Clustering in Big Data

Min Chen[1], Simone A. Ludwig[2] and Keqin Li[1]

[1]Department of Computer Science, State University of New York, New Paltz, NY

[2]Department of Computer Science, North Dakota State University, Fargo, ND

Abstract

The need to understand large, complex, information rich data sets is common to all fields of studies in this current information age. Given this tremendous amount of data, efficient and effective tools need to be present to analyze and reveal valuable knowledge that is hidden within the data. Clustering analysis is one of the popular approaches in data mining and has been widely used in big data analysis. The goal of clustering involves the task of dividing data points into homogeneous groups such that the data points in the same group are as similar as possible and data points in different groups are as dissimilar as possible. The importance of clustering is documented in pattern recognition, machine learning, image analysis, information retrieval, etc. Due to the difficulties of parallelization of the clustering algorithms and the inefficiency at large scales, challenges for applying clustering techniques in big data has arisen. The question is how to deploy clustering algorithms for this tremendous amount of data to get the clustering result within a reasonable time. This chapter provides an overview of the mainstream clustering techniques proposed over the past decade and the trend and progress of clustering algorithms applied in big data. Moreover, the improvement of clustering algorithms in big data are introduced and analyzed. The possible future for more advanced clustering techniques are illuminated based on today's information era.

1.1 Introduction

1.1.1 Application background

An overwhelming flow of data in structured, unstructured or heterogeneous format has been accumulated due to the continuous increase in the volume and detail of data captured by organizations, such as social media, government, industry and science. These massive quantities of data are produced because of the growth of the web, the rise of social media, the use of mobile, and the information of Internet of Things (IoT) by and about people, things, and their interactions. The big data era has arrived. Big data becomes the most influential force in daily life. According to the IDC reports, the digital universe is doubling in size every two years and it will reach 44 zettabytes by 2020 [1].

How to store huge amounts of data is not the biggest problem anymore. But how to design solutions to understand this big amount of data is a major challenge. Operations such as analytical operations, process operations, retrieval operations, are very difficult and hugely time consuming because of this massive volume of data. One solution to overcome these problems is the use of data mining techniques in discovering knowledge from big data. Data mining [2] is called exploratory data analysis, among other things. It is an analytic process designed to explore data. Data mining aims to search for consistent patterns or systematic relationships between variables. It then validates the findings by applying the detected patterns to new subsets of data. Although the hidden patterns are derived from heterogeneous data in big data mining, these hidden patterns can still be reviewed as structured knowledge. The structured knowledge is combined with human knowledge of decision makers that are heterogeneous or unstructured and upgraded into intelligent knowledge [3].

1.1.2 Big data challenge

The term of big data [4] that refers to large database has a comprehensive definition through the 3Vs of big data: volume, variety, and velocity. Garter [5] extended this 3Vs model to a 4Vs model by including a new "V": value. More recently, this model has been updated to 5Vs [6]: volume, velocity, variety, veracity and value. Where *volume* refers to the vast amounts of data generated every second. *Velocity* refers to the speed at which new data is generated and the speed at which data moves around. *Variety* refers to the different types of data, such as structured, unstructured and heterogeneous data, and different sources of data that can be used. *Veracity* refers to the messiness or trustworthiness of the data. Quality and accuracy are less controllable with different forms of big data. *Value* refers to how the data can be turned into value. It is the process of discovering hidden values of big data sets, which then documents and socializes as realized values. The challenges of big data have been arisen because of its 5Vs characteristics [7]:

Volume: masses of data has been produced by the increased use of emails, twitter messages, photos, video clips and sensor data. This massive data is too large to store and is hard to analyze using traditional database technology. The challenge is how to determine the relevance within large data volumes and how to extract valuable information from the relevant data.

Velocity: another change of big data is how to respond quickly to data and deal with it in a reasonable time. Techniques on analyzing the data while it is being generated without ever putting it into databases is urgent in the field of big data study.

Variety: data comes from different sources with different specifications such as Twitter, Facebook, LinkedIn and instant messaging in complex and heterogeneous format. Another challenging issue is how to manage, merge and govern the different forms of data.

Complexity: quality and accuracy are less controllable due to the different sources and different structures of data. It becomes really complicated to connect to and associate from heterogeneous data in order to extract useful information and thus improvements on exploiting this huge amount of data are very wide and dispersed. Data complexity increases with the increase in volume. Traditional data management with relational database tools are no longer sufficient to meet the requirements to capture, store and further analyze big data.

1.1.3 Big data clustering analysis

The speed of information growth exceeds the Moore's Law at the beginning of this new century. Given this tremendous amount of data, efficient and effective tools need to be present to analyze and reveal valuable knowledge that is hidden within the data. Clustering is one of the popular approaches in data mining and has been widely used in big data analysis. The goal of clustering involves the task of dividing data points into homogeneous groups such that the data points in the same group are as similar as possible and data points in different groups are as dissimilar as possible.

However, conventional clustering techniques cannot cope with this huge amount of data because of their high complexity and computational cost [8]. The question for big data clustering is how to scale up and speed up clustering algorithms with minimum sacrifice to the clustering quality. Therefore, an efficient processing model with a reasonable computational cost of this huge, complex, dynamic and heterogeneous data is needed in order to exploit this huge amount of data. Single machine clustering techniques and multiple machine clustering techniques are two most popular big data clustering techniques. Single-machine clustering algorithms run in one machine and can use resources of just one single machine while the multi-machine clustering [8] techniques can run in several machines with access to more resources. Multi-machine clustering techniques have become more popular due to the better scalability and faster user response time.

1.1.4 Chapter organization

The rest of the chapter is organized as follows: the main stream clustering algorithms and key technologies for clustering in big data are introduced in Chapter 1.2. The instances of clustering techniques which have been used in single-machine clustering and multi-machine clustering are illustrated in Chapter 1.3. The applications of big data clustering include image segmentation, load balancing in parallel computing, genetic mapping and community detention are discussed in Chapter 1.4. Finally, the chapter is concluded in Chapter 1.5.

1.2 Overview of Clustering Techniques in Big Data

1.2.1 General Information of Clustering Analysis

Clustering is one of the most fundamental tasks in exploratory data analysis that groups similar data points in an unsupervised process. Clustering techniques have been exploited in many fields including in many areas, such as data mining, pattern recognition, machine learning, biochemistry and bioinformatics [9]. The main process of clustering algorithms is to divide a set of unlabeled data objects into different groups. The cluster membership measure is based on a similarity measure. In order to obtain a high quality partition, the similarity measure between the data objects in the same group is to be maximized, and the similarity measure between the data objects from different groups is to be minimized. Most of the clustering task uses an iterative process to find locally or globally optimal solutions from a high-dimensional data sets. In addition, there is no unique clustering solution for real-life data and it is also hard to interpret the 'cluster' representations [9]. Therefore, the clustering task requires much experimentation with different algorithms or with different features of the same data set. Hence, how to save computational complexity is a significant issue for the clustering algorithms. Moreover, clustering very large data sets that contain large numbers of records with high dimensions is considered a very important issue nowadays. Most conventional clustering algorithms suffer from the problem that they do not scale with larger sizes of data sets, and most of them are computationally expensive with regards to memory space and time complexities. For these reasons, the parallelization of clustering algorithms is a solution to overcome the aforementioned problems, and the parallel implementation of clustering algorithms is inevitable.

More importantly, clustering analysis is distinguished from other analysis [9]. Clustering analysis belongs to the so called unsupervised learning category. The main goal is to divide a set of unlabeled data sets into several groups based on the conceptual or hidden properties of the input data sets. In other words, clustering analysis is unsupervised 'nonpredictive' learning. It divides the data sets into several clusters based on a subjective measurement. Clustering analysis is unlike supervised learning and it is not based on a 'trained characterization'. In general, there is a set of desirable features for a clustering algorithm [9,10]: *scalability***,** the temporal and spatial complexity of the algorithm should not explode on large data sets. *Robustness*, the outliers in the data set should be detected during the process. *Order insensitivity,* the ordering of the input data should not affect the outcome of the algorithm. *Minimum user-specified input,* the number of user-specified parameters should be minimized. *Arbitrary-shaped clusters,* the clusters can be shaped arbitrarily. *Point proportion admissibility,* different clustering algorithms produce different results with different features. Hence, a clustering algorithm should be chosen such that duplicating the data set and the re-clustering task should not change the clustering results.

1.2.2 Mainstream Clustering Algorithms

Depending on the data properties or the purpose of clustering, different types of clustering algorithms have been developed [10]:

Partitioning clustering requires a fixed number of clusters to be specified a priori. Objective functions such as the square error function are used as a criterion in the optimization process of data partitioning. Partitioning clustering uses an iterative process to optimize the cluster centers, as well as the number of clusters.

Hierarchical clustering does not specify the number of clusters, and the output is independent of the initial condition. However, the hierarchical clustering is static, i.e., the data points assigned to a cluster cannot be reassigned to another cluster. In addition, it will fail to separate overlapping clusters due to the lack of information regarding the global shape or size of the clusters.

Density-based clustering separates data objects based on their regions of density, connectivity and boundary. The clusters are defined as connected dense component which can grow in any direction the density leads to. Density-based clustering is good for discovering clusters of arbitrary shapes and it can provide a natural protection against outliers.

Grid-based clustering divides the space of data objects into grids. It is capable to go through the dataset when computing the statistical values for the grids with a fast processing time. However, the performance of grid-based clustering depends on the size of the grid, it is insufficient to obtain the required clustering quality for highly irregular data distributions.

Model-based clustering assumes that the data is generated by a mixture of underlying probability distributions. It is a method which is used to optimize the fit between the given data and the predefined mathematical model. One advantage of model-based clustering is that it can automatically determine the number of clusters based on standard statistics.

Evolutionary clustering approaches use genetic algorithm, particle swarm optimization and other evolutionary approach for clustering task. For example, genetic algorithm uses evolutionary operators (such as selection, crossover and mutation) and a population to obtain the optimal partition of the input data. Evolutionary approaches are stochastic and use an iterative process. These algorithms start with a random population of solutions which is a valid partition of data with a fitness value. During the iterative step, the evolutionary operators are applied to generate the new population. A population's likelihood of surviving into the next iteration is determined by a fitness function. The iterative step is repeated until it finds the required solution meeting some stop criteria.

1.2.3 Key Technologies for clustering in big data

Unlike traditional clustering algorithms, volume of data must be taken into account when clustering big data because this requires substantial changes in the architecture of storage system. In addition, most of the traditional clustering algorithms are designed to handle either numeric or categorical data with limited size. On the other hand, big data clustering deals with different types of data such as image, video, sensors, mobile devices, text etc. [11, 12]. Moreover, the velocity of big data requires the big data clustering techniques to have a high demand for online processing of data. At the high level of the five category clustering algorithms, most of the algorithms have a similar procedure. These algorithms start with some random initialization and follow some iterative process until some convergence criteria are met. For example, partitioned clustering such as the k-means algorithm, starts with randomly chosen k centroids and reassigns each data point to the closest cluster centroids in an iterative process. Thus, the issue of big data clustering is how to speed up and scale up the clustering algorithms with the minimum sacrifice to the clustering quality. There are three ways to speed up and scale up big data clustering algorithms.

The first way is to reduce the iterative process using sampling-based algorithms. Sampling-based algorithms perform the clustering based on a sample of the datasets instead of using on the whole dataset. Complexity and memory space needed for the process decreases in sampling-based algorithms because computation only needs to take place for smaller sample dataset. PAM, CLARA and CLARANS [9-12] are proposed to fight with the exponential search space in the K-medoid clustering problem.

The second way is to reduce the data dimension using randomized techniques. Dimensionality of the dataset is another aspect which influences the complexity and speed of clustering algorithms. Random projection and global projection are used to project dataset from a high dimensional space to a lower dimensional space [8, 12]. CX/CUR, CMD and Colibri [8, 12] are dimension reduction techniques which are proposed to reduce long execution times of big data clustering.

The last way is to apply parallel and distributed algorithms use multiple machines to speed up the computation in order to increase the scalability. Parallel processing applications include conventional parallel applications and data-intensive applications. The conventional parallel applications assume that data can be fit into the memory of distributed machines. Data intensive applications are I/O bound and devote the largest fraction of execution time to the movement of data. OpenMP, MPI [13], and MapReduce are common parallel processing models for computing data-intensive applications.

1.3 Instances of Clustering Techniques in Big Data

In this section, we will list some clustering techniques which are designed for large-scale data sets. There are mainly two types of techniques based on the number of computer nodes that have been used: single-machine techniques and multi-machine techniques. Due to the nature of scalability and faster response time to the users, multi-machine clustering techniques have attracted more attention. A list of common big data clustering techniques is shown in Figure 1.

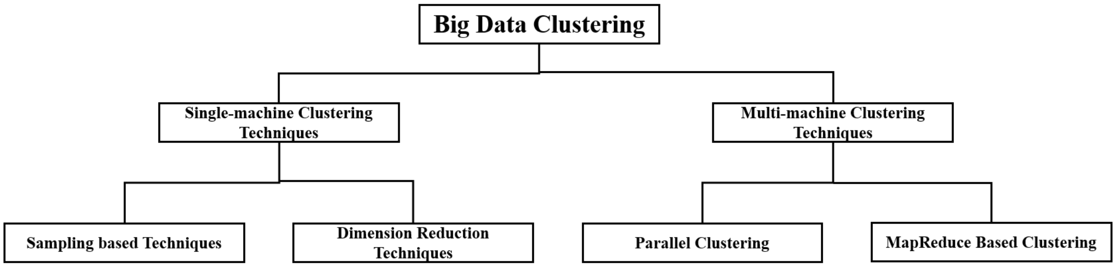
Big Data Clustering
Single-machine Clustering Techniques
Multi-machine Clustering Techniques
Sampling based Techniques
Dimension Reduction Techniques
Parallel Clustering
MapReduce Based Clustering

Figure 1: A list of big data clustering techniques.

1.3.1 Single-machine clustering techniques

Single-machine clustering techniques run in one machine and the data has been store in one machine. Sampling based techniques and dimension reduction techniques are two common strategies for single-machine techniques.

1.3.1.1 Sampling Based Techniques

The problem of scalability in clustering algorithms is based on computing time and memory requirements. Sampling based algorithms handle one sample of the dataset at a time, and then generalize it to whole dataset. Most of these algorithms are partitioning based algorithms.

CLARANS is an example of the K-medoids method which has been extended to spatial very large database (VLDB) [9-12]. Before CLARANS, PAM (Partitioning around medoids) and CLARA were the two early version of k-medoid method. PAM was one of the first k-medoids algorithms introduced. It is more robust than k-means in presence of noise and outlier detection. PAM combines relocation of points between perspective clusters with re-nominating the points as potential medoids in an iterative process. The objective function is adopted to guide the process. The space complexity of PAM is O(n^2) because it needs to store the entire pair-wise dissimilarity matrix between objects in the central memory. Hence, PAM becomes impractical in large datasets. Unlike PAM, CLARA is proposed to avoid this problem. CLARA uses five samples and each sample has O(k) points. PAM will be applied to each sample and CLARA retains the best medroids using the objective function. The whole data set is assigned to resulting medoids for final partition. Both space complexity and time complexity are linear, not quadratic. CLARANS is proposed to improve the efficiency in comparison to CLARA. CLARANS use random search to generate neighbors by starting with a set of medroids. If a neighbor represents a better partition, the process continues with the same set of medoids. Otherwise, the algorithms restarts until a local minimum is found. The best medoids are returned for the formation of a resulting partition. The time complexity of CLARANS is O(n^2).

1.3.1.2 Dimension Reduction Techniques

The complexity and speed of a clustering algorithm are influenced by the number of instances in the dataset. However, objects in data mining could consist of hundreds of attributes. High dimensionality of the data set is another influential aspect and clustering in such high dimensional spaces requires more complexity and longer execution time [8, 12]. One approach to dimension reduction is projection. A dataset can be projected from a high dimensional space to a lower dimensional space. PCA (Principal Component Analysis) is one method used to reduce the dimensionality of a data set [14]. It provides simpler representation of data, reduction in memory and faster execution time. One approach to dimension reduction is subspace clustering. Subspace clustering seeks to find clusters in different subspaces within a dataset. Many dimensions are irrelevant in high dimensional data. Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple subspaces. CLIQUE is one of the algorithms proposed to find clusters within subspaces of a dataset [15]. CLIQUE combines density and grid based clustering. CLIQUE operates on high-dimensional data by not operating all the dimensions at once but by processing a single dimension at the first step and then grows upward to the higher one. The last way to address the problem of dimension reduction is co-clustering. Traditional clustering focuses on only features while co-clustering focus on both data points and features. The Dual Regularized Co-Clustering (DRCC) [16] method based on seminonnegative matrix tri-factorization is a co-clustering algorithm. It generates a data graph and a feature graph to explore the geometric structure of data manifold and feature manifold.

1.3.2 Multiple machine clustering techniques

In this age of data explosion, parallel processing is essential to processing a massive volume of data in a timely manner. Due to the growth of data size and memory and processor advancements, single-machine clustering techniques with a single processor and a memory cannot handle the tremendous amount of data. Algorithms that can be run on multiple machines are needed. Unlike single-machine techniques, multiple machine clustering techniques divide the huge amount of data into small pieces. These small pieces of data can be loaded on different machines and the huge problem can be solved using the processing power of these machines. Parallel processing applications include conventional parallel applications and data-intensive applications. The conventional parallel applications assume that data can be fit into the memory of distributed machines. Data intensive applications are I/O bound and devote the largest fraction of the execution time to the movement of data. OpenMP, MPI [13], and MapReduce are common parallel processing models for computing data-intensive applications. In here, we only discuss the conventional parallel and MapReduce clustering algorithms.

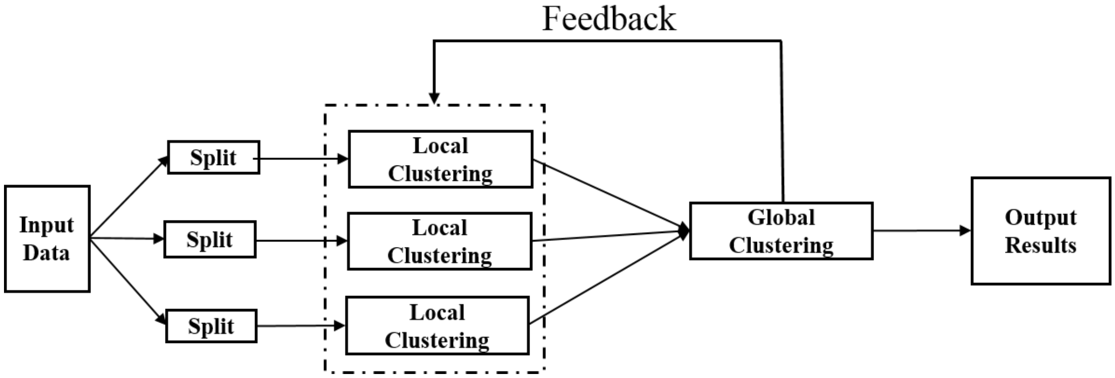

Figure 2: The general framework of most parallel and MapReduce clustering.

Both parallel and MapReduce clustering algorithms follow the general framework illustrated in Figure 2. Firstly, the input data are partitioned and distributed over different machines. Secondly, each machine performs local clustering on its own split of the input data. Thirdly, the information of machines is aggregated globally to produce global clusters for the whole data set. The global clusters are sent back to each machine as a start point for new local clustering. This process continues until the stop criteria have been met. Finally, the final results of the global clusters is generated.

1.3.2.1 Parallel Clustering

Three main strategies in the parallelism used in data mining algorithms can be identified as the following [10]: 1) Independent parallelism, the whole data is operated in each processor and no communication between processors. 2) Task parallelism, different algorithms are operated on each processor. 3) SPMD (Single Program Multiple Data) parallelism, the same algorithm is executed on multiple processors with different partitions. The results are exchanged in order to cooperate each other. The combination of tasks and SPMD parallelism with the master-slave architecture is the common strategy.

Parallel Partitioning Clustering Algorithms

The k-means algorithm is one of the most popular partitioning clustering algorithms. A parallel implementation of k-means is introduced in [17]. The parallel K-means algorithm is developed on the message passing model of a network of workstations (NOWs). Beside parallel k-means, a parallel CLARANS algorithm using PVM (Parallel Virtual Machine) is introduced in [18]. PVM uses a master/slave paradigm. The master program can assign tasks to other slaves and the communication between computers is based on message-passing.

Parallel Hierarchical Clustering Algorithm

The network topology of processors and the splitting of data accessing have been used in the parallelization of hierarchical clustering algorithms. Parallel BIRCH [19] called PBRICH is a hierarchical clustering algorithm applied to the SPMD model with message-passing. PBIRCH divides the data equally into the processors and the k-means algorithm is applied in the initial stage. The k initial centroids are broadcasted to each processor and a CF-tree is constructed accordingly. The centroids of the local clusters are used to compute the global centroids. Each processor recalculates the centroids and the processes are repeated until it converges. In Figure 3, BIRCH scans the input data and builds the CF-tree. The initial CF-tree might not be accurate due to the skewed input order or the splitting effect by the page size. A tree condensing step might be applied to address this issue. The global clustering clusters all the sub-clusters in the leaf nodes which is done by an agglomerative hierarchical clustering algorithm. The clustering results reassigns all the data points based on the results by the global clustering step.

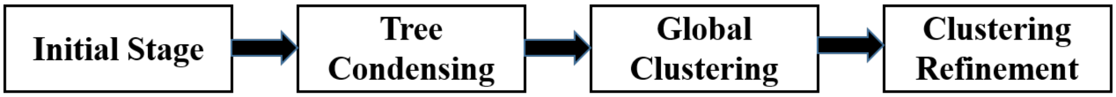

Figure 3: The flow chart of BIRCH algorithm.

Parallel Density based Clustering Algorithm

PDBSCAN [20] is a parallel version of the DSBSCAN algorithm. DSBSCAN is a density-based clustering algorithm. The main objective of density based clustering is the discovery of clusters of arbitrary shapes. PDBSCAN with master-slave configuration includes three steps. The first step divides the input data into several partitions and distributes them to different nodes. The second step concurrently clusters the partition using DBSCAN. The last step accumulates the local clusters and calculates the global clusters for the whole data.

Parallel Graph-based Clustering Algorithm

The goal of graph partitioning is to find the good cluster of vertices. The parallelization of METIS is ParMETIS which is a multilevel partitioning algorithm [21]. ParMETIS includes three phases. The first phase is the coarsening phase. It tries to find the maximal matching on the original graph such that the vertices that are matched create a small enough graph. The second phase is the partitioning phase. The coarsened graph from the previous phases is clustered in k-way using the multilevel recursive bisection algorithm. A greedy algorithm is applied to project back the partitioning from the second phase to the original graph in the last phase.

GPU based parallel clustering

The use of GPU instead of CPU speeds up the computation in parallel computing. GPUs consiste of thousands of cores and CPUs only have several processing cores (see Figure 4). GPUs are much more powerful and faster than CPUs. A G-DBSCAN [22] is a GPU-based parallel algorithm. G-DBSCAN has two parallelized steps. A graph where the edges are created based on a predefined threshold is constructed in the first step. The second step uses the Breath-First Search (BFS) to traverse the graph to identify the clusters. The results shows that G-DBSCAN is 112 times faster than DBSCAN.

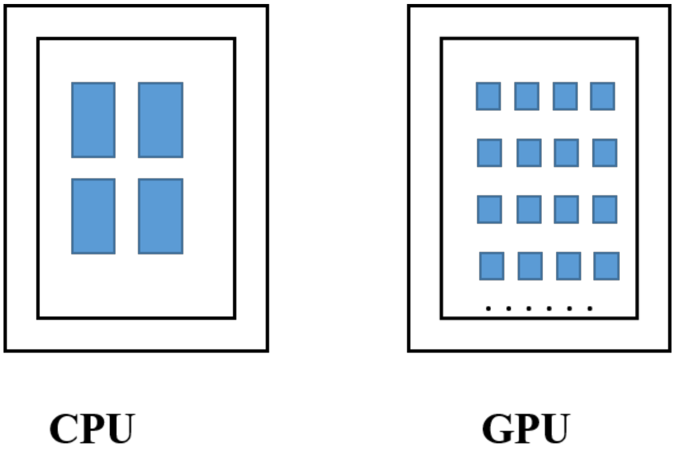

Figure 4: Difference between CPU and GPU. CPU has several cores while GPU are consists of hundreds of cores.

1.3.2.2 MapReduce based Clustering

MapReduce is one of the most efficient big data solutions which enables to process a massive volume of data in parallel with many low-end computing nodes. This programming paradigm is a scalable and fault-tolerant data processing technique that was developed to provide significant improvements in large-scale data-intensive applications in clusters. MapReduce has gained significant momentum from industry and academia by virtue of its simplicity, scalability, and fault-tolerance [23].

The MapReduce model hides the details of the parallel execution, which allows users to focus only on data processing strategies. The MapReduce model consists of two basic elements [24]: mappers and reducers. The idea of this programming model is to design mappers (or map function), which can be used to generate a set of intermediate key/value pairs. The reducer (or reduce function) are used as a shuffling or combining function to merge all of the intermediate values which are associated with the same intermediate key. The main aspect of the MapReduce algorithm is that if every map and reduce is independent of all other ongoing maps and reduces, then the operations can be run in parallel on different keys and lists of data.

The process of the MapReduce approach can be decomposed as follows: 1) Prepare input data: MapReduce utilizes the Google File System (GFS or HDFS) as an underlying storage layer to read input and store output [59]. GFS is a chunk-based distributed file system that supports fault-tolerance by data partitioning and replication. The big data is divided into small chunk on different worker node. 2) The map step: the map function of each node is applied to local data and the output is written to a temporary storage space. 3) The sort and send step: the output from step 2) is sorted with key such that all data belonging to one key are located on the same node. The sorted results are sent to the reduce processors. 4) The reduce step: each group of output data (per key) is processed in parallel on each reduce node. The user-provided reduce function is executed once for each key value produced by the map step. 5) Produce the final output: The MapReduce system collects all of the reduce outputs and sorts them by key to produce the final output. The results are stored in the GFS.

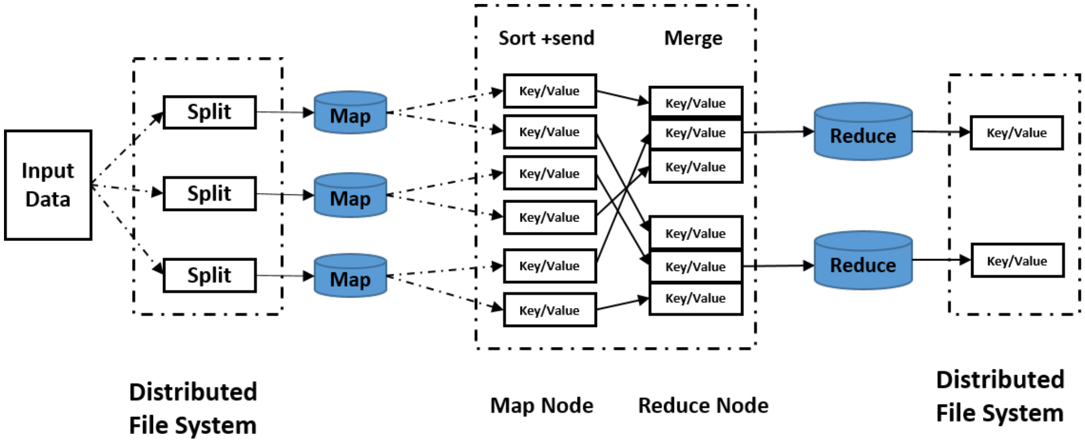

Figure 5: A general framework of a MapReduce system.

MapReduce based Partitioning Clustering Algorithms

PKMeans [25] is a distributed version of the k-means algorithm using the MapReduce framework to speed up and scale up the process. The PKMeans follows the general framework illustrated in Figure 5. The data is implicitly partitioned in the distributed file system. The local clustering is performed in the map step using the k-means algorithm. The global clustering is performed in the merge and reduce step. The outputs of the reduce step are the centroids of clusters which are sent back to the map step for the next iteration. The process is iterated until it converges. The speed up and size up of PKMeans are near linear and the scale up is also good.

MapReduce based DBSCAN

MapReduce-based DBSCAN [26] is a density-based clustering algorithm implemented using MapReduce. Existing parallel DBSCAN have three major drawbacks. First, they cannot properly balance the load among parallel tasks. Second, the scalability of these algorithms is limited because all critical sub procedures are not parallelized. Third, these algorithms are not less portable to emerging parallel processing paradigms because they are not designed for shared-nothing environments. In MR-DBSCAN, all the critical sub-procedures are parallelized to ensure scalability. A novel cost-based data partitioning method is proposed to achieve desirable load balancing on heavily skewed data. Results show that the MR-DBSCAN has good speed up and scale up. It is a lot faster than the exiting parallel DBSCAN.

MapReduce based Evolutionary Algorithm

The inherent parallel nature of evolutionary algorithms makes them optimal candidates for parallelization. A scalable MR-CPSO algorithm using the MapReduce framework to overcome the inefficiency of PSO clustering for large data sets is proposed in [27]. MR-CPSO algorithm using the MapReduce methodology has been successfully parallelized. MR-CPSO is a partitioning clustering algorithm similar to the k-means approach. The clustering task in MR-CPSO is formulated as an optimization problem to obtain the best solution based on the minimum distances between the data points and the cluster centroids. Another approach based on a Glowworm Swarm Optimization Clustering Algorithm using MapReduce was introduced in [28]. The experimental results showed that the algorithm scales very well with increasing data set sizes and achieves a very close to linear speedup while maintaining the clustering quality.

MapReduce based Fuzzy C-Means Algorithm

A fuzzy clustering algorithm was implemented using the MapReduce paradigm in [29]. A validity analysis was conducted in order to show the correct working of the implementation as well as to evaluate the clustering results. A scalability analysis was conducted to demonstrate the performance of the parallel fuzzy clustering algorithm implementation with increasing number of computing nodes used.

1.4 Application

1.4.1 Image segmentation

Image segmentation is a necessary first process in image understanding and computer vision by correctly classifying the pixels of an image in decision-oriented applications [30]. The essential goal of image segmentation is to partition an image into uniform and homogeneous attribute regions based on some likeness measure, such as gray level, color, tone, texture, etc. Due to the variety and complexity of images, image segmentation is still a very challenging research topic. Basically, segmentation approaches for images are based on the discontinuity and similarity of image intensity values. Discontinuity is an approach which partitions an image based on abrupt changes. According to the predefined criteria, the similarity approach is based on partitioning an image into similar regions.

Big data image segmentation is a strategy in image analysis, which emerged especially in the medical area [31]. Different algorithms for big data image segmentation have been executed on the parallel computing strategy to achieve effective results. Researchers have proposed a variety of techniques to tackle the challenging problem of image segmentation. Because of the complexity of image segmentation and given that only partial prior knowledge is provided, the segmentation result would be poor if a supervised method was adopted. Thus, the unsupervised method is a better choice to solve such a problem. Clustering as an unsupervised learning strategy has been used for solving automatic grouping of images and image segmentation. In addition, clustering can also organize and retrieve image databases in an efficient way.

Partitioning clustering is the most popular technique for big data image segmentation due to the simplicity of implementation. In [32], a distributed c-means algorithm is proposed for big data image segmentation and has been applied for Magnetic Resonance Images (MRI) segmentation. The algorithm is implemented on a parallel and distributed machine based on mobile agents. The approach is implemented on a multi-agent platform. The proposed algorithm is executed on Mobile Classification Agents on different nodes on their data at the same time. The results are provided to the Mobile Host Agent to further compute the global results until the stopping criteria have been met. The output of the segmented images is provided by the Mobile Classification Agents. The algorithm is applied on MRI cerebral image and the results show that the complexity of the parallel program has been reduced using the multi-agent platform. In addition, the approach overcomes the big data challenges and offers a high-performance computing tool using a multi-agent system.

1.4.2 Load Balancing in Parallel Computing

In the big data era, the rapid and scalable deployment of virtual Web stores, media outlets, and other on-line sites or services is a problem of considerable practical interest. In a cloud-based architecture, a set of hosted resources such as processors, operating systems, software and other components can be combined or strung together to form virtual machines. The key is how to allocate the resources to support these virtual machine and reduce the response time. Load balancing [33] is an important problem of heterogeneous computer networks in cloud computing. The main concern of load balancing is how to distribute resources among the users such that no node is overloaded or sitting idle. Traditional parallel computing and grid computing environments load balancing algorithms can be classified into three ways [34]:

1) Static load balance algorithms: these algorithms are suitable for small distributed environments. The decisions related to the balancing of the load will be made at compile time when resource requirements are estimated. The advantage of static load balancing algorithms is the simplicity with regards to both implementation and overhead.
2) Dynamic load balance algorithms: the distribution of work load for dynamic load balancing algorithms is changed at run-time. The communication delays and execution time have been reduced since they use current load information when making distribution decisions.
3) Mixed load balance algorithms: These algorithms are a mix of static and dynamic load balance algorithms. They focus on how to symmetrically assign computing tasks and how to reduce communication cost of symmetrical distributed computing nodes.

In a distributed system environment, load balancing algorithms focus on reducing the job response time by distributing the total load of system. There are many load balancing algorithms available. Depending on the requirements, clustering algorithms play an important role in realizing cloud computing to implement the load balancing of accessing resources. Active clustering [35] is a clustering based dynamic load balancing algorithm. It groups similar nodes together using local re-wiring and then work on these clusters. It uses the concept of match maker node to create a cluster. The active clustering adopts the iterative nature of hierarchical clustering algorithms. In the iterative process, the algorithm assumes there are two types of nodes. The first node selects a matchmaker node, which is of a different type. This matchmaker node connects with its neighbor node, which is of the same type as the first node. The process repeats until the matchmaker node gets detached. The performance of the system is enhanced with high resources thereby increasing the throughput by using these resources effectively. However, the algorithm is degraded due to the increase of system diversity. Hence, there is strong need for an efficient load balancing mechanism.

1.4.3 Genetic Mapping

Genetic maps are important components in plant science. A genetic map serves many practical biological purposes and is a key tool in both classical and modern plant research. A genetic map is a list of genetic elements ordered according to their co-segregation patterns [36]. The essential concept of a genetic map is the linkage group. The linkage group collects genetic markers that are found on a single chromosome. The number of linkage groups equals the number of chromosomes in the species. Genetic markers are clustered into the linkage groups. In order to group genetic makers into linkage groups, it needs to compute pairwise similarities between all pairs of markers and then use various standard clustering algorithms for clustering. A similarity matrix with the similarity scores is an input for clustering algorithms with a complexity of O(m^2) for m markers.

The bottleneck for genetic mapping therefore is how to efficiently find the linkage groups using clustering algorithms. Moreover, conventional genetic mapping tools were designed for small data sets and can only handle up to 10,000 markers. In [37], a fast clustering algorithm called BubbleCluster is proposed to exploit the structure of genetic makers. The BubbleCluster algorithm consists of three phases. The first phase is the most important. It exploits the structure of genetic linkage groups to quickly cluster high-quality makers as a skeleton. The second phase uses the skeleton obtained from the first phase to cluster more noisy low-quality makers. Small clusters found from previous phases are merged with large clusters. This algorithm exploits the underlying linear structure of chromosomes to avoid the quadratic pairwise calculation.

1.4.4 Community Detection

Community detection is a useful tool to partition the nodes of an observed network into groups. A "network community" is typically thought of as a group of nodes such that the connection of the nodes is dense within groups but sparser between them [38]. Community detection is potentially very useful. A good partition can be considered as a hint of underlying semantic structure or possible mechanisms of network formation. Researchers in computer science, mathematics, physics, statistics and bioinformatics use community detection algorithms to better understanding the large-scale structure of the interaction between groups in social and biological systems. For example, knowing the groups or communities within a social network such as LinkedIn can be used to infer about the trends of collaboration between individuals in industry as well as academia. It helps to better understand the function of key biological processes by uncovering the nature of interactions between groups of proteins in a Protein-Protein Interaction network. Hence, community detection has received a great deal of attention in real-world graphs such as large social networks, web graphs and biological networks.

Community detection is similar to clustering where both use a data mining technique to partition a network into groups based on the similarity measurement. In [39], a fast parallel modularity optimization algorithm (FPMQA) for community detection is introduced. An interesting social network called Interest Network links two IDs if both IDs have participated to the discussion about one or more topics. The initial network is updated using the attitude consistency of the connected ID pairs. FPMQA is then used to conduct community detection. FPMQA assumes that each node is a separate community in a network that finds the pairs of communities with the local maximum. FPMQA uses the parallel technique to merge the corresponding community pairs into one community. Compared to conventional approaches, FPMQA reduces the running time of community detection and uses the reliable ground truths to evaluate detected communities.

1.5. Conclusion Remarks

In this chapter, we reviewed the literature about clustering in big data and the corresponding clustering algorithms. Clustering algorithms are categorized into six classifications: partitioning, hierarchical, density based, model-based, grid-based algorithms and evolutionary algorithms. The most popular clustering algorithms are partitioning and hierarchical algorithms and many of the parallel versions of the algorithms have been developed. The traditional single-machine techniques are not powerful enough to handle the huge volume of data accumulated nowadays. Parallelism in clustering algorithms is important for multi-machine techniques. Parallel clustering is potentially useful for big data clustering. However, the complexity of implementing such algorithms is a challenge. Hence, MapReduce has gained significant momentum from industry and academia by virtue of its simplicity, scalability, and fault-tolerance. Big data clustering as an essential task in big data mining and is not limited to image segmentation, load balancing, genetic maps and community detection.

References

[1] Executive Summary Data Growth, Business Opportunities, and the IT Imperatives
An ICD report. Retrieved from www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm.

[2] H. A. Edelstein. Introduction to data mining and knowledge discovery (3rd ed). Potomac,

MD: Two Crows Corp. 1999.

[3] Z. Xu and Y. Shi, Exploring Big Data Analysis: Fundamental Scientific Problems. Annals of Data Science, 2(4), 363-372, 2015.

[4] C. P. Chen and C. Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 275, 314-347, 2014.

[5] D. Laney: 3D Data Management Controlling-Data Volume, Velocity and Variety (February 2001).

[6] Y. Demchenko, P. Grosso, C. De Laat and P. Membrey, Addressing big data issues in scientific data infrastructure. In Collaboration Technologies and Systems (CTS), 2013 IEEE International Conference on (pp. 48-55).

[7] A. Fahad, N. Alshatri, , Z.Tari, A. Alamri, I.Khalil, A. Y. Zomaya, and A. Bouras, A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE transactions on emerging topics in computing, 2(3), 267-279, 2014.

[8] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, Big data clustering: a review. In International Conference on Computational Science and Its Applications (pp. 707-720). Springer International Publishing, 2014.

[9] L. Kaufman, P. J. Rousseeuw, Finding groups in data: an introduction to cluster analysis (Vol. 344). John Wiley & Sons, 2009.

[10] W. Kim, Parallel clustering algorithms: survey. Parallel Algorithms, Spring, 2009.

[11] C. C.Aggarwal, C. K. Reddy, Data clustering: algorithms and applications. CRC Press, 2013.

[12] R. T. Ng, J. Han, CLARANS: A method for clustering objects for spatial data mining. IEEE transactions on knowledge and data engineering, 14(5), 1003-1016, 2002.

[13] J. A. Zhang, Parallel Clustering Algorithm with MPI–Kmeans. Journal of computers 8.1 (2013): 10-17.

[14] I. Jolliffe, Principal component analysis. John Wiley & Sons, Ltd, 2002.

[15] J. Yadav, D. Kumar, Subspace Clustering using CLIQUE: An Exploratory Study.

[16] Q. Gu, J. Zhou, Co-clustering on manifolds. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 359-368), 2009.

[17] S. Kantabutra and A. L. Couch, Parallel K-means clustering algorithm on NOWs. NECTEC Technical journal, 1(6), 243-247, 2000.

[18] Y. P. Zhang, J. Z. Sun, Y. Zhang, and X. Zhang, Parallel implementation of CLARANS using PVM. In Machine Learning and Cybernetics, 2004. Proceedings of 2004 IEEE International Conference on (Vol. 3, pp. 1646-1649).

[19] A. Garg, A. Mangla, N. Gupta, and V. Bhatnagar, PBIRCH: a scalable parallel clustering algorithm for incremental data. In 2006 10th IEEE International Database Engineering and Applications Symposium (IDEAS'06) (pp. 315-316).

[20] X.Xu, J. Jäger, and H. P. Kriegel, A fast parallel clustering algorithm for large spatial databases. In High Performance Data Mining (pp. 263-290). Springer US, 1999.

[21] G. Karypis, K. Schloegel, and V. Kumar, Parmetis. Parallel graph partitioning and sparse matrix ordering library. Version, 2, 2003.

[22] G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, L. Rocha, G-dbscan: A gpu accelerated algorithm for density-based clustering. Procedia Computer Science, 18, 369-378, 2013.

[23] K. Shim, MapReduce algorithms for big data analysis. Proceedings of the VLDB Endowment, 5(12), 2016-2017, 2012.

[24] X.Wu, X. Zhu, G. Q. Wu, and W. Ding, Data mining with big data. IEEE transactions on knowledge and data engineering, 26(1), 97-107, 2014.

[25] W. Zhao, H. Ma, Q. He, Parallel k-means clustering based on mapreduce. In IEEE International Conference on Cloud Computing (pp. 674-679). Springer Berlin Heidelberg, 2009.

[26] 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.

[27] I. Aljarah, and S. A. Ludwig, Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on (pp. 104-111). IEEE.

[28] N. Al-Madi, I. Aljarah, S. A. Ludwig, Parallel Glowworm Swarm Optimization Clustering Algorithm based on MapReduce, IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, USA, December 2014.

[29] S. A. Ludwig, MapReduce-based Fuzzy C-Means Clustering Algorithm: Implementation and Scalability, International Journal of Machine Learning and Cybernetics, Springer, vol. 6, no. 6, pp. 923-934, 2015.

[30] M. Gong, Y. Liang, J. Shi, W. Ma, and J. Ma, Fuzzy c-means clustering with local

information and kernel metric for image segmentation. IEEE Transactions on Image Processing, 22(2), 573-584, 2013.

[31] C. P. Chen and C. Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 275, 314-347, 2014.

[32] F. Z. Benchara, M. Youssfi, O. Bouattane, H. Ouajji, and M. O. Bensalah, Distributed C-means algorithm for big data image segmentation on a massively parallel and distributed virtual machine based on cooperative mobile agents. Journal of Software Engineering and Applications, 8(03), 103, 2015.

[33] N. J. Kansal and I. Chana, Cloud load balancing techniques: A step towards green computing. IJCSI International Journal of Computer Science Issues, 9(1), 238-246, 2012.

[34] S. S. Moharana, R. D. Ramesh, D. Powar, Analysis of load balancers in cloud computing. International Journal of Computer Science and Engineering, 2(2), 101-108, 2013.

[35] J. Uma, V.Ramasamy and A. Kaleeswaran, Load Balancing Algorithms in Cloud Computing Environment-A Methodical Comparison. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume, 3, 2014.

[36] J. Cheema and J. Dicks, (2009). Computational approaches and software tools for genetic linkage map estimation in plants. Briefings in bioinformatics, 10(6), 595-608.

[37] V. Strnadová, A. Buluç, J. Chapman, J. R. Gilbert, J. Gonzalez, S. Jegelka and L.Oliker, Efficient and accurate clustering for large-scale genetic mapping. In Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on (pp. 3-10). IEEE.

[38] J. Leskovec, K. J. Lang and M. Mahoney, Empirical comparison of algorithms for network community detection. In Proceedings of the 19th ACM international conference on World wide web (pp. 631-640), 2010.

Min Chen is now an Assistant Professor at SUNY at New Paltz. She received her bachelor‘s degree in mathematics and physics from College of St. Benedict in 2009, and earned her master’s degree in computer science and doctoral degree in software engineering at North Dakota State University in 2011 and 2015, respectively. Her current research interests include artificial intelligence, machine learning and big data computing.

Simone A. Ludwig is an Associate Professor of Computer Science at North Dakota State University, USA. She received her PhD degree and MSc degree with distinction from Brunel University, UK, in 2004 and 2000, respectively. Before starting her academic career she worked several years in the software industry. Her research interests lie in the area of computational intelligence including swarm intelligence, evolutionary computation, neural networks, and fuzzy reasoning. In particular, computational intelligence methods and algorithms are applied to optimization problems in areas such as data mining (including big data), image processing, and cloud computing.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 320 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, Journal of Parallel and Distributed Computing. He is an IEEE Fellow.