MPI IMPLEMENTATION OF K-MEANS CLUSTERING

Sepideh Hayati
Alireza Khornegah

Spring 2025

WHAT IS K-MEANS CLUSTERING?

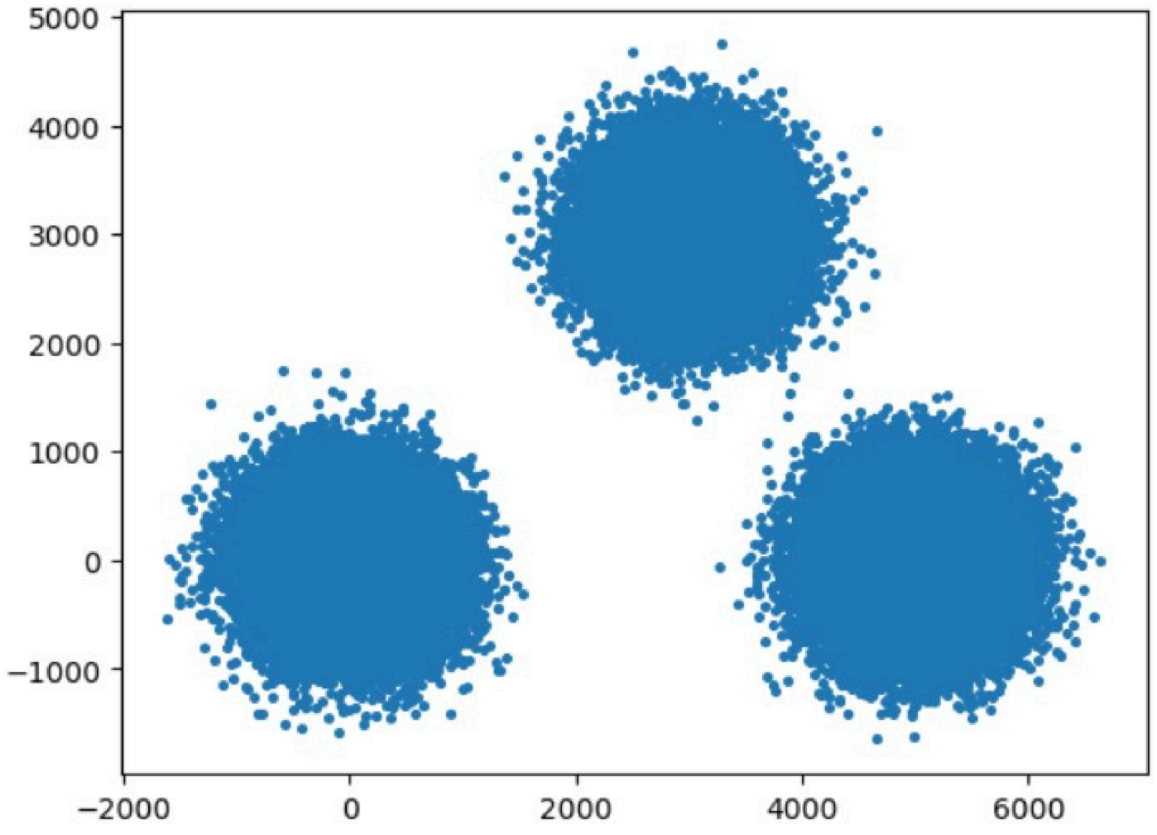
K-means is an unsupervised learning method for clustering data points.

Defenition

How doese it work?

Its Goal

DATASETS GENERATION



How to Generate Datasets

In Python using NumPy

Feature of Datasets

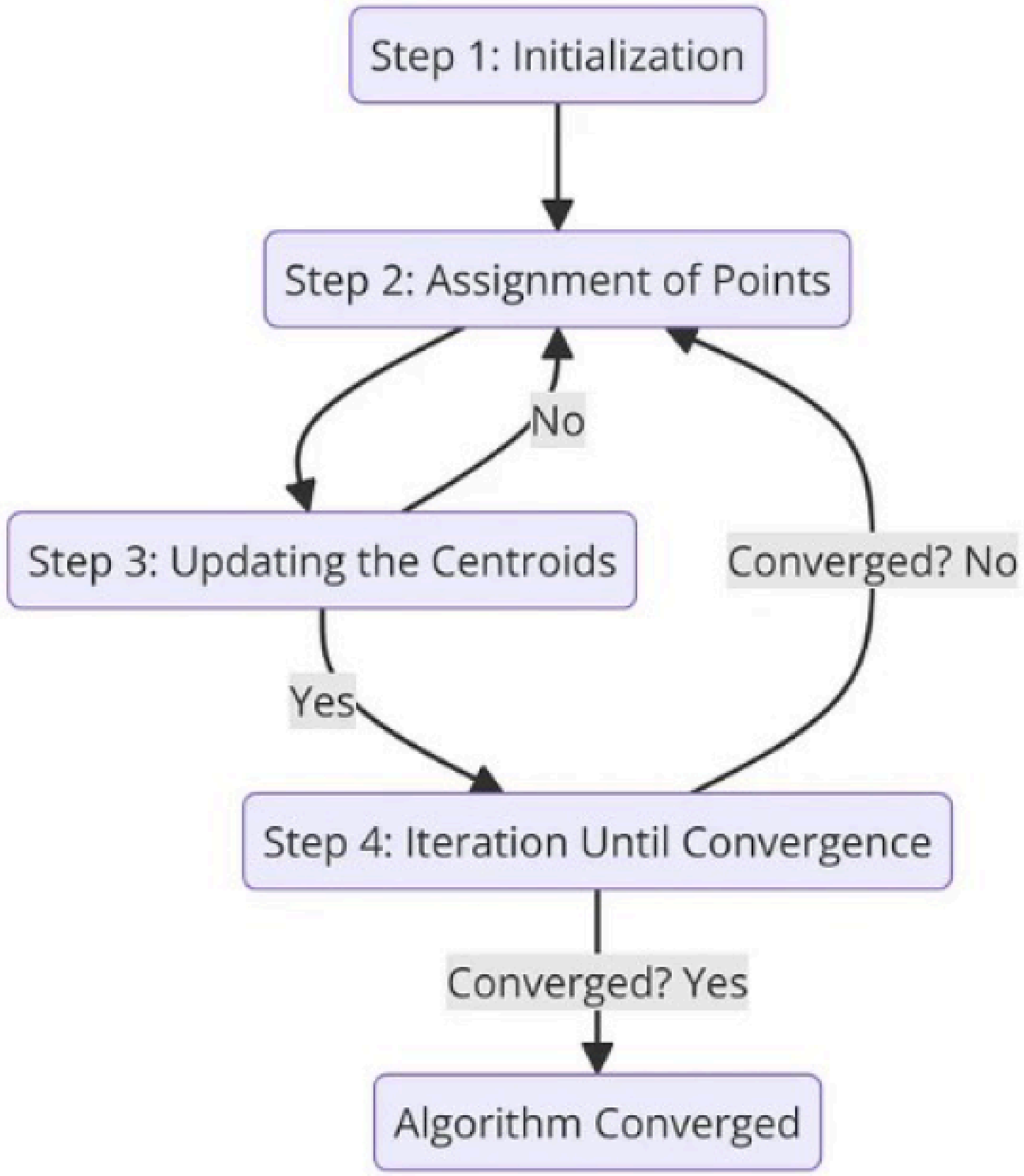
Three cluster centers
With a standard deviation of 400
2D random points

Size of Datasets

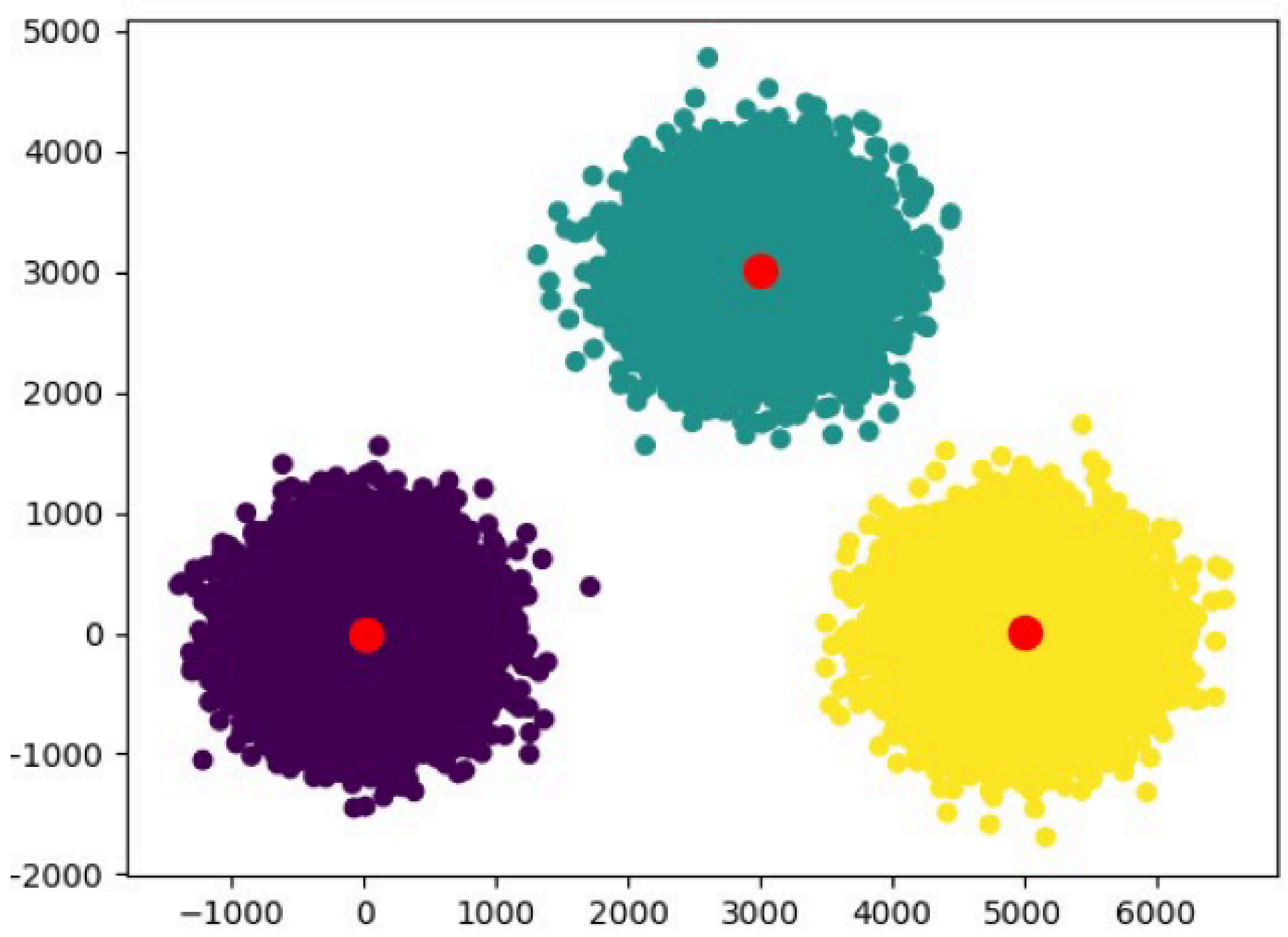
20K, 40K, 80K

SERIAL IMPLEMENTATION

Algorithm Steps



Given a dataset and the parameters below, the goal is to assign each point to one of K clusters and compute the final centroids.



OBSERVATION FOR PARALLELIZATION

- Data Distribution and Broadcasting
- Initial Centroid Selection and Distribution
- Workload Partitioning, Distance Calculation and Cluster Assignment
- Parallel Centroid Update with Reduction

Data Distribution and Broadcasting

Parallel Version

- Only one processor reads the dataset from the file shares the data with all the others using MPI.
- MPI Functions Used: MPI Bcast

Serial Version

- Entire data is read by one process, no sharing or distribution needed.

Initial Centroid Selection and Distribution

Parallel Version

- Root process selects the initial centroids (starting points for clusters) and shares them.
- MPI Functions Used: MPI Bcast

Serial Version

- Initial centroids are selected and used by one process, without sharing.

Workload Partitioning, Distance Calculation and Cluster Assignment

Parallel Version

The workload is divided among all processes:

- Each process handles a subset of the data points.
- Every process calculates distances only for its own chunk.
- Each point is assigned to the nearest cluster locally.
- MPI Functions Used: MPI Comm rank, MPI Comm size, MPI Allgatherv

Serial Version

- It loops through the entire dataset.
- For each point, it calculates the distance to all centroids.
- It then assigns that point to the closest cluster.
- stores the result in one process—no need for communication or data distribution.

Parallel Centroid Update with Reduction

Parallel Version

- Each process works on its own portion of the dataset
- Computes the local sum of coordinates for each cluster.
- Computes the local count of points assigned to each cluster.
- MPI Functions Used: MPI Allreduce

Serial Version

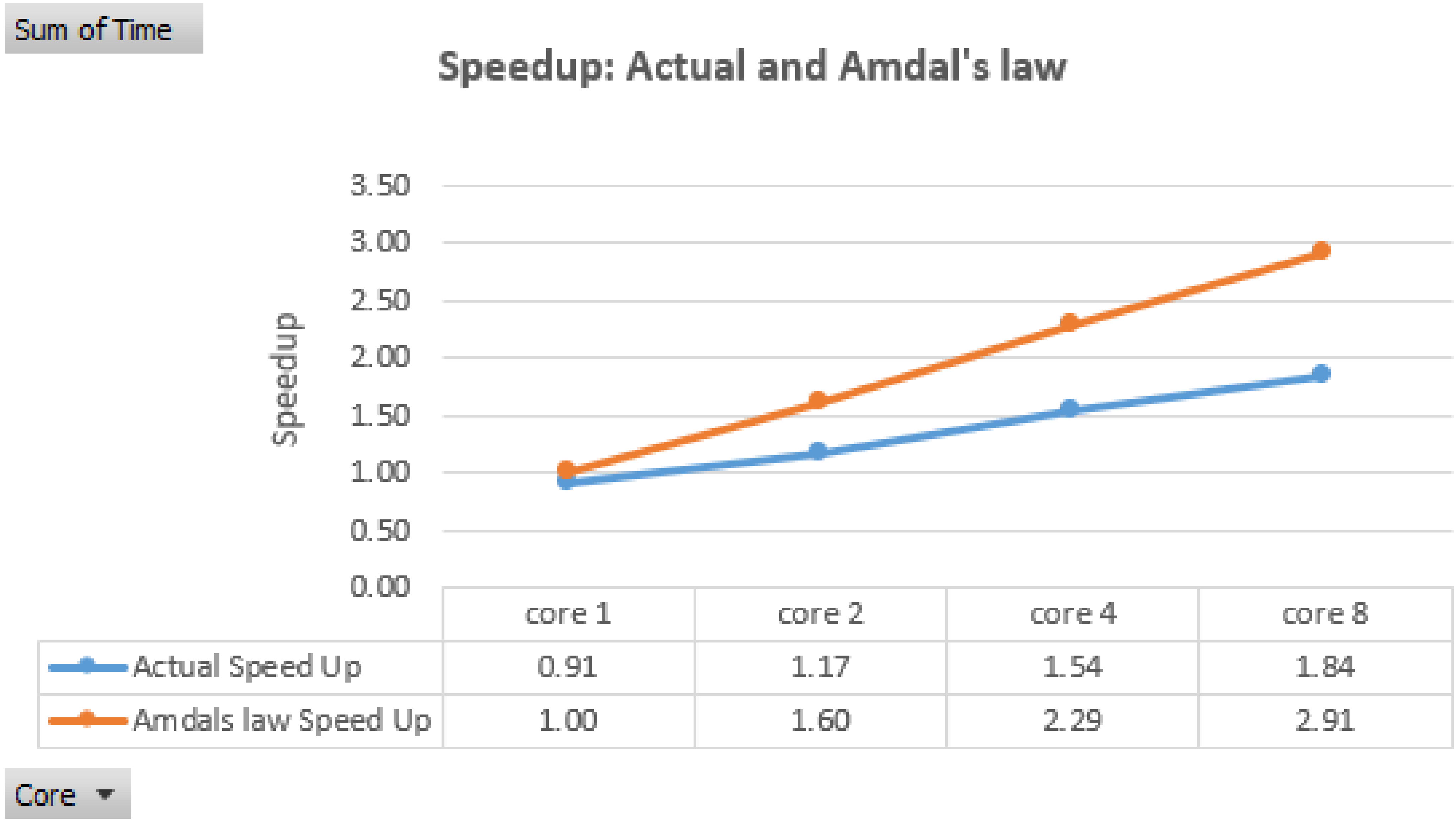
In the serial version, one process:

- Adds up all data points assigned to each cluster.
- Calculates the mean (average) to find the new centroid.

AMDAHL’S LAW

To guess how much faster it can get, Amdahl’s Law is used:

$$Speedup(N) = \frac{1}{S + \frac{P}{N}}$$



There is a GAP!

This highlights the influence of factors such as communication overhead, memory bandwidth limitations, and hardware inefficiencies.

PERFORMANCE ANALYSIS

Machines

To measure the performance of the MPI-based solution, two types of clusters were set up on GCP:

Fat Clusters:

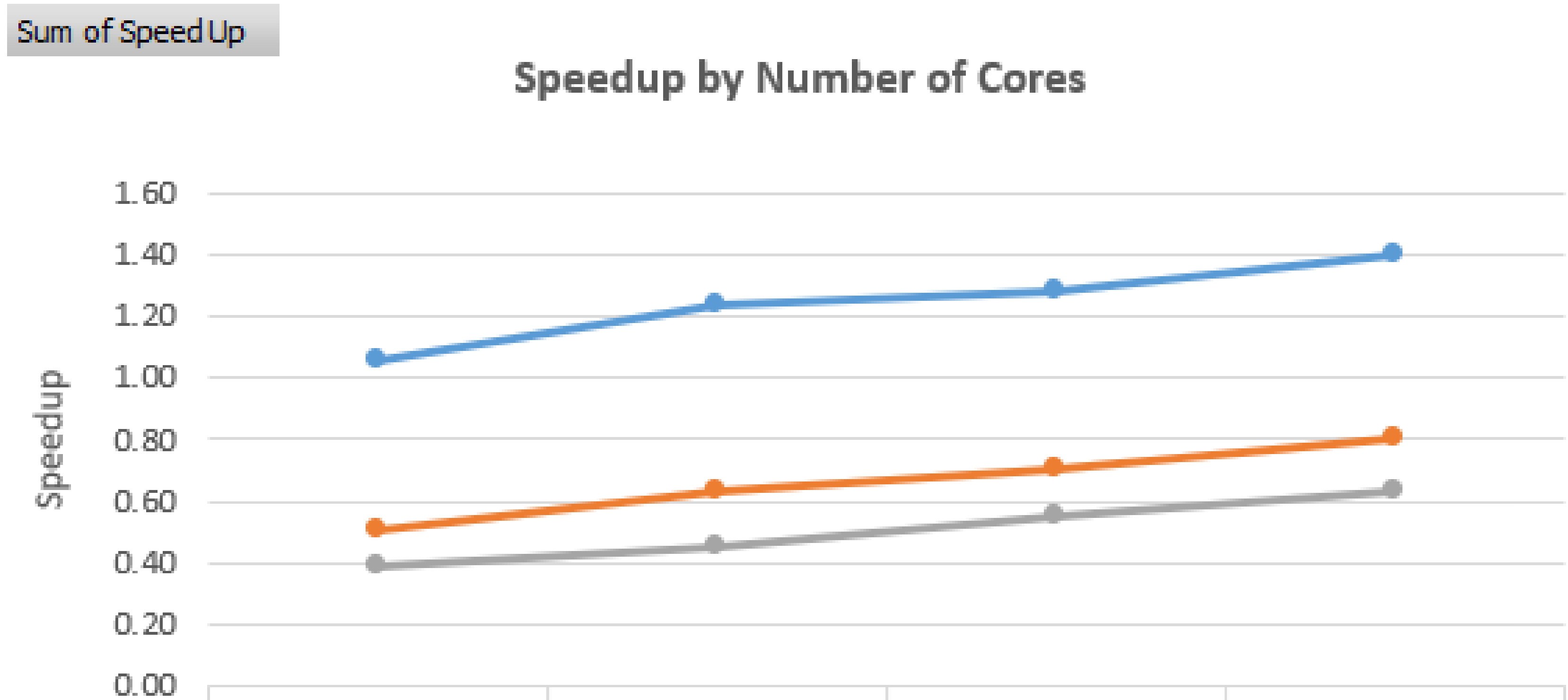
- Two nodes
- Each with 4 vCPUs
- Intra-regional, both in the same region
- Location: us-central1-a and us-central1-b

Light Clusters:

- Four nodes
- Each with 2 vCPUs
- Intra-regional, all in the same region
- Location: us-central1-a

Fat Clusters:

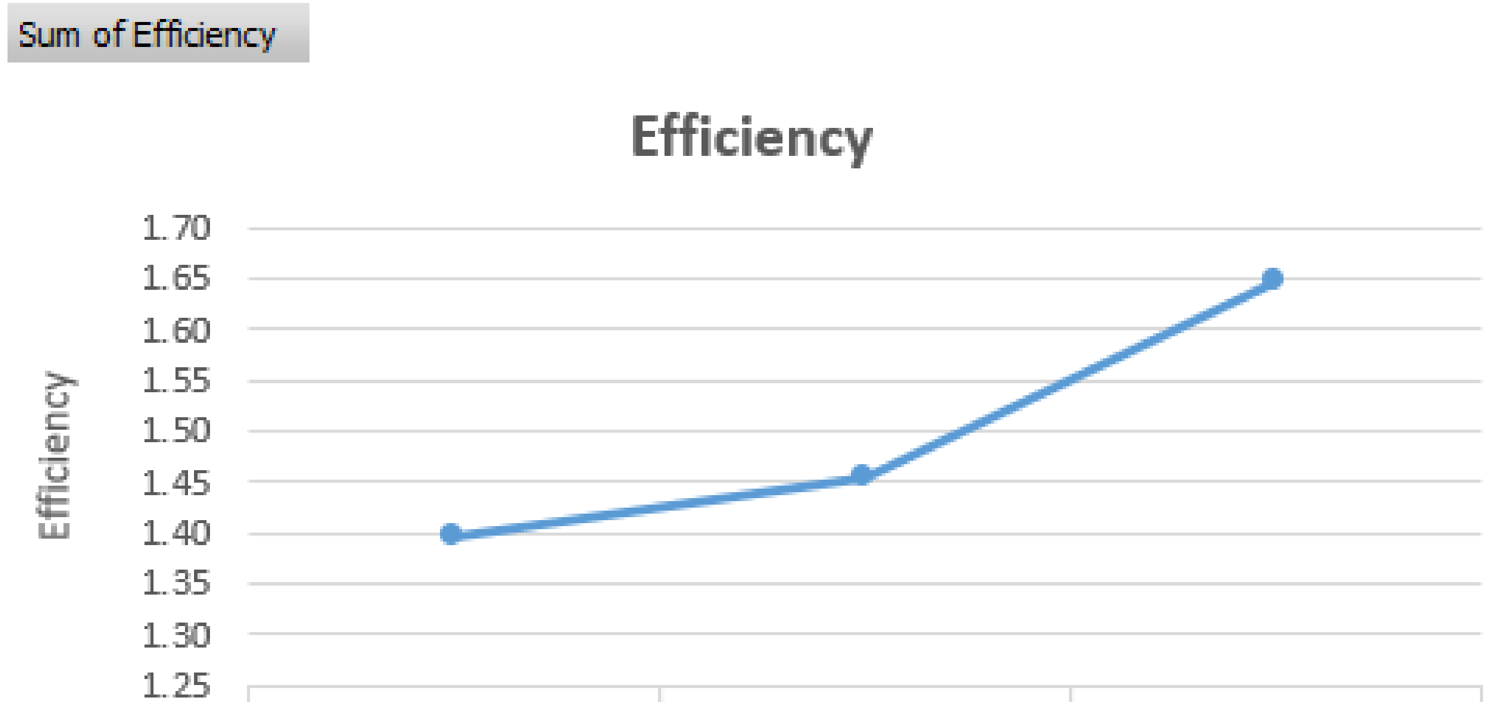
Strong scalability



	Core 1	Core 2	Core 4	Core 8
20 k	1.06	1.24	1.29	1.40
40 k	0.50	0.63	0.70	0.80
80 k	0.38	0.45	0.55	0.63

- By increasing the number of cores, the speedup improves.
- The speedup increases more slowly for larger datasets like 40K and 80K.
- This is likely due to MPI communication overhead, which grows as more cores are added, reducing the benefits of parallelism for larger datasets.

Weak scalability



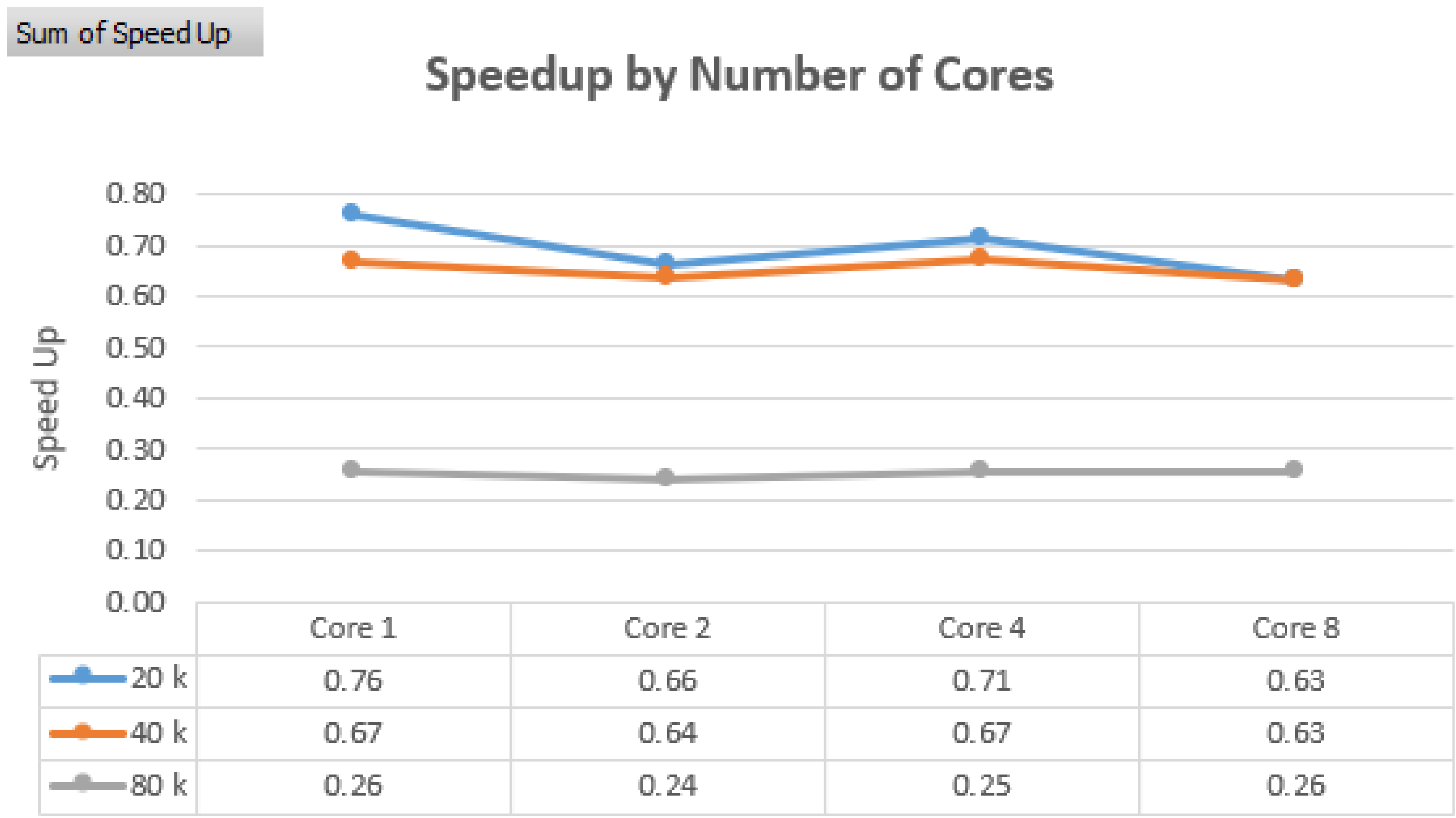
	Core 2 20 k	Core 4 40 k	Core 8 80 k
Total	1.40	1.46	1.65

Dataset size ▾ Num.P ▾

- As we increase both the dataset size and the number of cores proportionally, efficiency improves.
- This shows that the system handles larger workloads well when more resources are added, indicating good weak scalability on fat clusters.

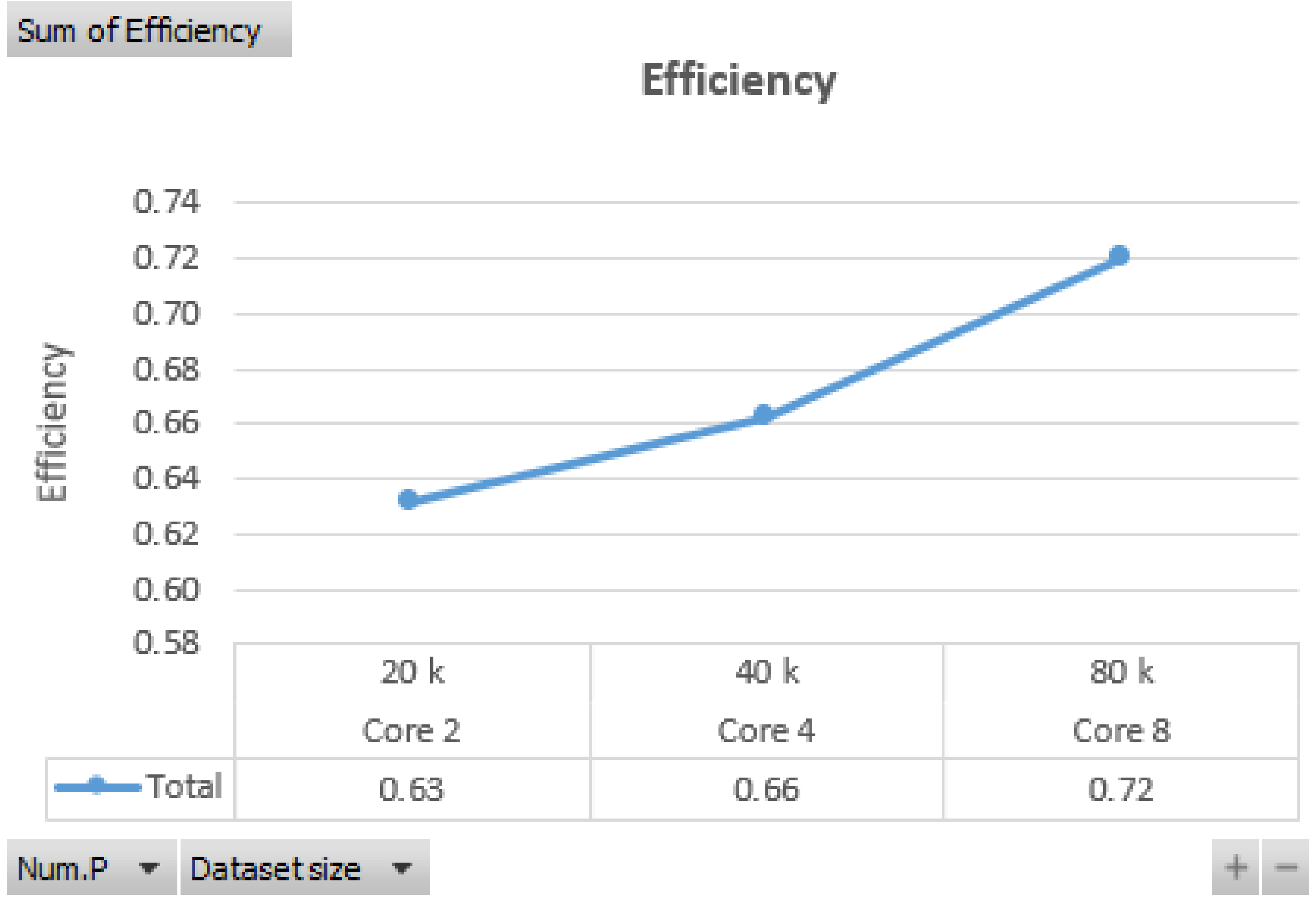
Light Clusters:

Strong scalability



- As the number of cores increases from 1 to 8, the speedup remains low and even drops slightly for all dataset sizes.
- This indicates poor strong scalability, due to the limited processing power of light clusters or Increased communication overhead

Weak scalability



- Efficiency improves steadily as both dataset size and number of cores increase proportionally .
- The system handles larger workloads more efficiently when more resources are available.
- The system shows moderate weak scalability