Big Data Clustering: A Review

Ali Seyed Shirkhorshidi[1], Saeed Aghabozorgi[1], Teh Ying Wah[1] and Tutut Herawan[1,2]

[1]Department of Information Systems
Faculty of Computer Science and Information Technology
University of Malaya
50603 Pantai Valley, Kuala Lumpur, Malaysia
[2]AMCS Research Center, Yogyakarta, Indonesia

shirkhorshidi_ali@siswa.um.edu.my, saeed@um.edu.my,
tehyw@um.edu.my, tutut@um.edu.my

Abstract. Clustering is an essential data mining and tool for analyzing big data. There are difficulties for applying clustering techniques to big data duo to new challenges that are raised with big data. As Big Data is referring to terabytes and petabytes of data and clustering algorithms are come with high computational costs, the question is how to cope with this problem and how to deploy clustering techniques to big data and get the results in a reasonable time. This study is aimed to review the trend and progress of clustering algorithms to cope with big data challenges from very first proposed algorithms until today's novel solutions. The algorithms and the targeted challenges for producing improved clustering algorithms are introduced and analyzed, and afterward the possible future path for more advanced algorithms is illuminated based on today's available technologies and frameworks.

Keywords: Big Data; Clustering; MapReduce, Parallel Clustering.

1 Introduction

After an era of dealing with data collection challenges, nowadays the problem is changed into the question of how to process these huge amounts of data. Scientists and researchers believe that today one of the most important topics in computing science is Big Data. Social networking websites such as Facebook and Twitter have billions of users and they produce hundreds of gigabytes of contents per minute, retail stores continuously collect their customers' data, You Tube has 1 billion unique users which are producing 100 hours of video each an hour and its content ID service scans over 400 years of video every day [1], [2]. To deal with this avalanche of data, it is necessary to use powerful tools for knowledge discovery. Data mining techniques are well-known knowledge discovery tools for this purpose [3]–[9]. Clustering is one of them that is defined as a method in which data are divided into groups in a way that objects in each group share more similarity than with other objects in other groups [1]. Data clustering is a well-known technique in various areas of computer science and related domains. Although data mining can be considered as the main origin of clustering, but it is vastly used in other fields of study such as bio informatics, energy studies, machine learning, networking, pattern recognition and therefore a lot of research works has been done in this area [10]–[13]. From the very beginning researchers were dealing with clustering algorithms in order to handle their complexity and

computational cost and consequently increase scalability and speed. Emersion of big data in recent years added more challenges to this topic which urges more research for clustering algorithms improvement. Before focusing on clustering big data the question which needs to be clarified is how big the big data is. To address this question Bezdek and Hathaway represented a categorization of data sizes which is represented in table 1 [14].

Table 1. Bezdek and Hathaway categorization for big data

			Big data		
Bytes	10^6	10^8	10^{10}	10^{12}	$10^{>12}$
"Size"	Medium	Large	Huge	Monster	Very large

Challenges of big data have root in its five important characteristics [15]:

- **Volume:** The first one is Volume and an example is the unstructured data streaming in form of social media and it rises question such as how to determine the relevance within large data volumes and how to analyze the relevant data to produce valuable information.
- **Velocity:** Data is flooding at very high speed and it has to be dealt with in reasonable time. Responding quickly to data velocity is one of the challenges in big data.
- **Variety:** Another challenging issue is to manage, merge and govern data that comes from different sources with different specifications such as: email, audio, unstructured data, social data, video and etc.
- **Variability:** Inconsistency in data flow is another challenge. For example in social media it could be daily or seasonal peak data loads which makes it harder to deal and manage the data specially when the data is unstructured.
- **Complexity:** Data is coming from different sources and have different structures; consequently it is necessary to connect and correlate relationships and data linkages or you find your data to be out of control quickly.

Traditional clustering techniques cannot cope with this huge amount of data because of their high complexity and computational cost. As an instance, the traditional K-means clustering is NP-hard, even when the number of clusters is k=2. Consequently, scalability is the main challenge for clustering big data.

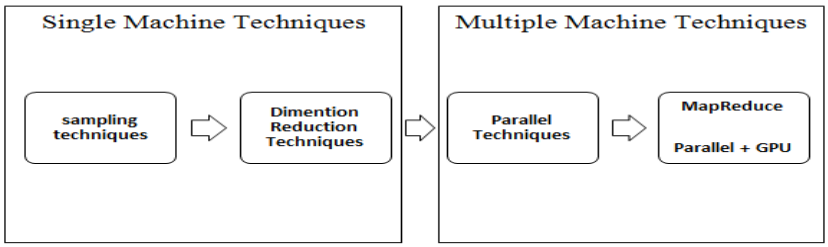

Fig. 1. Progress of developments in clustering algorithms to deal with big data

The main target is to scale up and speed up clustering algorithms with minimum sacrifice to the clustering quality. Although scalability and speed of clustering algorithms were always a target for researchers in this domain, but big data challenges underline these shortcomings and demand more attention and research on this topic. Reviewing the literature of clustering techniques shows that the advancement of these techniques could be classified in five stages as shown in Figure 1.
In rest of this study advantages and drawbacks of algorithms in each stage will be discussed as they appeared in the figure respectively. In conclusion and future works we will represent an additional stage which could be the next stage for big data clustering algorithms based on recent and novel methods.
Techniques that are used to empower clustering algorithms to work with bigger datasets trough improving their scalability and speed can be classified into two main categories:

- Single-machine clustering techniques
- Multiple-machine clustering techniques

Single-machine clustering algorithms run in one machine and can use resources of just one single machine while the multiple-machine clustering techniques can run in several machines and has access to more resources. In the following section algorithms in each of these categories will be reviewed.

2 Big Data Clustering

In general, big data clustering techniques can be classified into two major categories: single-machine clustering techniques and multiple-machine clustering techniques. Recently multiple machine clustering techniques has attracted more attention because they are more flexible in scalability and offer faster response time to the users. As it is demonstrated in **Fig. 2** single-machine and multiple-machine clustering techniques include different techniques:

- Single-machine clustering
    - Sample based techniques
    - Dimension reduction techniques
- Multiple-machine clustering
    - Parallel clustering
    - MapReduce based clustering

In this section advancements of clustering algorithms for big data analysis in categories that are mentioned above will be reviewed.

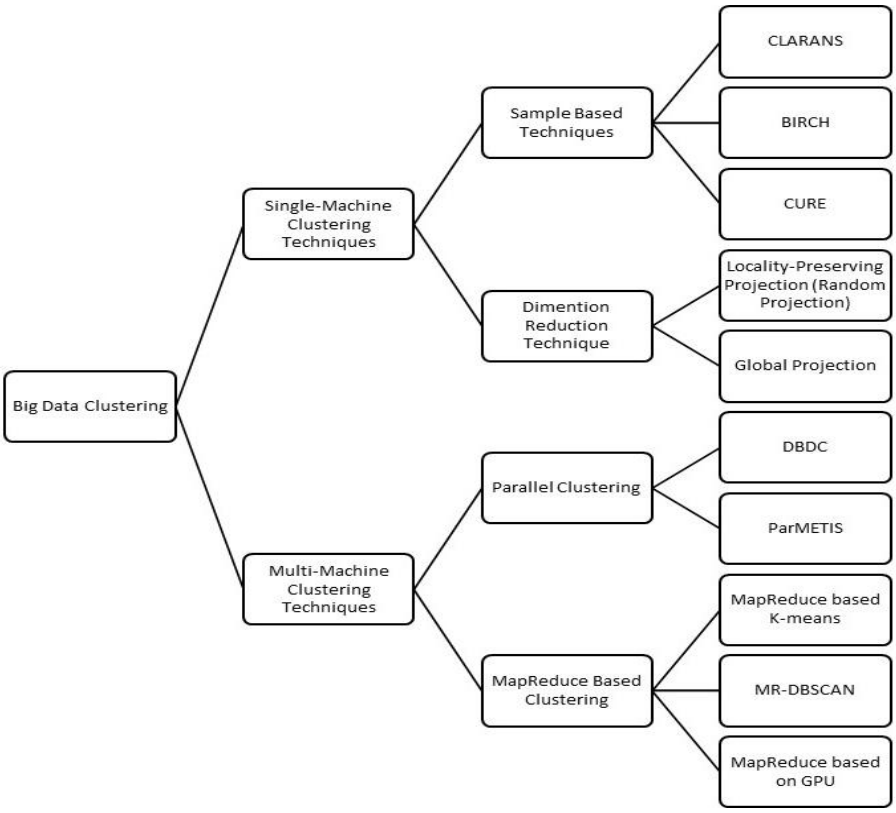

Fig. 2. Big data clustering techniques

2.1 Single-machine clustering techniques

2.1.1 Sampling Based Techniques

These algorithms were the very first attempts to improve speed and scalability of them and their target was dealing with exponential search space. These algorithms are called as sampling based algorithms, because instead of performing clustering on the whole dataset, it performs clustering algorithms on a sample of the datasets and then generalize it to whole dataset. This will speed up the algorithm because computation needs to take place for smaller number of instances and consequently complexity and memory space needed for the process decreases.

Clustering Large Applications based on Randomized Sampling (CLARANS).
Before introducing CLARANS [16] let us take a look at its predecessor Clustering Large Applications (CLARA) [17]. Comparing to partitioning around medoids (PAM) [17], CLARA can deal with larger datasets. CLARA decreases the overall quadratic complexity and time requirements into linear in total number of objects. PAM calculates the entire pair-wise dissimilarity matrix between objects and store it in central memory, consequently it consume O (n^2) memory space, thus PAM cannot

be used for large number of n. To cope with this problem, CLARA does not calculate the entire dissimilarity matrix at a time. PAM and CLARA can be regarded conceptually as graph-searching problems in which each node is a possible clustering solution and the criteria for two nodes for being linked is that they should be differ in 1 out of k medoids.
PAM starts with one of the randomly chosen nodes and greedily moves to one of its neighbors until it cannot find a better neighbor. CLARA reduce the search space by searching only a sub-graph which is prepared by the sampled O (k) data points.
CLARANS is proposed in order to improve efficiency in comparison to CLARA. like PAM , CLARANS, aims to find a local optimal solution by searching the entire graph, but the difference is that in each iteration, it checks only a sample of the neighbors of the current node in the graph. Obviously, CLARA and CLARANS both are using sampling technique to reduce search space, but their difference is in the way that they perform the sampling. Sampling in CLARA is done at the beginning stage and it restricts the whole search process to a particular sub-graph while in CLARANS; the sampling is conduct dynamically for each iteration of the search procedure. Observation results shows that dynamic sampling used in CLARANS is more efficient than the method used in CLARA [18].

BIRCH
If the data size is larger than the memory size, then the I/O cost dominates the computational time. BIRCH [19] is offering a solution to this problem which is not addressed by other previously mentioned algorithms. BIRCH uses its own data structure called clustering feature (CF) and also CF-tree. CF is a concise summary of each cluster. It takes this fact into the consideration that every data point is not equally important for clustering and all the data points cannot be accommodated in main memory.
CF is a triple <N,LS,SS> which contains the number of the data points in the cluster, the linear sum of the data points in the cluster and the square sum of the data points in the cluster, the linear sum of the data points in the cluster and the square sum of the data points in the cluster. Checking if CF satisfies the additive property is easy, if two existing clusters need to be merged, the CF for the merged cluster is the sum of the CFs of the two original clusters. The importance of this feature is for the reason that it allows merging two existing clusters simply without accessing the original dataset.
There are two key phase for BIRCH algorithm. First, it scans the data points and build an in memory tree and the second applies clustering algorithm to cluster the leaf nodes. Experiments conducted in [20] reveal that in terms of time and space, BIRCH performs better than CLARANS and it is also more tolerable in terms of handling outliers. **Fig. 3** represents a flowchart which demonstrates steps in BIRCH algorithm.

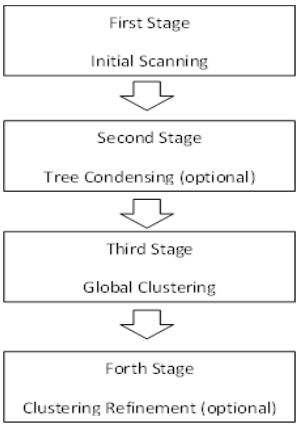

Fig. 3. BIRCH algorithm flowchart

CURE

Single data point is used to represent a cluster in all previously mentioned algorithms which means that these algorithms are working well if clusters have spherical shape, while in the real applications clusters could be from different complex shapes. To deal with this challenge, clustering by using representatives (CURE) [21] uses a set of well-scattered data points to represent a cluster. In fact CURE is a hierarchical algorithm. Basically it considers each data point as a single cluster and continually merges two existing clusters until it reaches to precisely k clusters remaining. The process of selecting two clusters for merging them in each stage is based on calculating minimum distance between all possible pairs of representative points from the two clusters. Two main data structures empower CURE for efficient search. Heap is the first, which is used to track the distance for each existing cluster to its closest cluster and the other is k-d tree which is used to store all the representative points for each cluster.

CURE also uses sampling technique to speed up the computation. It draws a sample of the input dataset and run mentioned procedure on the sample data. To clarify the necessary sample size Chernoff bound is used in the original study. If the dataset is very large, even after sampling, the data size is still big and consequently the process will be time consuming. To solve this issue, CURE uses partitions to accelerate the algorithm. If we consider n as the original dataset and the n' as the sampled data, then CURE will partition the n' into p partitions and within each partition it runs a partial hierarchical clustering until either a predefined number of clusters is reached or the distance between the two clusters to be merged exceeds some threshold. Then another clustering runs and passes on all partial clusters from all the p partitions. At the final stage all non-sampled data points will assign to the nearest clusters. Results [21] represent that in comparison to BIRCH, execution time for CURE is lower while it maintain the privilege of robustness in handling outliers by shrinking the representative points to the centroid of the cluster with a constant factor. **Fig. 4** demonstrate the flowchart of CURE algorithm.

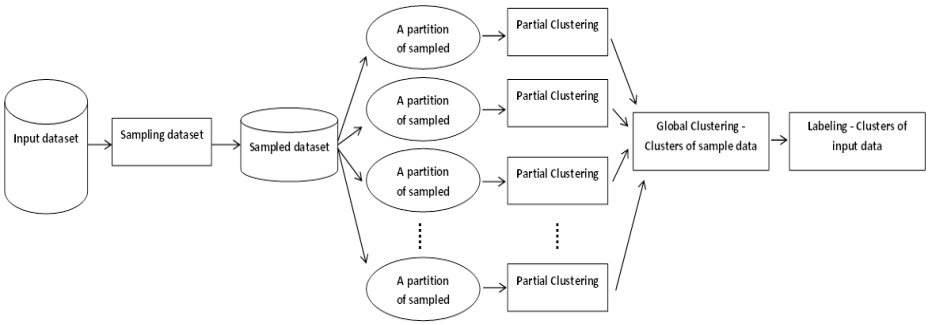

Fig. 4. CURE flowchart

2.1.2 Dimension Reduction Techniques

Although the complexity and speed of clustering algorithms is related to the number of instances in the dataset, but at the other hand dimensionality of the dataset is other influential aspect. In fact the more dimensions data have, the more is complexity and it means the longer execution time. Sampling techniques reduce the dataset size but they do not offer a solution for high dimensional datasets.

Locality-Preserving Projection (Random Projection)
In this method, after projecting the dataset from d-dimensional to a lower dimensional space, it is desired that the pairwise distance to be roughly preserved. Many clustering algorithms are distance based, so it is expected that the clustering result in the projected space provide an acceptable approximation of the same clustering in the original space. Random projection can be accomplished by a linear transformation of the data matrix A, which is the original data matrix. If R be a $d \times t$ and rotation matrix ($t << d$) and its elements $R(i,j)$ be independent random variables, then $A' = A.R$ is the projection of matrix A in t dimensional space. It means each row in A' has t dimensions. Construction of rotation matrix is different in different random projection algorithms. Preliminary methods propose Normal random variables with mean 0 and variance 1 for $R(i,j)$ although there are studies which represent different methods for allocating values to $R(i,j)$ [22]. After generating the projected matrix into lower dimensional space, clustering can be performed. Samples of algorithm implementation is illustrated in [23], [24] and recently a method is proposed to speed up the computation [25].

Global Projection
In global projection the objective is for each data point it is desirable that the projected data point to be as close as possible to original data point while in local preserving projection the objective was to maintain pairwise distances roughly the same between two pairs in projected space. In fact if the main dataset matrix is considered to be A and A' be the approximation of it, in global projection aim is to minimize $||A' - A||$. Different approaches are available to create the approximation matrix such as SVD (singular value decomposition) [26], CX/CUR [27], CMD [28] and Colibri [29].

2.2 Multi-Machine clustering techniques

Although sampling and dimension reduction methods used in single-machine clustering algorithms represented in previous section improves the scalability and speed of the algorithms, but nowadays the growth of data size is way much faster than memory and processor advancements, consequently one machine with a single processor and a memory cannot handle terabytes and petabytes of data and it underlines the need algorithms that can be run on multiple machines. As it is shown in **Fig. 5**, this technique allows to breakdown the huge amount of data into smaller pieces which can be loaded on different machines and then uses processing power of these machines to solve the huge problem. Multi machine clustering algorithms are divided into two main categories:

- Un-automated distributing– parallel
- Automated distributing– MapReduce

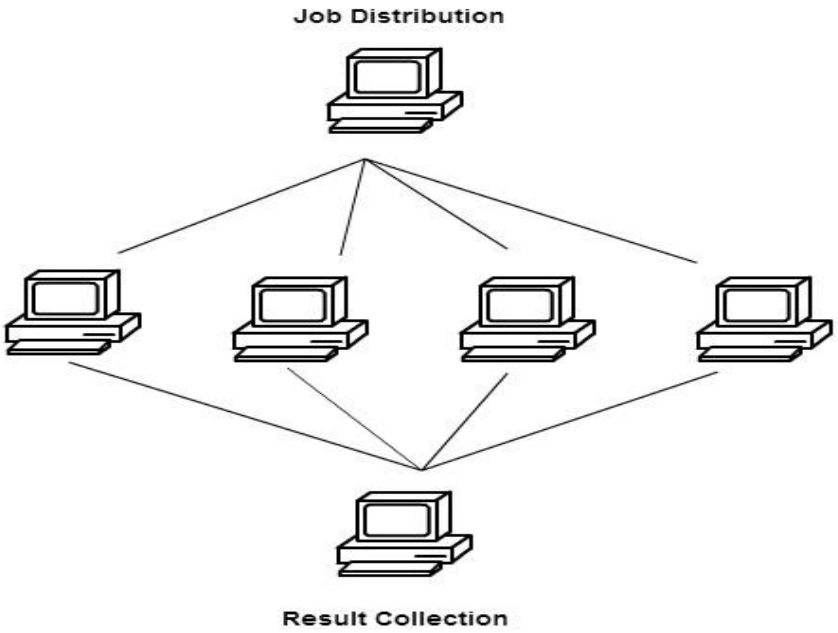

Fig. 5. General concept of multi-machine clustering techniques

In parallel clustering, developers are involved with not just parallel clustering challenges, but also with details in data distribution process between different machines available in the network as well, which makes it very complicated and time consuming. Difference between parallel algorithms and the MapReduce framework is in the comfortless that MapReduce provides for programmers and reveals them form unnecessary networking problems and concepts such as load balancing, data distribution, fault tolerance and etc. by handling them automatically. This feature allows huge parallelism and easier and faster scalability of the parallel system. Parallel and distributed clustering algorithms follows a general cycle as represented below:

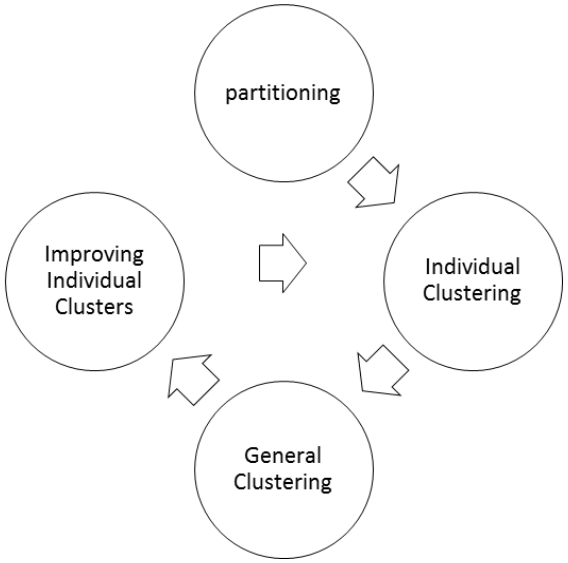

Fig. 6. General cycle for multi-machine clustering algorithms

In the first stage, data is going to be divided into partitions and they distribute over machines. Afterward, each machine performs clustering individually on the assigned partition of data. Two main challenges for parallel and distributed clustering are minimizing data traffic and its lower accuracy in comparison with its serial equivalent. Lower accuracy in distributed algorithms could be caused by two main reasons, first, it is possible that different clustering algorithms deploy in different machines and secondly even if the same clustering algorithm is used in all machines, in some cases the divided data might change the final result of clustering. In the rest of this study, parallel algorithms and MapReduce algorithms will be discussed subsequently, then more advanced algorithms proposed recently for big data will be covered.

2.2.1 Parallel clustering

Although parallel algorithms add difficulty of distribution for programmers, but it is worthfull because of the major improvements in scaling and speed of clustering algorithms. At the following parts some of them will be reviewed.

DBDC

DBDC [30], [31] is a distributed and density based clustering algorithm. Discovery of clusters of arbitrary shapes is the main objective of density based clustering. The density of points in each cluster is much higher than outside of the cluster, while the density of the regions of noise is lower than the density in any of the clusters. DBDC [32] is an algorithm which obeys the cycle mentioned in Figure 2 . At the individual clustering stage, it uses a defined algorithm for clustering and then for general clustering, a single machine density algorithm called DBSCAN is used for finalizing the results.

The results show that although DBDC maintain the same clustering quality in comparison to its serial interpretation, but it runs 30 times faster than that.

ParMETIS

ParMETIS [33] is the parallel version of METIS [34] and is a multilevel partitioning algorithm. Graph partitioning is a clustering problem with the goal of finding the good cluster of vertices. METIS contains three main steps. First step is called as coarsening phase. In this stage maximal matching on the original graph is done and then the vertices which are matched create a smaller graph and this process is iterated till the number of vertices become small enough. The second stage is partitioning stage in which k-way partitioning of the coarsened graph is performed using multilevel recursive bisection algorithm. Finally in third un-coarsening stage, a greedy refinement algorithm is used to project back the partitioning from second stage to the original graph.

ParMETIS is a distributed version of METIS. Because of graph based nature of ParMETIS it is different from clustering operations and it does not follow the general cycle mentioned earlier for parallel and distributed clustering. An equal number of vertices are going to distribute initially, then a coloring of a graph will compute in machines. Afterward, a global graph incrementally matching only vertices of the same color one at a time will be computed. In partitioning stage this graph broadcast to machines and recursive bisection by exploring only a single path of the recursive bisection tree performs in each machine. Finally un-coarsening stage is consisting of moving vertices of edge-cut. Experiments represent that ParMETIS was 14 to 35 times faster than serial algorithm while maintaining the quality close to the serial algorithm.

GPU based parallel clustering

A new issue is opened recently in parallel computing to use processing power of GPU instead of CPU to speed up the computation. G-DBSCAN [35] is a GPU accelerated parallel algorithm for density-based clustering algorithm, DBSCAN. It is one of the recently proposed algorithms in this category. Authors distinguished their method by using a graph based data indexing to add flexibility to their algorithm to allow more parallelization opportunities. G-DBSCAN is a two-step algorithm and both of these steps have been parallelized. The first step constructs a graph. Each object represents a node and an edge is created between two objects if their distance is lower than or equal to a predefined threshold. When this graph is ready, the second step is to identify the clusters. It uses breath first search (BFS) to traverse the graph created in the first step. Results show that in comparison to its serial implementation, G-DBSCAN is 112 times faster.

2.2.2 MapReduce

Although parallel clustering algorithms improved the scalability and speed of clustering algorithms still the complexity of dealing with memory and processor distribution was a quiet important challenge. MapReduce is a framework which is illustrated in **Fig. 7** initially represented by Google and Hadoop is an open source version of it [36]. In this section algorithms which are implemented based on this framework are reviewed and their improvements are discussed in terms of three features:

- **Speed up:** means the ratio of running time while the dataset remains constant and the number of machines in the system is increased.
- **Scale up:** measures if x time larger system can perform x time larger job with the same run time
- **Size up:** keeping the number of machines unchanged, running time grows linearly with the data size

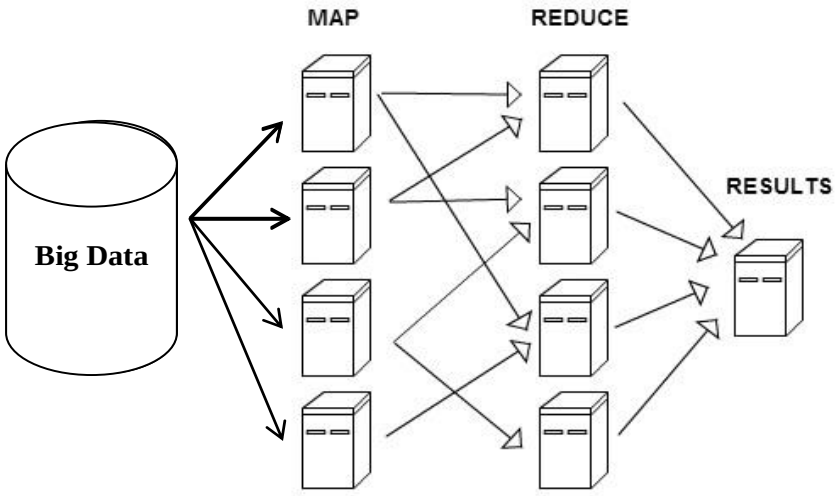

Fig. 7. MapReduce Framework

MapReduce based K-means (PK-means)

PKMeans [37] is distributed version of well-known clustering algorithm K-means [38], [39]. The aim of k-means algorithm is to cluster the desire dataset into k clusters in the way that instances in one cluster share more similarity than the instances of other clusters. K-means clustering randomly choose k instance of the dataset in initial step and performs two phases repeatedly: first it assigns each instance to the nearest cluster and after finishing the assignment for all of the instances in the second phase it updates the centers for each cluster with the mean of the instances.

PKMeans distributes the computation between multiple machines using MapReduce framework to speed up and scale up the process. Individual clustering which contain the first phase happens in the mapper and then general clustering perform second phase in the reducer.

PKMeans has almost linear speed up and also a linear size up. It also has a good scale up. For 4 machines it represented a scale up of 0.75. At the other hand, PKMeans is an exact algorithm, it means that it offer the same clustering quality as its serial counterpart k-means.

MR-DBSCAN

A very recent proposed algorithm is MR-DBSCAN [40] which is a scalable MapReduce-based DBSCAN algorithm. Three major draw backs are existed in parallel DBSCAN algorithms which MR-DBSCAN is fulfilling: first they are not successful to balance the load between the parallel nodes, secondly these algorithms are limited in scalability because all critical sub procedures are not parallelized, and finally their architecture and design limit them to less portability to emerging parallel processing paradigms.
MR-DBSCAN proposes a novel data partitioning method based on computation cost emission as well as a scalable DBSCAN algorithm in which all critical sub-procedures are fully parallelized. Experiments on large datasets confirm the scalability and efficiency of MR-DBSCAN.

MapReduce based on GPU
As it discussed in G-DBSCAN section, GPUs are much more efficient than CPUs. While CPUs have several processing cores GPUs are consisted of thousands of cores which make them much more powerful and faster than CPUs. Although MapReduce with CPUs represents very efficient framework for distributed computing, but if GPUs is used instead, the framework can improve the speed up and scale up for distributed applications. GPMR is a MapReduce framework to use multiple GPUs. Although clustering applications are not still implemented in this framework, but the growth of data size urge researcher to represent faster and more scalable algorithms so maybe this framework could be the appropriate solution to fulfill those needs.

3 Conclusion and Future Works

Clustering is one of the essential tasks in data mining and they need improvement nowadays more than before to assist data analysts to extract knowledge from terabytes and petabytes of data. In this study the improvement trend of data clustering algorithms were discussed. to sum up, while traditional sampling and dimension reduction algorithms still are useful, but they don't have enough power to deal with huge amount of data because even after sampling a petabyte of data, it is still very big and it cannot be clustered by clustering algorithms, consequently the future of clustering is tied with distributed computing. Although parallel clustering is potentially very useful for clustering, but the complexity of implementing such algorithms is a challenge. At the other hand, MapReduce framework provides a very satisfying base for implementing clustering algorithms. As results shows, MapReduce based algorithms offer impressive scalability and speed in comparison to serial counterparts while they are maintaining same quality. Regarding to the fact that GPUs are much powerful than CPUs as a future work, it is considerable to deploy clustering algorithms on GPU based MapReduce frameworks in order to achieve even better scalability and speed.

Acknowledgments. This work is supported by University of Malaya High Impact Research Grant no vote UM.C/625/HIR/MOHE/SC/13/2 from Ministry of Education Malaysia.

References

1. T. C. Havens, J. C. Bezdek, and M. Palaniswami, "Scalable single linkage hierarchical clustering for big data," in *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on. IEEE*, 2013, pp. 396–401.
2. "YouTube Statistic," 2014. [Online]. Available: http://www.youtube.com/yt/press/statistics.html.
3. P. Williams, C. Soares, and and J. E. Gilbert, "A Clustering Rule Based Approach for Classifi-cation Problems," *Int. J. Data Warehous. Min.*, vol. 8, no. 1, pp. 1–23, 2012.
4. R. V. Priya and A. Vadivel, "User Behaviour Pattern Mining from Weblog," *Int. J. Data Warehous. Min.*, vol. 8, no. 2, pp. 1–22, 2012.
5. T. Kwok, K. A. Smith, S. Lozano, and D. Taniar, "No Title," in *Parallel Fuzzy c-Means Clustering for Large Data Sets*, 2002, pp. 365–374.
6. H. Kalia, S. Dehuri, and A. Ghosh, "A Survey on Fuzzy Association Rule Mining," *Int. J. Data Warehous. Min.*, vol. 9, no. 1, pp. 1–27, 2013.
7. O. Daly and D. Taniar, "Exception Rules Mining Based on Negative Association Rules," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2004)*, 2004, pp. 543–552.
8. M. Z. Ashrafi, D. Taniar, and K. A. Smith, "Redundant association rules reduction techniques," *Int. J. Bus. Intell. Data Min.*, vol. 2, no. 1, pp. 29–63, 2007.
9. D. Taniar, W. Rahayu, V. C. S. Lee, and O. Daly, "Exception rules in association rule mining," *Appl. Math. Comput.*, vol. 205, no. 2, pp. 735–750, 2008.
10. Meyer, F. G., and J. Chinrungrueng., "Spatiotemporal clustering of fMRI time series in the spectral domain," *Med. Image Anal.*, vol. 9, no. 1, pp. 51–68, 2004.
11. J. Ernst, G. J. Nau, and Z. Bar-Joseph, "Clustering short time series gene expression data.," *Bioinforma. 21*, vol. 21, no. suppl 1, pp. i159 – i168, Jun. 2005.
12. F. Iglesias and W. Kastner, "Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns," *Energies*, vol. 6, no. 2, pp. 579–597, Jan. 2013.
13. Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Mach. Learn.*, vol. 55, no. 3, pp. 311–331, 2004.
14. R. Hathaway and J. Bezdek, "Extending fuzzy and probabilistic clustering to very large data sets," *Comput. Stat. Data Anal.*, vol. 51, no. 1, pp. 215–234, 2006.
15. "Big Data, What is it and why it is important." [Online]. Available: http://www.sas.com/en_us/insights/big-data/what-is-big-data.html.
16. R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.
17. L. Kaufman and peter J. Rousseeuw, *Finding Groups in Data: An Introduction on Cluster Analysis*. John Wiley and Sons, 1990.
18. R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, 2002.
19. T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large database," in *SIGMOD Conference*, 1996, pp. 103–114.
20. T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large database," in *SIGMOD Conference*, 1996, pp. 103–114.
21. S. Guha and R. Rastogi, "CURE: An efficient clustering algorithm for large database," *Inf. Syst.*, vol. 26, no. 1, pp. 35–58, 2001.
22. D. Achlioptas and F. McSherry, "Fast computation of low rank matrix approximations," *J. C=ACM*, vol. 54, no. 2, p. 9, 2007.

23. X. Z. Fern and C. E. Brodley, “Random projection for high dimensional data clustering: A cluster ensemble approach,” in *ICML*, 2003, pp. 186–193.
24. S. Dasgupta, “Experiments with random projection,” *UAI*. pp. 143–151, 2000.
25. C. Boutsidis, C. Chekuri, T. Feder, and R. Motwani, “Random projections for k-means clustering,” in *NIPS*, 2010, pp. 298–306.
26. G. H. Golub and C. F. Van-Loan, *Matrix computations*, 2nd ed. The Johns Hopkins University Press, 1989.
27. P. Drineas, R. Kannan, and M. W. Mahony, “Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition,” *SIAM J. Comput.*, vol. 36, no. 1, pp. 132–157, 2006.
28. J. Sun, Y. Xie, H. Zhang, and C. Faloutsos, “Less is More: Compact Matrix Decomposition for Large Sparse Graphs.,” in *SDM*, 2007.
29. H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos, “Colibri: Fast mining of large static and dynamic graphs,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 686–694.
30. E. Januzaj, hans P. Kriegel, and M. Pfeifle, “DBDC: Density based distributed clustering,” in *Advances in Database Technology-EDBT 2004.*, 2004, pp. 88–105.
31. Aggarwal, C. C., C. K. Reddy, and Eds, *Data Clustering: Algorithms and Applications*. 2013.
32. M. Ester, hans P. Kriegel, J. Sander, and X. Xui, “A density-based algorithm for discovering clusters in large spatial database with noise,” in *KDD*, 1996, pp. 226–231.
33. G. Karypis and V. Kumar, “Parallel multilevel k-way partitioning for irregular graphs,” *SIAM Rev.*, vol. 41, no. 2, pp. 278–300, 1999.
34. G. Karypis and V. Kumar, “Multilevel k-way partitining scheme for irregular graphs,” *J. Parallel Disteributed Comput.*, vol. 48, no. 1, pp. 96–129, 1998.
35. G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, and L. Rocha, “G-DBSCAN: A GPU Accelerated Algorithm for Density-based Clustering,” *Procedia Comput. Sci.*, vol. 18, pp. 369–378, 2013.
36. P. P. Anchalia, A. K. Koundinya, and S. NK., “MapReduce Design of K-Means Clustering Algorithm,” in *Information Science and Applications (ICISA), 2013 International Conference on*, 2013, pp. 1–5.
37. W. Zhao, H. Ma, and Q. He, “parallel k-means clustering based on MapReduce,” in *Cloud Computing*, 2009, pp. 674–679.
38. J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
39. B. Mirkin, *Clustering for data mining a data recovery approach*. CRC Press, 2012.
40. Y. He, H. Tan, W. Luo, S. Feng, and J. Fan, “MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data,” *Front. Comput. Sci.*, vol. 8, no. 1, pp. 83–99, 2014.