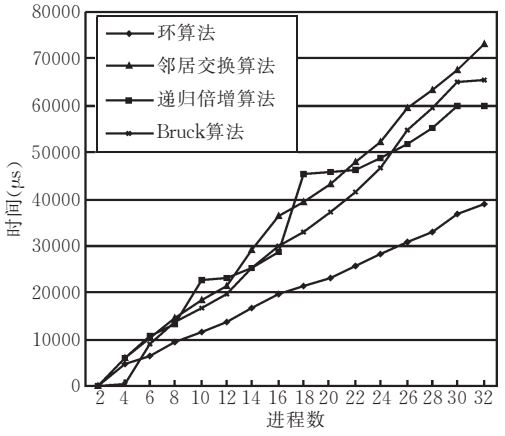

图 8　深腾 6800 以太网上 Allgather 算法的中等长度消息(8KB)通信性能

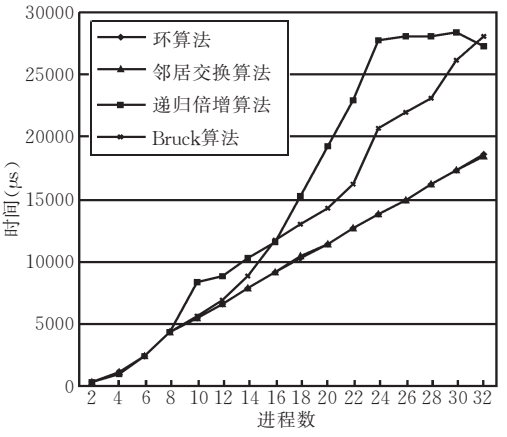

图 9　曙光 4000A 以太网上 Allgather 算法的中等长度消息(8KB)通信性能

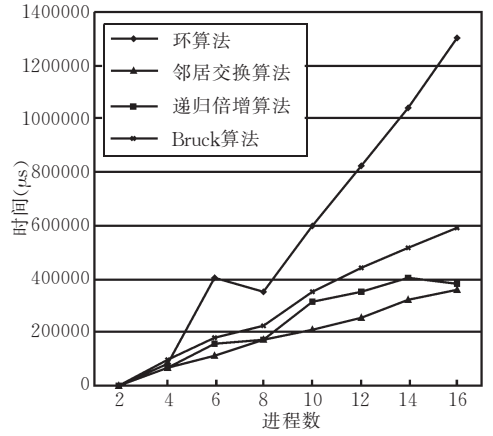

图 10　深腾 6800 以太网上 Allgather 算法的长消息(120KB)通信性能

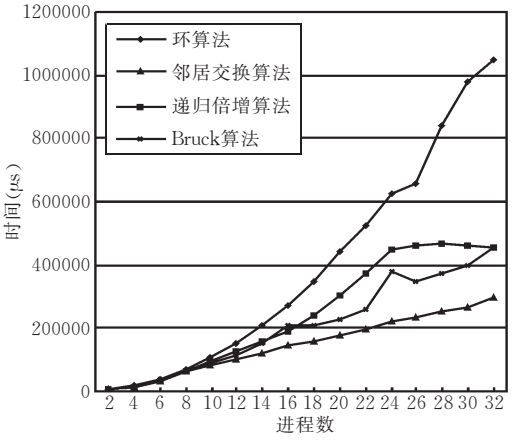

图 11　曙光 4000A 以太网上 Allgather 算法的长消息(120KB)通信性能

6 结论及未来工作

对集群通信算法进行性能测试是一项较为艰苦的工作,因为算法的性能取决于消息的长度、参与进程的数量、网络带宽及网络延迟.通过选择不同的机器、不同的网络、不同的进程数、进程映射以及不同的消息长度对不同的算法进行测试,我们看到:不同的集合算法适用于不同的网络和不同的消息长度,没有一个统一的算法能适用于所有情况;在处理器数目增多的情况下,算法的性能优劣会发生变化,原来不重要的性能因素会变得很重要,从而影响对合适的算法的选择.

测试结果证明,在 QsNET 上,本文所提出的邻居交换算法对于中长消息及长消息通信性能均为最优,但优势并不明显.在快速以太网上,邻居交换算法的长消息通信性能明显优于其它算法,中长消息通信性能表现不稳定,短消息通信性较差.测试结果表明,在进程数增多,通信量增大的情况下,通信局部性好的算法能够获得更好的性能,也表明本文提出的平均逻辑通信距离确实能够有效衡量算法的通信局部性.

在测试的时候,进程数和处理器数是一一对应的,即每个进程分配一个处理器.目前的测试在每 4 个处理器的结点上只分配一个进程,另 3 个处理器空闲.由于邻居交换算法具有近邻通信的特性,如果在每个结点上分配 2 个或者 4 个进程,邻居交换算法可能会取得更好的性能.我们计划进一步扩展测试工作,测试每结点分配 2 个或 4 个进程的 Allgather 算法通信性能.如果有机会还会在更多的大型机上进行测试工作.

致 谢 Argonne 国家实验室数学与计算机科学部 Rajeev Thakur 博士提供了环、递归倍增及 Bruck 算法的实现代码,并对不同的 Allgather 算法提供了深刻的讨论;本文的工作得到了中国科学院计算机网络信息中心超级计算中心的支持,在此表示感谢!

参 考 文 献

1 Benson G. D., Chu C. W. *et al*. A comparison of MPICH Allgather algorithms on switched networks. In: Proceedings of the 10th European PVM/MPI Users' Group Meeting, Venice, 2003, 335～343

2 Hensgen D., Finkel R., Manbet U.. Two algorithms for barrier synchronization. International Journal of Parallel Programming, 1988, 17(1): 1～17

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

4 Chan E. W., Heimlich M. F., Purakayastha A. *et al*. On optimizing collective communication. In: Proceedings of the 2004 IEEE International Conference on Cluster Computing, San Diego, 2004, 145～155

5 Thakur R., Rabenseifner R. *et al*. Optimization of collective communication operations in MPICH. International Journal of High Performance Computing Applications, 2005, 19(1): 49～66

6 Johnson K. L.. The impact of communication locality on large-scale multiprocessor performance. ACM SIGARCH Computer Architecture News, 1992, 20(2): 392～402

CHEN Jing, born in 1981, M. S. candidate. Her research interests include parallel software design and performance evaluation.

ZHANG Yun-Quan, born in 1973, Ph. D., associate professor. His research interests include performance evaluation, parallel software design, parallel computational model and parallel data mining.

ZHANG Lin-Bo, born in 1962, Ph. D., professor. His research interests include computational mathematics, parallel computing.

YUAN Wei, born in 1979, M. S., engineer. His research interests include parallel data mining, parallel software design and performance evaluation.

Background

This work is supported by the National Natural Science Foundation of China under grant No. 60303020 and No. 60533020. Research contents of these two projects include research on key techniques of performance evaluation, analysis and optimization for teraflops parallel computing systems and study on parallel algorithm and its application of the contemporary parallel computers.

In the past several years, the authors have proposed a new parallel computational model DRAM(h), which is based on the memory complexity concept. This paper focuses on optimizing the performance of MPI collective communication for terascale Linux clusters. In order to analyze and compare the complexity of various MPI Allgather algorithms, the authors propose the 'Average Logical Communication Distance (ALCD)' concept in this paper, it can be added to the DRAM(h) model as a new evaluation metric for collective communication on terascale parallel computing systems. This would increase the performance analysis accuracy of DRAM (h) model on terascale parallel computing systems.