COMS3008 - Parallel Computing

Seale Rapolai (1098005)
Mahlekenyane Ts'eole (1118248)
Meriam Elabor (1076589)
Nkosikhona Sibisi (915702)

October 24, 2017

1 Introduction

This report focuses on the common problem of clustering data. It involves classifiying 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. One of the problems that arises in trying to cluster data includes the amount of time required to classify each data item into a certain group (cluster). This report will focus on solving the performance problem related to clustering.

To solve the clustering problem introduced, an algorithm that classifies and groups the data based on their characteristics should be developed. Since the process of clustering is quite time consuming in itself, a proposed solution to this is to parallelise the entire process (i.e the process of clustering the input data). Parallelising this process will allow us to divide the amount of work involved in clustering the data amongst different processing items and hence classify multiple data items concurrently.

There were different algorithms to consider in order to solve the problem. Due to the fact that exclusive clustering is a requirement in our solution to the problem, we will focus on the *K-Means* algorithm, instead of counterparts like *C-Means* and *Hierarchial* clustering algorithms. The *K-Means* works by picking k initial centroids then through iterating will produce *k* exclusive clusters. It is apparent that when the size of the dataset becomes increasingly large, the *k*-means and its clustering process will become increasingly slow. This will lead us to computing sections of the algorithm simultaneously through parallelisation techniques.

2 Solution technicalities

There are various clustering algorithms that could be used to cluster set of data into various clusters. In order to solve the above mentioned problem, the k-means algorithm was chosen to be parallelised since it can be partitioned into various tasks that can be run concurrently relatively easily.

In order to parallelise this algorithm, various factors had to be taken into consideration, such as the dependencies and interactions between the various tasks. From this we will be able to determine the various factors such as the degree of concurrency.

2.1 Task Decomposition

In order 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. Keeping in mind that not everything in the algorithm can be be broken into multiple smaller tasks.

The K-means algorithm takes in a large set of data items and then classifies the data items into k separate clusters. It does this by defining k cluster center at random places and assigning each data item to the closest center. From this, k clusters will be formed. Once this is done, it then takes the average of each cluster, reassigns the center of the cluster as this mean and then repeats the process.

From this, it is clear to see that we can decompose the input data and classify multiple data items in into k clusters simultaneously. This decomposition technique is referred to as input data partitioning. The input data is partitioned across various tasks where each task classifies each data item in its dataset into one of the k clusters.

Furthermore, since the algorithm has to calculate a the averages of each cluster, once all the data items have been classified under the k clusters, and reassign the centers of the clusters as these means, it is clear that this part of that algorithm must be performed in serial.

2.2 Task Dependencies and Interactions

In order to describe the dependencies and interactions between the tasks, a task dependency graph and a task interactions graph will created. Both those graphs will give us useful information about the parallelizability of the algorithm as well as help us determine any overheads or extra computations that we may arise.

2.2.1 Task Dependencies

The task dependency graph will depict the dependencies between the various tasks. Where each note represents a task and each edge (arrow) represents a dependency. The task dependency graph will allow us determine the order in which tasks can be completed. That is, whether all the tasks can be ran concurrently or if certain tasks need to be run before others.

Figure 1 represents the task dependency graph of the algorithm with input data partitioning. The input data is then partitioned into smaller data sets and the items in each dataset are classified under one of the k clusters. Once all the tasks complete classifying the data items into clusters the results are then combined and the new cluster centers are obtained.

2.2.2 Task Interactions

The task interactions graph will depict the interactions between the various tasks. Where each note represents a task and each edge representing a interaction between tasks. An interaction between two tasks will represent communication between the two tasks (i.e. data exchange). This graph will mainly help us determine/calculate any overheads that may arise from that tasks sharing data with each other. This is that cost of communication.

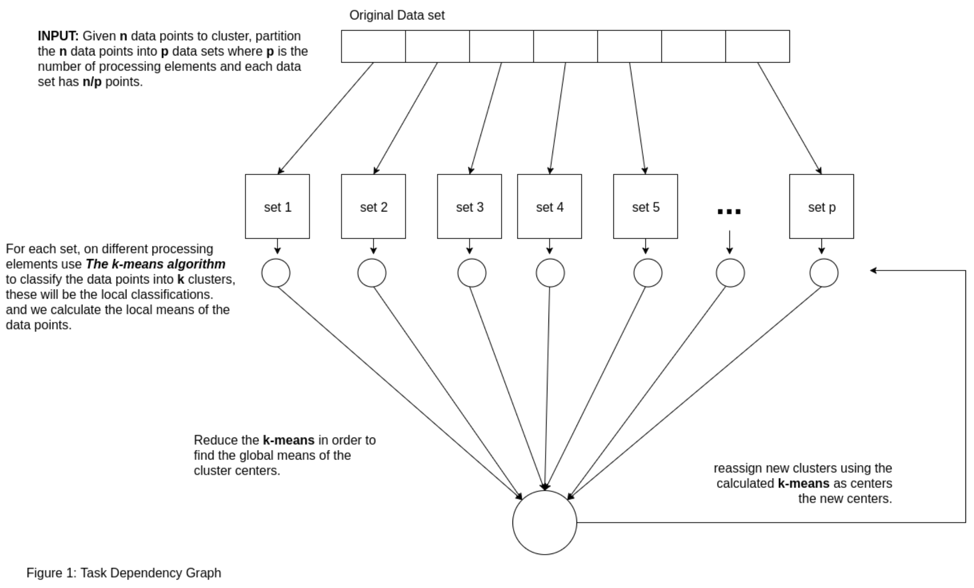
Original Data set
INPUT: Given n data points to cluster, partition the n data points into p data sets where p is the number of processing elements and each data set has n/p points.
set 1
set 2
set 3
set 4
set 5
...
set p
For each set, on different processing elements use The k-means algorithm to classify 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

Figure 1: Task Dependency graph

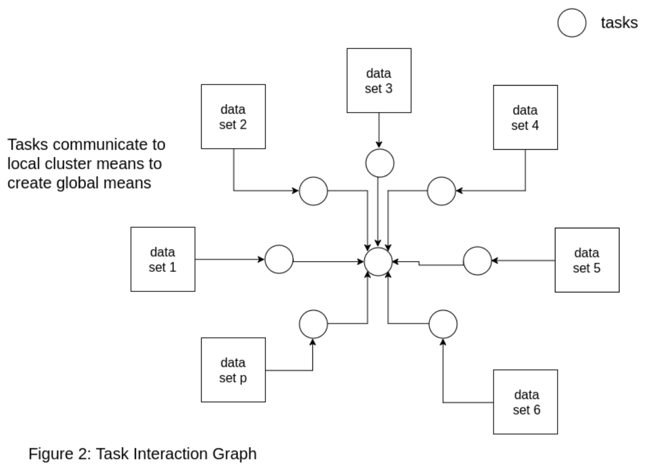
tasks
data set 3
data set 2
data set 4
Tasks communicate to local cluster means to create global means
data set 1
data set 5
data set p
data set 6
Figure 2: Task Interaction Graph

Figure 2: Task Interaction graph

Figure 2 represents the task interaction graph. The graph depicts the data exchange between the tasks. Each task is given a set of data to cluster as can be seen in Figure 2. The tasks also compute the local means of the clusters and once done this data is send to the main process. The main process then computes the global means and sends the data back to the tasks.

2.3 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.

2.4 Limitation

Although the parallelising of the algorithm will enhance the performance of the algorithm, there are certain factors that limit the performance. These limitations restrict various aspects of the parallel algorithm such as the number of tasks that we can partition the algorithm into. 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 this limitation, 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.

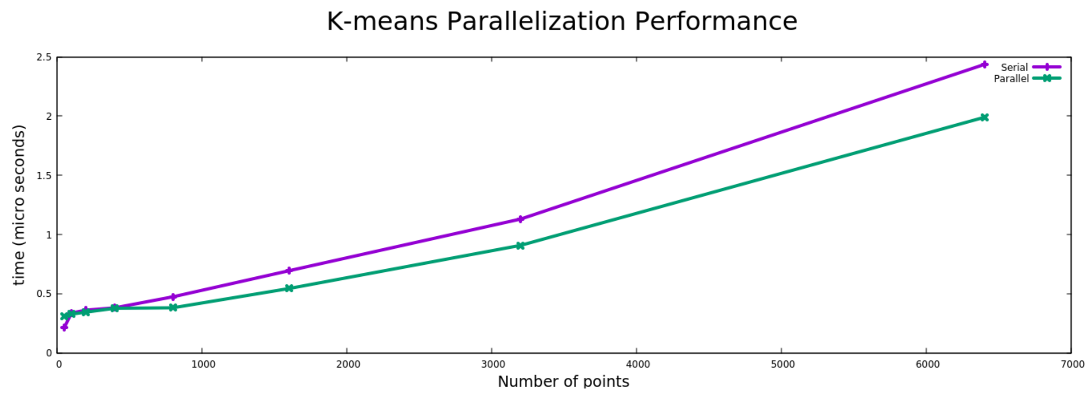

Figure 3: Performance of parallel vs serial

3 Results analysis

Once the algorithm was parallelised, its performance was evaluated and compared to that of the serial version. This evaluation was carried out by comparing the performance of both algorithms with varying input data sizes. This was done in order to determine if the parallelised version has indeed solved the problem as well as to what degree this improvement was.

3.1 Perfomance Analysis

Figure 3 summarises the performance of the parallel algorithm compared to the serial version. For small number of input data items, the serial performs faster, but as the size of the input data increases the parallel algorithms begins to perform faster. From this we can see that for very large input data, the parallel algorithm will perform significantly faster.

3.2 Speedup

From the results obtained from the performance of the parallel algorithm we calculate the speedup of the parallel algorithm with increasing input data. This speedup is the increase in speed of the parallel algorithm as compared to the serial version. This speedup is defined as (Speed of

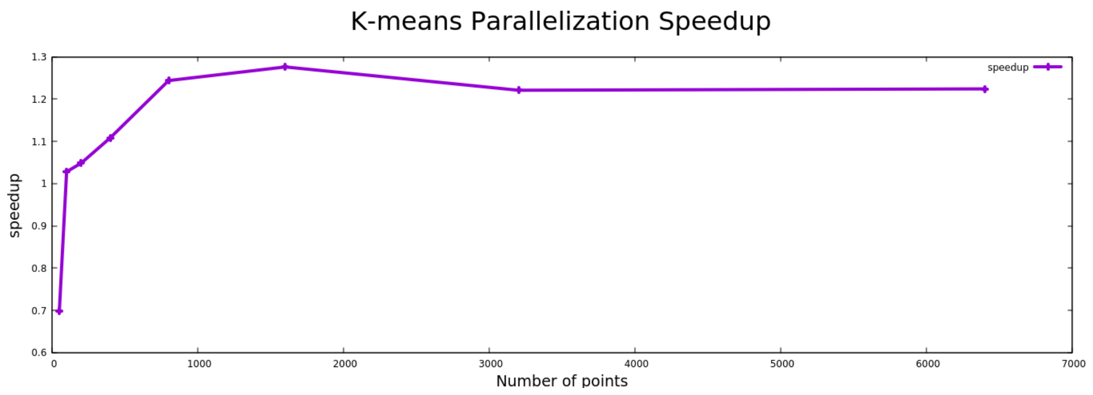

Figure 4: Performance of parallel vs serial

serial)/(speed of parallel). Figure 4 summarises the speedup of the algorithm with increasing input data sizes.

4 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.