计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2014.20121117
51(8):1863-1870,2014

Infiniband 网络中 All_to_All 通信性能优化

陈淑平　卢德平　陈忠平
(江南计算技术研究所　江苏无锡　214000)
(chenshuping_hit@163.com)

Optimizing All_to_All Communication in Infiniband

Chen Shuping, Lu Deping, and Chen Zhongping
(*Jiangnan Institute of Computing Technology*, *Wuxi*, *Jiangsu* 214000)

Abstract　All_to_All operation is an important collective function. No effective congestion control mechanism exists in current commercial Infiniband networks. Two typical All_to_All algorithms, the make-pair algorithm and the post-all algorithm, are studied in this paper. We found that their utilization of network bandwidth were between 30% and 70% when sending the messages which are larger than 32 KB. Further analysis and experiments demonstrate that it is the result of heavy congestion in the networks. In this paper, we adopt a novel algorithm to alleviate the congestion. It splits the large message into many small sized messages and sends them independently by an efficient schedule scheme. It creates one reliable connection for each process pair, and maintains a counter for each connection, which counts for the outstanding send requests. When the counter exceeds a predefined threshold value, congestion occurs between the two processes. Then it pauses the posting of send requests to the send queue of the congested connection in order to avoid the congestion spreading out through the entire network. The experiment results demonstrate that the new algorithm can alleviate the congestion effectively and improve All_to_All operation performance. Compared with the existed algorithms, its utilization of network bandwidth can improve 10% at least and 20% at most.

Key words　All_to_All algorithm; congestion control; message split; message schedule; Infiniband

摘　要　All_to_All 操作是一种重要的集合操作.目前的商用 Infiniband 网络中没有有效的拥塞控制机制.通过实验研究了 2 种典型的 All_to_All 算法在 Infiniband 网络中的性能,发现这些算法在传输大于 32 KB 的大消息时会在网络中产生严重的拥塞,从而导致网络带宽利用率仅有 30%～70%.尝试通过将大消息拆分成小消息、调度小消息的发送来减少网络拥塞.在任意 2 对进程间都建立可靠的连接,为每个连接都维护一个正在处理的发送请求计数器.当该计数器超过某个阈值后,认为这 2 个进程间的通信链路上发生了拥塞,此时停止向该连接的发送队列投递新的发送请求,以避免拥塞扩散到整个网络.实验结果表明该优化算法可以改善网络的拥塞程度;相比现有算法带宽利用率可以提高 10%以上,最多可以提高 20%.

关键词　All_to_All 算法;拥塞控制;消息拆分;消息调度;Infiniband

中图法分类号　TP393

收稿日期:2012-12-03;**修回日期**:2013-03-06

基金项目:国家科技支撑计划基金项目(2011BAH04B03)

All_to_All 函数是 MPI 及并行 C 中一种重要的集合操作[1-3]. 该操作包含 N 个进程,每个进程有 N 个长度相等的数据缓冲区,编号分别为 1,2,…, N;对任意 $1 \leqslant i, j \leqslant N$,进程 P_i 将其编号为 j 的缓冲区中的数据发送给进程 P_j,存放在进程 P_j 的编号为 i 的缓冲区中. 由此可见,All_to_All 操作是在做一个分布式的置换操作. All_to_All 操作的性能对并行程序来说非常关键. 在理想状态下,All_to_All 操作中每个进程的带宽等于网卡的链路带宽. 但实际上,各个进程在同时发送数据时会相互竞争通信路径,从而使网络中产生拥塞,导致进程的带宽明显低于链路带宽. 优秀的 All_to_All 算法可以通过各种方法避免网络中的消息相互竞争,从而提高网络的带宽利用率.

Infiniband[4]是由 Infiniband Trade Association 提出的一种基于交换的互连架构,具有低延迟、高带宽的特性. 在 2012 年 6 月份公布的 Top 500 高性能计算机中,超过 40%的系统采用 Infiniband 技术构建互连网络[5]. 本文将研究各类 All_to_All 算法在 Infiniband 中传输大消息时的性能,并根据 Infiniband 的特性提出一种优化的 All_to_All 实现方法.

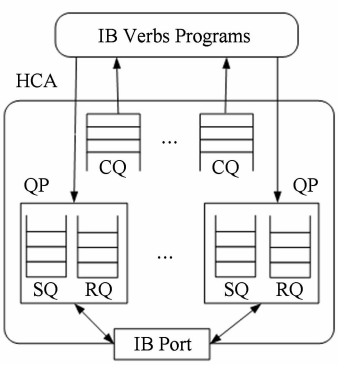
1 相关研究

1.1 Infiniband 简介

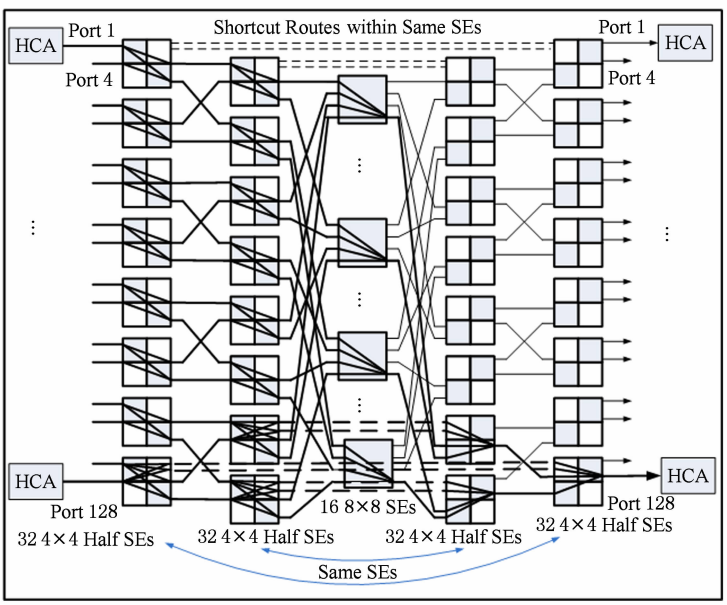
Infiniband 网络中,CPU 通过主通道适配器(host channel adapter, HCA)连接到 Infiniband 交换机上. HCA 结构如图 1(a)所示. 每块 HCA 允许创建多个 QP(queue pair, QP),每对 QP 可以实现两块 HCA 卡间的可靠数据传输. 在 Infiniband 中,用户使用 IB Verbs 消息库进行编程,收发操作都是非阻塞的. 发送消息时,用户首先向 QP 投递发送请求,然后不用等待发送完成即可返回继续做其他的工作. HCA 根据用户投递的发送请求将消息拆分成多个固定大小的数据包传递到 Infiniband 网络中. 当有多个 QP 时,HCA 采用循环调度的方式发送不同 QP 上的数据包,即先发送一个 QP 的数据包,然后切换到下一个 QP 发送同样数量的数据包. 当发送请求中的所有数据都传送完毕后,HCA 会向完成队列(complete queue, CQ)中写入一个完成 CQE (complete queue element). 用户程序可以通过查询 CQ 来获取发送请求的完成情况.

1.2 网络中的拥塞问题

Infiniband 利用基于信用的机制防止接收缓冲区发生溢出[6]. 每条物理链路都有一个信用值,表示该链路的接收缓冲还可以容纳多少数据,当信用为 0 时该链路不能再发送数据. 当网络中一条链路发生拥堵后,可能会很快传播至整个网络. 如图 1(b)所示[6],图左半部分的所有 HCA 同时向最右下角的那个 HCA 发数据,导致发往该 HCA 的数据量超过了其接收能力,在很短的时间内这些数据包就会



(a) The HCA structure



(b) The congestion tree

Fig. 1 The HCA structure and congestion tree.

图 1 HCA 结构及拥塞树

填满所流经的所有链路的接收缓冲,形成一棵以最右下角 HCA 为根的“拥塞树”.此时如果有其他数据包流经该“拥塞树”,就会因缺少信用而被堵塞,从而降低网络的整体性能.在 All_to_All 操作中,如果算法对消息发送顺序的调度不合理,就会出现多个进程同时向一个进程发消息的情况,使网络中产生严重的拥塞,导致 All_to_All 性能严重下降.

现有文献中有 4 种方法可以在一定程度上减少网络的拥塞程度:1)拥塞控制机制[6].当硬件检测到网络中某条通信链路存在堵塞时,会通知导致拥塞的发送源减少数据包的发射速率.2)动态路由算法[7].当网络中产生拥塞后,通过修改路由表将热点数据路由到其他空闲的通路来传送.3)LMC(LID mask count)机制[8].通过为一个 HCA 分配多个 LID,可以使两个 HCA 间有多条路由路径;当一条路径发生拥堵后,可以使用另外的路径传送数据.4)虚通道机制[9].每个虚通道都有独立的信用值,当一个虚通道被堵后,可以使用其他虚通道进行数据传输.上述 4 种机制都需要硬件提供支持,而目前的商用 Infiniband 网络不支持这些功能或仅提供了极其有限的支持.因此目前广泛使用的 All_to_All 算法均是通过软件方法来控制消息的发送、降低网络的拥塞.

1.3 现有的 All_to_All 算法

本文将 All_to_All 操作中一个进程发给另一个进程的整块数据称为一条消息.每次 All_to_All 操作中,每个进程需要创建 $N-1$ 个 QP,发送 $N-1$ 条消息.现有文献中的算法都是将消息作为一个发送请求处理的.最经典的算法有两种:配对发送算法和批量发送算法[10].

1.3.1 配对发送算法

该算法将 All_to_All 操作分为 $N-1$ 个步骤,在步骤 k 中,进程 P_r 仅向进程 $P_{(r+k)\%N}$ 发送数据,同时仅从进程 $P_{(r-k)\%N}$ 接收数据,以减少消息间的相互冲突,降低网络拥塞.该算法的性能依赖于进程的排序方法.该算法描述如下:

算法 1. 配对发送算法.

输入:进程个数 N;本进程编号 r.

① *barrier*();

② for i from 1 to N, do

③ 投递发往进程 $P_{(r+i)\%N}$ 的请求;

④ 查询 CQ,等待该请求完成;

⑤ *barrier*();

⑥ endfor

1.3.2 批量发送算法

该算法中,进程并不对消息的发送顺序进行调度,而是将 $N-1$ 个发送请求一起投递给 HCA,由 HCA 调度各数据包的发送.该算法描述如下:

算法 2. 批量发送算法.

输入:进程个数 N;本进程编号 r.

① *barrier*();

② for i from 1 to N, do

③ 若 $i\neq r$,则投递发往进程 P_i 的工作请求 WR_i;

④ endfor

⑤ 查询 CQ,等待这 $N-1$ 个工作请求全部完成;

⑥ *barrier*().

2 现有算法实验结果及分析

本节将在一个具有胖树结构的 Infiniband 网络中测试上述算法的性能.首先定义几个概念:

定义 1. All_to_All 带宽.All_to_All 操作中单个进程发送的总数据量/All_to_All 操作的总时间.

定义 2. All_to_All 带宽利用率.All_to_All 带宽/HCA 卡链路带宽.

2.1 实验环境

网络拓扑结构如图 2 所示.这是一棵两层胖树,每层有 16 个 32 端口的交换机.上层交换机除了连接下层节点的端口外,其余端口未使用.下层每个交换机可以连接 16 块 HCA 卡,因此该网络最多可以连接 256 块 HCA 卡.每个 CPU 上安装一块 HCA 卡,可运行 4 个虚拟机.采用最小跳路由算法(一种静态路由算法).测试代码使用 RDMA write[4] 操作实现消息收发.该实验环境中,两块 HCA 卡对连测试时理论上的最大带宽为 4.85 GBps.

由于 All_to_All 性能受底层路由影响很大,选择不同的节点,其拓扑结构就会不同.为了便于对各种算法进行比较,本文在进行每组测试时,各类算法都使用相同的节点.选择节点的方法是:先以交换机为单位将进程平均分配给各个交换机,然后在每个交换机中随机选出所需数目的 CPU.另外,该实验在测试配对发送算法的性能时,并没有根据网络拓扑结构对进程进行最优的排序,而是以一种随机的方式进行排序.

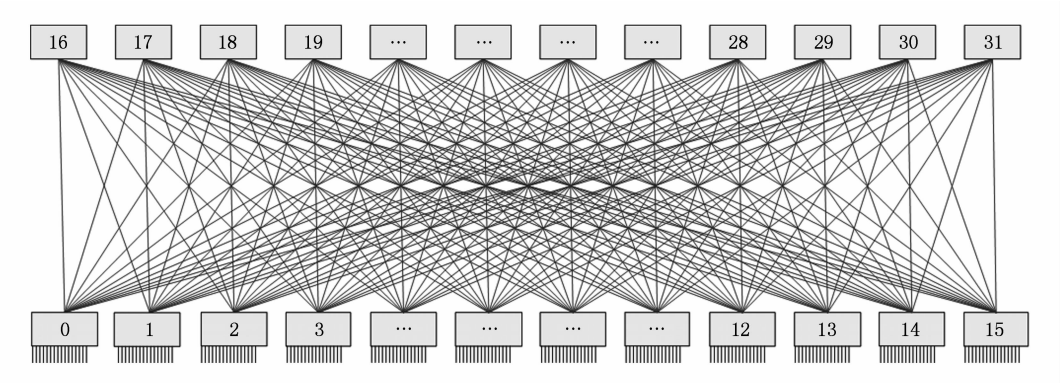


Fig. 2 The fat tree structure.

图 2 胖树结构

2.2 实验结果

图 3 分别显示了进程数为 64,128,256 时,配对发送算法和批量发送算法在发送不同长度的消息时的带宽利用率. 测试中每个 CPU 上运行一个进程.

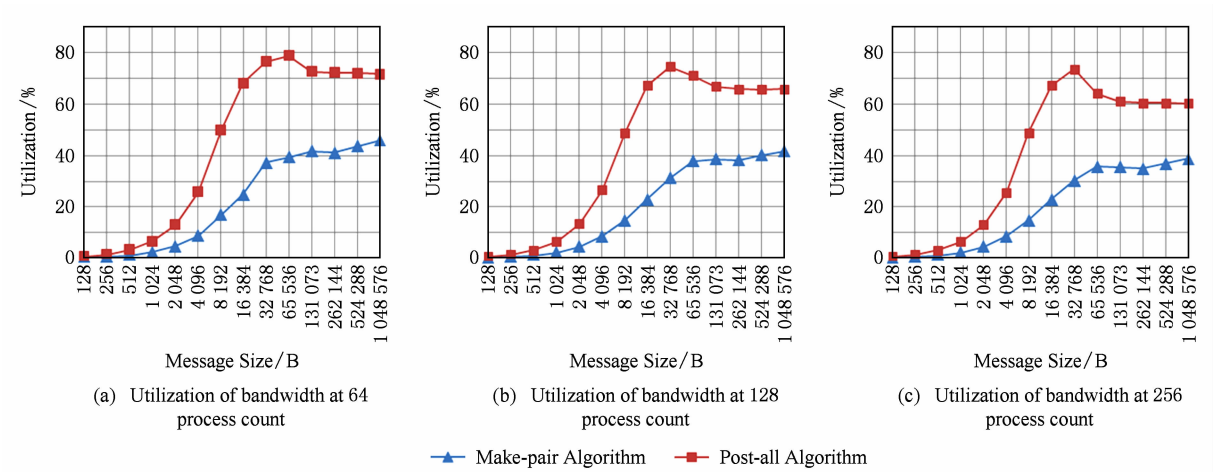


Fig. 3 Bandwidth utilization of existing algorithms.

图 3 现有算法的带宽利用率

从图 3 中可以看出,配对发送算法的带宽利用率随消息长度的增加而增加;在消息长度为 1 MB 时,带宽利用率达到最大值. 批量发送算法在消息长度不大于 32 KB 时,随着消息长度的增加,带宽利用率逐渐提高;当消息长度为 32 KB 时,带宽利用率达到最大值;此后带宽利用率呈下降趋势并趋于稳定. 本文仅研究长度大于 32 KB 的大消息的 All_to_All 传输性能. 从图 3 中可以看出,在各种进程数下,2 种算法在传送大消息时的带宽利用率仅有 35%~72%.

为了检测网络中的拥塞程度,在上述实验过程中记录了每个交换机端口的 PortXmitWait 计数器值. 在 IBA 规范中,该计数器定义为网络端口由于缺少信用或仲裁而导致不能发送数据的时间(以时钟滴答为单位). 实验结果表明,在这两种算法下,每个交换机端口的 PortXmitWait 计数器值都很大,这表明网络中产生了严重的拥塞. 批量发送算法中,由 HCA 调度数据包的发送,当多块 HCA 调度的数据包都流经同一条物理链路时,就会产生拥塞,从而降低了 All_to_All 的带宽利用率. 配对发送算法仅仅考虑了每个步骤中发出的消息在发送端及接收端是否产生冲突,而没有考虑在网络中是否会产生冲突,因此该算法也会产生严重的拥塞.

3 优化算法

现有算法都是将消息作为一个工作请求投递给 HCA. 本文尝试将消息拆分成小消息,通过对小消息的发送进行调度来达到降低网络拥塞、提高 All_to_All 带宽的目的.

3.1 理想的 All_to_All 通信模型

在 All_to_All 通信中,每个进程发送 $N-1$ 条消息,N 个进程共发送 $N(N-1)$ 条消息. 只要拓扑结构和使用的节点不变,不论采用何种算法,网络中

每条链路上传输的消息数都是固定不变的.假设每条链路最多传输 M 条消息($M \geqslant N$),则理论上 All_to_All 算法的最大带宽利用率为 N/M.如果控制每条消息的发射速率恒为 $Bandwidth/M$,那么即使所有消息同时发送,每条链路的最大负载也不会超过链路的传输能力,网络中就不会产生拥塞,从而可以获得最大的带宽利用率.

3.2 对理想模型的近似模拟

实际上,HCA 发送数据时是以数据包为单位进行调度的,并且在一段时间内会连续调度同一个 QP 上的数据包,因此消息的发射频率不会恒为 $Bandwidth/M$,这就会导致网络拥塞现象的发生.

为了减少网络拥塞的发生,HCA 要尽可能频繁地切换 QP 的调度.用户程序可以将大消息拆分成很多条小消息,通过控制这些小消息的发送,保证在任何时刻每个 QP 上仅有少数几个数据包等待发送.这样 HCA 在发送完一个 QP 上的所有数据包后就切换到下一个 QP,网络上产生拥塞的概率就会降低.为了充分利用网络带宽,需要同时向多个 QP 投递发送请求.在投递请求时维护一个发送窗口,每完成一个发送请求就以循环方式选择下一个 QP 补投新的请求,保证 HCA 上未完成的工作请求数等于该窗口大小.

3.2.1 小消息长度的选择

拆分后的消息长度越小,QP 的切换就越频繁,但投递发送请求、查询请求完成等的开销也越大.需要通过实验确定一个合适的小消息大小.

3.2.2 发送窗口大小的选择

发送窗口的大小对 All_to_All 算法的性能具有重要的影响.如果发送窗口太小,则不能充分利用链路的传送能力;如果发送窗口太大,则在一个发送窗口内同一个 QP 上会有很多个发送请求,导致每个 QP 上都有很多数据包等待发送.理想的发送窗口大小跟网络的传输能力、HCA 的调度方法有关.本文将通过实验确定一个合适的发送窗口.

3.2.3 拥塞避免

为了监测网络的拥塞程度,每个进程为每个 QP 都维护了一个计数器,该计数器记录了未完成请求的个数.设发送窗口大小为 $window$,进程个数为 N,则发送窗口中平均每个 QP 有 $window/(N-1)$个发送请求.当 QP 的未完成请求计数器大于等于 $window/(N-1)$时,该 QP 发出的消息肯定发生了拥塞.在补投新的发送请求时,不应该选择那些已经发生了拥塞的 QP.

3.2.4 虚拟机环境下的优化

在虚拟化环境中,一个 CPU 上可能运行多个虚拟机,每个虚拟机上都运行一个进程,两个 CPU 间可能有多对 QP 进行通信.如果这些 QP 同时进行通信,就可能产生冲突.为了避免冲突,可以采用下面的方法.令每个 CPU 上运行的虚拟机个数为 m,则两个 CPUX 与 Y 之间所有的通信包括:X_1Y_1,X_1Y_2,…,X_1Y_m,X_2Y_1,…,X_mY_m(X_i,Y_i 是虚拟机编号).将 All_to_All 操作分为 m^2 个步骤进行,在步骤 k 中,每个 CPU 的第(k/m)个进程向其他 CPU 的第$(k\%m)$个进程发送数据,保证每个步骤中每对 CPU 间最多有一对 QP 在通信.

3.2.5 伪代码描述的算法

为了简化算法,我们假设每个 CPU 上运行的虚拟机个数相等.算法如下所示:

算法 3. 优化的 All_to_All 算法.

输入:每个处理器上的进程个数 m(即每个 CPU 上的虚拟机个数);发送窗口大小 $window$;本进程在所属处理器上的编号 $myID$.

① for $step$ from 0 to (m^2-1), do
② if($step\%m==0$) do
③ $barrier()$;
④ endif
⑤ if ($myID==step/m$) do
⑥ 找出各个处理器上的第$(step\%m)$个进程,存放在数组 QPS 中;并将发往这些进程的消息拆分成小消息;
⑦ 依次向 QPS 中的 QP 投递发送请求,直到填满发送窗口为止;
⑧ 查询 CQ,每完成一个请求就从 QPS 中选一个 QP 补投新的请求.选择 QP 时,若该 QP 上所有数据已经发送完毕,或者未完成的工作请求数大于等于$(window/m)$,则跳过该 QP;若所有 QP 上的数据都发送完毕,则退出步骤⑧;
⑨ endif
⑩ endfor

4 实验结果及分析

4.1 实验中使用的参数

图 4 分别显示了进程数为 64,128,256 时,小消息长度及发送窗口大小对 All_to_All 性能的影响.

测试中每个 CPU 上运行一个进程，消息长度为 256 KB. 可以看出：

1) 在同一小消息长度下，发送窗口在 40～100 之间时，带宽达到最大值. 发送窗口增大到 200 后，带宽逐渐下降并趋于平稳. 这是因为发送窗口增大后，在一个发送窗口内同一个 QP 上有多个发送请求，从而容易造成拥塞.

2) 小消息长度为 16 KB 和 32 KB 时，相比小消息长度为 8 KB 和 64 KB 时带宽明显提高. 消息长度为 8 KB 时，由于传输的都是小消息，系统开销较大，带宽较低. 消息长度为 64 KB 时，没有充分将大消息拆分成可分离发送的小消息，带宽也比较低.

根据上述实验数据结果，我们选择小消息大小为 16 KB，发送窗口大小为 80.

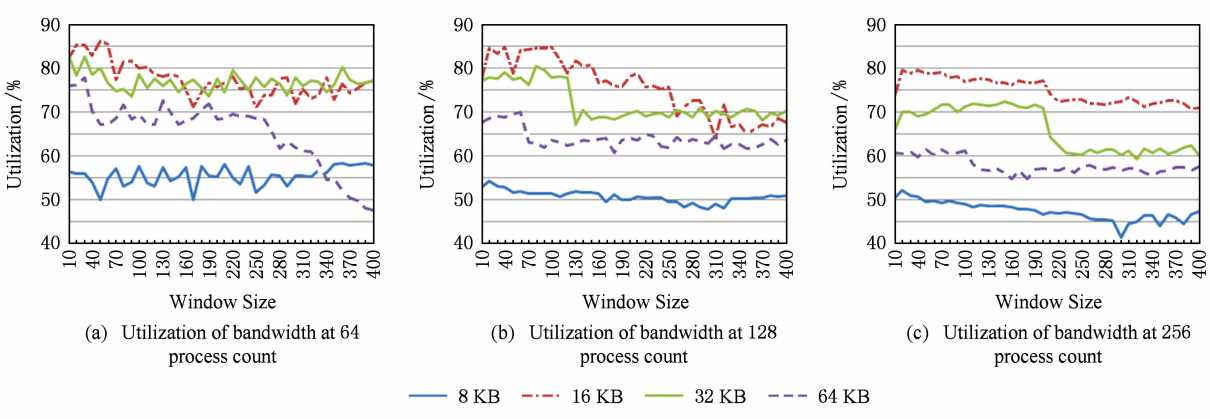


Fig. 4 Impacts of message size and window size on bandwidth utilization.

图 4 小消息长度和发送窗口大小对 All_to_All 性能的影响

4.2 实验结果

图 5 分别显示了 3 种算法在 64，128，256 个进程下传输不同长度消息时的带宽利用率，可以看出：

1) 配对发送算法的性能明显低于其他 2 种算法. 这是因为在该算法中每个 CPU 在每个步骤中仅向一个固定的 CPU 发数据，当该通路发生拥塞时，不能充分利用带宽向其他 CPU 发数据.

2) 当消息长度小于 64 KB、并且进程个数小于 256 时，新算法的性能相比批量发送算法没有优势. 这是因为在此条件下，网络中可供拆分调度的小消息个数比较少，不足以发挥调度的优势.

3) 当消息长度大于 64 KB 时，新算法的性能比批量发送算法有明显的提升，例如，在 128 KB 大小下，带宽利用率提高 10%左右，而在 1 MB 大小下，提高约 18%.

4) 进程个数相等时，批量发送算法中，消息长度越大，带宽利用率越低；而新算法中，消息长度越大，带宽利用率越高. 这是因为新算法将大消息拆分成了小消息，消息长度越大，越利于发挥调度的优势. 当消息长度大于 1 MB 时，新算法的带宽利用率会趋于平稳.

图 6 显示了每个 CPU 上分别运行 1，2，3，4 个虚拟机时各算法的性能. 测试中每个虚拟机上运行一个进程，共使用了 256 个 CPU，消息大小为 256 KB. 为便于比较，图 6 中显示的是 CPU 上所有虚拟机的聚合带宽. 可以看出随着虚拟机数的增加，进程个数

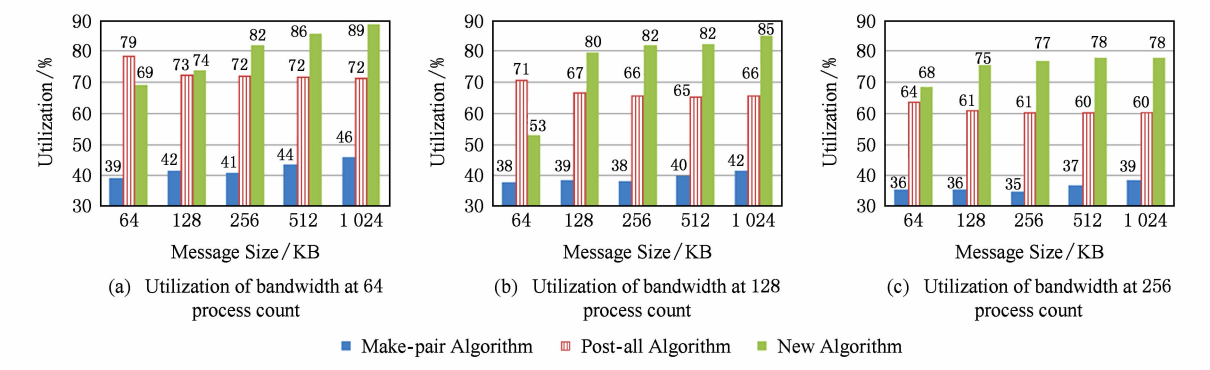


Fig. 5 Bandwidth utilization of the three algorithms at different process counts.

图 5 不同进程个数下各算法带宽利用率

随之增大,新算法的带宽利用率稍有下降,但仍然远高于其他 2 种算法. 随着虚拟机个数的增加,配对发送算法的带宽利用率反而增加. 出现这种现象的原因是:在虚拟机个数为 1 时,每个 CPU 在每个步骤中仅向一个固定的 CPU 发数据,当该通路发生拥塞时,不能充分利用剩余带宽向其他 CPU 发数据;而随着虚拟机个数的增加,每个 CPU 可以同时向多个其他 CPU 发数据,当一个虚拟机使用的通信链路发生拥塞后,其他的虚拟机还可以继续进行数据传输,从而提高了带宽利用率. 可以预见,当虚拟机个数增大到一定数值后,配对发送算法跟批量发送算法的行为实质上是一样的(都由硬件调度数据包的发送),此时其带宽利用率约等于批量发送算法,仍低于优化算法.

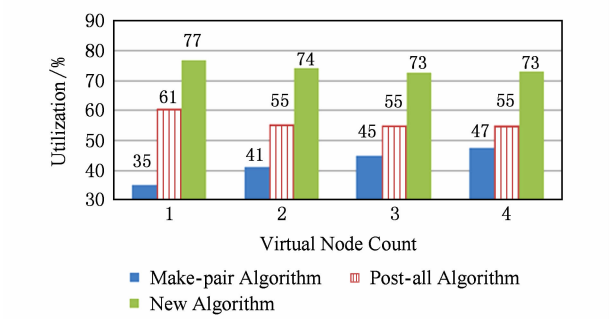


Fig. 6 Bandwidth utilization of the three algorithms in virtual environment.

图 6 虚拟机环境下 3 种算法的带宽利用率

我们的实验环境中每个处理器最多支持 4 个虚拟机,因此没有获得更大的虚拟机个数条件下的性能,但可以通过计算得出一个理论值.

设处理器个数为 N,每个处理器上虚拟机个数为 m. 每次 All_to_All 通信中,时间开销分为 2 个部分:1)同步时间;2)通信时间. 假设每个步骤中的通信时间固定不变(设为 X),则总的通信时间为 m^2X. 一次同步的时间开销跟 m 的大小有关,不妨记为函数 $B(m)$. 共需要进行 m 次同步,故同步的总时间为 $mB(m)$. 因此总时间开销为 $m^2X+mB(m)$. 设 2 个进程间交换的消息大小为 D,则每个 CPU 上所有进程传送的总数据量为 $m(Nm-1)D$,则单个处理器的带宽为 $\frac{m(Nm-1)D}{m^2X+mB(m)}$. 如果 $B(m)$ 随着 m 呈线性增长或对数增长,则上面计算出的带宽随 m 的增大而增大,并趋向于 ND/X.

可见,如果能够优化同步开销,随着 m 的增大,带宽利用率应该升高(这跟在处理器个数不变的情况下增大消息长度带宽利用率会随之提高的现象是一致的). 我们测出的实验结果中,随着 m 增大,带宽利用率会略有下降,这是因为在实际环境中,一次同步的开销可能随着 m 呈非线性增长.

5 结 论

本文提出了一种将大消息拆分成小消息、通过调度小消息的发送来减少网络拥塞、提高带宽利用率的 All_to_All 算法. 实验结果表明,在发送大消息时,该算法相比现有的算法,带宽利用率可以提高 10%以上,最多可以提高 20%.

参 考 文 献

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

[2] Sur S, Jin H W, Panda D K. Efficient and scalable All-to-All personalized exchange for InfiniBand-based clusters [C] // Proc of the 2004 Int Conf on Parallel Processing. Los Alamitos, CA: IEEE Computer Society, 2004: 275-282

[3] Mamidala A R. Scalable and high performance collective communication for next generation multicore Infiniband clusters [D]. Columbus, OH: Ohio State University, 2008

[4] Infiniband Trade Association. Infiniband architecture specification volume 1, Release 1.2.1 [EB/OL]. (2007-12-06) [2013-02-15]. http://www.infinibandta.org

[5] Top500 Supercomputer sites. Top500 list-June 2012 [EB/OL]. [2013-02-15]. http://www.top500.org/list/2012/06/

[6] Pfister G, Gustat M, Denzel W, et al. Solving hot spot contention using Infiniband architecture congestion control [C/OL] //Proc of the High Performance Distributed Computing, 2005. Los Alamitos, CA: IEEE Computer Society, 2005 [2013-02-15]. http://www.cercs.gatech.edu/hpidc2005/presentations/GregPfister.pdf

[7] Hoefler T, Schneider T, Lumsdaine A. Optimized routing for large-scale Infiniband networks [C] //Proc of the 2009 17th IEEE Symp on High Performance Interconnects. Los Alamitos, CA: IEEE Computer Society, 2009: 103-111

[8] Vishnu A, Koop M, Moody A, et al. Hot-Spot avoidance with multi-pathing over InfiniBand: An MPI perspective [C] //Proc of the 7th IEEE Int Symp on Cluster Computing and the Grid. Los Alamitos, CA: IEEE Computer Society, 2007: 479-486

[9] Subramoni H, Lai P, Sur S, et al. Improving application performance and predictability using multiple virtual lanes in modern multi-core Infiniband clusters [C] //Proc of the 39th Int Conf on Parallel Processing. Los Alamitos, CA: IEEE Computer Society, 2010: 462-471

[10] Rao Li, Zhang Yunquan, Li Yucheng, et al. Performance test and analysis of Alltoall collective communication on domestic hundred trillion times cluster system [J]. Computer Science, 2010, 37(8): 186-188 (in Chinese)
(饶立, 张云泉, 李玉成, 等. 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)

Chen Shuping, born in 1983. Received his master degree from the National University of Defense Technology. Assistant engineer. His main research interests include high performance computing and interconnect networks.

Lu Deping, born in 1966. Master, senior engineer. His main research interests include high performance computing and interconnect networks.

Chen Zhongping, born in 1975. Master, engineer. His main research interests include high performance computing and interconnect networks.