3.2 (Task 2.2) Execution Time Analysis with Varied CPU Utilization

Following are the results observed with different number of CPUs :

1 CPU: 395.10 seconds.
2 CPUs: 277.55 seconds.
3 CPUs: 229.26 seconds.
4 CPUs: 221.68 seconds.

Ratio between 1 CPU and 2 CPUs: 1.424
Ratio between 2 CPUs and 3 CPUs: 1.21
Ratio between 3 CPUs and 4 CPUs:1.034

The **initial reduction in execution time from using 1 CPU to 2 CPUs is significant**. This improvement is because of the **effective parallelization capabilities of Ray and Modin**, as they can distribute tasks across multiple CPUs, thus reducing the overall execution time.

However, as we increase the number of CPUs beyond 2, the decrease in execution time becomes less pronounced. The transition from **2 CPUs to 3 CPUs and then to 4 CPUs shows a diminishing return in performance improvement**. This observation does not indicate a linear speedup with the addition of more CPUs.

Some reasons for less significant raise can be due to -

Overhead of Parallelization: As more CPUs are added, the overhead associated with managing parallel tasks and synchronising data across multiple workers may counterbalance the benefits of parallel execution.

In conclusion, while increasing the number of CPUs can significantly reduce execution time, especially when moving from a single CPU to multiple CPUs, there are diminishing returns as more CPUs are added. This behaviour of nonlinear speedup could be due to overhead introduced by parallel computing frameworks when optimising performance for large-scale data processing tasks.

4.2 (Task 3.2) Theoretical Maximum Speedup vs. Observed Speedup

I would like to use two Laws, to derive the theoretical Maximum

Amdahl's Law - states that the speedup achievable in parallel computing is limited by the proportion of the computation that remains sequential, expressed as S(N) = 1/((1-P)+(P/N)) , where P represents the proportion of the computation that can be parallelized and N is the number of processors.

Gustafson's Law states that the speedup scales linearly with the size of the problem, formulated as $S(N)=N-\alpha(N-1)$, where
α represents the proportion of the computation that is inherently sequential and
N is the number of processors.

Points considered:

- The sequential part of the merge sort algorithm mainly involves merging the sorted sublists.
- The time complexity of the merging step is O(n), where n is the total number of elements across all sublists.

- The parallelizable part of the merge sort algorithm involves sorting individual sublists in parallel.
- Assuming perfect parallelization, each sublist can be sorted independently.
- The time complexity of sorting each sublist is O(n/p), where p is the number of partitions and n is the size of the sublist.

Theoretical Maximum Speedup:

- Since the merging step is inherently sequential and cannot be parallelized further, Amdahl's Law applies to this part.
- Let's denote the proportion of time spent on merging as Q =0,
- Assuming Q=0 (perfect parallelization of sorting sublists), Amdahl's Law yields the theoretical maximum speedup for the merging step (pairwise merging), which is half of the number of workers, i.e. 2 in this case.

- For the sorting sublists step, Gustafson's Law applies as we scale the problem size with the number of workers.
- Gustafson's Law suggests that the speedup can scale linearly with the number of workers as the problem size increases.
- Therefore, the theoretical maximum speedup for sorting sublists can be approximated as N=4.

Total Theoretical Maximum Speedup:

- Since merging and sorting sublists are both critical parts of the merge sort algorithm, the total theoretical maximum speedup possible would be

- Total theoretical maximum speedup = Speedup from sorting sublists * Speedup from merging.
- Total theoretical maximum speedup = 4 × 2 = 8

Comparing Theoretical and Observed Speedup

The difference between the theoretical maximum speedup and the observed speedup with Ray can be attributed to several practical factors:

- Real-world overheads, like managing parallel tasks and data synchronisation, are often more significant than what theoretical models anticipate.
- Shared resources, such as memory bandwidth and CPU cache, can lead to contention among parallel tasks, reducing the effective speedup observed in practice.

Therefore its unlikely to reach the theoretical maximum speedup.