浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2020 年 4 月 2 日

课程名称：并行计算与多核编程　指导老师：楼学庆　电子邮件：wutongcs@zju.edu.cn
实验名称：MPI 编程　实验类型：综合型　联系电话：18888922355

一、 实验目的

- 利用 MPI 编程进行数据排序实验
- 比较不同程序选项下不同算法的程序性能

二、 实验原理

1. MPI

MPI 是由一组来自学术界和工业界的研究人员建立在各种并行计算体系结构设计的一个标准化的和便携式的消息传递系统。MPI 现已成为并行编程的事实标准，具有很高的可移植性，几乎所有平台都可以使用 MPI 实现。

mpi4py 是一个构建在 MPI 之上的 Python 库，实现了 MPI 标准中几乎所有的接口，包括点对点通信，集合通信、阻塞 / 非阻塞通信、组间通信等。

mpi4py 提供了接口 MPI.Init()，MPI.Init_thread() 和 MPI.Finalize() 来初始化和结束 MPI 环境。但是 mpi4py 通过在 init.py 中写入了初始化的操作，因此在我们 from mpi4py import MPI 的时候就已经自动初始化了 MPI 环境。MPI_Finalize() 被注册到了 Python 的 C 接口 Py_AtExit()，这样在 Python 进程结束时候就会自动调用 MPI_Finalize()，因此不再需要我们显式调用。

MPI 的集合通信允许在一个通信组内的多个进程之间同时传递数据。Scatter 和 Gather 都是集合通信操作。Scatter 可以向不同的进程发送不同的数据，且不是完全复制。Gather 是 Scatter 的逆过程，每个进程将发送缓冲区的消息发送给 0 号进程，0 号进程根据发送进程的进程号将各自的消息存放到自己的消息缓冲区中。

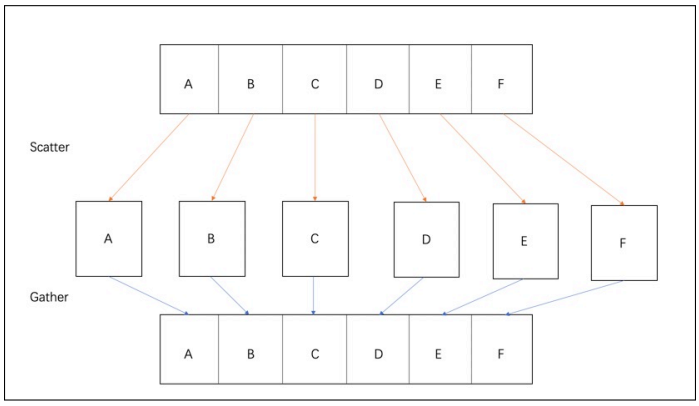

图 1: Scatter 和 Gather 的过程示意图

2. 冒泡排序

冒泡排序的基本思想就是：从数组的头部开始，逐个与相邻元素比较，根据大小交换位置，直到最后将最大的数据元素交换到了无序队列的队尾；下一次继续这个过程，直到所有数据元素都排好序。算法的核心在于每次通过两两比较交换位置，选出剩余无序数组里最大的数据元素放到队尾。

进行多线程优化时，首先进行数组的划分，每个子列在一个线程上进行小规模的冒泡排序，形成多个有序的子列。完成后将这若干个有序子列合并，即为排序完成。

冒泡排序的时间复杂度为 $O(N^2)$，优化后的时间复杂度为 $O(\frac{N^2}{p^2})$。

3. 归并排序

归并排序是一种采用分治策略的排序算法。归并排序中，数组并逐渐划分为更小的数组，再从小到大，对数组进行排序。每一级所要排序的数组都是下一级已经进行局部排序的子数组。

进行多线程优化时，首先将数组划分为若干个子列，分别进行归并，每个子列运行在不同线程上。

归并排序的时间复杂度为 $O(NlogN)$，优化后的时间复杂度为 $O(\frac{N}{p}logN)$。

4. 桶排序

桶排序的原理是首先确定数组的最大值和最小值，然后根据每个元素的值，将数组划分为若干个子列。再对每个子列进行排序，排序完成后直接按顺序合并这些子列，即为排序完成。

桶排序的特性非常适合多线程的实现。进行多线程优化时，将每个子列放在不同的线程上并行执行。

桶排序的时间复杂度为 $O(N)$，优化后的时间复杂度为 $O(\frac{N}{p})$。

三、 实验过程

本实验使用 MPI 的 Python 接口实现。首先将 mpi4py 引入进来，这一过程自动完成了环境的初始化。

```
from mpi4py import MPI
```

生成测试数据。根据传入的数据规模，生成一个包含如此规模数据的随机数组。

```
dataset_size = int(sys.argv[1])
data = []
for _ in range(dataset_size):
    data.append(random.randint(1, dataset_size))
```

记录开始时间，获取 MPI 环境参数，将数组划分成若干子列。在冒泡排序和归并排序中，根据其初始位置划分；在桶排序中根据其值划分。

```
start = MPI.Wtime()
comm = MPI.COMM_WORLD
thread_num = comm.size

# 冒泡排序或归并排序
new_list = []
```

```
for rank in range(thread_num):
   new_list = np.array_split(data, thread_num)
# 桶排序
data_min = min(data)
data_max = max(data)
bucket_size = int((data_max - data_min) / thread_num) + 1
new_list = []
for rank in range(thread_num):
   new_list.append([x for x in data if (x >= data_min + bucket_size *
       rank) and (x < data_min + bucket_size * (rank + 1))])
```

将划分后的数据分发到各个线程，各个线程内部调用相应的排序方法进行排序。

```
v = comm.scatter(new_list, 0)

# 冒泡排序
for i in range(len(v)):
   for j in range(len(v) - i - 1):
      if v[j] > v[j + 1]:
         v[j], v[j + 1] = v[j + 1], v[j]
# 归并排序
v = merge_sort(v)
# 桶排序
v = sorted(v)

g = comm.gather(v, 0)
```

将各个部分排序好的子列进行拼接，程序结束。

```
data_sorted = []
if comm.rank == 0:
   for i in range(len(g)):

      # 冒泡排序或归并排序
      data_sorted = list(heapq.merge(data_sorted, g[i]))
      # 桶排序
      data_sorted.extend(g[i])
   end = MPI.Wtime()
```

四、 实验结果

测试环境

- CPU：Intel(R) Core(TM) i5-7267U CPU @ 3.10GHz
- 操作系统：macOS Catalina 10.15.4

1. 冒泡排序

冒泡排序的测试用时和加速比如下所示：

运行时间 (s) 线程数 / 数据规模	1	2	3
10	0.000377	0.000901	0.006658
100	0.003148	0.001532	0.008744
1000	0.39051	0.092478	0.072355
10000	30.859356	10.292342	5.60702

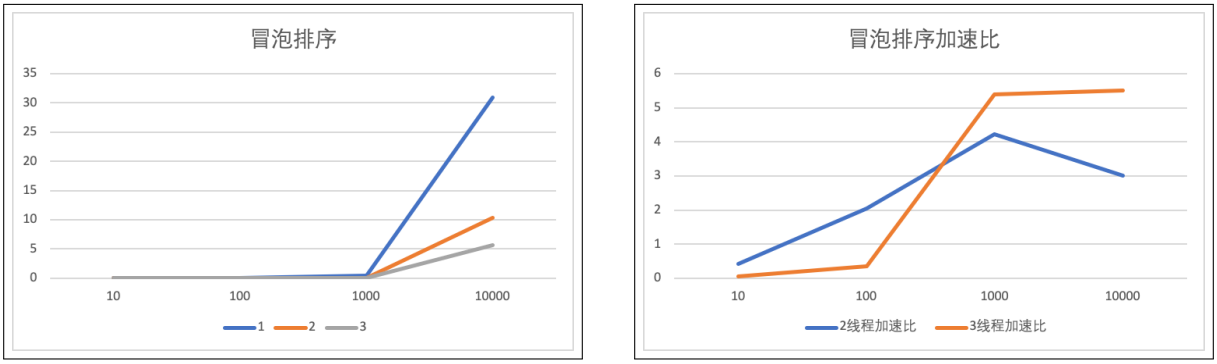

图 2: 冒泡排序测试用时和加速比

2. 归并排序

归并排序的测试用时和加速比如下所示：

运行时间 (s) 线程数 / 数据规模	1	2	3
10	0.000371	0.001064	0.00521
100	0.001527	0.001234	0.003943
1000	0.009784	0.006158	0.021999
10000	0.107708	0.0824	0.07041
100000	1.256005	0.675637	0.708869
1000000	14.559129	10.532762	7.976281
10000000	166.398289	90.149739	90.116808

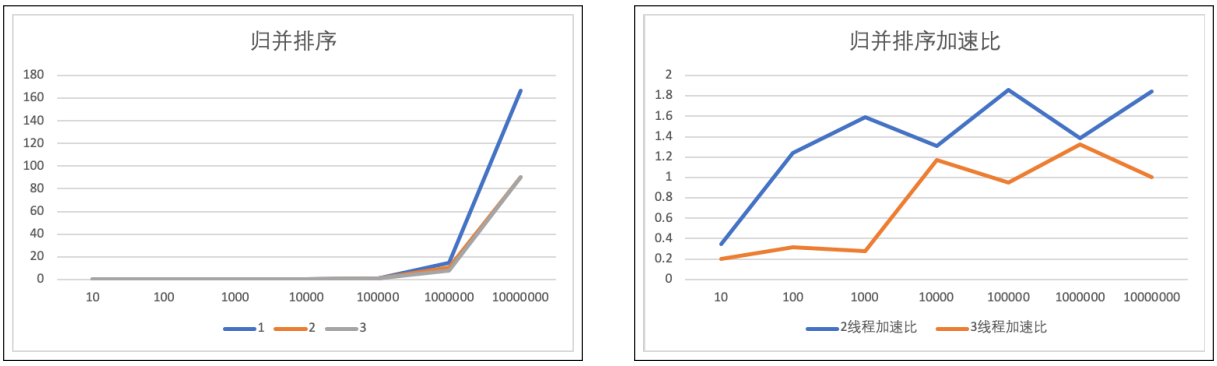

图 3: 归并排序测试用时和加速比

3. 桶排序

桶排序的测试用时和加速比如下所示：

运行时间 (s) 线程数 / 数据规模	1	2
10	0.000128	0.000531
100	0.000202	0.001457
1000	0.000748	0.001536
10000	0.007098	0.008363
100000	0.087463	0.085003
1000000	1.169745	1.329691
10000000	14.988145	11.509621
100000000	299.679751	265.53675

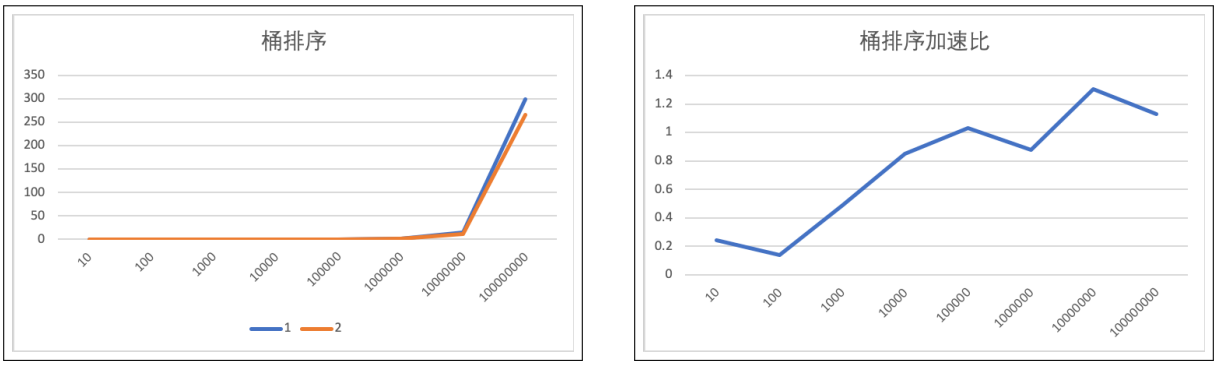

图 4: 桶排序测试用时和加速比

五、 分析讨论

实验的测试环境的 CPU 为双核，原则上 MPI 只能使用两个线程，运行时加上–use-hwthread-cpus 选项可最高达到三个线程。

在上述的三种排序算法中，当数据规模较小时，多线程的用时都要高于单线程的用时，这是由于在线程创建、通信上的开销高于执行算法的开销。当数据规模足够大时，多线程可体现出效果。在冒泡排序中，数据规模达到 100 时双线程就有加速效果，数据规模达到 1000 时，三线程的加速效果就已经优于双线程。在归并排序中，双线程的加速效果始终高于三线程的效果，且双线程在数据规模为 100 时就已经有效果，三线程要达到 10000 才有效果。在桶排序中，数据规模达到 1000 以上是，双线程体现出了加速效果。在以上三种排序算法中，冒泡排序的多线程加速效果最好，其次是归并排序，桶排序在数据规模较小时多线程的意义不大。

分析加速比随数据规模的变化趋势，三种排序算法中，总体而言加速比都是在增加的，但波动较大。导致这一现象的因素是复杂的。首先，测试环境的 CPU 只有双核，虽然物理上有四个线程，但操作系统会进行进程的调度，该程序的两个或三个线程并不是时时刻刻都同时运行在 CPU 上的。第二，影响 CPU 性能的因素主要有核心数和主频，而主频受温度的影响较大。当测试的数据规模较大时，CPU 的温度会升高，导致 CPU 频率降低，这在很大程度上影响了程序的运行时间。由于受到硬件条件的限制，只能定性地得出加速比随数据规模增加的初步结论，但无法进行定量研究。

通过本实验，我初步了解了 MPI 的特点和用法，并对操作系统和体系结构的知识有了更深入的理解。MPI 作为当下主流的并行计算标准，其功能甚为强大，值得进一步学习和研究。