PARALLEL COMPUTING PROJECT

SIEVE OF ERATOSTHENES

Instructor

Dr. Manish Kumar Bajpai

Team Members

Murarishetty Santhosh Charan (2019287)

Repala Nagaraju (2019127)

Eraganaboina Venkata Srujan (2019250)

Veerisetty Subramanyam(2019172)

Introduction

The aim of this project is to parallelize sieve of Eratosthenes algorithm. The Sieve of Eratosthenes is an ancient mathematical algorithm of finding prime numbers between two sets of numbers. It is an almost mechanical procedure for separating out composite numbers and leaving the primes. It was invented by the Greek scientist and mathematician Eratosthenes who lived approximately 2,300 years ago.

The algorithm makes work easier by eliminating complex looping divisions or multiplications.

For a given upper limit n the algorithm works by iteratively marking the multiples of primes as composite, starting from 2. Once all multiples of 2 have been marked composite, the muliples of next prime, ie 3 are marked composite.

This process continues until $p \leqslant \sqrt{n}$, where p is a prime number.

The advantage of sieving is that any computing effort is replaced by counting.

Serial Implementation

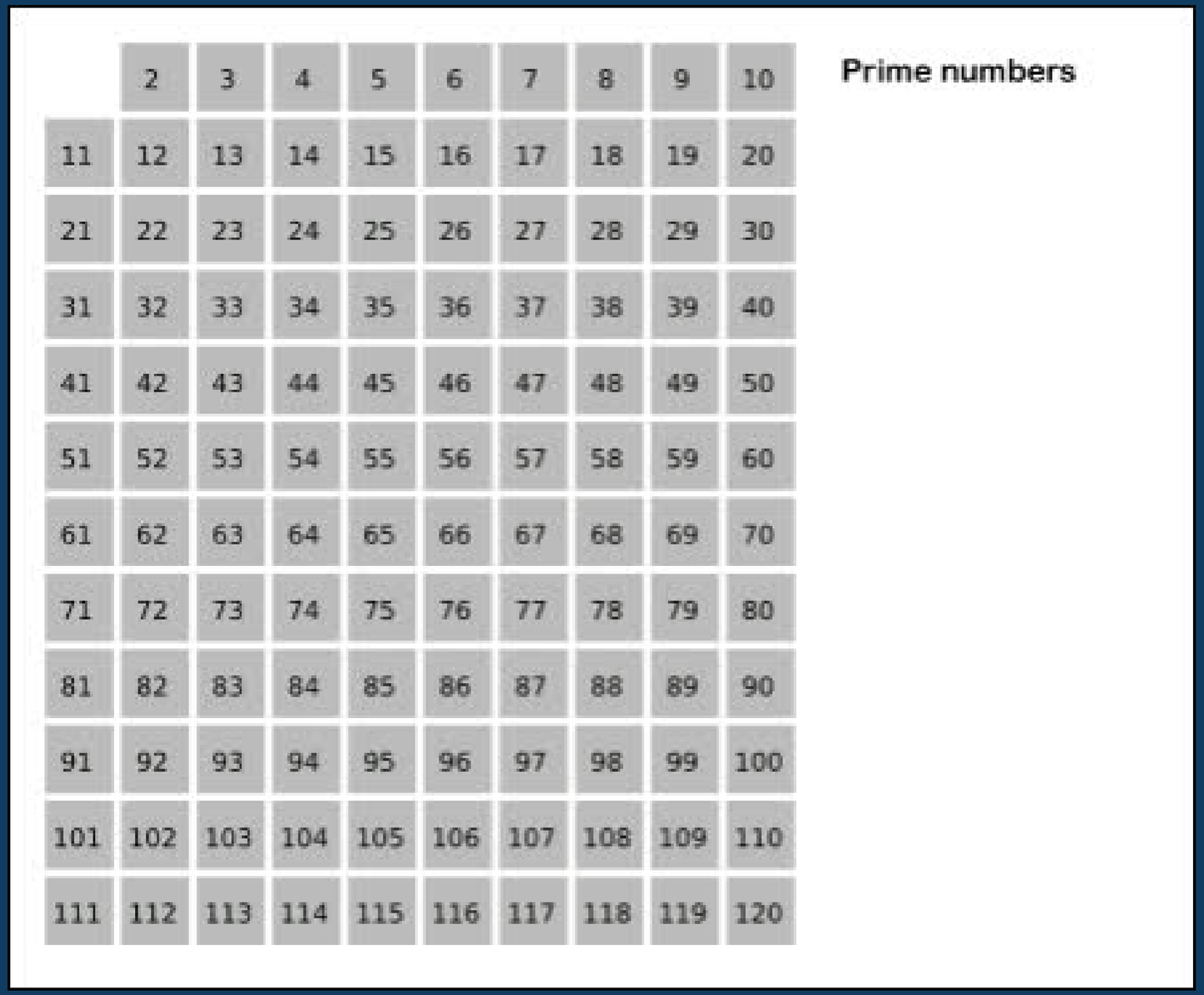
This Algorithm consists of going through a table of numbers using the following steps :

- List all consecutive numbers from 2 to N, i.e. (2, 3, 4, 5, ……., N) .
- Assign the first prime number to a variable p .
- Beginning with p*p, perform an incremental of p and mark the integers equal or greater than p*p in the algorithm. And those integers will be p*(p + 1), p*(p + 2), p*(p + 3), p*(p + 4) ……

- The first unmarked number greater than p is identified from the list. If the number does not exist in the list, the procedure is halted. p is equated to the number and the above step is repeated.

- The Sieve of Eratosthenes is stopped when the square of the number being tested exceeds the last number on the list.Assign the first prime number to a variable p .

- All numbers in the list left unmarked when the algorithm ends are referred to as prime numbers.



Working of the Sieve of Eratosthenes

Parallel Implementation

- In parallel processing, rather than having a single program execute tasks in a sequence (like the tasks of the algorithm above), parts of the program are instead split such that the program is executed concurrently (i.e. at the same time), by multiple entities. The entities that execute the program can be called either threads or processes depending on how memory is mapped to them.
- In shared memory parallelism, threads share a memory space among them. Threads will be able to read and write to and from the memory of other threads.

Steps involved in Parallel Implementation

- If a primitive task represents each integer, then two communication are needed to perform the repeat part each iteration of the repeat, until loop.
- Reduction needed each iteration in order to determine the new value of k.
- Then we broadcast to inform all the tasks of the new value of k.
- This will take many reduction and broadcast operations.
- New version of the parallel algorithm that requires less computation and less communication than original parallel algorithm.

Complexity Analysis

In Serial Implementation:

- Time Complexity : O(N*log(log(N))
- Space Complexity : O(N)

In Parallel Implementation:

- Time Complexity : O(N)
- Space Complexity : O(N)

Advantages

SPEEDUP

The advantage of speedup is that it allows a problem to be solved faster. If multiple processes or threads are able to work at the same time

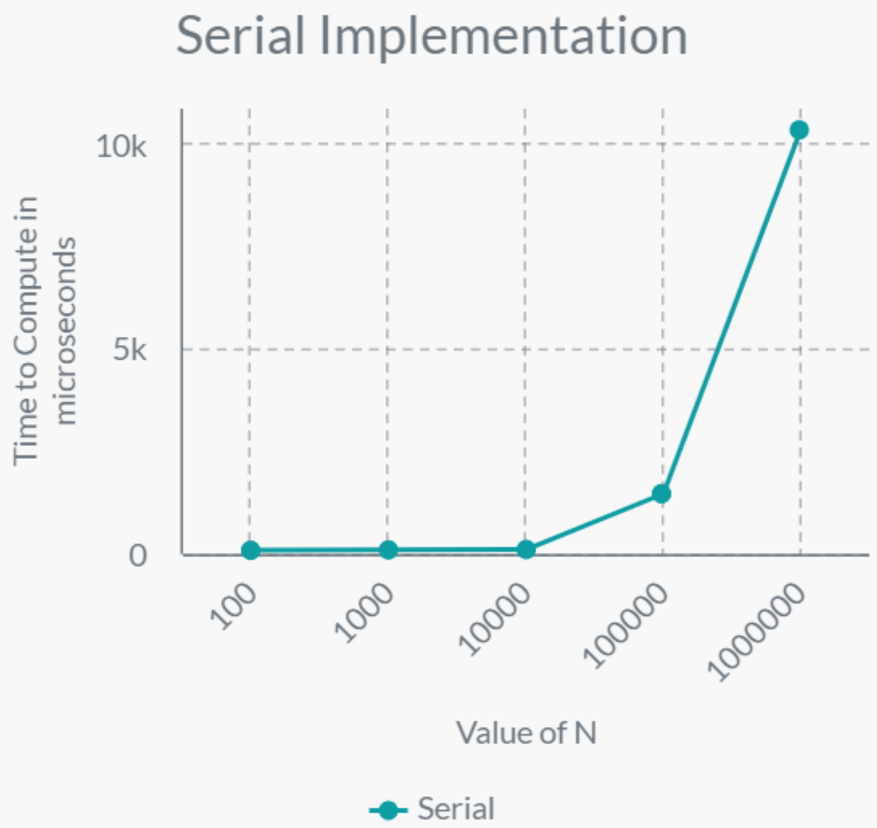
ACCURACY

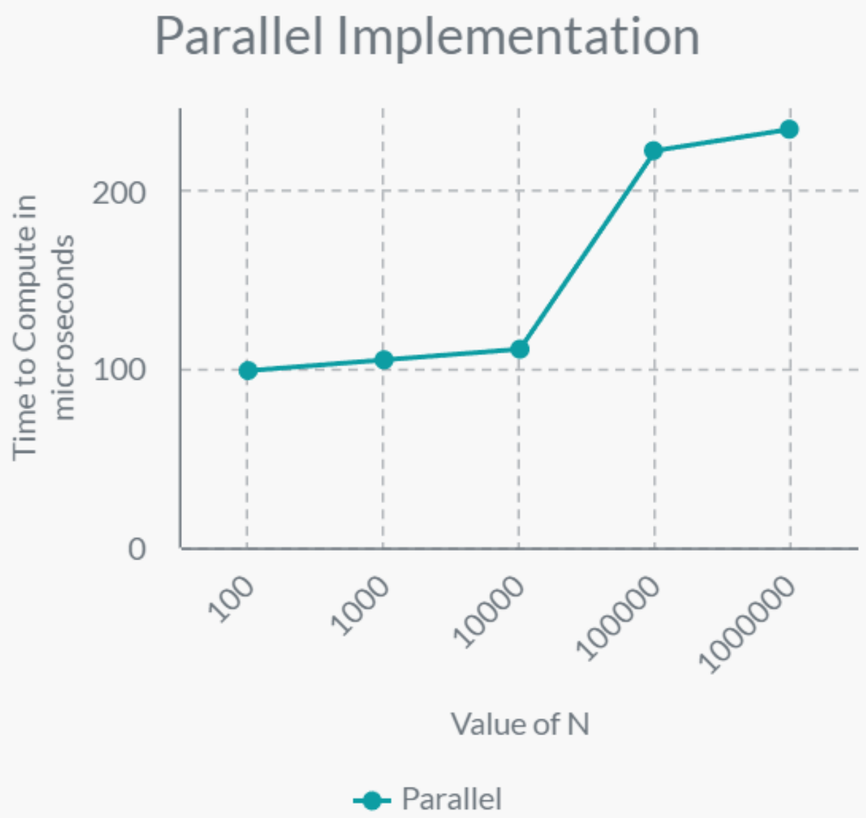
Accuracy is the idea of forming a better model of a problem. If more processes or threads are assigned to a task, they can spend more time doing error checks or other forms of diagnostics to ensure that the final result is a better

WEAK SCALING

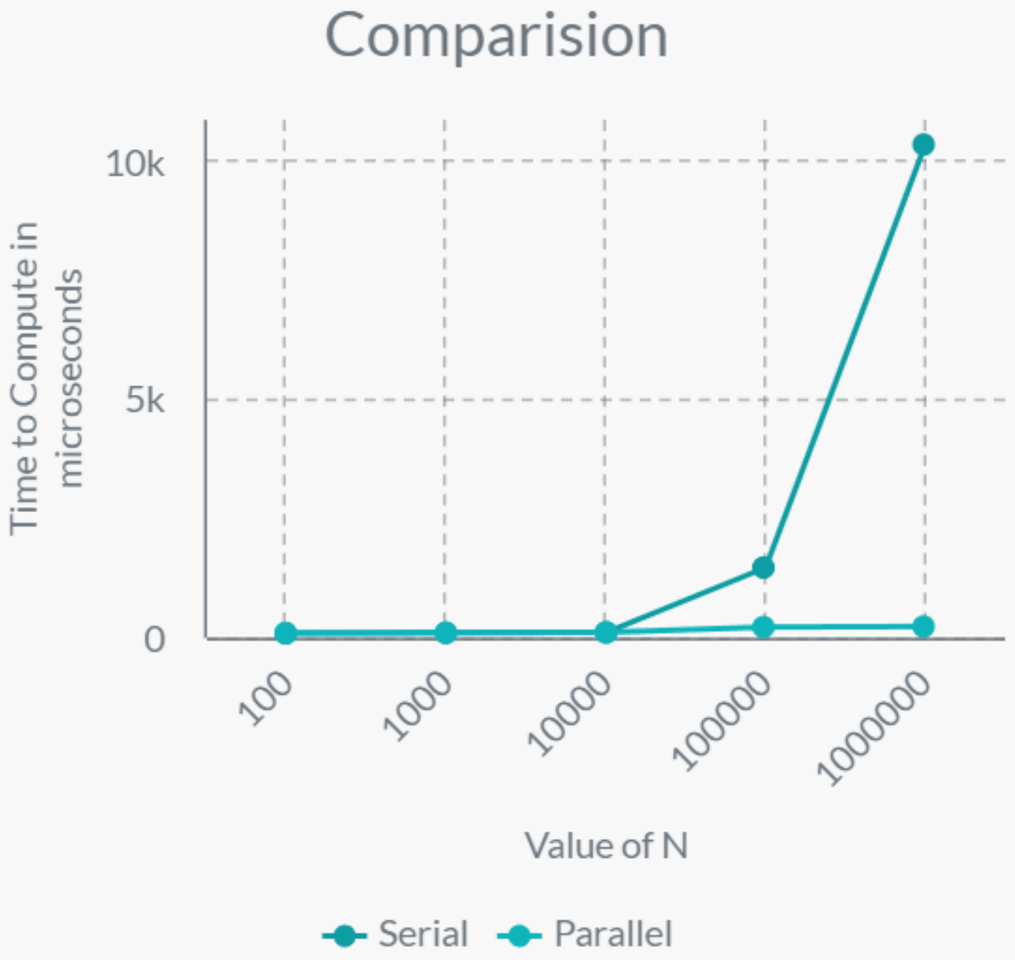
Weak scaling says that more processes and threads can be used to solve a bigger problem in the same amount of time it would take fewer processes and threads to solve a smaller problem.

Graphs


Serial Implementation
Time to Compute in microseconds
10k
5k
0
100
1000
10000
100000
1000000
Value of N
Serial


Parallel Implementation
Time to Compute in microseconds
200
100
0
100
1000
10000
100000
1000000
Value of N
Parallel

Comparision



Results

Time Taken by Serial and Parallel Implementations

Value of N / Time in microseconds	Serial	Parallel
100	88	99
1000	100	105
10000	110	111
100000	1467	222
1000000	10319	234

Conclusion

The sequential algorithm developed has taken considerably large amount of time to generate the prime numbers. But when same task is taken up with parallel approach we could achieve better performance as far as speedup is considered. Here we used data parallel approach to Sieve of Eratosthenes and compared it with sequential algorithm. The analysis and computational results obtained show that efficient usage of resources to contribute towards performance of parallel computing system. The computational results obtained here show that how GPU is supportive in parallel computation and in turn contributes to speed up in the parallel computing system.


Thank you