Design and Performance Report

On

# Parallel Sieve of Eratosthenes

by


| | |
|---|---|
| SHIVANKIT GAIND | 2015A7PS0076P |
| ROHIT GHIVDONDE | 2015A7PS0077P |


For the partial fulfilment of the
course on

## Parallel Computing

Submitted to
Prof. Shan Sundar Balasubramaniam



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**
**February, 2018**

# Design

1. The task for finding all prime numbers less than N is divided among processes.

2. The design involves using each process to apply the sieve of eratosthenes algorithm independently on the range of values assigned to it.

3. The range of numbers from 1 to N is divided into two ranges:
    a. 1 to √N (range 1)
    b. (√N + 1) to N (range 2)

4. Each process is allotted two range of numbers:
    a. Range 1
    b. (Range 2)/(Number of processes)

5. The reasons behind dividing the ranges this way:
    a. All the non primes till number N can be sieved by using numbers between 1 to √N . Therefore each process uses this range independently to sieve the rest of the numbers.
    b. Remaining numbers need to be divided equally among all the processes to balance the work done by each process.

6. Each process traverses range 1 and uses the first unmarked number (a prime number) to sieve the rest of the numbers to the right of it in range 1 and all the numbers in range 2.

7. Step 5 continues till the end of range 1 (i.e √N) is reached.

8. When the sieving process ends, all the unmarked numbers are primes and all the marked numbers are non primes.

# Performance Evaluation

**A.** The performance measurements for different values of p (where p is the number of nodes used) at every value of N from 10^5 to 10^10 is given below:

### ⇒ If only one core per node was used (means p = 2 corresponds to 2 processes):

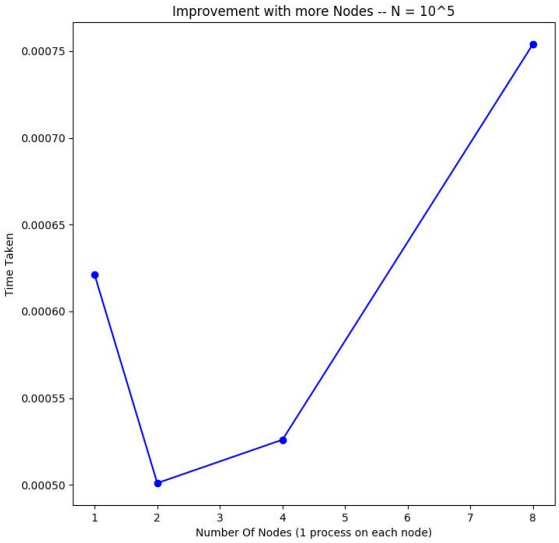
| p/N | 10^5 | 10^6 | 10^7 | 10^8 | 10^9 | 10^10 |
|---|---|---|---|---|---|---|
| P = 1 | 0.000621 | 0.007040 | 0.077178 | 0.961129 | 10.545066 | 125.850458 |
| P = 2 | 0.000501 | 0.003405 | 0.037459 | 0.493092 | 5.346813 | 59.399659 |
| P = 4 | 0.000526 | 0.002003 | 0.018236 | 0.225241 | 2.650449 | 28.195559 |
| P = 8 | 0.000754 | 0.001277 | 0.010046 | 0.111235 | 1.375004 | 14.554182 |

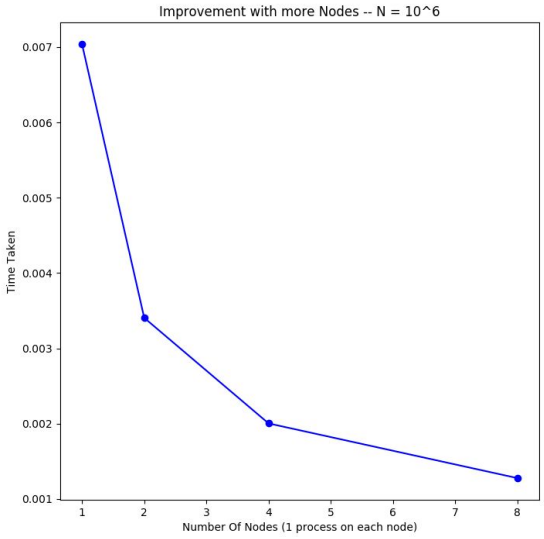
### ⇒ If all 4 cores per node were used (means p = 2 corresponds to 8 processes):

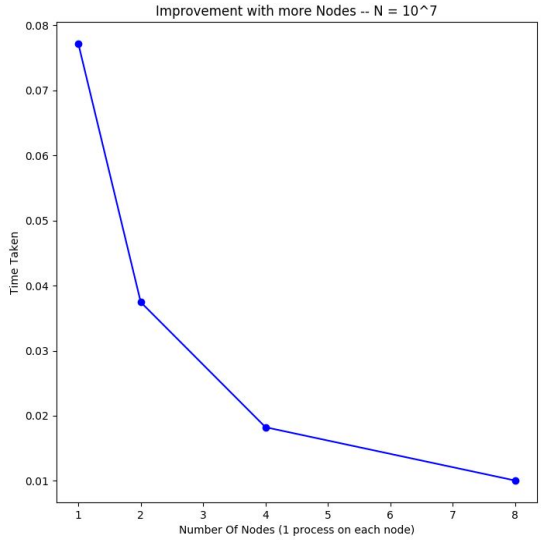
| p/N | 10^6 | 10^7 | 10^7 | 10^8 | 10^9 | 10^10 |
|---|---|---|---|---|---|---|
| P = 1 | 0.00020504 | 0.00182884 | 0.019794 | 0.407233 | 4.66749 | 54.191013 |
| P = 2 | 0.000432014 | 0.00111485 | 0.018520 | 0.195226 | 2.35306 | 27.704572 |
| P = 4 | 0.000618935 | 0.000979 | 0.00494409 | 0.0858421 | 1.19708 | 13.550571 |
| P = 8 | 0.000918 | 0.001006 | 0.002478 | 0.041884 | 0.576007 | 6.618218 |

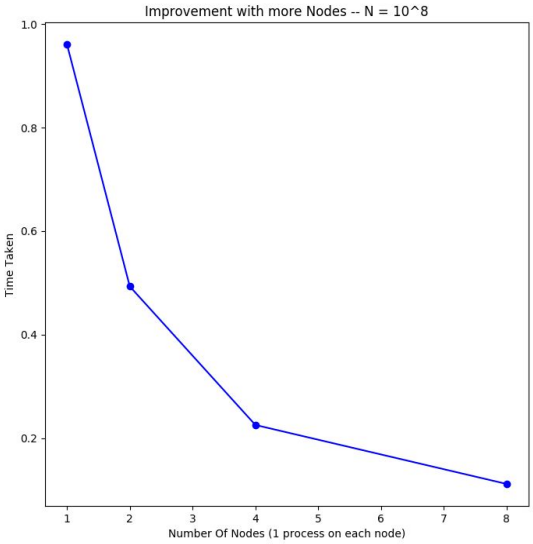
The plots showing how performance improves with increase in the number of nodes at each value of N are given below:

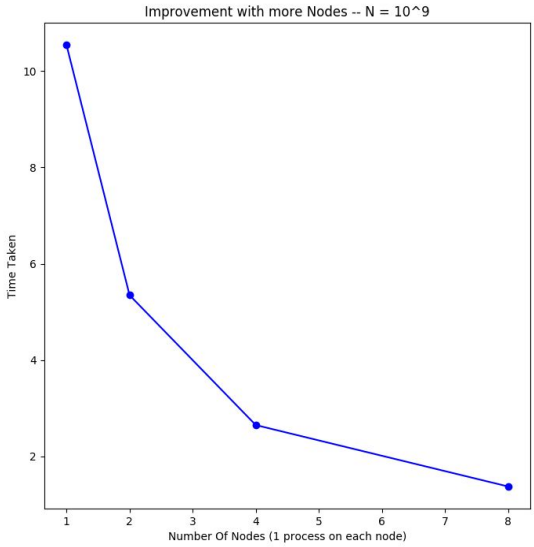
### ⇒ If only one core per node was used:

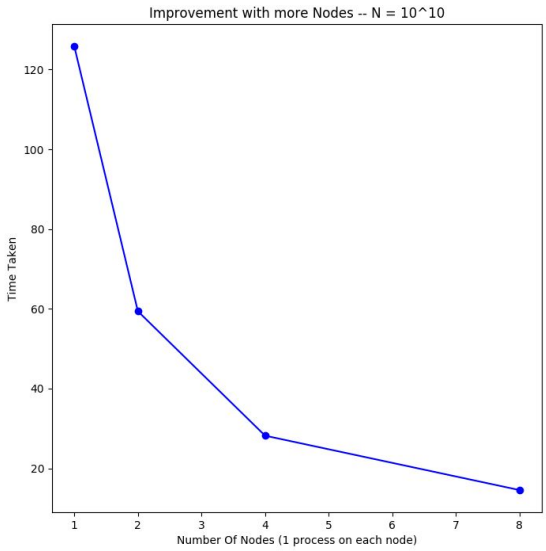

Improvement with more Nodes -- N = 10^5
Time Taken
Number Of Nodes (1 process on each node)
0.00075
0.00070
0.00065
0.00060
0.00055
0.00050
1
2
3
4
5
6
7
8


Improvement with more Nodes -- N = 10^6
Time Taken
Number Of Nodes (1 process on each node)
0.007
0.006
0.005
0.004
0.003
0.002
0.001
1
2
3
4
5
6
7
8

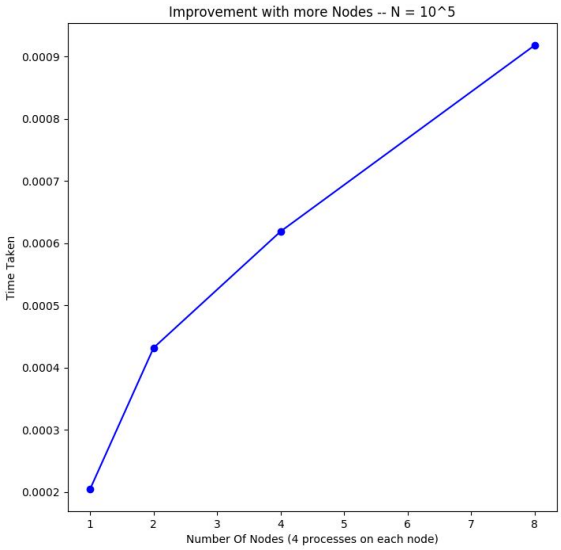

Improvement with more Nodes -- N = 10^7
Time Taken
Number Of Nodes (1 process on each node)
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
1
2
3
4
5
6
7
8

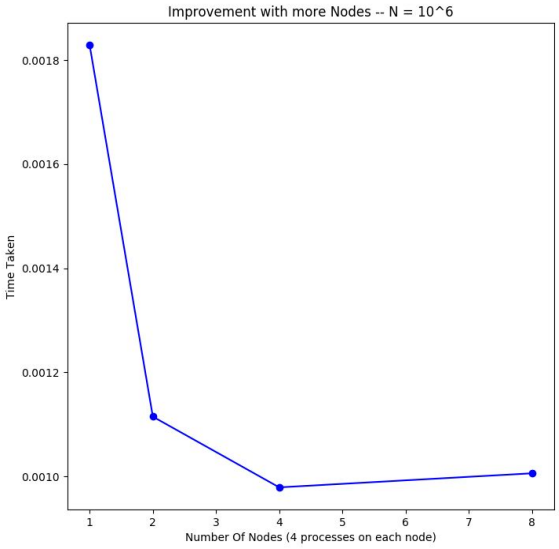

Improvement with more Nodes -- N = 10^8
Time Taken
Number Of Nodes (1 process on each node)
1.0
0.8
0.6
0.4
0.2
1
2
3
4
5
6
7
8

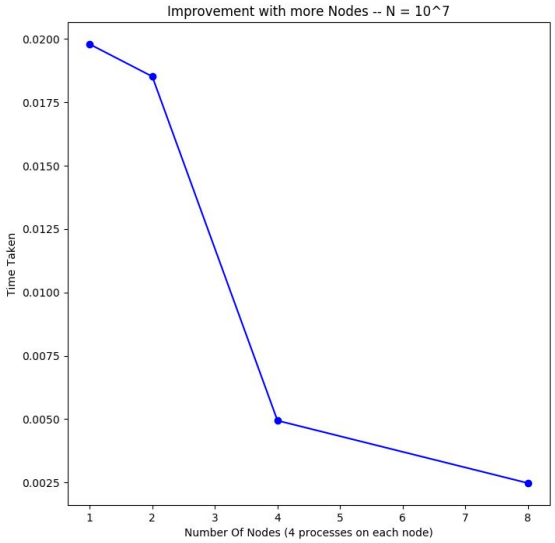


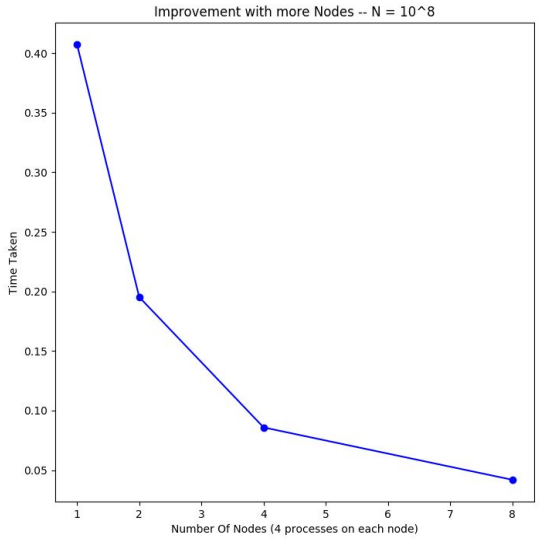



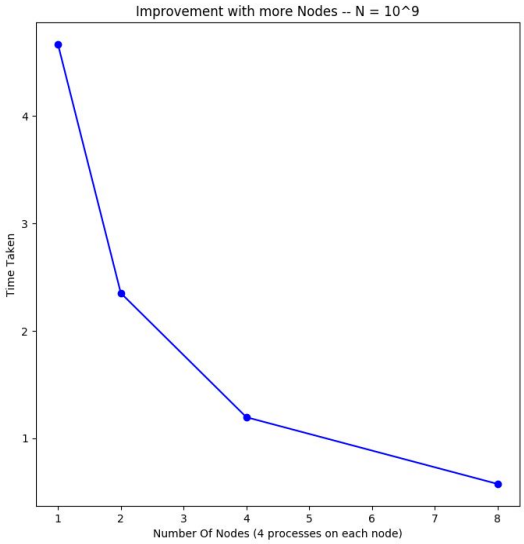

## ⇒ If all 4 cores per node were used:

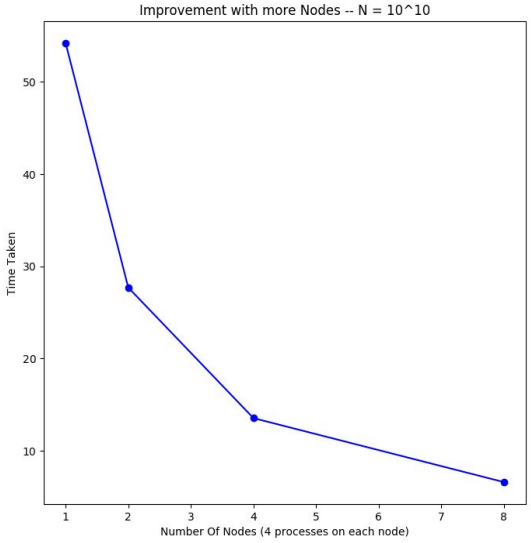




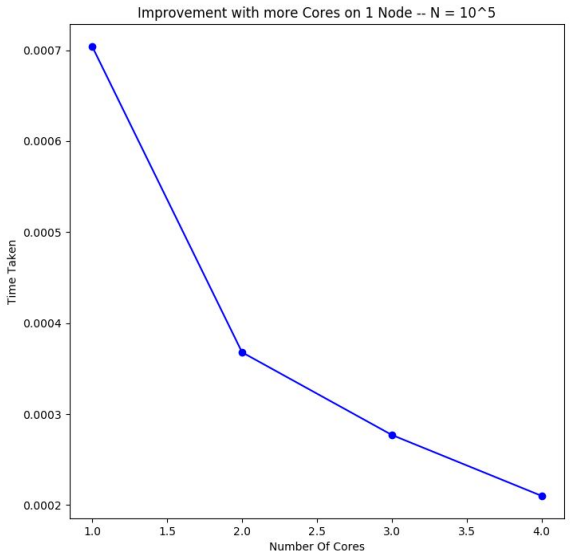



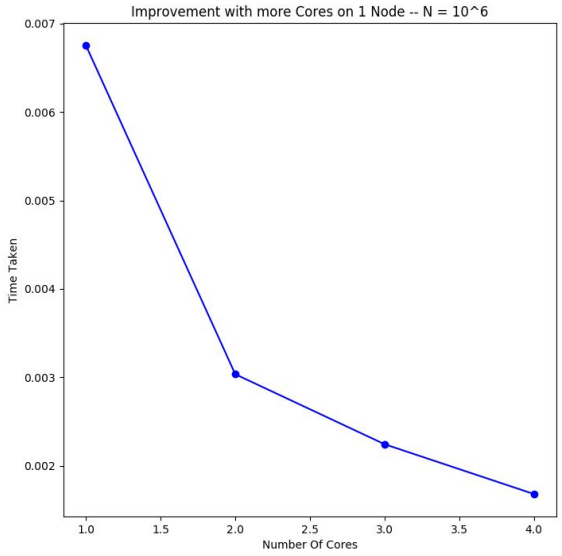

A different behaviour is observed in case of 10^5 (and 10^6 also to some extent) because the overhead of distributing the tasks among processes is more than the work done per process.

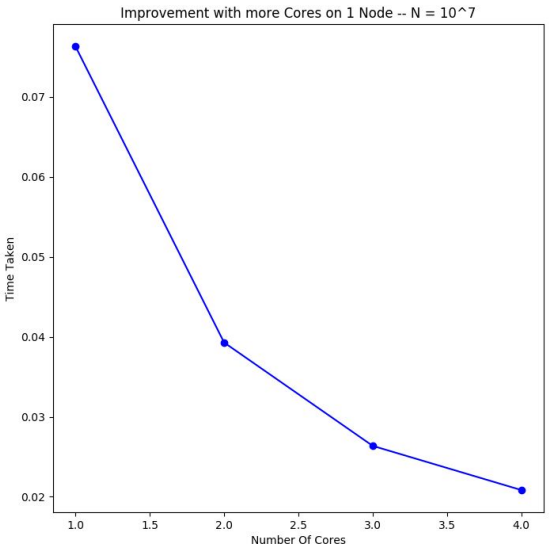
**B. The Performances also improved with increase in the number of cores used per node (from 1 to 4) when we tested the algorithm on one node for different values of N.**

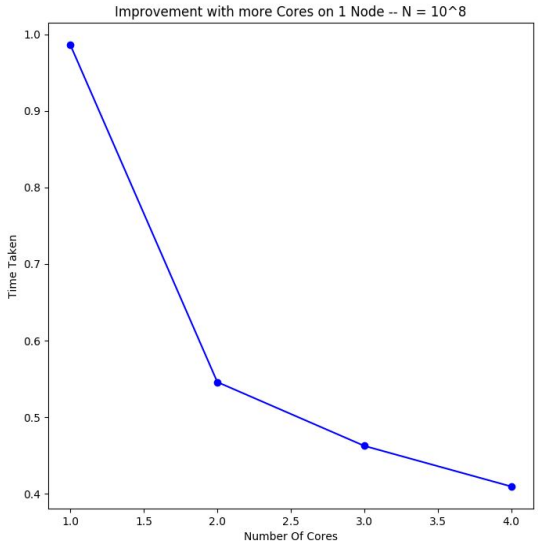
The measurements and plots are given as below:

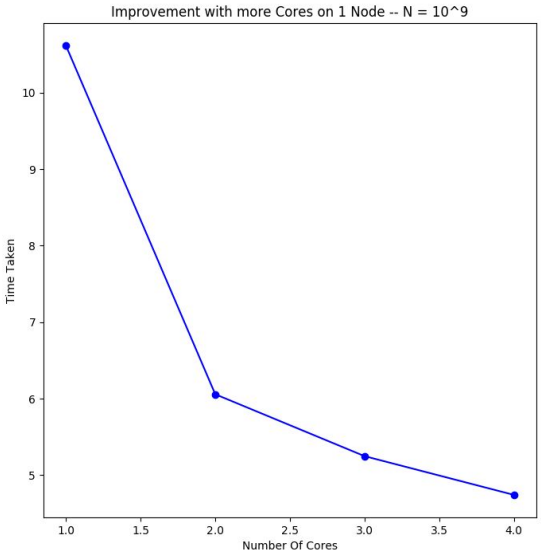
| N/cores | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 10^5 | 0.000704 | 0.00036788 | 0.000277042 | 0.000210047 |
| 10^6 | 0.006752 | 0.00303602 | 0.00224495 | 0.00168204 |
| 10^7 | 0.0763001 | 0.0392799 | 0.0263629 | 0.0208402 |
| 10^8 | 0.986114 | 0.545881 | 0.462561 | 0.409527 |
| 10^9 | 10.615196 | 6.057869 | 5.249028 | 4.742663 |
| 10^10 | 124.571731 | 74.564781 | 59.930163 | 54.191013 |

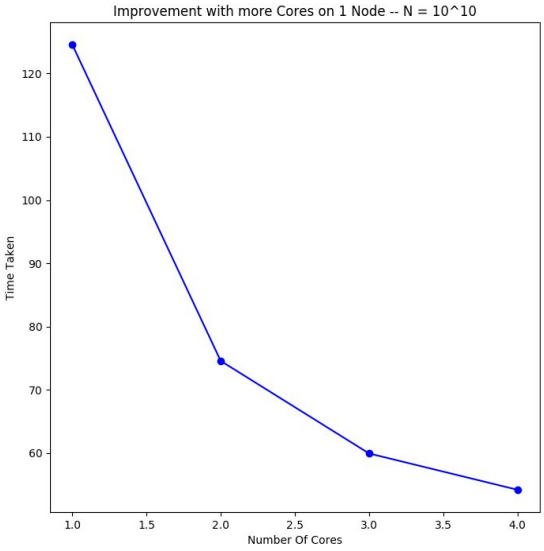


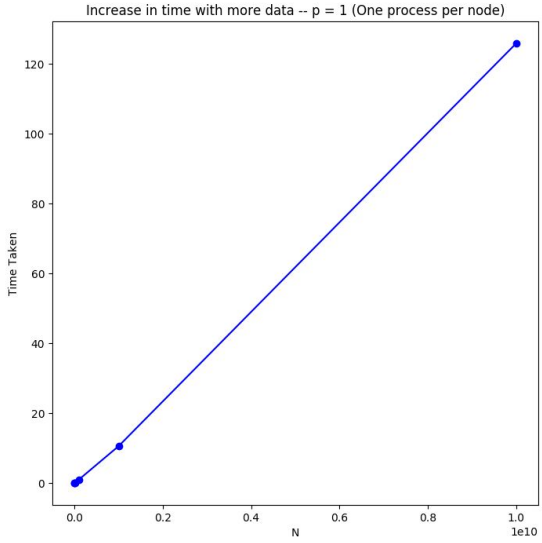


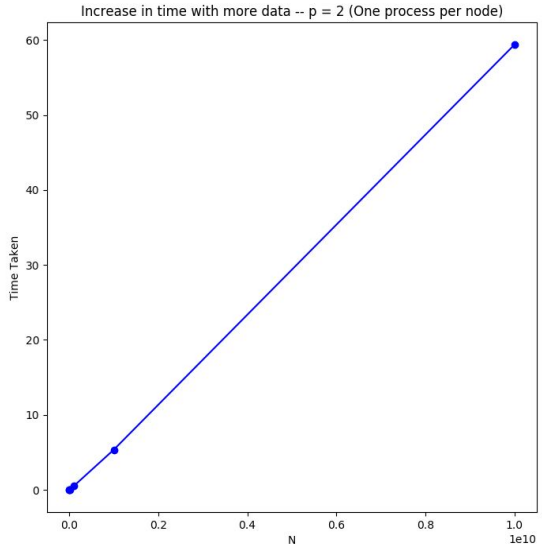



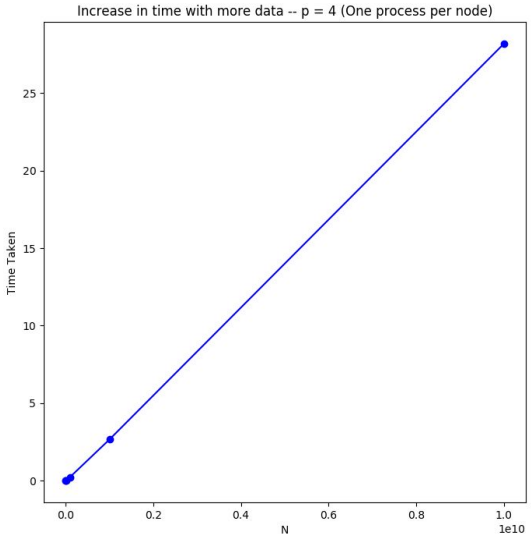



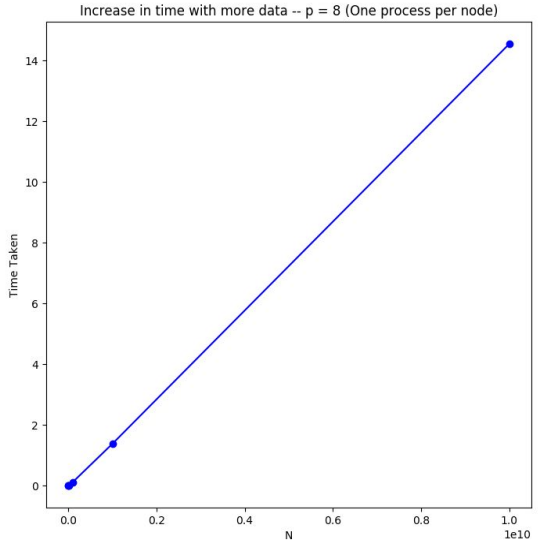




**C**. **The time increased for increase in value of N (from 10^5 to 10^10) at every value of p (where p is the number of nodes)**. It can be observed from the first table. The plots are shown below:


Increase in time with more data -- p = 1 (One process per node)
Time Taken
N
1e10


Increase in time with more data -- p = 2 (One process per node)
Time Taken
N
1e10


Increase in time with more data -- p = 4 (One process per node)
Time Taken
N
1e10


Increase in time with more data -- p = 8 (One process per node)
Time Taken
N
1e10