文章编号：2096-1618(2016)03-0277-08

基于 MPI 和 OpenMP 的排序算法并行优化研究

王 帅，喻 歆，何 嘉
(成都信息工程大学计算机学院，四川 成都 610225)

摘要：排序是计算机程序设计中的一项重要操作，其性能好坏决定整个程序性能的优劣。针对常见的快速排序、冒泡排序、归并排序、计数排序和选择排序这5种排序算法，分别用 MPI(message passing Interface)和 OpenMP(open multi-processing)并行化编程环境对其进行并行程序优化，研究分析 MPI 和 OpenMP 并行优化时的优缺点，并对比不同并行化技术下的加速比和开销等性能，为更高效的排序算法的并行程序设计奠定基础。

关 键 词：计算机应用；高性能计算；排序；并行编程；加速比；MPI；OpenMP
中图分类号：TP302.7 **文献标志码**：A

0 引言

排序算法是计算机科学、模式识别、人工智能、商业服务、高性能计算和大数据等领域最常用的知识[1-2]。基于排序算法的重要性，科研工作者不断在实践中开发出各种各样的排序算法。但是，由于各个排序算法的时间复杂度、空间复杂度和稳定性的不同与限制，使其使用范围有所不同。而随着计算机应用与发展的领域越来越广，再加之计算机硬件的速度和存储空间的局限性，怎样提高计算机性能并节省存储空间渐渐地成为高性能计算发展的方向。其中，数据处理和排序算法也已成为程序设计人员考虑的主要因素之一，排序算法的选择恰当与否直接影响程序的执行效率和内外存储空间的占用量，从而影响整个软件的性能。

而 MPI 消息传递模型[3]与 OpenMP 共享内存的应用程序编程接口[4-5]，能够很好地实现排序算法的并行化设计，以此提高排序算法的计算效率。所以，在针对某一问题时要选择合适的算法、设计高效的并行化策略才能够提高效率、节省资源，满足用户需求。

对常见的快速、冒泡、归并、计数和选择这5种排序算法分别用 MPI 编程环境实现进程之间的并行化，以及用 OpenMP 实现多线程之间的并行化，通过数据试验对比，总结 MPI 与 OpenMP 的优缺点、对排序算法的适用性进行分析。

1 算法描述

1.1 冒泡排序

冒泡排序的基本思想是[6]：对待排序列 $A[i]$，第一次遍历时，对要排序的序列进行两两比较得到最大值，并将最大值放到 $A[i]$ 位置，第二次遍历将剩下元素的最大值放到 $A[i-1]$ 位置，以此类推，直到待排序列为有序的序列。

冒泡排序的伪代码用过程 BUBBLE-SORT 表示，具体如表1所示。

表1 冒泡排序 BUBBLE-SORT 伪代码

```
BUBBLE-SORT(A)
for i ← 1to Length[A]
    do for j ← Length[A] decline to i +1
        do if A[j]<A[j-1]
            then change A[j]↔A[j-1]
```

冒泡排序是稳定的。时间复杂度为 $O(n^2)$，空间复杂度为 $O(1)$。

1.2 快速排序

快速排序的基本思想是[7-8]：通过一趟快速排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据要小，然后再按照按此方法对这两部分数据分别进行快速排序。

快速排序的伪代码用过程 QUICKSORT 表示，具体如表2所示。

表2 快速排序 QUICKSORT 伪代码

```
QUICKSORT(A,m,n)
if m<n
  then k←PARTITION(A,m,n)
       QUICKSORT(A,m,k-1)
       QUICKSORT(A,k+1,n)
```

收稿日期：2016-04-11
基金项目：四川省科技计划资助项目(2012GZ0111)

快速排序的关键部分是 PARTITION 过程,其功能是找到划分数据的位置,伪代码如表 3 所示。

表 3 快速排序 PARTITION 伪代码

```
PARTITION(A,m,n)
p←A[n]
i←m-1
for j←m to n-1
  do if A[j]≤p
    then i←i+1
      change A[i]↔A[j]
change A[i+1]↔A[n]
return i+1
```

快速排序是一个不稳定的排序算法,其最优时间复杂度为 $O(n\log_2{}^n)$,最坏的时间复杂度为 $O(n^2)$。空间复杂度为 $O(\log_2{}^n)$。

1.3 归并排序

归并排序的基本思想[9-10]:归并排序通常也采用递归实现,算法首先将待排序序列划分 2 个子序列,按此方法在对子序列划分,直到划分为只有一个一组的时候开始归并这些分组,不断的递归并直到原序列为有序。

归并排序的伪代码用过程 MERGE-SORT 表示,具体如表 4 所示。

表 4 归并排序 SORT 伪代码

```
MERGE-SORT(A,p,r)
if p<r
  then q←[(p+r)/2]
    MERGE-SORT(A,p,q)
    MERGE-SORT(A,q+1,r)
MERGE(A,p,q,r)
```

其中 MERGE 为归并 2 个排好序的子序列,伪代码如表 5 所示。

表 5 归并排序 MERGE 伪代码

```
MERGE(A,p,q,r)
m1←q-p+1
m2←r-q
create arrays L[1..m1+1] and R[1..m2+1]
for i←1 to m1
  do L[i]←A[p+i-1]
for j←1 to m2
  do R[j]←A[q+j]
L[m1+1]←∞
R[m2+1]←∞
i←1 and j←1
for k←p to r
  do if L[i]<R[j]
    then A[k]←L[i]
      i←i+1
    else A[k]←R[j]
      j←j+1
```

归并排序是稳定的排序算法。时间复杂度为 $O(n\log_2{}^n)$,空间复杂度为 $O(n)$。

1.4 计数排序

计数排序的基本思想是[11]:对于给定的待排序列中的每一个元素 x,统计该序列中值小于 x 元素的个数。一旦确定,就可以将 x 直接存放到最终的输出序列的正确位置上。

计数排序的伪代码用过程 COUNTING-SORT 表示,具体如表 6 所示。

表 6 计数排序 COUNTING-SORT 伪代码

```
COUNTING-SORT(A,B,m)
for i←0 to m
  do C[i]←0
for j←1 to length[A]
  do C[A[j]]←C[A[j]]+1
for i←1 to m
  do C[i]←C[i]+C[i-1]
for j←length[A] decline to 1
  do B[C[A[j]]]←A[j]
    C[A[j]]←C[A[j]]-1
```

计数排序是稳定的排序。其时间复杂度 $O(n+k)$,空间复杂度 $n+O(k)$(其中 k 是整数的范围)。

1.5 选择排序

选择排序的基本思想[12]:第 1 趟,在待排序的序列中选出最小的数据,将他与第一个元素交换;第 2 趟,在剩下待排序序列中选出最小的记录,将它与第二个元素交换;以此类推,直到倒数第二个与最后一个元素比较为止。

选择排序的伪代码用过程 SELECT-SORT 表示,具体如表 7 所示。

表 7 选择排序 SELECT-SORT 伪代码

```
SELECT-SORT(A)
for i←1 to n-1
  min←i
  for j←i+1 to n
    if A[min]>A[j]
      min←j
  change A[min]↔A[i]
```

选择排序是不稳定的排序,时间复杂度为 $O(n^2)$,空间复杂度为 O(1)。

2　编程环境与算法设计

2.1　MPI 编程环境

MPI(message passing interface)是一种消息传递编程模型实现的代表,图 1 为 MPI 消息传递模型。

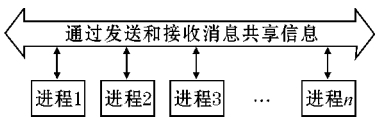

图 1　MPI 消息传递模型

在执行 MPI 程序时,进程之间执行实际上是进程之间发送和接收消息,其主要操作就是消息传递。消息传递的方式已经被广泛地运用于并行计算机上,尽管在实现方式上有所差异,但是通信方式采用消息传递已经成为了一种共识。虽然在 MPI 的函数库中包含有很多的消息传递函数,然而对于所有通信函数来说它们的基础是点对点通信。MPI 编程的优缺点[13]如表 8 所示。

表 8　MPI 编程的优缺点

优点	缺点
使用显示并行,提供更好的性能	进程通信开销大
可以静态任务调度	为避免较大的通信开销,并行化粒度较大
通信和计算能够同步进行	整体操作代价很高
数据的存储很少被检测到问题	并行化设计对串行代码的改动量较大
通信的同时产生同步,能够减少开销	动态负载很难均衡

MPI 程序设计一般采用主从模式和对等模式 2 种模式。

采用主从模式[13-14]设计的 MPI 程序中的每一个进程在执行任务时的功能、作用不一样,这些进程中的一个(或者多个)进程完成一类任务,而其余的进程则完成其他任务,并且执行不同任务的进程在程序中对应的代码也是不同的。

对等模式[13-14]指的是 MPI 程序中的每一个进程在执行任务时的作用和地位相同或相似,每个进程在 MPI 程序中对应的代码也是近似的,只是进程操作的数据不同。

并行化设计采用的对等模式,采用该模式可以减少子任务之间的数据相关性和通信开销,同时保证各个不同的排序算法之间的可对比性。

2.2　OpenMP 编程环境

OpenMP(open muti-processing)是专门为共享存储系统指定的标准,结合其计算性能和易用性,OpenMP 是在共享存储系统上编程的最优选择。图 2 为 OpenMP 共享内存模型。

OpenMP 编程环境设计的思路和 MPI 并行算法的设计思路基本一致,只是在 OpenMP 算法里用线程代替 MPI 中的进程,用共享变量代替 MPI 中消息传递的变量。OpenMP 的优缺点[13-17]如表 9 所示。

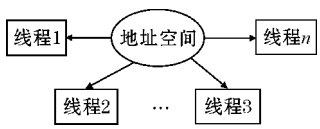

图 2　OpenMP 共享内存模型

表 9　OpenMP 编程的优缺点

优点	缺点
并行化程序容易实现	程序的可扩展性好
能够更好利用共享存储模型框架	进程或线程执行顺序是不确定的
允许运行调度	数据存储策略要合理
对细粒度和中粒度的并行都是有效的	需要显示同步机制
数据的存储问题很少被检测到	并行循环量太小时开销会成为很大的问题

2.3　MPI 并行化设计

基于 MPI 消息传递模型的算法设计,往往考虑到通信开销以及数据之间的相关性,为能够在同一条件下对冒泡、快排、归并、计数和选择排序算法进行性能分析和比较,采用的 MPI 并行化设计是对各排序算法进行进程之间的并行处理,MPI 并行化处理的思想如下:

(1)初始化待排序序列,并将序列由根进程根据进程数目均等地分给其他进程。(以数组形式 A[size][number]实现,size 为进程个数,number 为每个进程分配的数据数目)。

(2)并行化实现排序,各个进程独立地对各自分配的数据进行冒泡、快排、归并、计数和选择排序。该部分为主要的并行化实现部分。

(3)将各个进程排序得到的有序序列进行归并操作。

(4)归并结果由根进程输出。

MPI 并行化设计流程图如图 3 所示(以冒泡排序 Bubble_sort( )为例)。

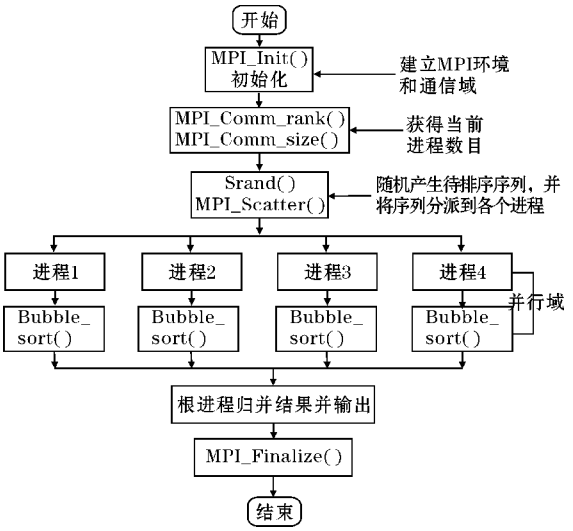

图 3 MPI 并行化设计流程图

2.4 OpenMP 并行化设计

OpenMP 并行化设计与 MPI 不同之处在于,MPI 是进程之间的并行,而 OpenMP 是线程之间的并行即将代码循环部分进行并行化处理,同时 OpenMP 提供了很多的编译指导语句从而使不同的 C 代码段进行循环并行化。

OpenMP 并行化设计思想如下:

(1)初始化操作,产生随机排序序列。

(2)利用#pragma omp parallel 和#pragma omp for 指导语句将待排序分给 4 个线程。

(3)利用#pragma omp parallel 和#pragma omp parallel sections 指导语句并行地对待排序列进行冒泡、快速、归并、计数和选择排序,该部分的功能是实现线程之间的并行排序。

(4)同样对排好的数据进行归并操作,并输出结果。

OpenMP 并行化设计流程图如图 4 所示(同样以冒泡 Bubble( )为例)。

在文中设计的进程[18]遵循 Linux 操作系统对进程的诠释,即进程是程序执行的一个实例。只有当程序被操作系统运行的时候,一个进程也就产生。

而线程[18]在 Linux 中则代表进程的一个执行流。为满足进程中多项工作可以并行工作,一般同一个进程中存在多个线程,每个线程代表程序的一部分工作,有时如果程序需要,也可以建立多个相同的线程执行类似的工作。

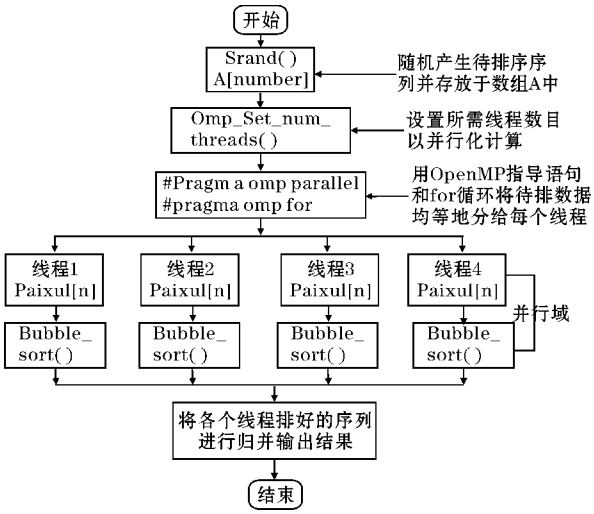

图 4 OpenMP 并行化设计流程图

3 实验运行

3.1 实验环境

进行的实验是在个人 PC 上运行的(四核 Inter(R) Core(TM) i5-4590S CPU @ 3.00 GHz 3.00 GHz 内存 8.00GB Windows7 系统),所用软件为 Microsoft Visual C++6.0 和 Microsoft Visual Studio2010、Matlab 以及 MPICH2、Microsoft Visual Studio2010 支持的 OpenMP 组件。

3.2 实验规模和参数设置

为更好地测试和分析排序算法的性能,该次实验过程中所使用的数据规模包括 500、600、800、1000、2000、3000、5000、8000、10000、20000、40000、60000、80000 和 100000。

在实验过程中分别在串行、MPI 并行和 OpenMP 并行情况下对 14 个不同规模的数据进行实验得出各自排序所需时间,计算 MPI 和 OpenMP 的各自加速比,并用 Matlab 画图,对两种并行化排序算法对比分析。

3.3 MPI 与 OpenMP 并行化时间复杂度分析

MPI 与 OpenMP 的时间复杂度与空间复杂度是大体相同的,而不同点是 MPI 是基于消息传递模型,不免会有通信开销,而 OpenMP 是基于共享内存的模型,对内存的使用和存储会有更高的要求,但却减少了通信开销。MPI 与 OpenMP 并行化时间复杂度、空间复杂度对比如表 10 所示。

由表 10 及以上分析可得,MPI 并行化更加适合粗粒度间的并行即节点与节点、进程与进程或 CPU 与

CPU,而 OpenMP 则适合细粒度或中粒度进程与进程或线程与线程之间的并行。实验使用的是单机多核机器,那么 OpenMP 并行化加速比的预期实验效果将比 OpenMP 并行化的效果要好一些(这里所说的效果指的是时间和加速比)。

说明:t_i 为 MPI 并行化编程设计的通信开销,k 为整数的范围。

表 10　MPI 与 OpenMP 并行化复杂度对比

排序算法	MPI 复杂度		OpenMP 复杂度	
	时间复杂度	空间复杂度	时间复杂度	空间复杂度
冒泡排序	$\frac{o(n^2)}{16}+t_i$	$o(1)$	$\frac{o(n^2)}{16}$	$o(1)$
快速排序	$\frac{n}{4}o(\log_2{}^{4n})+t_i$	$o(\log_2{}^{n})$	$\frac{n}{4}o(\log_2{}^{4n})$	$o(\log_2{}^{n})$
归并排序	$\frac{n}{4}o(\log_2{}^{4n})+t_i$	$\frac{o(n)}{4}$	$\frac{n}{4}o(\log_2{}^{4n})$	$o(n)$
计数排序	$o(\frac{n}{4}+k)+t_i$	$n+o(k)$	$o(\frac{n}{4}+k)$	$n+o(k)$
选择排序	$\frac{o(n^2)}{16}+t_i$	$o(1)$	$\frac{o(n^2)}{16}$	$o(1)$

4　实验结果与结果分析

4.1　实验运行结果

排序算法串行运行时间如表 11 所示,Matlab 曲线图如图 5 所示,时间结果为 20 次取平均值。

说明:由于快速排序的串行时间相对于其他排序的时间太小从图 5 看效果不是很明显,但从表 11 中也可以看出时间增加。

表 11　算法串行运行时间表/秒

问题规模	冒泡排序	快速排序	归并排序	计数排序	选择排序
500	0.000430	0.000110	0.000307	0.000590	0.000325
600	0.000608	0.000135	0.000310	0.000857	0.000491
800	0.001091	0.000179	0.000318	0.001452	0.000778
1000	0.001738	0.000277	0.000477	0.002303	0.001194
2000	0.006887	0.000472	0.004073	0.009323	0.004625
3000	0.018604	0.000827	0.009227	0.021263	0.010665
5000	0.057591	0.001395	0.024438	0.058007	0.028841
8000	0.151706	0.002766	0.056884	0.150195	0.072020
10000	0.258281	0.003022	0.094427	0.233305	0.114324
20000	1.044404	0.006019	0.351667	0.923614	0.450561
40000	3.000100	0.014038	1.397308	3.615806	1.798134
60000	9.424736	0.022615	3.394806	8.299873	4.185109
80000	15.01425	0.029955	6.037460	14.793053	7.309446
100000	24.99657	0.037942	8.890831	23.879600	11.62813

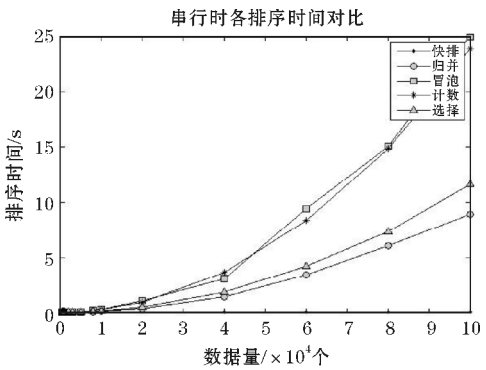

图 5　串行排序时间图

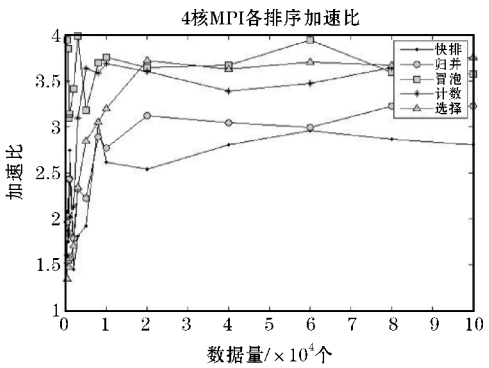

图 6　MPI 各排序加速比

由表 11 和图 5 可知,当数据规模小于 1000 时各排序算法时间相差很少,但随着数据量的增加,快速排序明显比其他排序算法要好,其次是归并、选择,冒泡和计数相对较差一点。

MPI 并行化设计之后的各排序算法时间表如表 12所示,Matlab 图形曲线如图 6 所示。

OpenMP 并行化设计之后的各排序时间表如表 13 所示,Matlab 图形如图 7 所示。

为便于分析将各个排序算法的 MPI 与 OpenMP 加速比情况进行分别对比,对比图如图 8 ~ 12 所示。

表 12 MPI 并行化各排序时间表/秒

问题规模	冒泡排序	快速排序	归并排序	计数排序	选择排序
500	0.000109	0.000053	0.000077	0.000367	0.000242
600	0.000158	0.000077	0.000093	0.000458	0.000318
800	0.000352	0.00091	0.000105	0.000793	0.000528
1000	0.000554	0.000101	0.000129	0.001144	0.000759
2000	0.002020	0.000326	0.000440	0.004360	0.002710
3000	0.004665	0.000456	0.000436	0.006864	0.004557
5000	0.018103	0.000726	0.000755	0.015963	0.010144
8000	0.041023	0.000911	0.000926	0.041912	0.023613
10000	0.068796	0.001154	0.001157	0.063230	0.035754
20000	0.286440	0.002369	0.002232	0.256258	0.120995
40000	0.817791	0.005011	0.004577	1.067099	0.496031
60000	2.388759	0.007635	0.007312	2.387043	1.130751
80000	4.182557	0.010455	0.009231	4.056434	1.990381
100000	6.991014	0.013529	0.011403	6.363970	3.097526

表 13 OpenMP 并行化各排序时间表/秒

问题规模	冒泡排序	快速排序	归并排序	计数排序	选择排序
500	0.000499	0.000102	0.000480	0.000191	0.000466
600	0.000521	0.000125	0.000700	0.000263	0.000570
800	0.000688	0.000140	0.000634	0.000455	0.000740
1000	0.000963	0.000205	0.000667	0.000669	0.001093
2000	0.002469	0.000588	0.003797	0.002648	0.003019
3000	0.005320	0.000669	0.007337	0.005776	0.006321
5000	0.018103	0.000741	0.009386	0.015682	0.012664
8000	0.042048	0.001411	0.028155	0.039476	0.024318
10000	0.080200	0.001885	0.046085	0.060409	0.039417
20000	0.303134	0.004495	0.106733	0.245778	0.137415
40000	0.871942	0.004961	0.420070	0.931799	0.469934
60000	2.538436	0.006795	0.958029	2.110588	1.071180
80000	4.186721	0.010334	1.625713	3.618784	1.841525
100000	7.345683	0.012040	2.502607	5.847106	2.962288

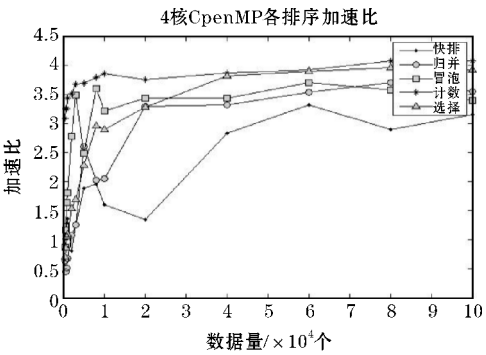

图 7 OpenMP 各排序加速比

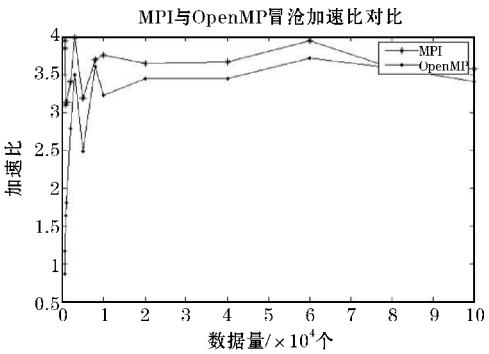

图 8 MPI 与 OpenMP 冒泡加速比对比

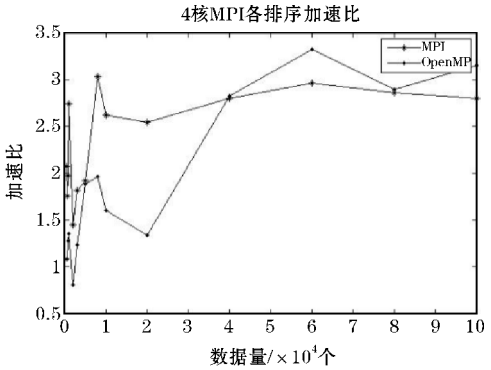

图 9　MPI 与 OpenMP 快速加速比对比

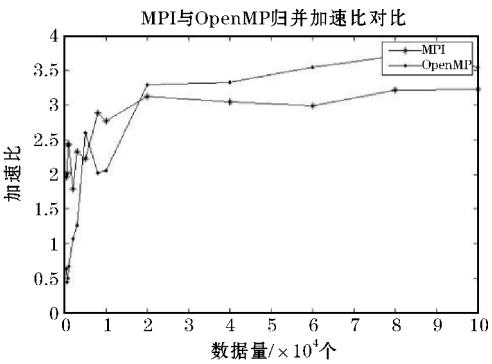

图 10　MPI 与 OpenMP 归并加速比对比

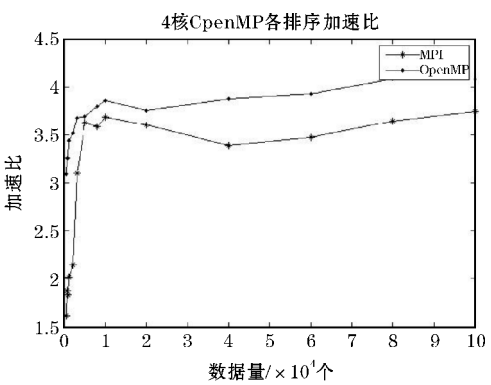

图 11　MPI 与 OpenMP 计数加速比对比

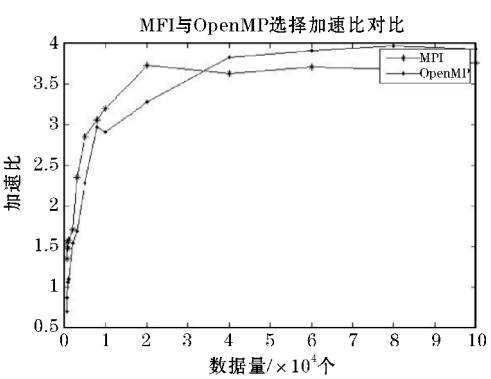

图 12　MPI 与 OpenMP 选择排序加速比对比

4.2　实验结果分析

根据 MPI 与 OpenMP 加速比对比图可以看到,用 MPI 和 OpenMP 将串行排序改为并行化实现后,计算效率得到很大地提升,而且 OpenMP 的加速比总体效果要比 MPI 的效果要好,这一点与预期效果一致。但是,由 MPI 与 OpenMP 各个排序算法的加速比对比曲线图可知,对于冒泡排序 MPI 并行化加速性能却一直好于 OpenMP 并行化加速性能且二者的加速比很接近,但并未达到预期效果。根据实验将 MPI 与 OpenMP 并行化性能分析归结为以下几点:

(1)MPI 与 OpenMP 并不是在所有的数据量下都能实现好的性能,而是在一定的数据范围内。从实验中整体来看,当数据量小于 40000 个时 MPI 并行化优于 OpenMP 并行化。而大于 40000 个时除了冒泡排序之外 OpenMP 并行化优化方法要好于 MPI 并行化优化。即:对于传统的串行排序利用 OpenMP 并行化方法进行优化,并不一定比 MPI 并行化优化所得到的加速性能要好,这要看选中的数据量范围以及采用的何种排序算法。

(2)在数据量小的时候,不管是 MPI 或者 OpenMP 优化,得到的加速比都不是很好,波动也大。而且数据量在小于 1000 时 MPI 与 OpenMP 并行化优化所得时间有时比串行时间很接近,优化效果不明显。但是到一定的数据规模时,MPI 与 OpenMP 优化整体上都能得到很好的并行加速比。

5　结束语

实验是在单核和 4 核的情况下对冒泡排序等 5 种排序算法进行串行、MPI 并行和 OpenMP 并行化设计,同时对各个排序在不同的优化方法下进行时间和加速比的对比,比较 MPI 与 OpenMP 的优缺点及实验所使用的优化方法对 5 种排序算法的适用性分析。从以上实验得出结论如下:对于文中的排序算法,在一定的数据规模内(>4×10^4 个),MPI 和 OpenMP 都能实现很好的并行优化;除冒泡排序之外,OpenMP 并行优化的各排序加速比要优于 MPI 并行优化;从时间复杂度和空间复杂度分析,OpenMP 并行优化所需时间要少于 MPI 并行优化,而所需的辅助空间是大致相同的。

对于采取的 MPI 与 OpenMP 并行化优化方法可得,在选择 MPI 和 OpenMP 进行并行化优化时要根据具体的情况进行分析,如实验环境、数据量级别、算法性能、客户需求等等。当然整体还有诸多缺点和不足,未来的研究方向是力求能够在更多核数、CPU 数大型集群上实验,而不是仅仅在单机多核上,同时要设计更加有效的并行化优化策略以及更多数量级的实验分析,以此满足未来高性能计算中对排序算法的高性能要求。

参考文献:

[1]　Cormen T H, Leiserson C E, Rivest R L, et al. In-

troduction to Algorithms[M]. 2nd ed. The MIT Press,2001.
[2] Dongarra J. The top 10 Algorithms[J]. IEEE Computing in Science & Engineering,2000,2(1):22-23.
[3] Innovative Computing Lab University of Tennessee. Advance MPI[R]. DavidCronk:Tennessee,2004.
[4] OpenMP Forum. OpenMP Fortran Application Program Interface[M]. 2000:15-25.
[5] David Kuek. OpenMP Overview[J]. WOMPAT, 2000:20-25.
[6] 郑国彪,曹侃宇. 冒泡排序法及其改进[J]. 青海大学学报:自然科学版,2002,20(3):43 -46.
[7] 谭浩强. C 语言程序设计[M]. 北京:清华大学出版社,2001.
[8] C A R Hoare. Quicksort[J]. The Computer,1962, 15(1):10-15.
[9] 严蔚敏,吴伟民. 数据结构(C 语言版)[M]. 北京:清华大学出版社,2014:283-284.
[10] Annay Levitin. The Design and Analysis of Algorithms[M]. Beijing:Tsinghua University Press,2004.
[11] 梁利刚,易超,杨绣丞,等. 静态排序算法设计与分析[J]. 计算机应用与软件,2012,29(3):283-286.
[12] Donald E, Knuth. Art of Computer Programming Volume3[M]. Sorting And Searching. (2nd Edition),2012.
[13] 张军. 多核集群下一种混合并行编程模型的研究[J]. 计算机应用,2009.
[14] 蒋沁谷. GRAPES 全球模式 MPI 和 OpenMP 混合并行方法[J]. 数值天气预报,2014.
[15] Rabenseifner R, Hager G, Jost G, et al. Hybrid MPI and OpenMP parallel programming[C]. 2006:11.
[16] Smith L, Bull M. Development of mixed model MPI/OpenMP Applications[J]. Scientific Programming,2001,9(2):83-98.
[17] 李建江,薛巍,张武生,等. 并行计算机及编程基础[M]. 北京:清华大学出版社,2011.
[18] 李晋. 进程的多对多(M:N)线程模型研究[J]. 计算机系统结构,2009.

The Performance Analysis of Sorting Algorithms based on MPI and OpenMP

WANG Shuai, YU Xin, HE Jia
(College of Computer Science and Technology, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Sorting is an important operation of designing a computer program. Its performance usually determines the quality of the whole program. This paper carries out parallel optimization of five common sorting algorithms, including quick sort, bubble sort, merge sort, counting sort and select sort, based on the MPI (Message Passing Interface) and OpenMP(Open Multi-Processing) parallel programming environment. The pros and cons using MPI and OpenMP are analyzed, and the performances including speedup and overhead are compared. The purpose of this paper is to lay the foundation for designing more efficient parallel sorting programs.

Key words: computer application; high performance computing; sorting; parallel programming; speedup; MPI; openMP