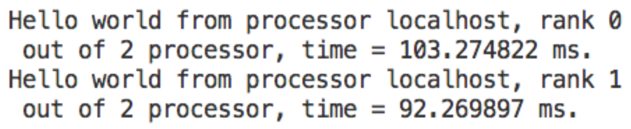
Technical Report for First Project in Parallel Computing

Yasheng Sun
117020910076
sunyasheng123@gmail.com

function	meaning
MPI_Comm_size	get number of processors
MPI_Comm_rank	get id of current processor
MPI_Get_processor_name	get name of current processor
MPI_Wtime	get the current time

Table 1. Basic function of MPI.



```
Hello world from processor localhost, rank 0
 out of 2 processor, time = 103.274822 ms.
Hello world from processor localhost, rank 1
 out of 2 processor, time = 92.269897 ms.
```

Figure 1. Ouput in the Hello World project.

Abstract

This report concludes the first project in parallel computing. This project involves the basic usage of MPI function and application of MPI in Mento Carto Method, which is illustrated through approximation of π. All my code is publicly available on my github https://github.com/sunyasheng/Parallel-Computing.

1. Hello World

This problem involves basic usage of MPI function as is shown in Table 1.

1.1. Result

The output of this first hello world MPI program is presented in Fig. 1. It is just some navie ouputs which notify us that there are actually multiple threads runing this program.

2. Computation of π through Mento Carto Method

This project involves a MPI program to approximate the π. Mento Carto method is a common simulation method to do approximation in computation. In this section, we will first introduce the method to finish the π approximation problem and then illustrated the result.

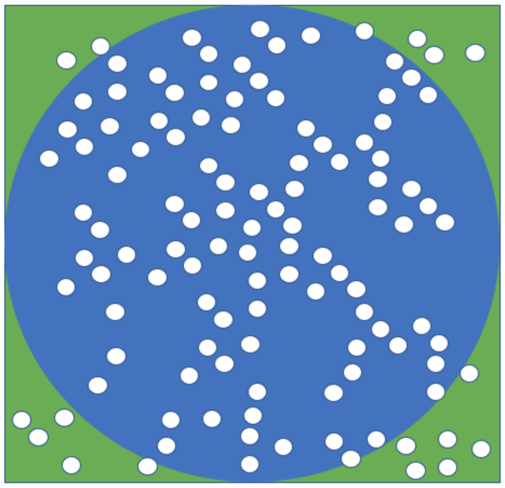


Figure 2. Illustration of π computation through Mento Carlo Method. π is approximated by the fraction of points located in the circle.

2.1. Methodology

Here we use the Mento Corto method to compute π. The basic principle to compute π is shown in Fig. 2. We sample the points randomly in the green square, whose locations are split into two subsets. π is approximated by calculating the fraction of all the points located inside the circle.

In this project, we just achieve the parallel vision of the above approach. Mutiple threads are launched and each of them holds a bunch of particles which will locate inside the square randomly. For each thread, we count the number of particles belong to the circle. After that, we sum up all these numbers in different processors and divide it by the total number of particles to obtain π in the reduced processor.

2.2. Result

The estimated value and consumed time under different number of processors when the load number of particles is 10000 is given in Table 2. The estimated value and con-

processor number	estimated value	consumed time
1	3.126	69.95ms
2	3.138	90.72ms

Table 2. Estimated value and consumed time with different number of particles.

load particles number	estimated value	consumed time
10000	3.134	98.14ms
100000	3.142	142.95ms

Table 3. Estimated value and consumed time under different load number of particles in each processors.

sumed time under different load number of particles when the processor number is 2 is given in Table 3. With the increasing number of the processors or increasing number of particles in every processors , the estimated number approximates true value more accurately since total number of particles here equals their mutiplication. More involved particles lead to a more precise result. In the meantime, longer time is required because time consumption in each processor is increased and the reduced processor has to wait all other processors finish before reduction.

3. Conclusion

In this project, we explore the basic usage of MPI function through a π estimation task. Experiment shows that the approximation accuracy can be improved by adding more particles with cost of time consumption. Future work will involve more complicated task under more advanced parallel framework.

References

[1] https://www.openmp.org/wp-content/uploads/Intro_To_OpenMP_Mattson.pdf