

Project 1

PRIME NUMBER

By

MISS. SASIRAT HARNWATTHANASIRI 6088024

MR. NATTAWAT LUMTANSAWAN 6088085

MR. WARIS VORATHUMDUSASEE 6088128

MISS. PANTITA WANG 6088219

MR. SIRICHOKE YOoyEN 6088232

MISS. PHUMMARAT YOSAMORNSUNTORN 6088233

A REPORT SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR
ITCS443 PARALLEL AND DISTRIBUTED SYSTEM

FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY
MAHIDOL UNIVERSITY 2019

This report described Sieve of Eratosthenes (finding the prime number) using MPI. There are 7 points that need to explain, which include algorithm, implementation, evaluation, performance results, discussion of the results, analysis of the algorithm, and further optimization.

1 Algorithm & Implementation

This algorithm was separated into 2 parts which are sequential and parallel version.

• Sequential Version Algorithm [1]

1. Create a list of unmarked natural number 2, 3,..., n
2. $k \leftarrow 2$
3. Repeat
 - Mark all multiples of k between k^2 and n.
 - $k \leftarrow$ smallest unmarked number $> k$ until $k^2 > n$
4. The unmarked numbers are primes

• Parallel Version Algorithm [1]

1. Create a list of unmarked natural number 2, 3,..., n. Each process creates its share of the list.
2. $k \leftarrow 2$ (All process performs this)
3. Repeat
 - Mark all multiples of k between k^2 and n. (Each process marks its share of the list)
 - $k \leftarrow$ smallest unmarked number $>$ (Process 0 only)
 - Process 0 broadcasts k to rest of processes until $k^2 > n$
4. The unmarked numbers are primes
5. Reduction to determine the number of primes

• Analysis of the Parallel Algorithm

1. Let α be the time to mark a cell.
2. Sequential execution time: $\alpha n \log \log n$
3. Computation time of parallel program: $\alpha n \log \log n / p$
4. Number of broadcasts: $\sqrt{n} / \log \sqrt{n}$
5. Broadcast time: $\lambda \lceil \log p \rceil$
6. Reduction time: $\lambda \lceil \log p \rceil$
7. Expected parallel execution time: $\alpha \frac{n \log \log n}{p} + \frac{\lceil \log \rceil}{\log \sqrt{n}} + \lambda \lceil \log p \rceil$

2 Evaluation

In this part, a comparison between sequential and parallel version. In the sequential, if n is equal to 100, the program will start at index 0 to index 99. In other words, sequential uses a global index to run the program. While parallel version, if n is equal to 100 and 4 processes are used, the block will contain four blocks, and each of them includes 25 indexes with the same number of size. So, it uses the local index.

3 Performance Results

From (a)Sequential in Figure 1, all answer is 78498 numbers of prime from 1-1,000,000 and it uses 0.610000 seconds to execute. On the other hand, (b)Parallel in Figure 2, all answer is 78498 numbers of prime from 1-1,000,000 and it uses 0.009813 seconds to execute. Therefore, that means the difference is about the execute time.

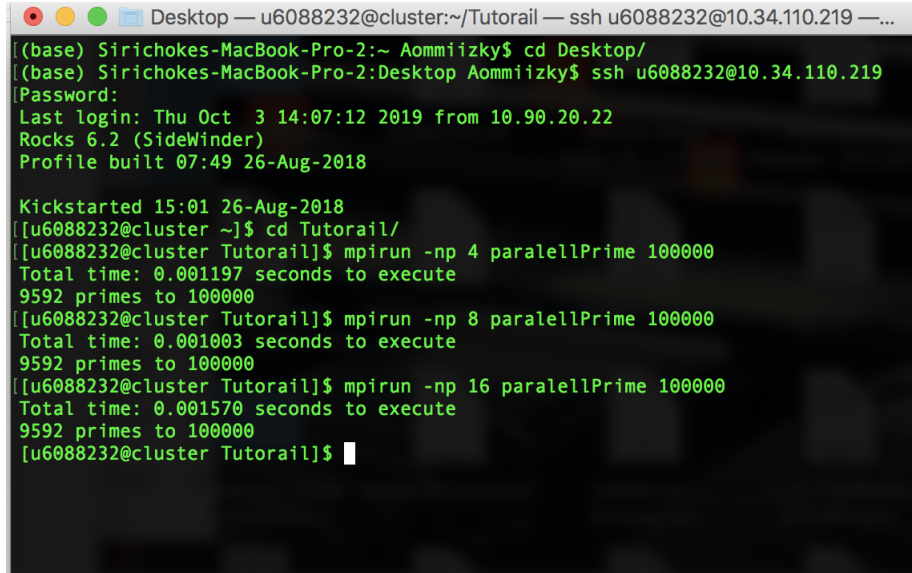
```
qewnattawat — u6088085@cluster:~ — ssh u6088085@10.34.110.219 — 80x...
999671
999683
999721
999727
999749
999763
999769
999773
999809
999853
999863
999883
999907
999917
999931
999953
999959
999961
999979
999983
All answer = 78498
Time = 0.610000 seconds to execute
Input the length of number
```

(a) Sequential

```
qewnattawat — u6088085@cluster:~ — ssh u6088085@10.34.110.219 — 80x...
[u6088085@cluster ~]$ mpicc -o paralellPrime paralellPrime.c
[u6088085@cluster ~]$ mpirun -np 4 paralellPrime 1000000
Total time: 0.009813 seconds to execute
78498 primes to 1000000
[u6088085@cluster ~]$
```

(b) Parallel

Figure 1: Compare between Sequential and Parallel

A terminal window titled 'Desktop — u6088232@cluster:~/Tutorail — ssh u6088232@10.34.110.219 —...' shows a series of commands and outputs. The user starts in a directory ~/Tutorail and runs 'cd Desktop/'. Then, they execute 'ssh u6088232@10.34.110.219', which prompts for a password. After login, the user runs 'cd Tutorail/' and then three MPI commands: 'mpirun -np 4 paralellPrime 100000', 'mpirun -np 8 paralellPrime 100000', and 'mpirun -np 16 paralellPrime 100000'. Each command outputs the total time in seconds and the number of primes found (9592). The times are approximately 0.001197s, 0.001003s, and 0.001570s respectively. The terminal text is as follows:

```
(base) Sirichokes-MacBook-Pro-2:~ Aommiizky$ cd Desktop/
(base) Sirichokes-MacBook-Pro-2:Desktop Aommiizky$ ssh u6088232@10.34.110.219
Password:
Last login: Thu Oct  3 14:07:12 2019 from 10.90.20.22
Rocks 6.2 (SideWinder)
Profile built 07:49 26-Aug-2018

Kickstarted 15:01 26-Aug-2018
[u6088232@cluster ~]$ cd Tutorail/
[u6088232@cluster Tutorail]$ mpirun -np 4 paralellPrime 100000
Total time: 0.001197 seconds to execute
9592 primes to 100000
[u6088232@cluster Tutorail]$ mpirun -np 8 paralellPrime 100000
Total time: 0.001003 seconds to execute
9592 primes to 100000
[u6088232@cluster Tutorail]$ mpirun -np 16 paralellPrime 100000
Total time: 0.001570 seconds to execute
9592 primes to 100000
[u6088232@cluster Tutorail]$
```

Figure 2: In the parallel version, compare between 4, 8 processes, and 16 processes.

According to the figure 2, from these 3 processes, which are 4, 8, 16 processes. It shows that if use more processes it will be faster. However, it depends on the internet because we connect to the Mahidol server.

4 Discussion about the results

The difficult part is writing code from sequential process to make it operate together or how to make each process independent since we split into blocks. Other outside blocks cannot know how far that process is running. Due to that, we need to use a broadcast command to find k value to other blocks that they can access the same value. Then, we need to MPI_reduce, to sum up, the results from all the blocks. Moreover, if the size of data is too large the runtime will be very slow.

5 Analysis of the algorithm

The algorithm that we used will balance the process and speed up the runtime. Moreover, it is very complicated to define which block belongs to what process.

6 Further optimization

Remove the even numbers that are not number 2 or not add to the block at all so it will be easier to search and limited the communication so runtime process will take shorter.

7 Responsibility for each member

- Sasirat Harnwatthanasiri 6088024
 - Analysis of the algorithm and do the slide
- Nattawat Lumtansawan 6088085
 - Find algorithm and debug the code
- Waris Vorathumdusadee 6088128
 - Find the way that can improve the code
- Pantita Wang 6088219
 - Find the information about MPI
- Sirichoke Yooyen 6088232
 - Writing code in sequential and parallel
- Phummarat Yosamornsuntorn 6088233
 - Compare the result between the sequential and parallel

Reference

- [1] mycourses.ict.mahidol.ac.th