Parallel Agglomerative Hierarchical Clustering Using Python

SUPERVISOR: Professor Andriy Miranskyy

PREPARED BY: Syed Mahbub Rafayal, Angelina Amadou

DEPARTMENT: Department of Data Science

COURSE NAME: Designs of Algorithms and Programming for Massive Data - F2017

COURSE CODE: DS 8001

DATE OF SUBMISSION: 20 December, 2017

Background Literature

Clustering is the grouping of a particular set of objects based on their characteristics, aggregating them according to their similarities. It is a statistical technique used on dataset to find pattern and trends that otherwise may be hidden. As one of simple and widely adopted machine learning models, clustering is performed for the diverse purpose such as statistical data analysis and exploratory data mining. The classical clustering algorithms developed in statistics assumed small datasets but with the advent of computing, users wish to apply cluster analysis to large datasets. Old methods have therefore been modified to accommodate large datasets and new methods are being developed (Firdaus & Uddin, 2015)
Data hierarchy consistently occurs in dataset, initial criteria of data hierarchy is not known, therefore, the goal of the clustering process is to determine, given a specified scope of bias, hierarchical binary-like patterns of the bias magnitude in the data (Dhillon & Modha, 2002). With the existence of numerous models such connectivity models, centroid models, distribution models, density model, advance in clustering research areas has been intensified nowadays. The two most prominent clustering methods are the K-means clustering method and the Hierarchical clustering method.

The K-Means concept enables a clustering algorithm by iteratively calculating the Euclidean mean between a data centroid and cluster candidates (Joshi, 2003).

Hierarchical clustering refers to a class of clustering methods that seek to build a hierarchy of clusters, in which some clusters contain others. Hierarchical clustering algorithms are either top-down or bottom-up (agglomerative). Bottom-up algorithms treat every single point as a cluster on its own and then iteratively merge closest pairs of clusters until all clusters have been merged into a single cluster that contains all points.

The hierarchical agglomerative clustering algorithm is usually performed in two phases. In the first phase, a similarity matrix and a nearest neighbour array based on the similarity index are formulated. In the second phase clusters are formed. Cluster formation involves a continual placement data entity in levels of nearest similarity. Considering a sequential computation approach, the worst case time complexity runs as **$O(n^3)$** ((Firdaus & Uddin, 2015)) making it suitable particularly suitable for small dataset. For large datasets, multiples classical clustering techniques have been implemented in a parallel computing environment (Dhillon & Modha, 2002). In particular, shared memory multiprocessor operating systems were developed for scalability (Unrau, Krieger, Gamsa, & Stumm, 1995).

Parallelization of clustering algorithm is challenging as it exhibits inherent data dependency during the hierarchical tree construction(Du & Lin, 2005; Jin et al., 2015).

Unrau (Unrau et al., 1995) argues that a key concept to good performance in non-uniform memory access is to provide concurrency that increases linearly with the number of processors. Furthermore, using standard message passing operation to reduces inter-process communication and maintaining efficient load balancing, a distributed memory parallel version of the average hierarchical agglomerative was developed (Cathey, Jensen, Beitzel, Frieder, & Grossman, 2007).

In this project, a parallelization of the hierarchical agglomerative clustering algorithms is presented using a standard multi-thread process.

The report is organized as follow: in section 2, a brief description of the sequential hierarchical algorithm pseudo code is presented and steps for parallelization are explored. In section 3, results and discussion of different simulation cases are presented. The paper then concludes with a brief summary in section 5.

Methodology

Sequential implementation

The goal of hierarchical clustering is to build a hierarchy of clusters in the dataset, a measure of similarity between points is selected. In single linkage clustering, the distance between two clusters is determined by a single element pair, namely those two elements (one in each cluster) that are closest to each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved (Murtagh & Contreras, 2012) Given a set of N items to be clustered, and a NxN distance (or similarity) matrix, the basic process is giving by the intuitive pseudo-code below (Borgatti, 1994):

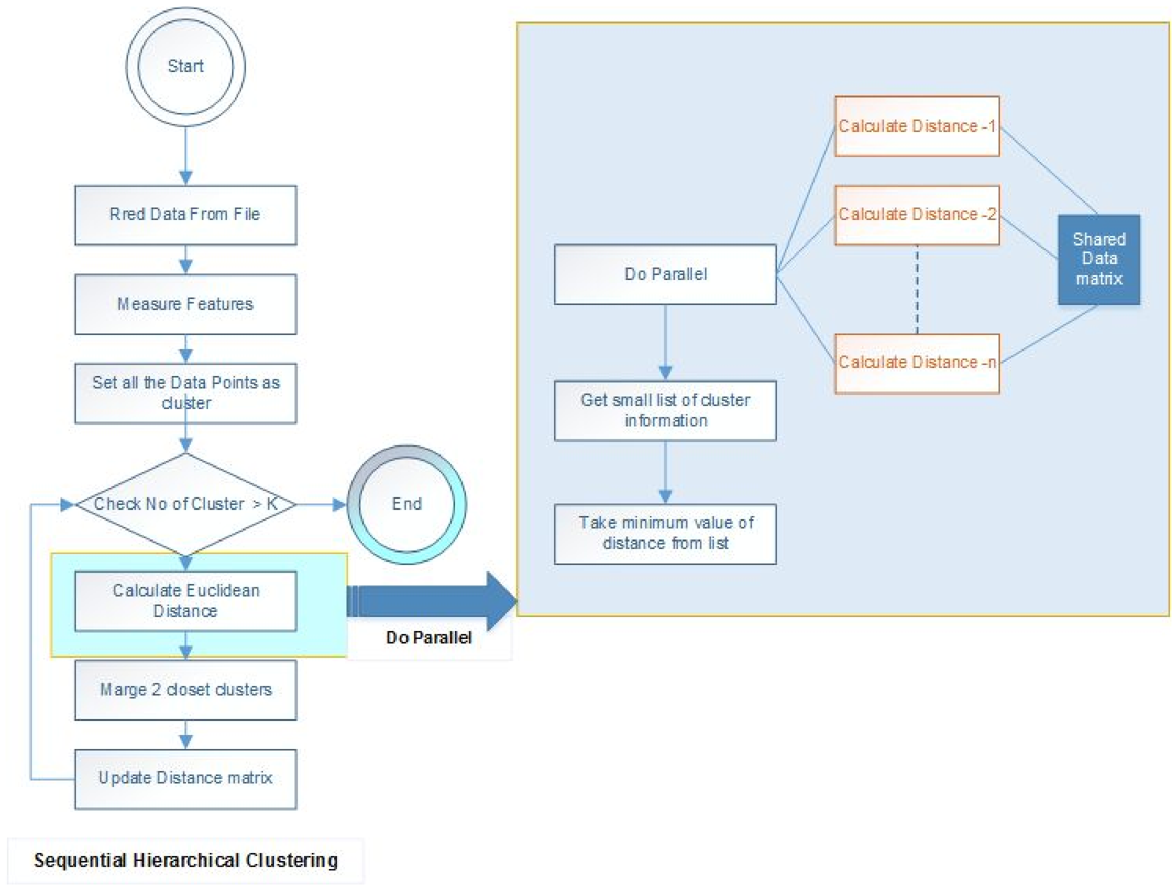
1. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contained.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.

3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Parallel implementation

For parallel computing, a multiprocessing library is used to create and coordinate multiple Python processes. The multiprocessing module aims at providing a simple API for the use of parallelism based on processes. Each one is scheduled independently on the CPU by the Operating System. Initially, the NxN matrix of the dataset is split into k numbers of partitions for each of the available cores (Figure 1). Each partition is processed independently (applying HAC algorithms). The Euclidian distance between neighbor points is calculated within the partitions. A single linkage type is applied and an index of the minimum distances within the partition is broadcast and made available to the remaining datasets. At a later stage, the dataset is merged and shuffle. Finally, the merged set is sorted and global minimum is retrieved (as stated in the HAC algorithm).

Fig.1. Dataflow diagrams for sequential and parallel code.



Results & Discussion

In this study, a sequential and parallel Hierarchical agglomerative clustering is tested on a machine with Intel(R) Core(TM)i7 - 4500U CPU @1.8Ghz, 8GB of RAM memory, 64-bit Windows 8 operating system. The task is implemented in Python (3.5). Two different sizes of “SHOPPING CART” datasets are used for implementation where one dataset has 220 instances and the other has 170 instances, and both dataset have 13 different attributes.

We run both the sequential and parallel Hierarchical agglomerative clustering method against 2 different datasets with different number of clusters (K) to measure the procession time. For the parallel methods, we generated seven parallel processes. In each step, we found that the parallel process time is relatively small compared to sequential process time (Table 1.). As expected, with a large number of clusters, the parallel algorithm outperforms the sequential (Fig. 2 and Fig.3) however as the number of clusters decreases, the run time of the two algorithms are similar on small datasets.

Table 1. Experiment results summary.

No of Cluster	Data Point	Sequential Process Time	Parallel Process Time
6	220	0:00:50.569830	0:00:44.297313
8	220	0:00:44.340709	0:00:40.155840
10	220	0:00:41.970808	0:00:39.038373
20	220	0:00:39.444105	0:00:31.811175
50	220	0:00:35.696559	0:00:19.883030
100	220	0:00:31.271879	0:00:15.356435
6	170	0:00:26.257964	0:00:09.457471
8	170	0:00:25.257964	0:00:09.409167
10	170	0:00:24.975979	0:00:09.332174

20	170	0:00:23.313438	0:00:08.802674
50	170	0:00:18.447078	0:00:07.003462
100	170	0:00:16.062154	0:00:04.299845

Fig 1: Experiment result - 200 Data Points

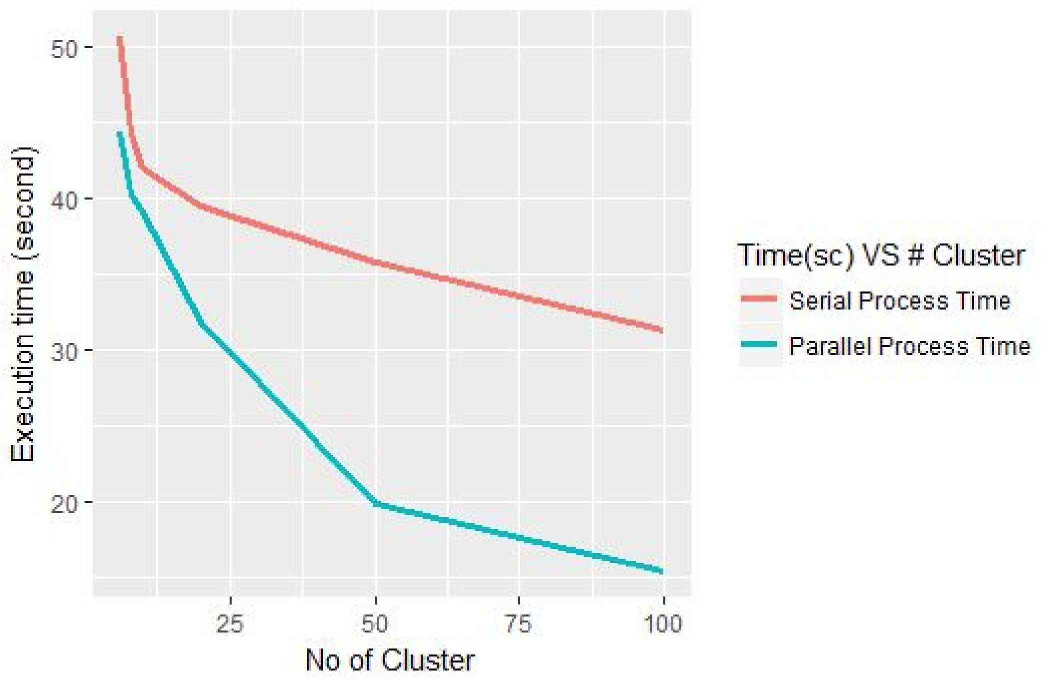
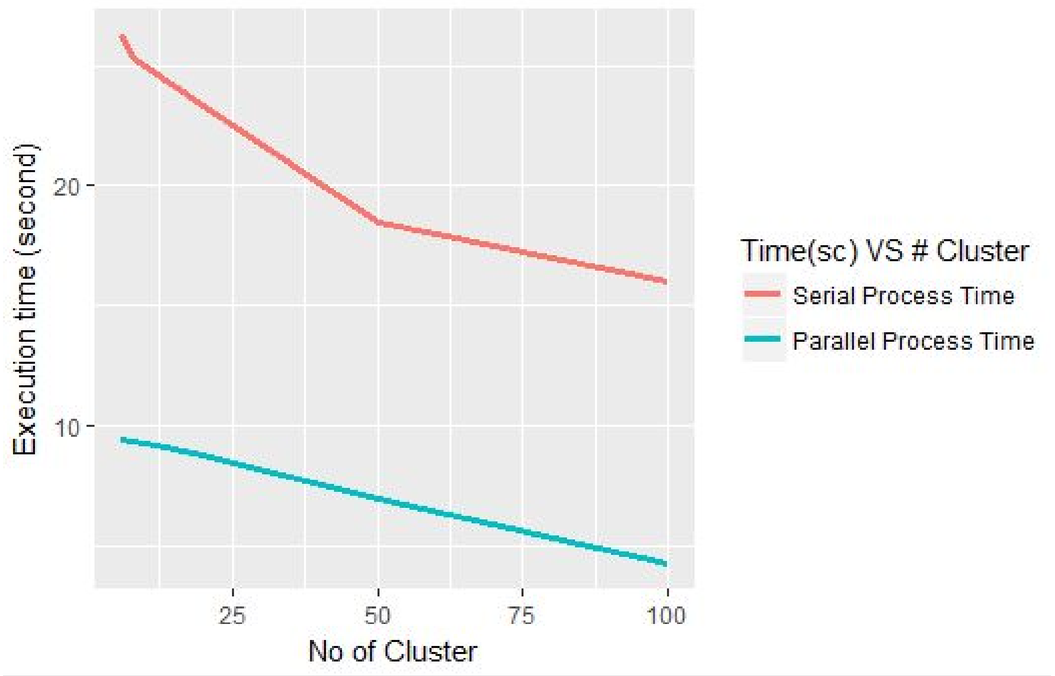


Fig 2: Experiment result -170 Data Points



Conclusion

In this project, a sequential implementation hierarchical agglomerative clustering with a single linkage was evaluated. Using a multi-threaded process, the algorithm was parallelized. The processing run time was recorded. In general, results suggest that the parallel algorithm runs faster than its sequential counterpart for each of the dataset. Therefore, one can expect a better performance on large scale dataset using a parallel framework.

References

Borgatti, S. (1994) http://www.analytictech.com/networks/hiclus.htm

Cathey, R. J., Jensen, E. C., Beitzel, S. M., Frieder, O., & Grossman, D. (2007). Exploiting parallelism to support scalable hierarchical clustering. *Journal of the American Society for Information Science and Technology*, *58*(8), 1207–1221.

Dhillon, I. S., & Modha, D. S. (2002). A Data-Clustering Algorithm on Distributed Memory Multiprocessors. *LargeScale Parallel Data Mining*, *1759*(802), 245–260.

Du, Z., & Lin, F. (2005). A novel parallelization approach for hierarchical clustering. *Parallel Computing*, *31*(5), 523–527.

Firdaus, S., & Uddin, A. (2015). A Survey on Clustering Algorithms and Complexity Analysis. *IJCSI International Journal of Computer Science Issues*, *12*(2), 62–85.

Jin, C., Liu, R., Chen, Z., Hendrix, W., Agrawal, A., & Choudhary, A. (2015). A Scalable Hierarchical Clustering Algorithm Using Spark. *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on*, 418–426.

Joshi, M. N. (2003). Parallel K- Means Algorithm on Distributed Memory Multiprocessors

Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(1), 86–97.

Unrau, R. C., Krieger, O., Gamsa, B., & Stumm, M. (1995). Hierarchical clustering: A structure for scalable multiprocessor operating system design. *The Journal of Supercomputing*, *9*(1–