Information Systems 60 (2016) 1–12

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Single-pass and linear-time k-means clustering based on MapReduce

Saeed Shahrivari, Saeed Jalili *

Computer Engineering Department, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO

Article history:
Received 25 May 2014
Accepted 23 February 2016
Recommended by: G. Vossen
Available online 4 March 2016

Keywords:
Distributed k-means
Data clustering
MapReduce-based clustering

ABSTRACT

In recent years, k-means has been fitted into the MapReduce framework and hence it has become a very effective solution for clustering very large datasets. However, k-means is not inherently suitable for execution in MapReduce. The iterative nature of k-means cannot be modeled in MapReduce and hence for each iteration of k-means an independent MapReduce job must be executed and this results in high I/O overhead because in each iteration the whole dataset must be read and written to slow disks. We have proposed a single-pass solution based on MapReduce called *mrk-means* which uses the reclustering technique. In contrast to available MapReduce-based k-means implementations, *mrk-means* just reads the dataset once and hence it is several times faster. The time complexity of *mrk-means* is linear which is lower than the iterative k-means. Due to usage of k-means++ seeding algorithm, *mrk-means* results in clusters with higher quality, too. Theoretically, the results of *mrk-means* are $O(\log^2 k)$−competitive to optimal clustering in the worst case, considering k as the number of clusters. During our experiments which were done on a cluster of 40 machines running the Hadoop framework, *mrk-means* showed both faster execution times, and higher quality of clustering results compared to available MapReduce-based and stream-based k-means variants.

1. Introduction

Large volumes of data are being produced nowadays and there is a high demand for methods and tools that can efficiently manage, mine, and process these large volumes of data, that is commonly known as *Big Data* [1]. Classical tools and management systems are not appropriate for big data. Hence, several new programming models and frameworks have been proposed for processing big data. The most well-known framework is MapReduce [2]. MapReduce is initially developed by Google and its popular open-source implementation is Hadoop [3]. MapReduce offers three main features in a single package: simple programming paradigm, linear and automatic scalability, and built-in fault tolerance. These three features make MapReduce an ideal framework for big data processing [4].

One of the most crucial tasks in data and knowledge discovery is *Data Clustering*. According to Jain's definition, "The goal of data clustering, also known as cluster analysis, is to discover the natural grouping(s) of a set of patterns, points, or objects" [5]. Data clustering has various applications in different fields. For example, in Computer Vision, Image Segmentation can be defined as a clustering problem [6]. In Information Retrieval, document clustering can provide hierarchical retrieval and improvements in flat retrieval performance [7]. In Bioinformatics, clustering is used for improving multiple sequence alignment [8]. Many

* Corresponding author.
E-mail addresses: s.shahrivari@modares.ac.ir (S. Shahrivari), sjalili@modares.ac.ir (S. Jalili).

http://dx.doi.org/10.1016/j.is.2016.02.007
0306-4379/

other important applications also exist in fields like: Medicine, Online Social Networks, etc. [5].

However, classic tools and methods for data clustering are not appropriate for big data. Most of the classic data clustering algorithms require the data to be stored in the main memory but a big dataset usually does not fit in the main memory. On the other hand, some data clustering methods like Single and Complete Linkage clustering algorithms cannot be applied to big data because the time complexity of these algorithms is $O(n^2)$ assuming n as the number of items [9]. When big data is the target, the volume of data is usually so big that just sub-linear, linearithmic, and linear time algorithms can be used.

k-means and its variants are the most well-known linear time algorithms for data clustering [5]. k-means has been successfully fitted into MapReduce and several implementations have been proposed [10–13]. However, the proposed solutions have some considerable shortcomings. Some shortcomings are inherited from the standard k-means algorithm. No guarantee of results quality and optimality, no guarantee of convergence, the requirement of providing the number of clusters as an input parameter, and high dependency of results quality to initial seeds, are some major shortcomings that are inherited from the standard k-means algorithm to available MapReduce-based k-means implementations [5].

On the other hand, a major performance penalty is the result of inherent conflict between the MapReduce framework and the k-means algorithm. k-means is an iterative algorithm and for reaching confident results, it needs to make some iterations. In contrast, MapReduce has a significant problem with iterative jobs [14]. MapReduce naturally does not support iteration and recursion and in order to execute iterative jobs, usually a driver program is used [15]. That is to say, a driver program executes the actual procedure several times in order to emulate iterative execution. This results in a major overhead for each iteration. In each iteration, the whole data must be loaded from the file system into the main memory. Then, after it is processed, the output must be written to filesystem again. Therefore, lots of I/O-related operations like disk I/O and data serialization occur during each iteration and this decelerates the whole execution [14].

In this paper, we present a novel single-pass MapReduce-based data clustering solution based on the well-known k-means algorithm that tries to overcome the mentioned shortcomings. Our solution is named *mrk-means* and delivers three main advantages. First, we have decreased both time and I/O complexity of clustering from $O(I \cdot n/p)$ to $O(n/p)$ assuming n as the number of items, I as the number of iterations, and p as the number of available processors. Second, we have given strong bounds for the optimality of the results. We have proven that *mrk-means* is an $O(\log^2 k)$ approximation to the optimal k-means solution.

We have performed several experiments on both real-world and synthesized datasets and the results show that *mrk-means* is superior to the state of the art MapReduce-based and single-pass stream-based k-means variants considering both speed and quality of results. Our experiments also show that *mrk-means* scales well when the number of machines or volume of data increases.

2. Related work

k-means is a very old and well-known clustering algorithm [5]. Several parallel versions have been proposed for parallel architectures [16–18]. Several versions have also been proposed for distributed systems [19,20]. However, as we mentioned before, less works have been proposed based on MapReduce. Besides k-means, there are other algorithms that are designed for large datasets like CURE which is based on sampling [21], DBSCAN which is a density based algorithm [22], and FDM which performs constrained clustering [23] but in this paper we focus on k-means variants.

Zhao et al. have proposed a MapReduce-based k-means clustering solution [12]. A fast solution is also available via the Mahout project that executes over the Hadoop framework [11]. Bahmani et al. have proposed a scalable k-means algorithm that extends the k-means++ technique for initial seeding [24]. There is also a distributed k-means++ implementation available from the GraphLab project [25]. However, all of the mentioned solutions need to process the whole dataset for several iterations in order to reach acceptable results and this makes them inappropriate for the MapReduce framework because MapReduce does not support iteration inherently. Hadian and Shahrivari have also proposed a parallel and single-pass clustering algorithm for parallel shared memory systems but it is not designed for the MapReduce framework [26].

Several extensions have been proposed for extending the standard MapReduce model to support recursion and iteration. Twister [14], Spark [27], and HaLoop [28] are some solution to be mentioned. The k-means algorithm has been implemented for all of the mentioned extensions. However, all of these solutions need the dataset to be small enough to be stored in the main memory. On the other hand, they are not stable and mature enough like the standard MapReduce implementations, i.e. Hadoop. Hence, they are not appropriate and cost-effective for very large datasets.

There are also some related works in the area of data stream clustering. These works are related because they try to cluster the given dataset in a single pass. Several, k-means variant have been proposed for stream data [29–32]. These solutions are able to cluster the dataset in a single pass but none of them are inherently capable and ready for execution in distributed systems and especially the MapReduce framework.

3. Preliminaries

If we have a set of n data items (also known as a dataset), each having d features, then this dataset can be represented by a matrix $M_{n\times d}$. We define $m_{i,j}$ as the element of row i and column j in M. Then, the ith item can be represented by vector $\vec{m}_i$. Given such a matrix, a partitional clustering algorithm should find a set $C = \{C_1, C_2, \ldots, C_k\}$ of clusters (partitions) as the output of clustering. Each cluster should preserve these two conditions:

1. Each cluster should have at least one data item assigned, i.e., $\forall C_i \in C: C_i \neq \emptyset$.