实验一

PB17081504 廖洲洲

实验题目

利用MPI,OpenMP 编写简单的程序,测试并行计算系统性能

1. 求素数个数
2. 求Pi 值

实验环境

操作系统：Win 10

IDE：Microsoft Visual Studio 2015

编译器：cl.exe

硬件配置：CPU：Intel Core i7-8550U；CPU核心数:4；内存:8G

实验内容

1.求素数个数

算法设计与分析

将待判断的n个整数分为N(N为进程或线程个数)份，每个进程（线程）计算自己所要判断的数据中有多少素数。最后将各个进程（线程）所得结果相加即得最终结果。

核心代码

MPI

```
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    startTime = MPI_Wtime();
    for (i = rank; i <= N; i = i + size) {
        localSum += isPrime(i);
    }
    MPI_Reduce(&localSum, &sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

OpenMP

```
#pragma omp parallel private(i,localSum)
    {
    int id = omp_get_thread_num();
    for (i = id,localSum=0; i <= N; i = i + NUM_THREADS) {
        localSum += isPrime(i);
    }
#pragma omp critical
    sum += localSum;
    }
```

实验结果

MPI

运行时间

规模\进程数	1	2	4	8
1000	0.000221s	0.001101s	0.001518s	0.002353s
10000	0.001274s	0.001857s	0.002291s	0.002721s
100000	0.026143s	0.017839s	0.013789s	0.008678s
500000	0.164712s	0.153874s	0.094654s	0.053361s

加速比

规模\进程数	1	2	4	8
1000	1	0.200727	0.145586	0.093923
10000	1	0.686053	0.556089	0.46821
100000	1	1.465497	1.895932	3.01256
500000	1	1.070434	1.740148	3.086749

OpenMP

运行时间（单位s）

规模\进程数	1	2	4	8
1000	0.000105	0.000328	0.000567	0.000834
10000	0.001255	0.002271	0.001259	0.001232
100000	0.021992	0.021136	0.026098	0.009860
500000	0.284949	0.199826	0.125164	0.080235

加速比

规模\进程数	1	2	4	8
1000	1	0.320122	0.185185	0.125899
10000	1	0.55262	0.996823	1.018669
100000	1	1.0405	0.84267	2.230426
500000	1	1.425986	2.276605	3.55143

分析与总结

分析与总结

- 从运行时间来看，当进程数（线程数）相同时，随着数据规模的增大，运行时间也相应增大
- 当数据规模较小时，随着进程数（线程数）的增多，程序运行时间反而增大，加速比小于1且越来越小，说明当数据规模较小时，进程（线程）通信时间或切换所占程序总运行时间比重大，因此增加进程数（线程数）不能带来更好的性能
- 当数据规模较大时，随着进程数（线程数）的增多，程序运行时间逐渐减少，加速比大于1且越来越大，说明当数据规模较大时，进程（线程）通信时间或切换所占程序总运行时间比重小，增加进程数（线程）能有效提高性能。但是，程序加速比并不是随着进程数的增加线性增加的，加速比/进程数在逐渐减小

2.计算PI值

算法设计与分析

利用公式$\pi = \int_0^1 \frac{4}{1+x^2} dx$，然后使用Simpson公式计算积分值。对于n个小的积分区间，MPI将其分为N(进程个数)等分，每个进程计算完相应的积分值后相加发送给root进程，则在root进程可得最终PI值。OpenMP使用N个线程共同计算积分值，最后由一个线程将所有的积分值累加。

核心代码

MPI

```
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    startTime = MPI_Wtime();
    for (i = rank; i < N; i = i + size) {
        temp = (i + 0.5)*w;
        local = 4.0 / (1 + temp*temp) + local;
    }
    MPI_Reduce(&local, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

OpenMP

```
#pragma omp parallel private(i,x,sum)
    {
        int id = omp_get_thread_num();
        for (i = id, sum = 0; i < num_steps; i = i + NUM_THREADS) {
            x = (i + 0.5)*step;
            sum += 4.0 / (1.0 + x*x);
        }
#pragma omp critical
        pi += sum*step;
    }
```

实验结果

MPI

运行时间

规模\进程数	1	2	4	8
1000	0.000165s	0.001116s	0.001565s	0.002595s
10000	0.000374s	0.001240s	0.001702s	0.002705s
50000	0.001369s	0.001632s	0.001948s	0.002623s
100000	0.002538s	0.002389s	0.001560s	0.002597s

加速比

规模\进程数	1	2	4	8
1000	1	0.147849	0.105431	0.063584
10000	1	0.301613	0.219741	0.138262
50000	1	0.838848	0.702772	0.521921
100000	1	1.062369	1.626923	0.977281

OpenMP

运行时间(单位s)

规模\进程数	1	2	4	8
1000	0.000035	0.000255	0.000483	0.000871
10000	0.000113	0.000330	0.000493	0.000954
50000	0.001230	0.000805	0.000934	0.001144
100000	0.002420	0.001588	0.001141	0.001119

加速比

规模\进程数	1	2	4	8
1000	1	0.137255	0.072464	0.040184
10000	1	0.342424	0.229209	0.118449
50000	1	1.52795	1.316916	1.075175
100000	1	1.523929	2.120947	2.162645

分析与总结

- 从运行时间来看，当进程数（线程数）相同时，随着数据规模的增大，运行时间也相应增大
- 当数据规模较小时，随着进程数（线程数）的增多，程序运行时间反而增大，加速比小于1且越来越小，说明当数据规模较小时，进程（线程）通信时间所占程序总运行时间比重大，因此增加进程数（线程）不能带来更好的性能,反而会降低程序性能
- 当数据规模较大时，随着进程数（线程数）的增多，程序运行时间逐渐减少，加速比大于1且越来越大，说明当数据规模较大时，进程通信时间所占程序总运行时间比重小，增加进程数（线程）能有效提高性能。但是，程序加速比并不是随着进程数的增加线性增加的，加速比/进程数在逐渐减小。甚至在迭代次数为100000，进程数增大为8时，加速比又重新变为小于1，说明进程数不是越多越好，进程数越大，随之而来的通信成本也越大。

实验部分截图

求素数个数

MPI N=500000

```
E:\Visual Studio Projects\MPI_CountPrimes\x64\Debug>mpiexec -n 1 MPI_CountPrimes.exe
素数个数为:41538
运行时间:0.096385s

E:\Visual Studio Projects\MPI_CountPrimes\x64\Debug>mpiexec -n 2 MPI_CountPrimes.exe
素数个数为:41538
运行时间:0.069572s

E:\Visual Studio Projects\MPI_CountPrimes\x64\Debug>mpiexec -n 4 MPI_CountPrimes.exe
素数个数为:41538
运行时间:0.036921s

E:\Visual Studio Projects\MPI_CountPrimes\x64\Debug>mpiexec -n 8 MPI_CountPrimes.exe
素数个数为:41538
运行时间:0.029593s
```

OpenMP n=500000

NUM_THREADS=1,2,4,8

```
素数个数为:41538
运行时间:0.102728
```

```
素数个数为:41538
运行时间:0.095102
```

```
素数个数为:41538
运行时间:0.083801
```

```
素数个数为:41538
运行时间:0.046637
```

计算PI值

MPI N=100000

```
E:\Visual Studio Projects\MPI_ComputePI\x64\Debug>mpiexec -n 1 MPI_ComputePI.exe
pi is 3.141593
运行时间:0.005802s

E:\Visual Studio Projects\MPI_ComputePI\x64\Debug>mpiexec -n 2 MPI_ComputePI.exe
pi is 3.141593
运行时间:0.001263s

E:\Visual Studio Projects\MPI_ComputePI\x64\Debug>mpiexec -n 4 MPI_ComputePI.exe
pi is 3.141593
运行时间:0.001249s

E:\Visual Studio Projects\MPI_ComputePI\x64\Debug>mpiexec -n 8 MPI_ComputePI.exe
pi is 3.141593
运行时间:0.001558s
```

OpenMP N=100000

NUM_THREADS=1,2,4,8

```
pi is 3.141593
运行时间为:0.001740
```

```
pi is 3.141593
运行时间为:0.001377
```

```
pi is 3.141593
运行时间为:0.000829
```

```
pi is 3.141593
运行时间为:0.000740
```