# PDC2014 Assignment 2

Yuxiang Wu

## Algorithm description

A boolean array of length *n* is evenly distributed to *p* processors. The processors are arranged in linear array. All processors begin by initializing their own portion to *true*. Then the first processor is responsible for searching the first prime, then broadcast it to all the others. Once each processor receives this value, it begins to sieve their local portion with it.

## Experiment Settings

The program is written with CUDA C, which is a parallel computing platform developed by NVIDIA. The architecture of CUDA is similar with PRAM. The CPU and its memory are called “host”, and the GPU and its memory are called “device”. The experiment is run on a ThinkPad T420 notebook, with Intel i5 CPU (2.8GHz), 8GB RAM and NVIDIA Quadro NVS4200M GPU. The GPU has one Stream Multiprocessor with 48 CUDA Cores. GPU clock rate is 1.62 GHz and its global memory is 1GB.

## Results

The program takes 3 input arguments: n, p and c, where n is the size of array, p is number of processors, and c is communication cost. For n=100, 10000, 1000000, and c = 10, 20, ..., 100, I run the program on p = 1, 3, 5, ..., 31 processors. Each test case is run 20 times and the average runtime is recorded.

The following Table 1 and Figure 1 shows results of experiment for c = 10. There is little difference between the runtime of n=100 and n=10000, because in these cases, the program startup takes most of the time. The acceleration by increasing p is not obvious, because the runtime plateaus too quickly. But for n=1000000, the acceleration is obvious. With p=7, it achieved 1.6X speedup, but it plateaus out afterwards.

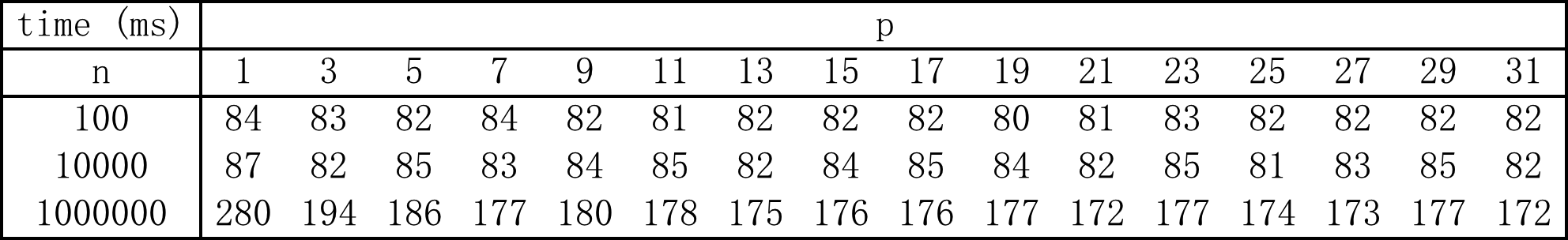


Table Average program runtime in milliseconds with c = 10

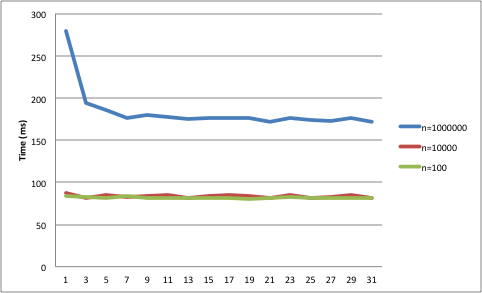


Figure Diagram of runtime for experiments with c = 10

To simulate the time for each inter-processor data transfer across the boundary of two adjacent processors in the array, each process will do c \* 100 \* idx local addition operations each time it receives message, where idx is the index of the process, idx = 1, 2, ..., p. The following table shows the experiment result for n=1000000.

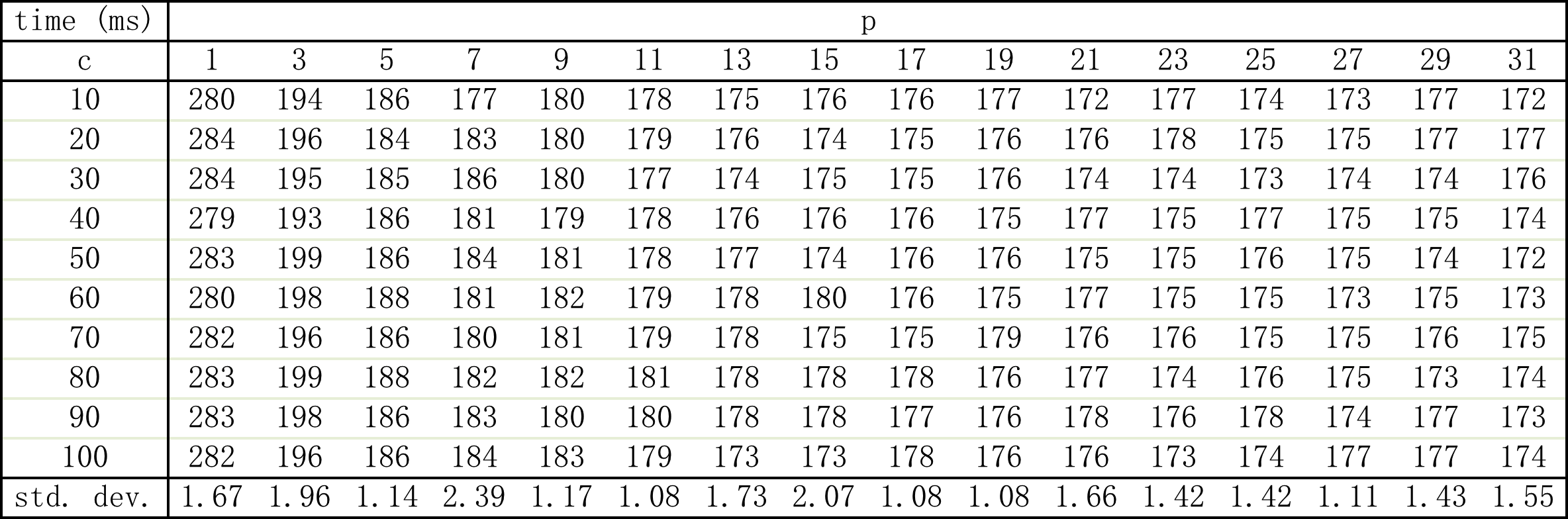


Table Runtime with different communication cost and number of processors, with n=1000000

The last row shows the standard derivations of the runtimes, which is relatively small compared to the runtimes. This indicates that the runtime is insensitive to communication cost. One reason is that cross boundary communication happens when broadcasting the prime number. Data size of prime is small and it’s quick to broadcast. Another reason is that local addition operation is relatively fast and takes a relatively small portion of time to complete.

## Extra Experiments

I did an extra experiment for n = 10000000, to see how much speedup it can gain with a large input size. Result is shown as follows.

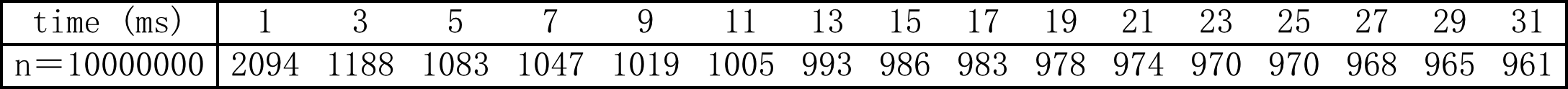


Table Runtime for n=10000000

Figure Diagram for experiment with n=10000000

It managed to achieve 2X speedup with p=7 and then runtime reaches a plateau. Comparing to the results when n=1000000, the speedup is more significant.