COMS3008-Parallel Computing : Clustering Problem

Seale Rapolai	●	1098005
Mahlekenyane Ts’eole	●	1118248
Meriam Elabor	●	1076589
Nkosikhona Sibisi	●	915702

Overview

About the project:

This project focuses on the common problem of clustering data.

It involves classifying different data items that share similar characteristics within the same data set, into groups. By doing this, various clusters of data items that share similar characteristics can be identified.

We plan to also solve the performance issue that arises

Introduction

Introduction

An issue that is introduced with clustering is that for large datasets, the performance of the algorithm becomes increasingly slow. This large dataset makes clustering a good candidate for applying parallelising techniques.

Clustering is the task of grouping a set of data into groups called ***clusters***. Each cluster is comprised of data points that are more similar to each other than data points in other clusters.

There are different algorithms to consider in order to cluster. Based on our experience with Machine Learning, we chose to focus on the *K-Means* algorithm , an exclusive clustering algorithm.

Solution Techniques

Solution Techniques

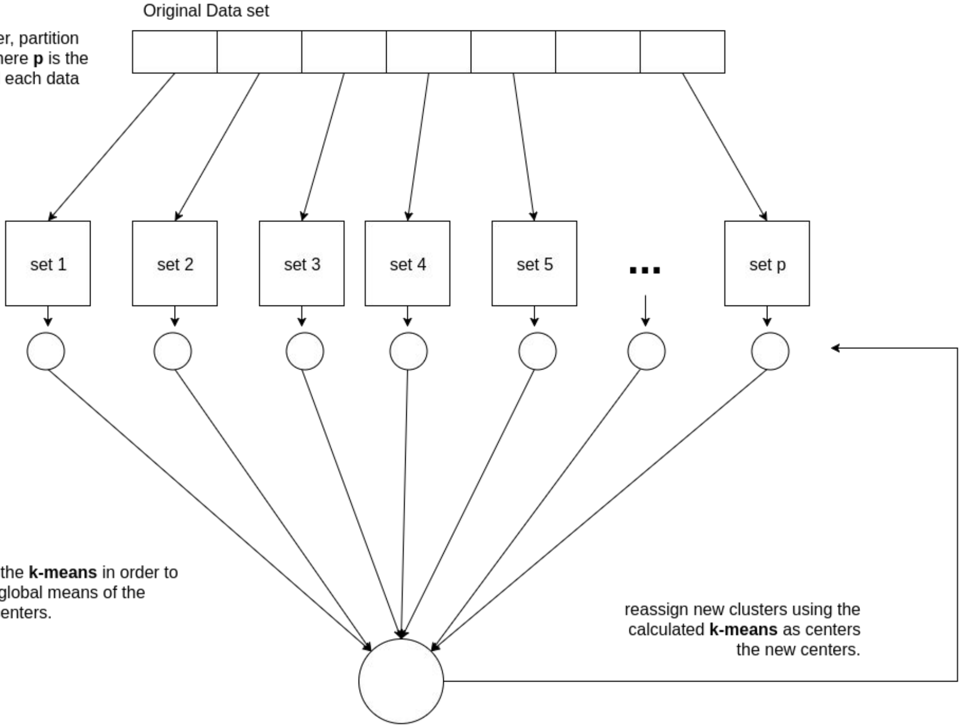
1. Task Dependencies
2. Task Interactions
3. Parallel Algorithm
4. Limitations

Task Decomposition

To parallelise the algorithm, we must first divide the computation into smaller tasks that can be run concurrently. This involves identifying parts/sections of the algorithm that can be broken down into multiple individual tasks that can be ran concurrently.

The K-means algorithm takes in a large set of data items and then classifies the data items into k separate clusters.
From this, it is clear to see that we can decompose the input data and classify multiple data items in into k clusters simultaneously.

Task Dependency



INPUT: Given **n** data points to cluster, partition the **n** data points into **p** data sets where **p** is the number of processing elements and each data set has **n/p** points.

For each set, on different processing elements use ***The k-means algorithm*** to classify the data points into **k** clusters, these will be the local classifications. and we calculate the local means of the data points.

Reduce the **k-means** in order to find the global means of the cluster centers.

reassign new clusters using the calculated **k-means** as centers the new centers.

Figure 1: Task Dependency Graph

Task Interactions

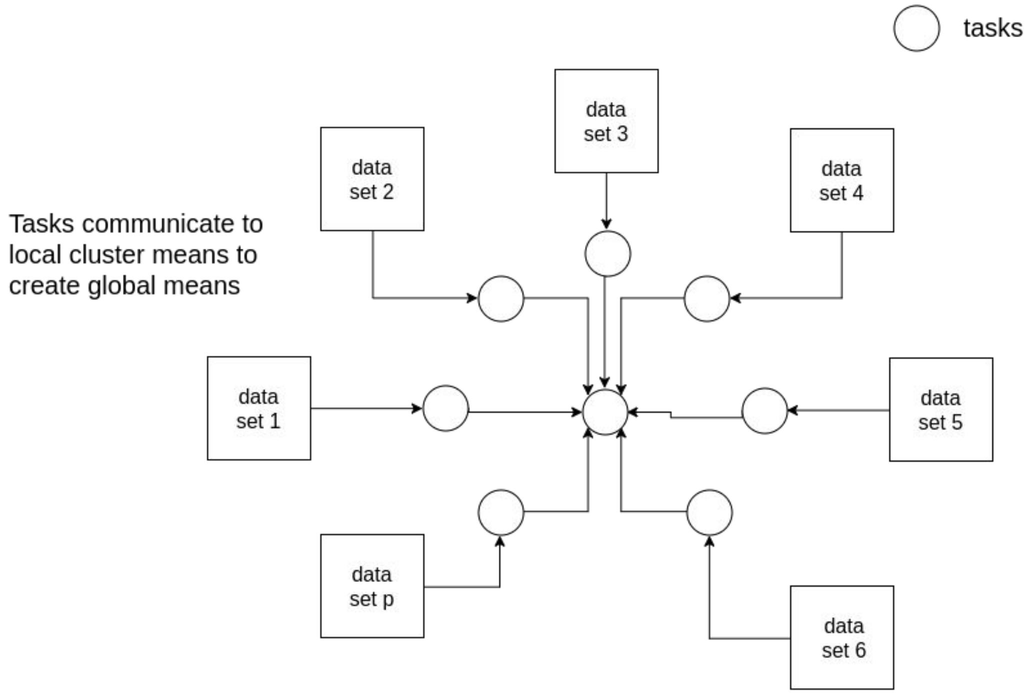


Figure 2: Task Interaction Graph

Parallel Algorithm

Various techniques were tested to parallelise the algorithm. With parallelising the *K-Means* itself, there was a noticeable slow down.
This led to the partitioning of the input data. The algorithm partitions the data and assigns them to a fixed number of processes.
We have our tasks as classifying each data point to a cluster. Our parallel algorithm also has the reducing of the averages of the data points in the *k* clusters parallelised.

The algorithm takes as input the *number of points*, *dimension of the points* and *k* as well as the *data list*. The data list has the format *(x,y,...,z,t)* i.e. the points and *t*, the cluster in which the point belongs. It then partitions and classifies the data as first come first serve. The algorithm then takes the averages in serial.

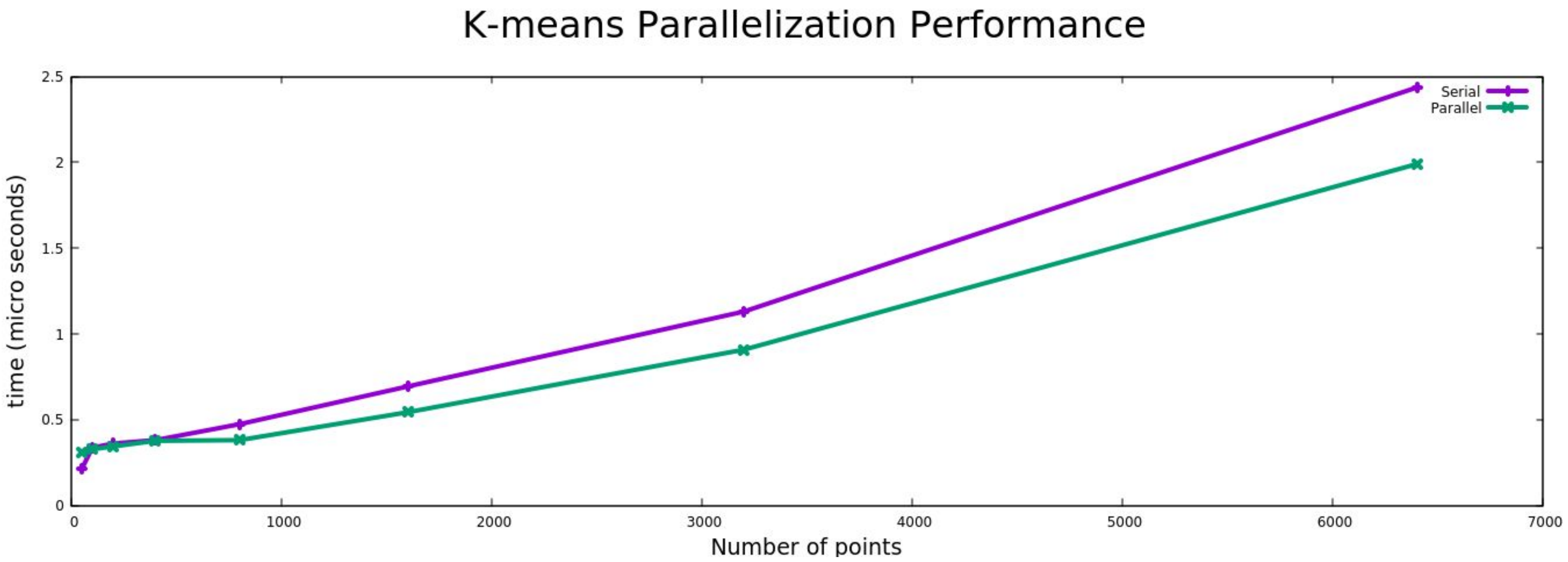
This process is continued a fixed number of times.

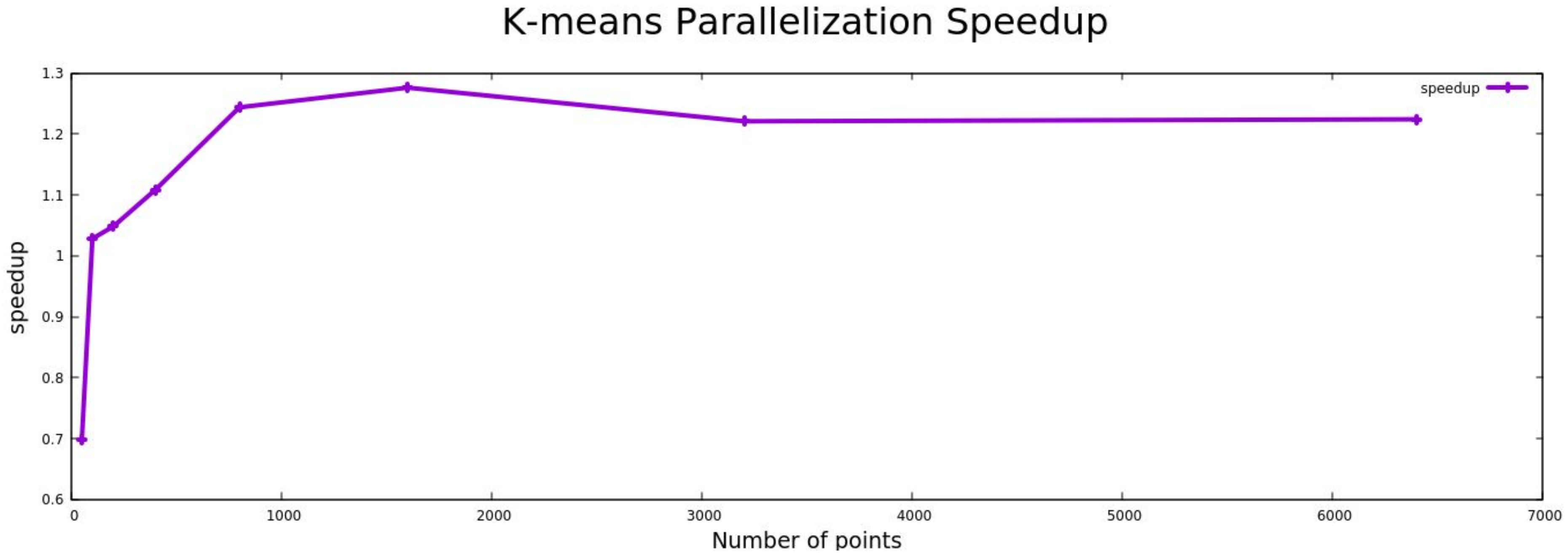
Limitations

Some of these limitations include the number of processors on the underlying machine as having too many tasks will lead to increased communication overheads and thus slow down the parallel algorithm.

As a result of limitations, the optimal number of tasks on the machine used to test the performance of the algorithm was 8 tasks. This optimal number of tasks will change depending on the capabilities of that machine.

Results Analysis


K-means Parallelization Performance
time (micro seconds)
Number of points
Serial
Parallel
0
0.5
1
1.5
2
2.5
0
1000
2000
3000
4000
5000
6000
7000



K-means Parallelization Speedup
speedup
Number of points
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
0
1000
2000
3000
4000
5000
6000
7000

Conclusion

Conclusion

In parallelising the k-means algorithm, various factors were taken into account and it was decomposed using input data partitioning. With this we had multiple tasks running concurrently to classify the data point into different clusters. The performance of the parallel algorithm was then evaluated and compared to the serial version. There was a noticeable improvement in the perform of the parallel version as compared to the serial one. With increasing sizes of input data points the parallel algorithms was faster in classifying the data points. As such we can conclude that the parallel version was able to solve the problem.