Abhijeet Nayak
UB ID: 50509142

Problem Statement:

Benchmarking a KDE estimator

Analysis:

Code Complexity: The code complexity for the KDE estimate would be ~ $O(n^2)$ if we are changing both the input parameters which are 'n' and 'k' (and are comparable in value), but for my analysis, I kept k: nearest neighbor constant and increased the value of 'n' by a factor of 10 (also k was much smaller than n) which makes it an $O(n)$ time complexity problem. One could observe the linear dependency of problem size on execution time, that is when we are increasing the value of problem size(n) by a factor of k, the computation time also increases by the equivalent factor.

Procedure:

Processor Metadata:

Model Name	Num of cores	Clock Speed	Memory	Cache	Compiler
Intel(R) Xeon (R) Gold 6230	32	2.10GHz	187G	L1(d):1.3MiB L1(i):1.3MiB L2:40MiB L3:55MiB	icpx

Execution Methods: For benchmarking an algorithm, the primary objective of the analysis was to compare sequential with parallel approaches and measure the effect of parallelization in terms of training speed, execution time, scale-up, and parallel efficiency.

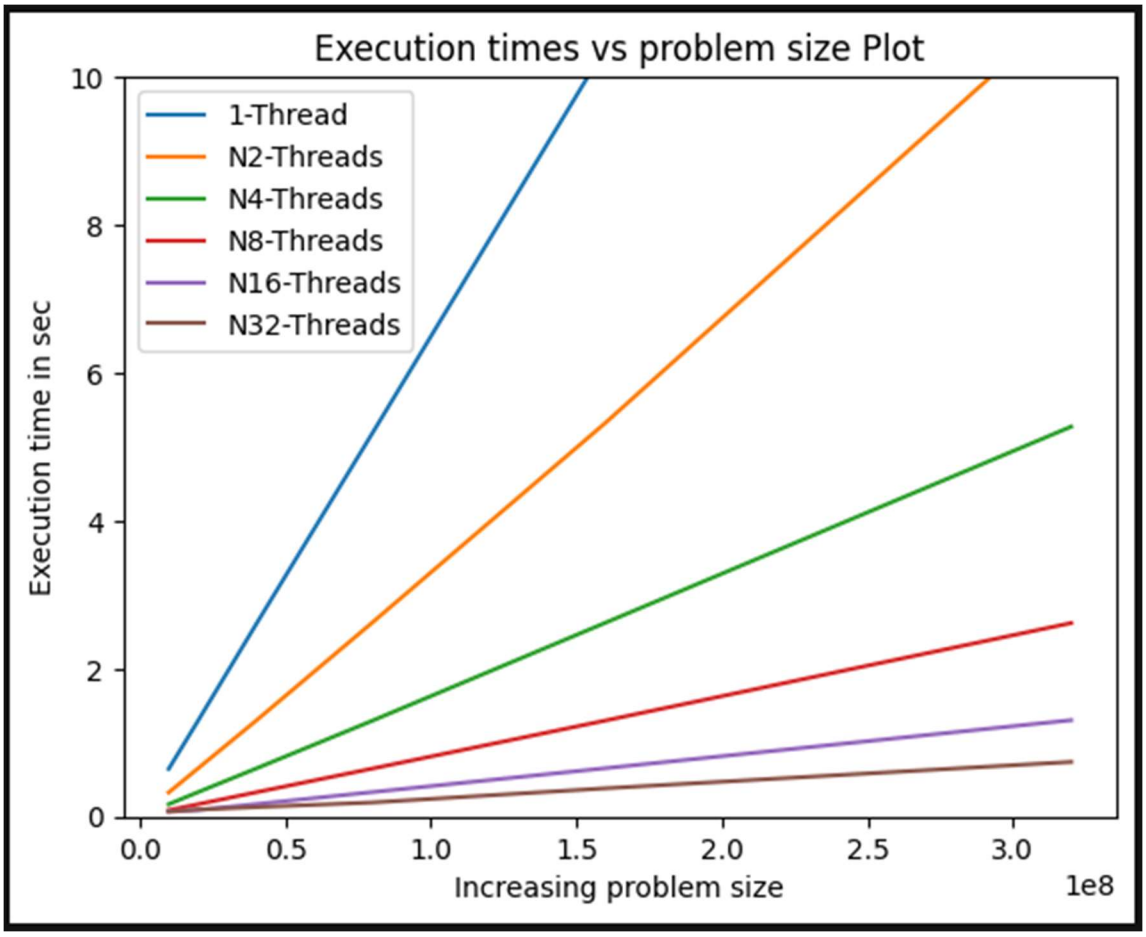
The problem size (n) was varied from $10x10^6$ to $32x10^6$ and the number of processors from sequential (with 1 proc) to 32 processors, and we are using OpenMP to parallelize the problem.

Results:

Execution time for different problem sizes and different numbers of processors used in seconds.

p/n	10,000,000	20,000,000	4,000,000	8,000,000	16,000,000	32,000,000
1	0.646274	1.29204	2.59973	5.18704	10.3952	20.8248
2	0.326704	0.650021	1.29704	2.62581	5.32301	10.9803
4	0.167344	0.329775	0.649697	1.2977	2.62071	5.2722
8	0.087497	0.166655	0.328533	0.650095	1.2988	2.61702
16	0.0721836	0.0883668	0.169203	0.331412	0.65452	1.30364
32	0.064533	0.0959619	0.128319	0.191361	0.382572	0.739231

Abhijeet Nayak
UB ID: 50509142



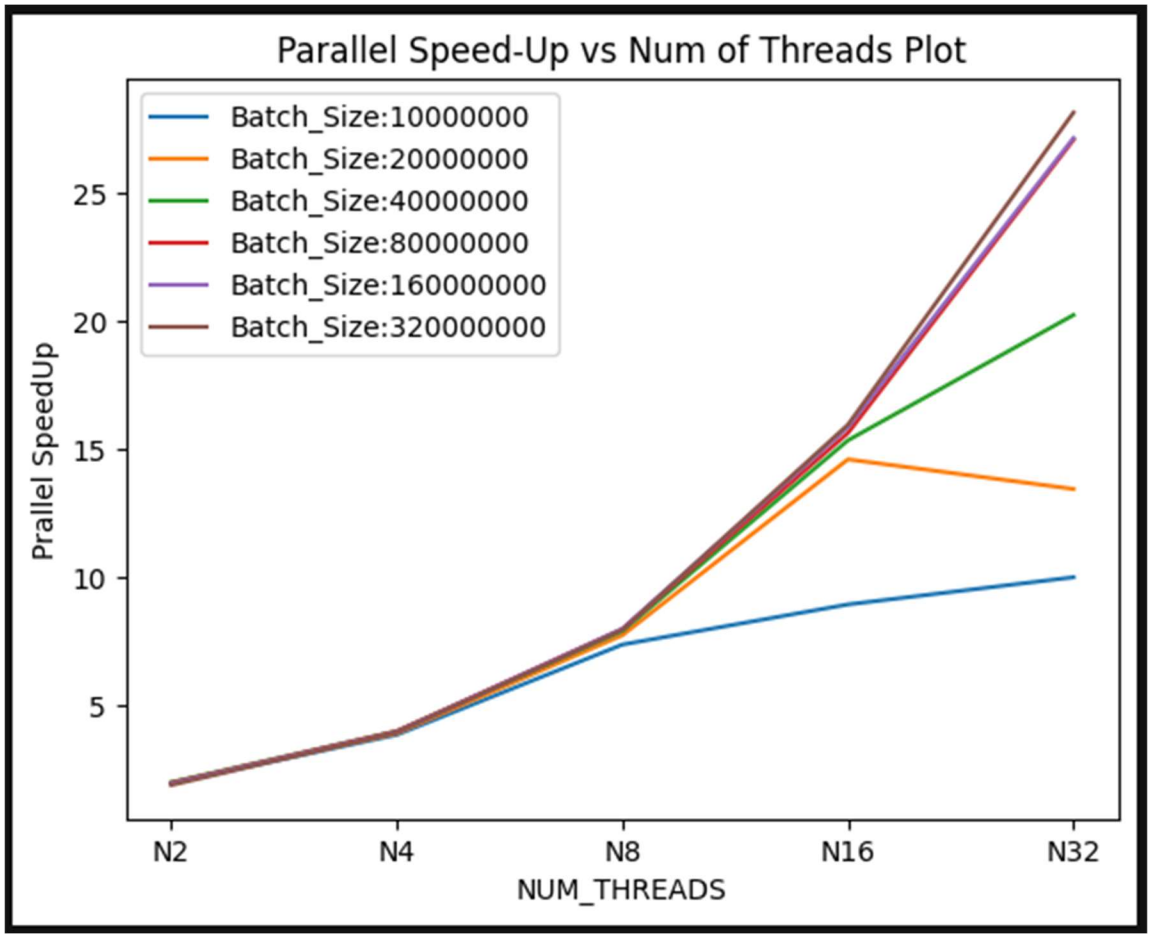
Speed Up:

Speed Up refers to the ability of a parallel computing system to handle the algorithm efficiently as we are increasing the system size or the number of processors. Numerically the speedup (S(p)) is calculated as:

$$S(p) = \frac{\text{Sequential execution time (using optimal implementation)}}{\text{Parallel execution time using p processors}}$$

Np	Batch Size 1	Batch Size 2	Batch Size 3	Batch Size 4	Batch Size 5	Batch Size 6
N2	1.978164	1.98769	2.004356	1.975406	1.95288	1.89656
N4	3.861949	3.917944	4.00145	3.997103	3.966559	3.949926
N8	7.386242	7.752783	7.913147	7.978895	8.003696	7.957448
N16	8.953197	14.62133	15.36456	15.65133	15.88217	15.97435
N32	10.01509	13.46409	20.2599	27.10605	27.17188	28.1709

Abhijeet Nayak
UB ID: 50509142



Parallel Efficiency:

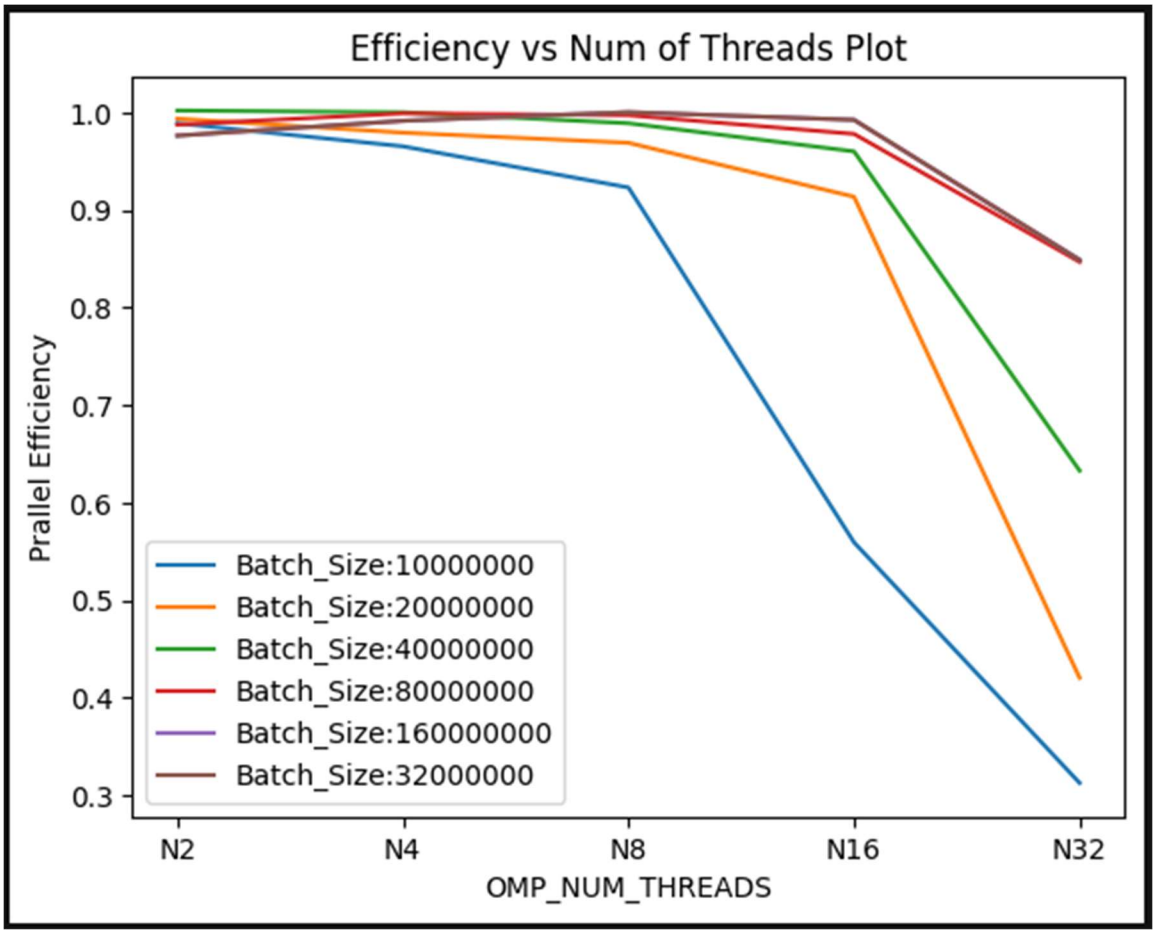
Parallel efficiency refers to how well the algorithm gets parallelized when employing parallel computing resources. It is a metric that assesses how well an algorithm or application runs in parallel on several processors or cores compared to the optimal scenario in which each processor operates independently and speeds up linearly.

The parallel efficiency (E) is calculated as:

$$E = \frac{\text{Speedup using p processors}}{p} * 100\ \%$$

Np	Batch Size 1	Batch Size 2	Batch Size 3	Batch Size 4	Batch Size 5	Batch Size 6
N2	0.989082	0.993845	1.002178	0.987703	0.97644	0.94828
N4	0.965487	0.979486	1.000362	0.999276	0.99164	0.987482
N8	0.92328	0.969098	0.989143	0.997362	1.000462	0.994681
N16	0.559575	0.913833	0.960285	0.978208	0.992636	0.998397
N32	0.312972	0.420753	0.633122	0.847064	0.849121	0.880341

Abhijeet Nayak
UB ID: 50509142



Critical Analysis:

We can observe the effect of multi-threading on execution time, speed up, and efficiency when performing strong scaling. For instance, we can observe that, when using a single thread for the problem size of 32,000,000, the time taken was 20.8248 sec and for 32 threads was 0.739231 which is significantly low.

We also observe that as we increase the number of processors after around 8~16 processors the gain in the speed up is not proportional to the increase in the number of processors for smaller problem sizes because then the communication cost starts adding up.

When we analyze the efficiency plot, we can observe that for larger problem/batch sizes the efficiency is at its peak when utilizing around 8-16 processors and 2-4 processors for smaller problem sizes, which validates the point that after a certain number of processors, the gain in computation time gets offset by the communication costs between them.