SIEVE PARALLEL ALGORITHM

CSE708 – Shivangi Mishra

Professor - Russ Miller

University at Buffalo The State University of New York

CONTENT

Sequential Algorithm

```
def FindPrime(n):
    prime = [True for i in range(n+1)]
    for i in range(2,n+1):
        for j in range(2,i):
            if i%j==0:
                prime[i]=False
                break
        prime[i] = True
```

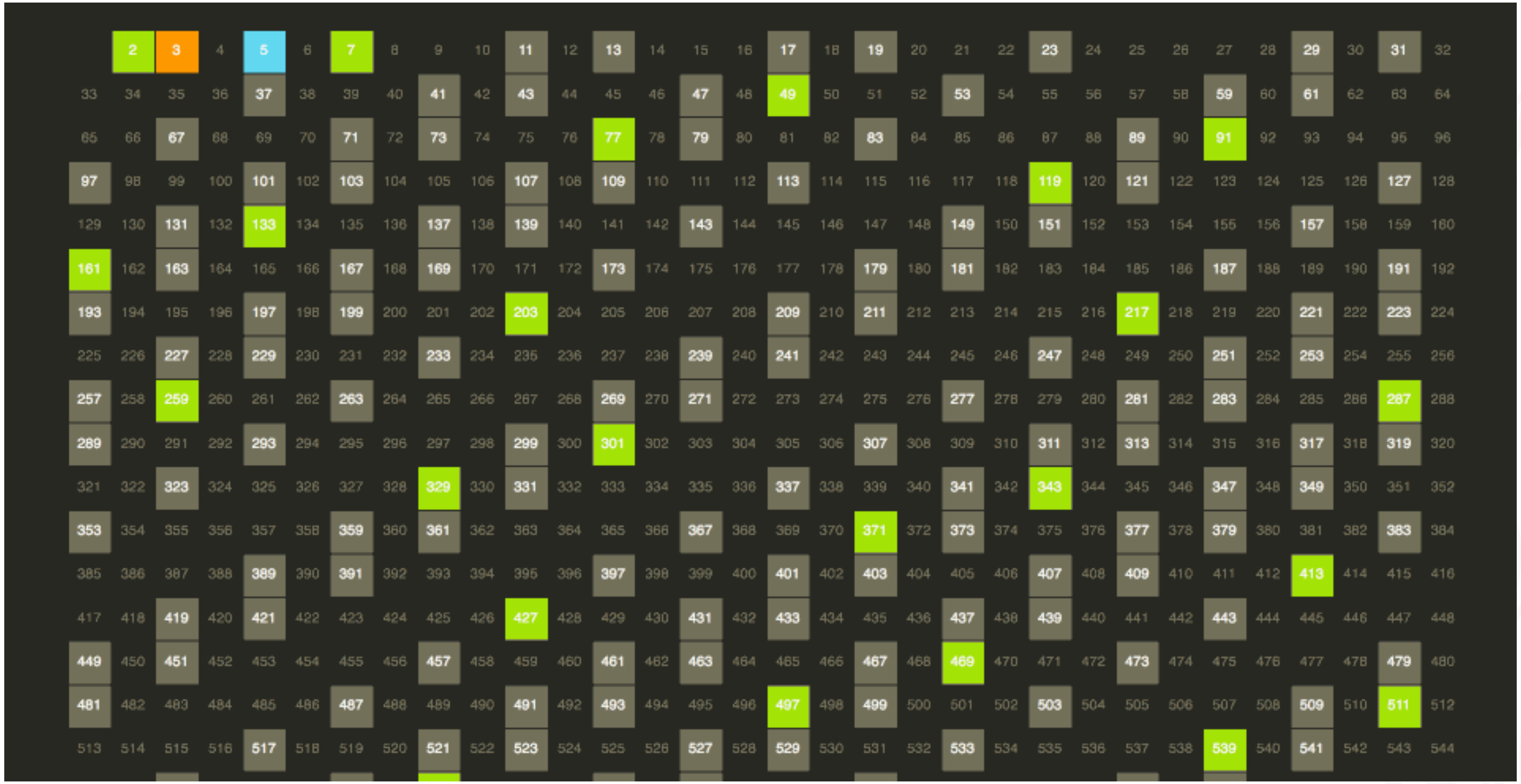
Time complexity: O(n^2)

- The prime number is a positive integer greater than 1 that has exactly two factors, 1 and the number itself.
  First few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23
- Except for 2, which is the smallest prime number and the only even prime number, all prime numbers are odd numbers.
- Every prime number can be represented in form of **6n + 1** or **6n – 1** except the prime numbers **2** and **3**, where n is any natural number.

Sieve of Eratosthenes

- The Sieve of Eratosthenes is a method used to find prime numbers.
- Prime numbers are important in modern encryption algorithms like sha256 that keep our digital transactions safe.
- Public-key cryptography also uses prime numbers to create specialized keys.
- The Sieve is also used in mathematics, abstract algebra, and elementary geometry to study shapes that reflect prime numbers.
- Biologists use the Sieve to model population growth, and composers use prime numbers to create metrical music.
- Olivier Messiaen, a French composer, used prime numbers to create unique rhythms in his music pieces.

Sieve Simulation



Sequential Sieve Algorithm

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

```
find primes up to N
For all numbers a : from 2 to sqrt(n)
     IF a is unmarked THEN
         a is prime
         For all multiples of a (a < n)
             mark multiples of as composite
All unmarked nummbers are prime!
```

Pseudo code

Time complexity: O(n*log(log(n)))

Parallel Sieve Implementation

- Split the array of length n between threads p each of size n/p.
- Utilize the #pragma omp parallel for directive to concurrently set the 'prime' array as 'True.' This directive distributes the workload among multiple threads for each array segment, enhancing efficiency.
- Simultaneously, multiple threads are employed to eliminate the non-prime multiples within the range of 2 to the square root of 'n.' This parallelization accelerates the process of finding prime numbers.
- Upon identifying the prime numbers in each thread, tally the count of primes from every thread and consolidate the results using #pragma omp parallel for reduction(+:primeCount).
- The master thread is responsible for displaying the final outcome, streamlining the presentation of prime numbers found through parallel computation.

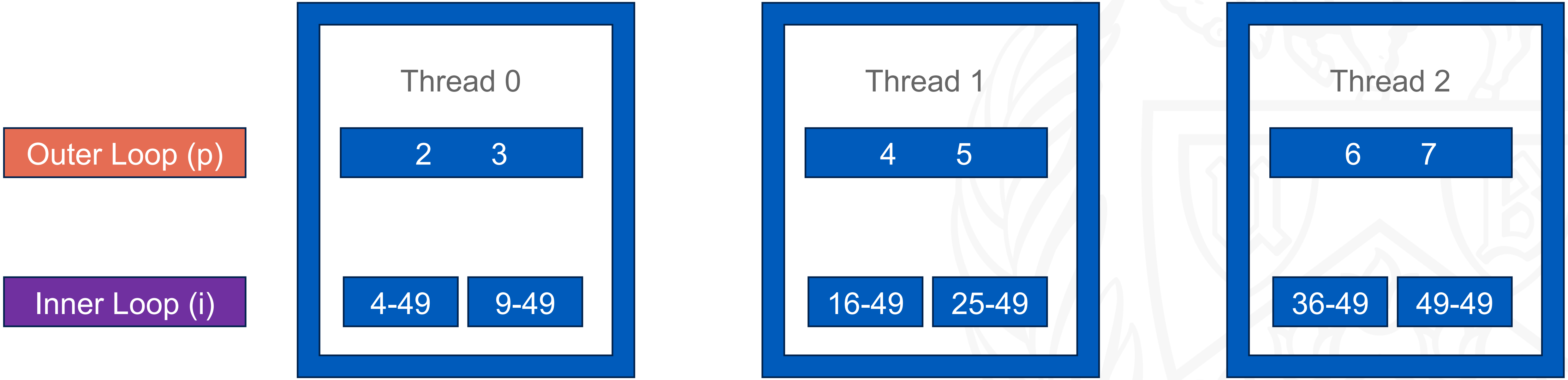
Marking array as True

Thread 1	**0**	**1**	**2**	**3**
Thread 2	4	5	6	7
Thread 3	8	9	10	11
Thread 4	12	13	14	15

Cancelling Out Multiples

```
#pragma omp parallel for
for (int p = 2; p <= sqrt_n; p++) {
    int thread_id = omp_get_thread_num();
    printf("\nFor thread_id %d, p = %d\n", thread_id, p);
    if (prime[p]) {
        #pragma omp parallel for
        for (int i = p * p; i <= n; i += p) {
            printf("\n For thread_id %d, p = %d, i = %d\n", thread_id, p, i);

            prime[i] = false;
        }
    }
}
```

Thread Distribution



Thread Distribution output

sieve_openmp.c | output.txt

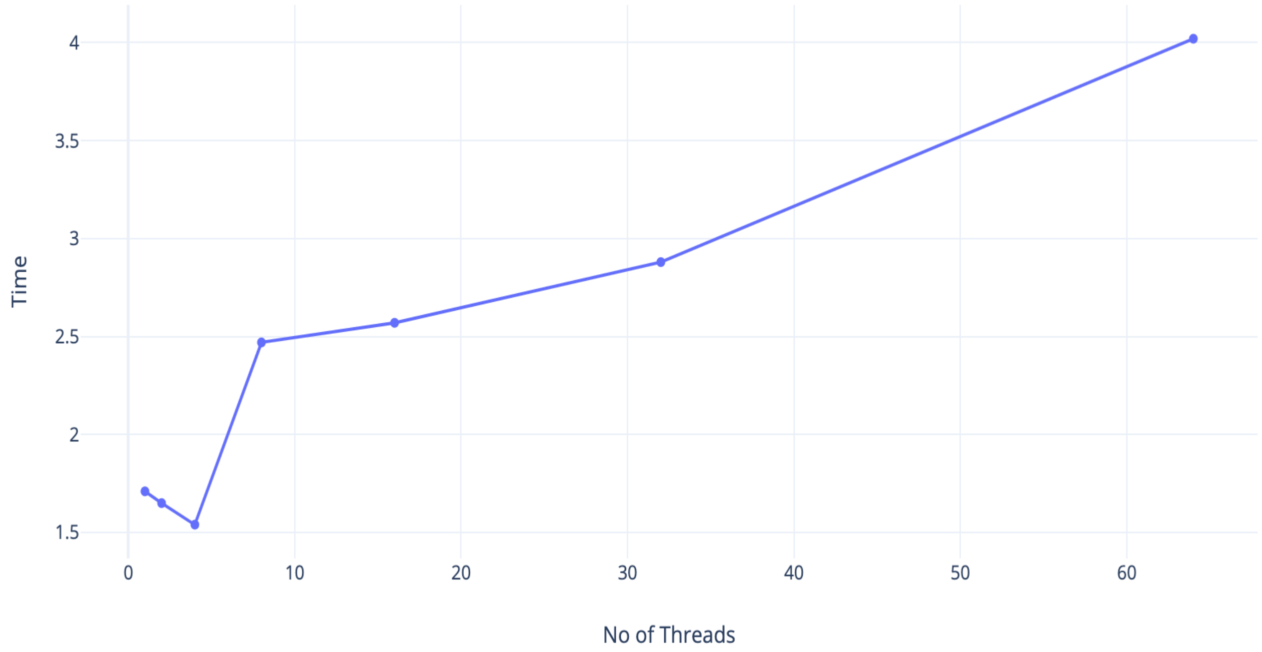
output.txt

```
Prime numbers up to 49 are:
 For thread_id 1, p = 5, i = 25

 For thread_id 1, p = 5, i = 30

 For thread_id 1, p = 5, i = 35

 For thread_id 1, p = 5, i = 40

 For thread_id 1, p = 5, i = 45

Total prime count: 15
Work took 0.000793 seconds
```

Result parallel

Input size: 10^8

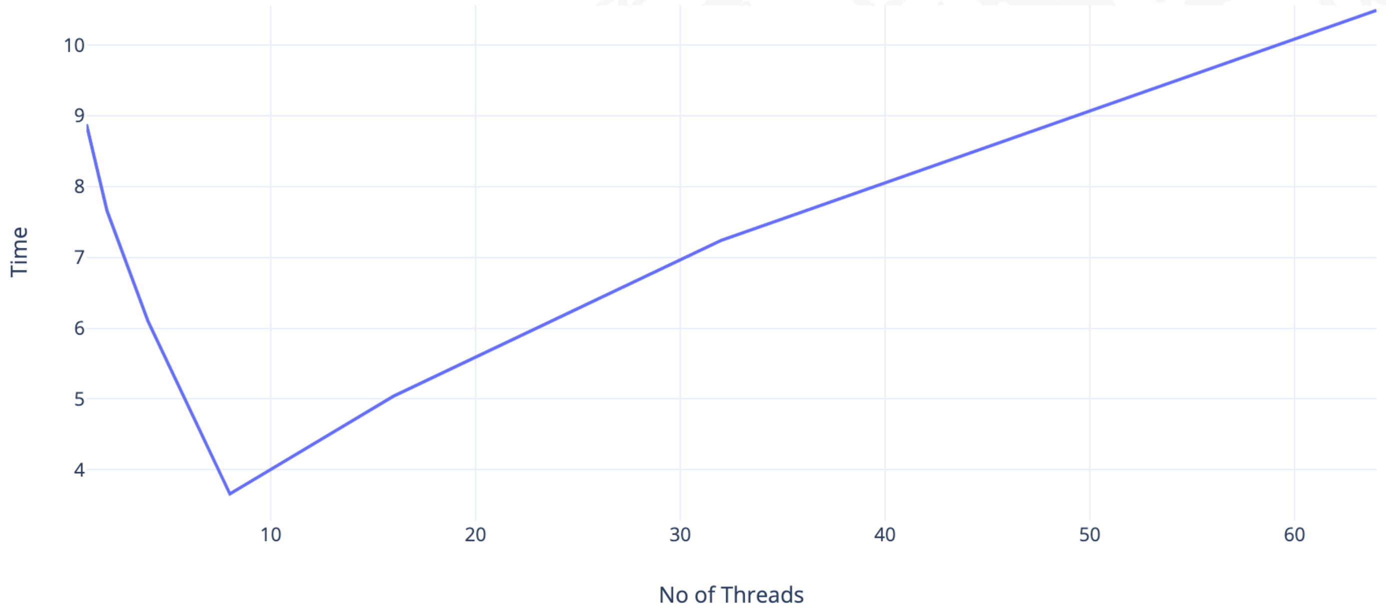
Threads	Time in sec
1	1.71
2	1.65
4	1.54
8	2.47
16	2.57
32	2.88
64	4.02



Result parallel

Input size: 10^10

Threads	Time in sec
1	8.88
2	7.66
4	6.10
8	3.66
16	5.04
32	7.24
64	10.49

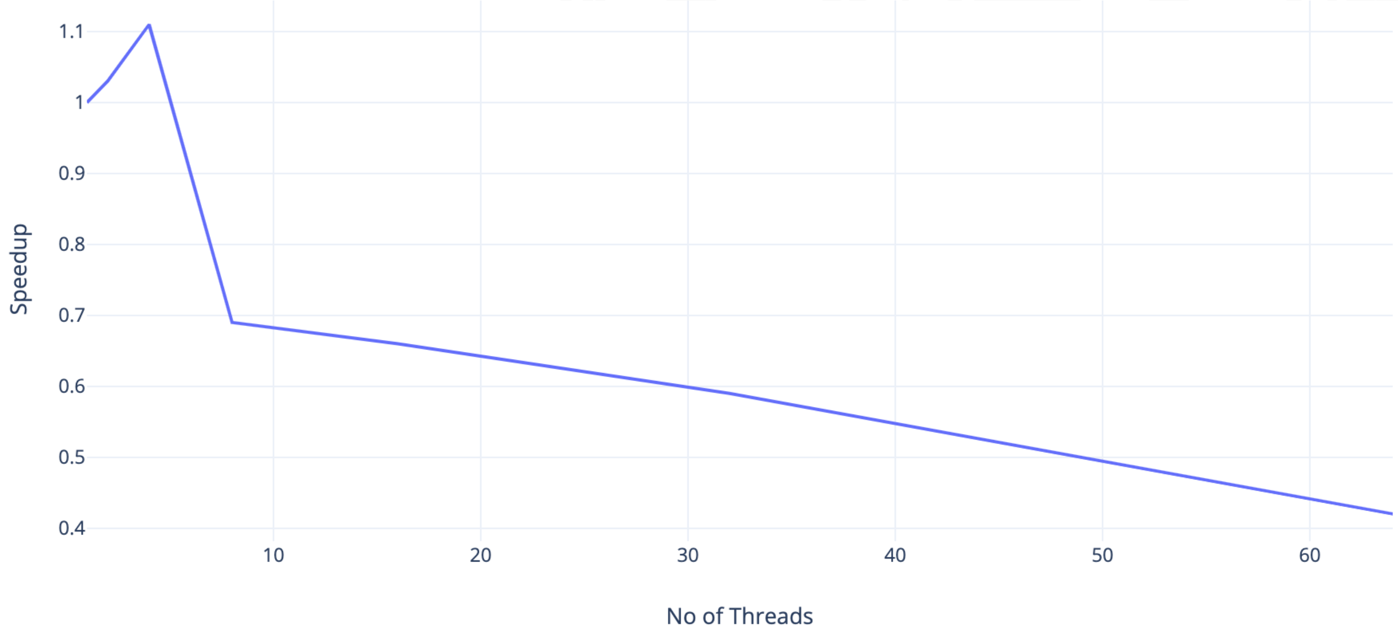


Speed-Up

Input size: 10^8

$\text{SpeedUp} = \frac{T_{seq}}{T_{parallel}}$ T_{seq} = 1.71 seconds.

Threads	Speedup
1	1
2	1.03
4	1.11
8	0.69
16	0.66
32	0.59
64	0.42



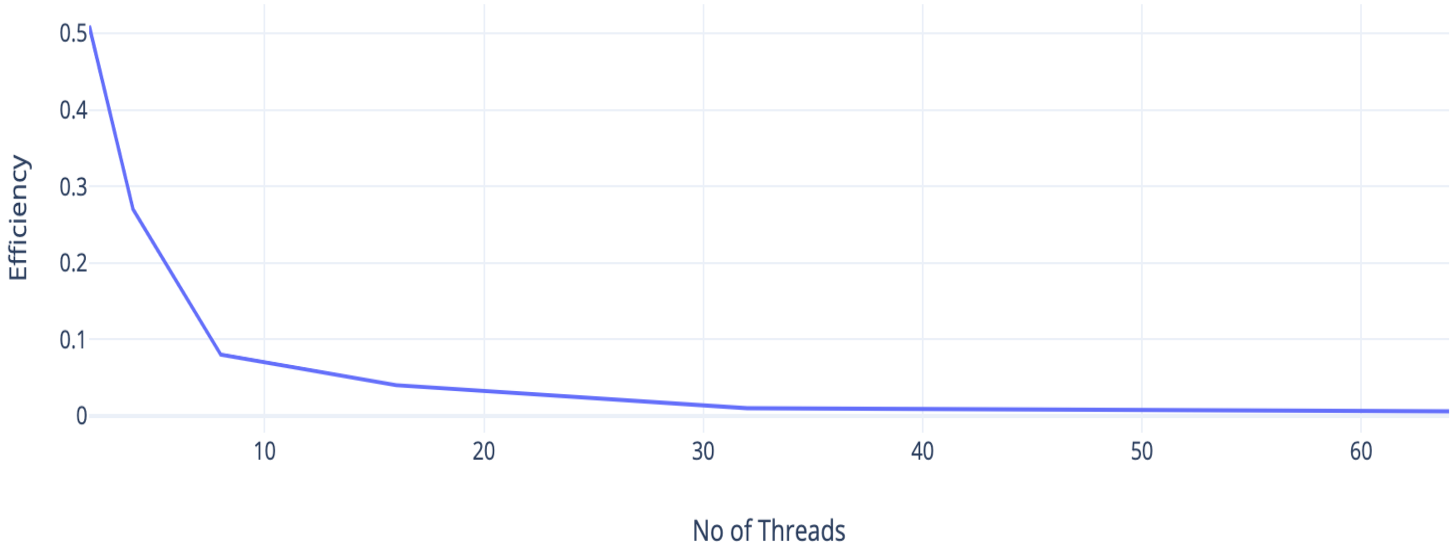
Efficiency

Input size: 10^8

Efficiency = $\frac{T_{seq}}{Cost}$

T_{seq} = 1.71 seconds.

Threads	Time	Cost	Efficiency
2	1.65	3.3	0.51
4	1.54	6.16	0.27
8	2.47	19.76	0.08
16	2.57	41.12	0.04
32	2.88	92.16	0.01
64	4.02	257.28	0.006

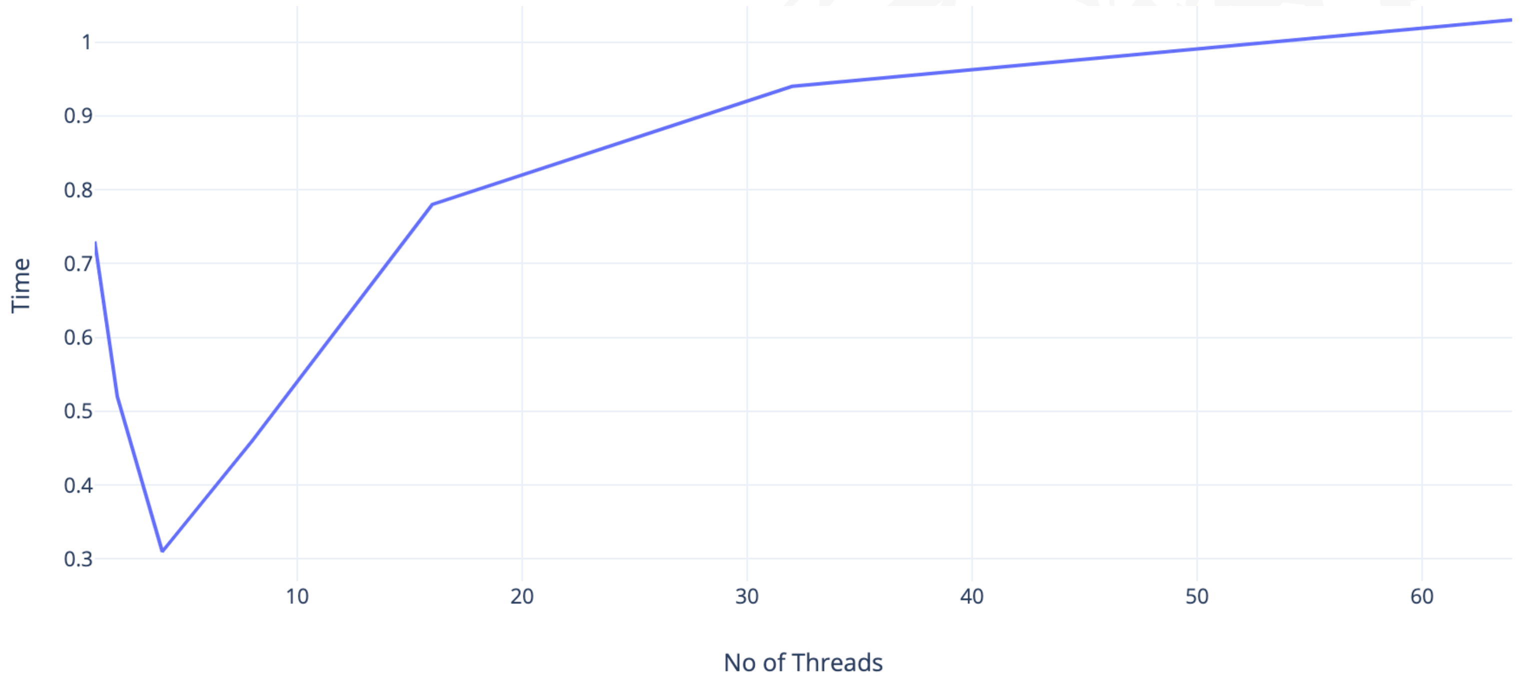


Scaled Result(Gustafson's law)

Input size: 10^6

Threads	Time	Input size
1	0.73	10^6
2	0.52	2*10^6
4	0.31	4*10^6
8	0.46	8*10^6
16	0.78	16*10^6
32	0.94	32*10^6
64	1.03	64*10^6

Data per thread = 10^6



Slurm Script

OpenMP configuration
1 node, 1 task & 64 cores requested

Specifying the total number of threads running for the parallel execution

$ sieve_openm.slurm ×

openmp_sieve > $ sieve_openm.slurm

```
#!/bin/bash

#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=64
#SBATCH --partition=general-compute
#SBATCH --qos=general-compute
#SBATCH --cluster=ub-hpc
#SBATCH --reservation=ubhpc-future
#SBATCH --exclusive
#SBATCH --cluster=ub-hpc
#SBATCH --time=01:00:00
#SBATCH --output=output_openmp.txt
#SBATCH --job-name="sieve"

## speedup runs


module load intel

gcc -fopenmp -o sieve_compiled sieve_openmp.c -lm

for nt in 1 2 4 8 16 32 64; do
export OMP_NUM_THREADS=$nt
./sieve_compiled
done
```

Script Execution

As requested by the slurm script, we got a single node “cpn-i14-06” which has 64 cores.

commands used:
squeue –u $USER - to see the list of active nodes
snodes cpn-i14-06 - to see the node configuration

14464496	sieve	mishra22	cse708f23	00:00:03	general-compute	Completed	UB-HPC Cluster

Completed **sieve 14464496**

Cluster	UB-HPC Cluster
Job Id	14464496
Job Name	sieve
User	mishra22
Account	cse708f23
Partition	general-compute
State	COMPLETED
Reason	None
Total Nodes	1
Node List	cpn-i14-06
Total CPUs	64
Time Limit	1:00:00
Time Used	0:03
Start Time	2023-12-11 19:22:18
End Time	2023-12-11 19:22:21
Memory	2800M

```
[mishra22@vortex1:~/Desktop/openmp_sieve]$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          14464496 general-c    sieve mishra22  R       0:02      1 cpn-i14-06
          1446448[illegible] scavenger ood-vsco mishra22  R       9:23      1 cpn-q07-18
[mishra22@vortex1:~/Desktop/openmp_sieve]$ snodes cpn-i14-06
HOSTNAMES     STATE     CPUS S:C:T    CPUS(A/I/O/T)   CPU_LOAD MEMORY    GRES                                    PARTITION          AVAIL_FEATURES
cpn-i14-06    resv      64   2:32:1   0/64/0/64       0.00     512000    (null)                                  general-compute*   AVX512,CPU-Gold-6448Y,INTEL,i14
,FUTURE
cpn-i14-06    resv      64   2:32:1   0/64/0/64       0.00     512000    (null)                                  scavenger          AVX512,CPU-Gold-6448Y,INTEL,i14
,FUTURE
[mishra22@vortex1:~/Desktop/openmp_sieve]$
```

References

- GFG
- OpenMP Slides

Thank You
Questions ?