CSCE 735 Parallel Computing

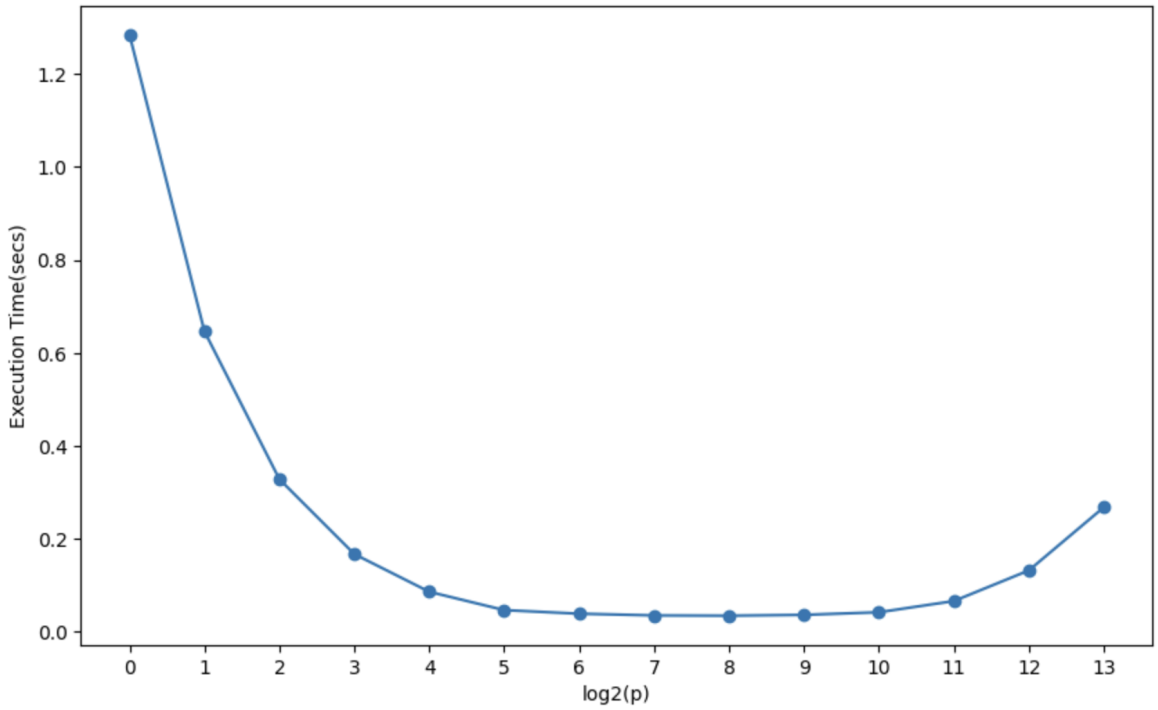
HW 1: Parallel Programming on a Multicore Multiprocessor

Part 1. Shared-Memory Programming with Threads

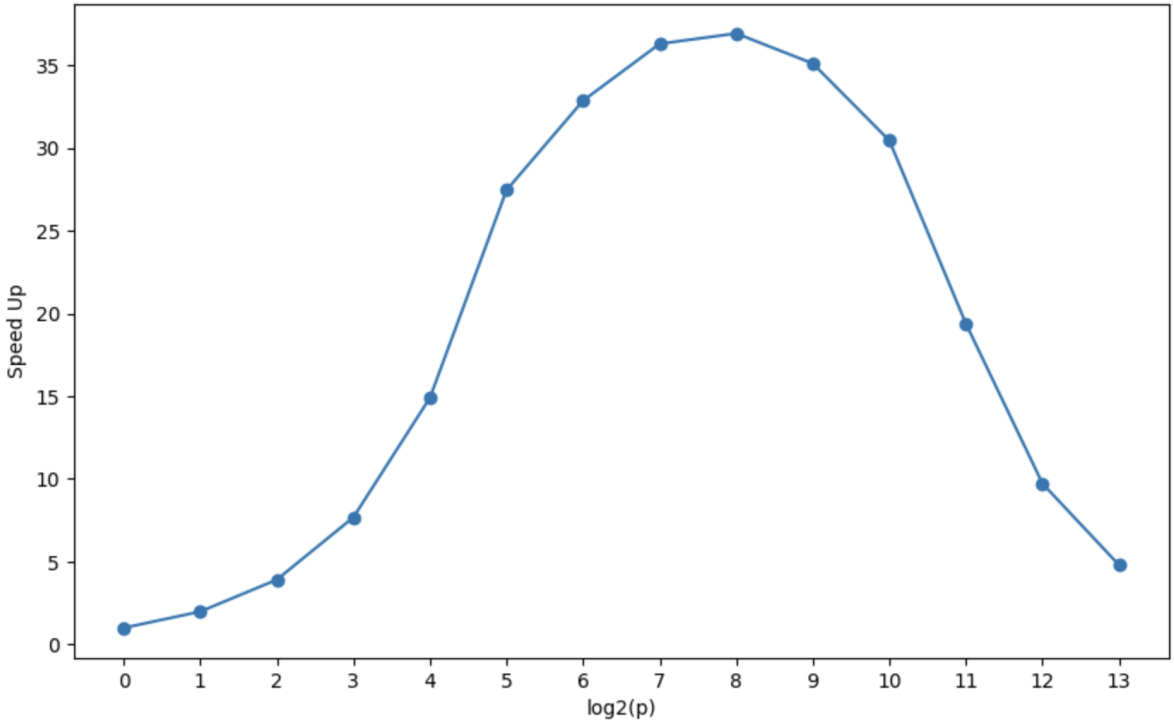
1. N = 10^8

Number of Processes (p)	Execution Time (secs)	Speed Up	Efficiency
1	1.2848	1.0	1.0
2	0.6465	1.987	0.994
4	0.3284	3.912	0.978
8	0.1673	7.68	0.96
16	0.0862	14.905	0.932
32	0.0468	27.453	0.858
64	0.0391	32.859	0.513
128	0.0354	36.294	0.284
256	0.0348	36.92	0.144
512	0.0366	35.104	0.069
1024	0.0422	30.445	0.03
2048	0.0663	19.379	0.009
4096	0.1322	9.719	0.002
8192	0.268	4.794	0.001

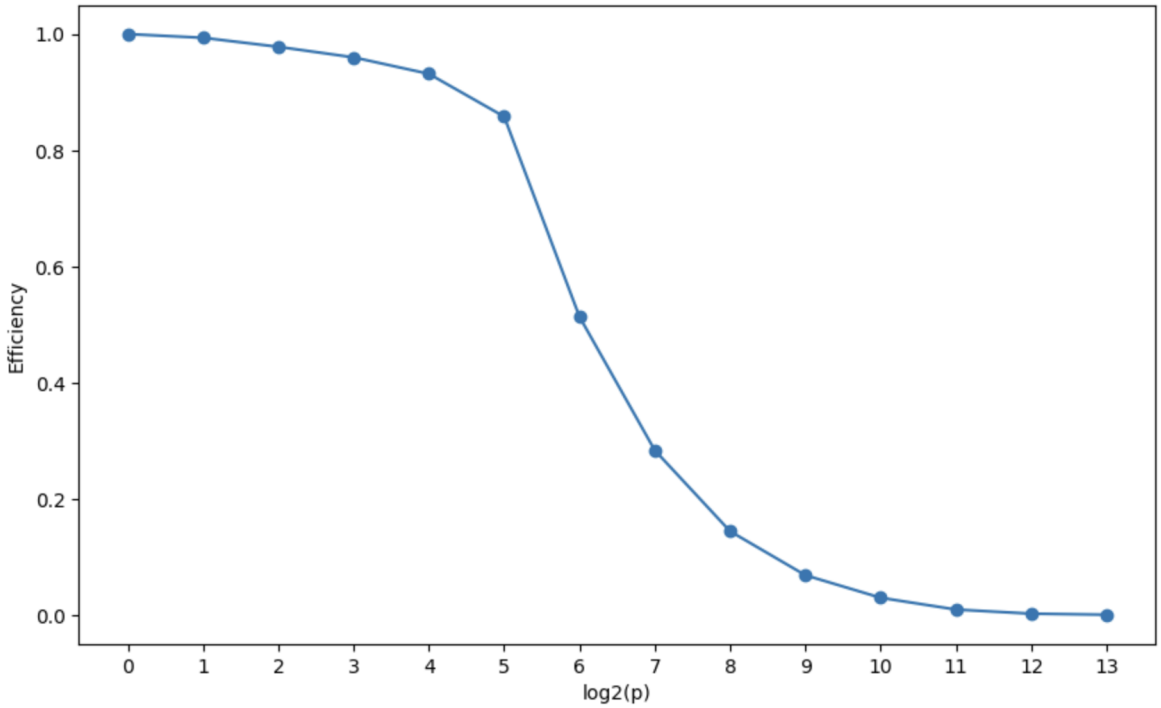
1.1 Execution Time vs log(P)



1.2 Speed Up vs log(P)



1.3 Efficiency vs log(P)

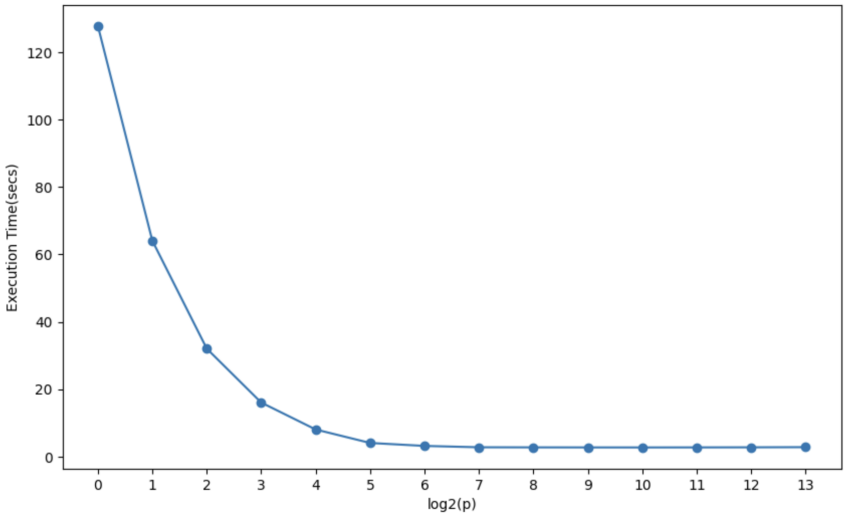


1.4 From the runs, it can be seen that **256** processes results in the least execution time of 0.0348 seconds.

2. N = 10^10

Execution time vs log(P)

Number of Processes (p)	Execution Time (secs)
1	127.9044
2	64.0983
4	32.0403
8	16.026
16	8.0214
32	4.0206
64	3.1414
128	2.7389
256	2.7128
512	2.6973
1024	2.6912
2048	2.7029
4096	2.7256
8192	2.7711



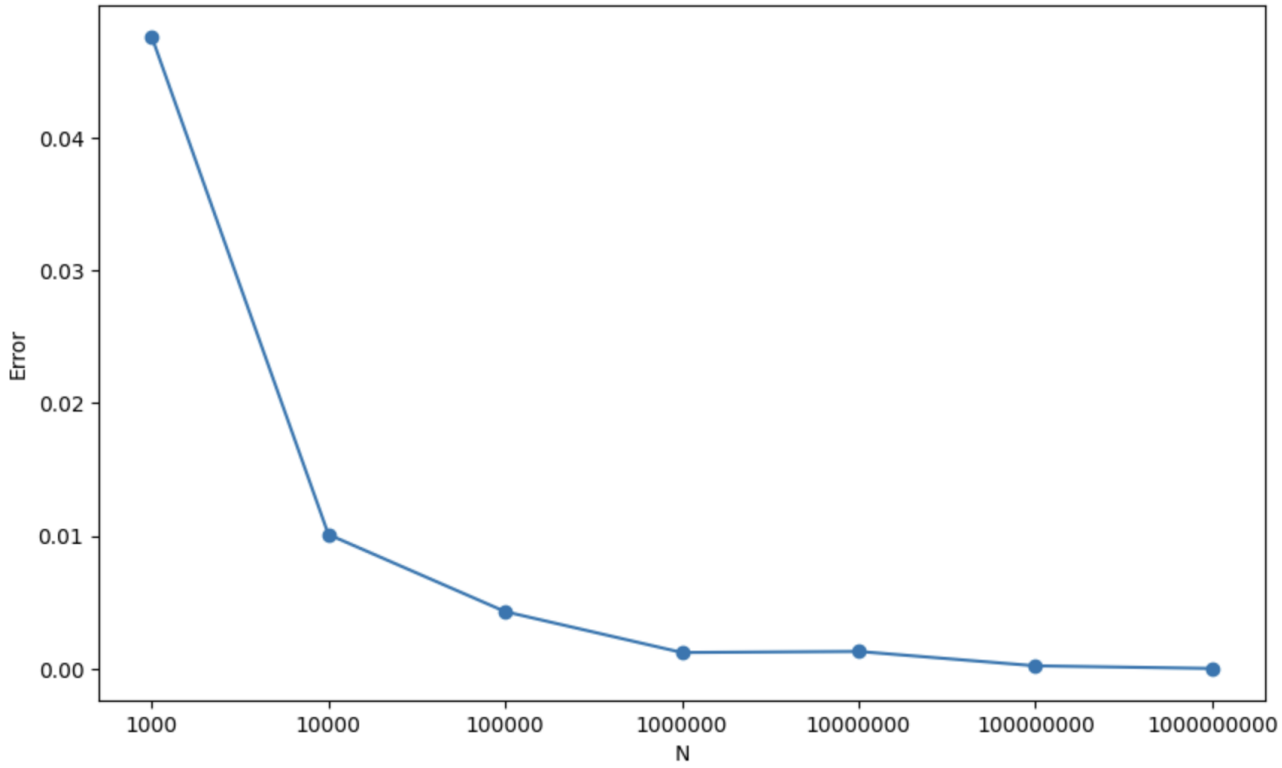
2.1 As we can see, at P = 1024, the execution time is the least which is 2.6973 seconds.

2.2 Yes, the run time seems to increase even beyond 8192 processes. This may be due to too many

threads and high context switching. The CPU would be more busy with context switching than on the task itself. Hence the increase in time. I could observe this experimentally with P = 16384, the execution time was 2.8723.

3. Yes, there was a difference in the optimal number of threads. N = 10^8 needed 256 processes and N = 10^10 needed 1024 processes. This is because, with increase in N, the task becomes more complex, and hence needs more processing. Therefore, with greater N value, the task requires more threads to reduce execution time.

4. Error vs N Where N is 10^k, k = 3..9



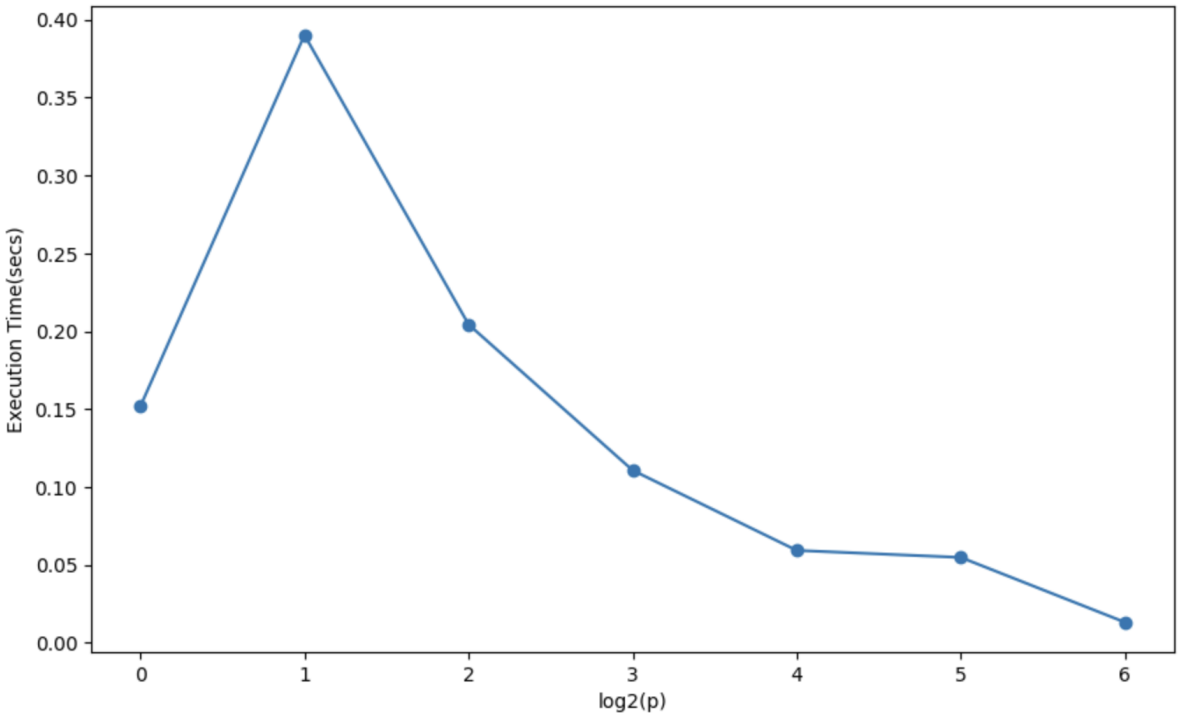
As N increases, the error reduces, which proves the accuracy of the algorithm is increasing.

Part 2. Distributed-Memory Programming with MPI

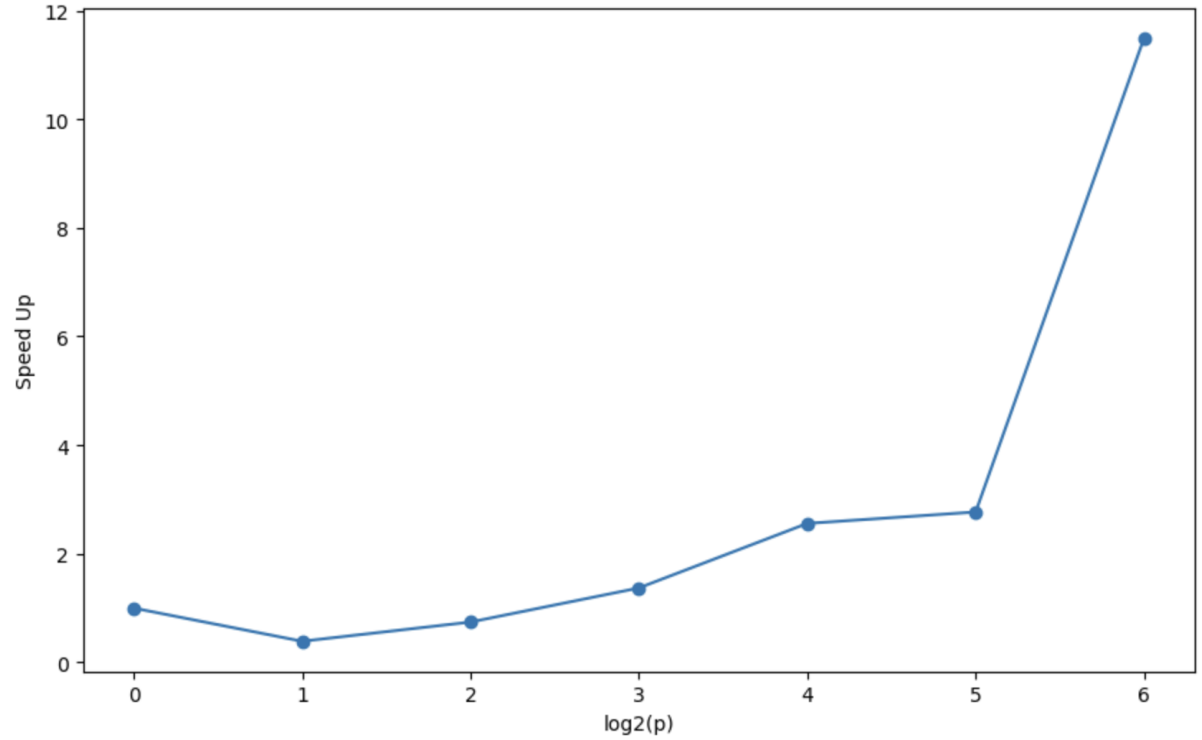
5. N = 10^8, tasks per node = 4

Number of Processes (p)	Execution Time (secs)	Speed Up	Efficiency
1	0.1517	1.0	1.0
2	0.39	0.389	0.194
4	0.2043	0.743	0.186
8	0.1107	1.37	0.171
16	0.0593	2.558	0.16
32	0.0548	2.768	0.087
64	0.0132	11.492	0.18

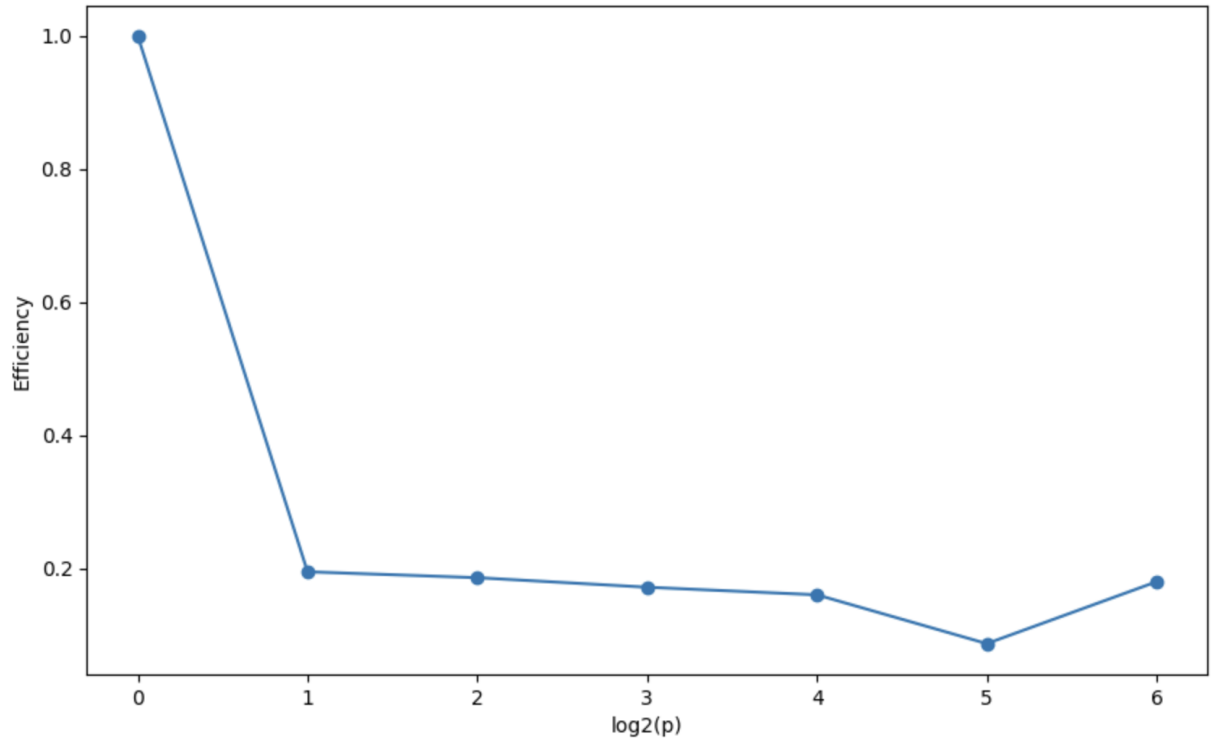
5.1 Execution Time vs log(P) with task per node as 4



5.2 Speed Up vs log(P)

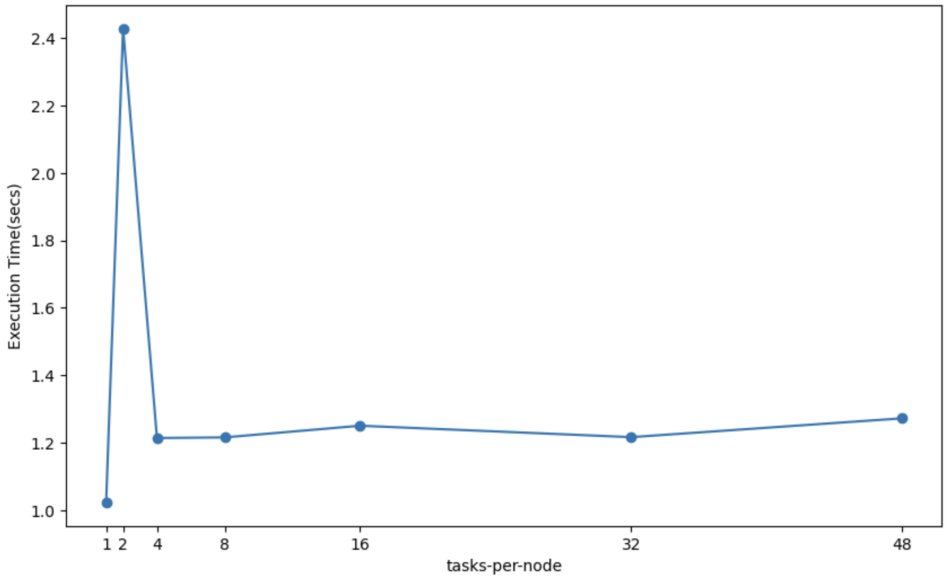


5.3 Efficiency vs log(P)



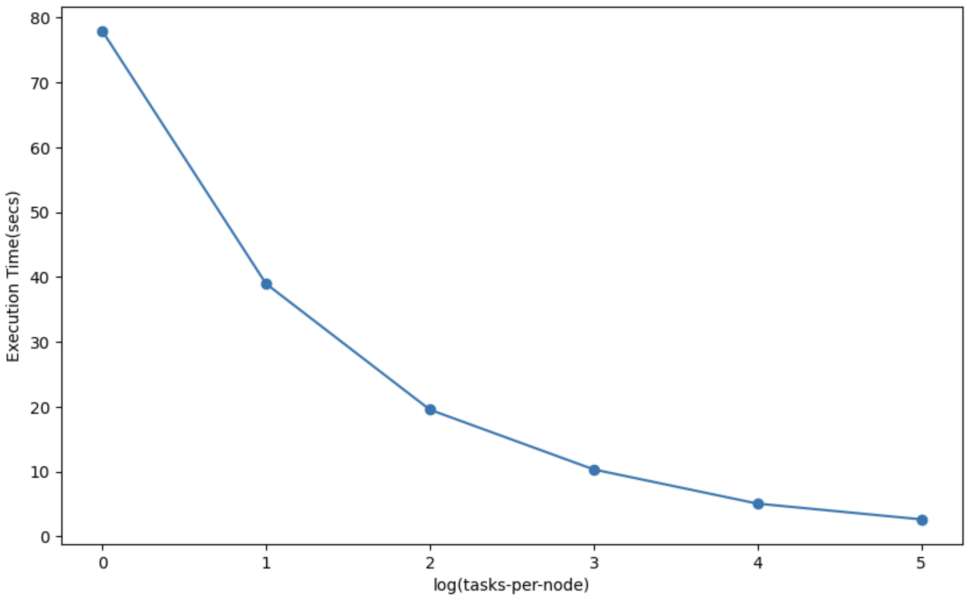
5.4 The execution time is the least (0.0132 seconds) when p is 64.

6. N = 10^10, p = 64, plotting execution times vs tasks-per-node, with number of nodes as 16.



There is very minimal difference on increasing the number of tasks per node. The minimal execution time is achieved at tasks-per-node=4.

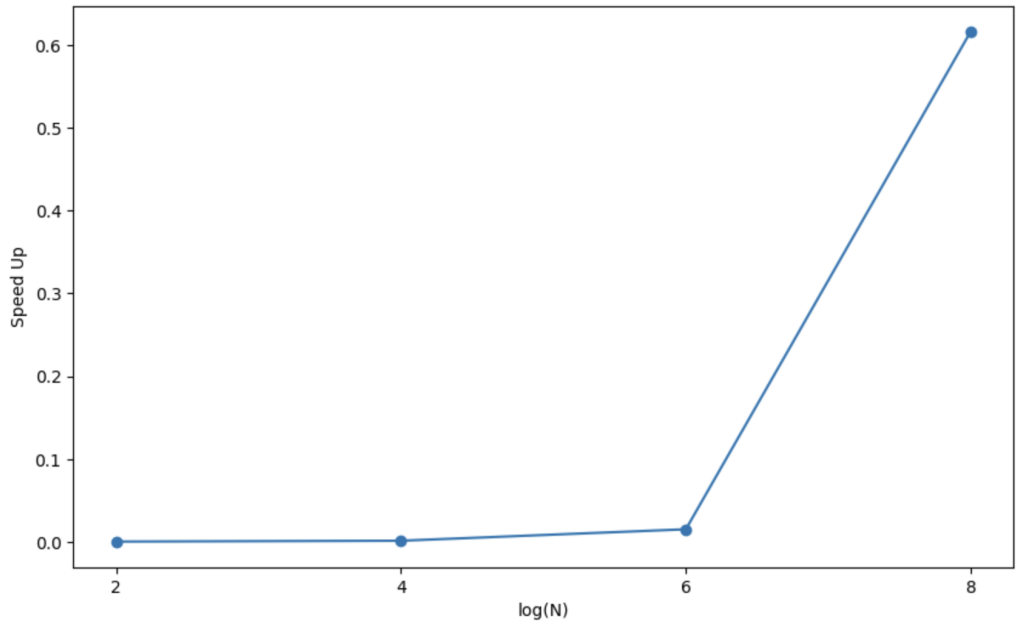
Plotting execution times vs tasks-per-node, with number of nodes as 1.



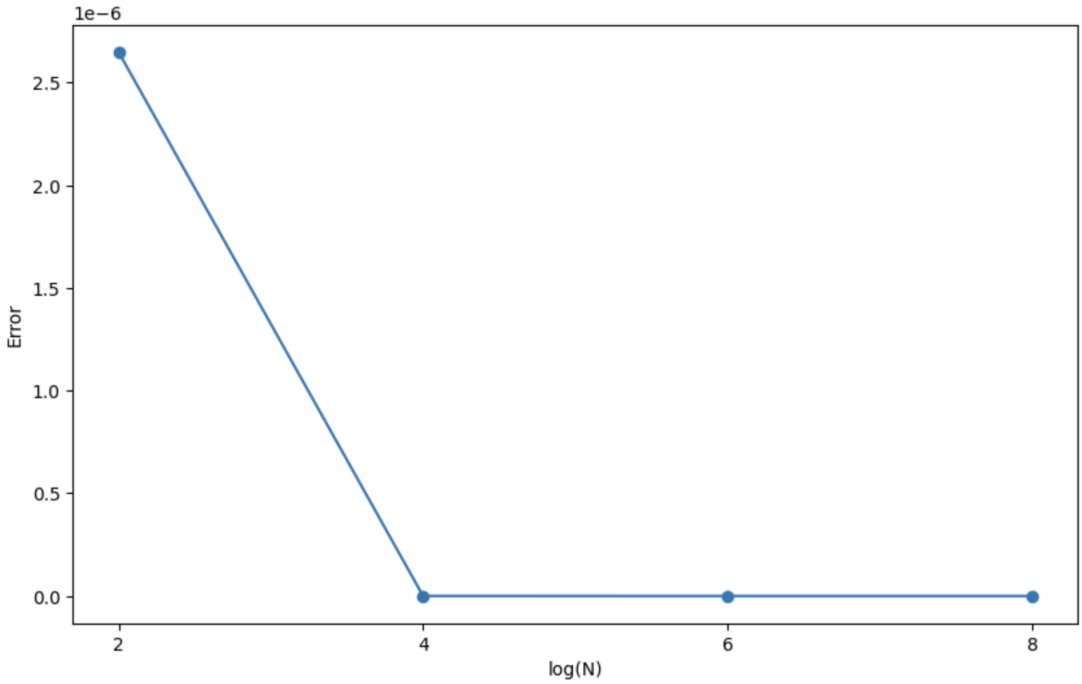
A significant difference can be seen if only one node is used, the least time being at 32 tasks per node.

7. p=64 for n=10^2, 10^4, 10^6 and 10^8, with ntasks-per-node=4.

7.1 SpeedUp vs log10(N)



7.2 Error vs log10(N)



The accuracy improves as N increases.