Fast Scalable k-means++ Algorithm with MapReduce

Yujie Xu, Wenyu Qu, Zhiyang Li, Changqing Ji, Yuanyuan Li, and Yinan Wu

1 School of Information Science and Technology, Dalian Maritime University, Dalian, China, 116026
{yujiex.dlmu,eunice.qu,lizy0205}@gmail.com
2 School of Physical Science and Technology, Dalian University, Dalian, China, 116622
jcqgood@gmail.com
3 School of Software, Dalian Jiaotong University, Dalian, China, 116028
lyy3232312@sohu.com
4 Department of Equipment, Unit 91550 of PLA, Dalian, China, 116023
wyn03021w@163.com

Abstract. K-means++ is undoubtedly one of the most important initializing algorithms for k-means owing to its provable approximation guarantee to the optimal solution. However, due to its sequential nature, k-means++ requires a large number of iterations to complete the initialization and it becomes inefficient as the size of data increase. Even though scalable k-means++ can drastically reduce the iterations and can be easily applied to the MapReduce systems, but due to its sequential nature, it still requires two MapReduce jobs in each round. Moreover, it takes a large number of I/O cost and it is time-consuming. In this paper, we propose Oversampling and Refining (OnR) method which can improve efficiency of scalable k-means++ by using only one MapReduce job to obtain \( \Omega(k) \) centers in each round. Except for the oversampling factor \( \ell \) of scalable k-means++, OnR uses another oversampling factor \( o \) to further increase the number of chosen centers. Oversampling is executed on the Mapper phase, and in Reducer phase, one Reducer is responsible for removing the oversampled centers generated from \( o \) and outputs a set of centers which is the same as the output of scalable k-means++. To reduce the expensive network cost caused by too large \( o \), OnR estimates the global cost by the local clustering cost and uses it to remove some wrong points in Mapper phase. Extensive experiments on real data are conducted and the performance results indicate that OnR outperforms scalable k-means++ in the aspect of I/O cost and running time.

1 Introduction

Clustering has been applied in many areas of computer science and its related fields, such as data mining, pattern recognition and image retrieval [1–4]. K-means
is one of the most widely used clustering methods, but it suffers from the well-known problem that converges to a local optimum. Due to the reason that it is highly dependent upon the chosen of initial centers. In recent years, many researches have focused on improving its initialization method [5][6]. An important piece of work in this direction is the k-means++ [7]. This algorithm is fast with small data in practice. Moreover, it obtains an $O(\log k)$ approximation solution to the optimal result of k-means and gives a theoretical guarantee firstly.

However, the era of big data poses new challenges for k-means++ algorithm. Although it can be run on the MapReduce [8], and there are also many clustering algorithms [9–12] run on MapReduce platform efficiently in practice, k-means++ is an exception. The fundamental reason is that k-means++ is a sequential algorithm and it is lack of scalability. That is the probability a point is chosen to be a center strongly depends on the previous centers. K-means++ algorithm chooses one center in each round and it needs $k$ rounds over the data to produce the expected initial centers. This requires many iterative computations. For a single computer, iterative computation is common and it is easily implemented. While for the MapReduce framework, it does not directly support these iterative data analysis applications. Instead, we must implement iterative programs by manually issuing multiple MapReduce jobs and this renders the data must be re-loaded and re-processed at each iteration, wasting I/O, network and CPU resources [13][14].

To reduce the number of rounds of k-means++, Bahman Bahmani et al. proposed scalable k-means++ algorithm [15]. We show it in Section 3 in more detail. It is a parallel version of the inherently sequential k-means++. Instead of choosing one point in each round, scalable k-means++ uses the oversampling method to choose $\ell = \Omega(k)$ points. Hence, it can drastically reduce the iteration rounds from $k$ to approximate $O(\log \psi)$. Scalable k-means++ enhances the scalability of k-means++ and it is easily paralleled in MapReduce framework. Another merit of it is that it achieves an $O(\log k)$ approximation to the k-means objective.

However, scalable k-means++ does not thoroughly break the inherent sequential nature of k-means++. Thus, it is embarrassingly parallel and can not be executed on MapReduce-based systems efficiently. Considering that there is no communication between Mappers, MapReduce scalable k-means++ requires two MapReduce jobs to complete in each round. The first job chooses $\ell$ centers and combines them. The second one is responsible for computing the clustering cost. Therefore, it has to iterate $O(\log \psi)$ rounds and at least $2 \times O(\log \psi)$ MapReduce jobs to choose the initial centers. As mentioned above, MapReduce does not directly support iterative analysis applications, when $\log \psi$ is large, it is time-consuming and we cannot put up with so many MapReduce jobs. In addition, it incurs large amount of network and I/O overhead.

This paper proposes an efficient parallel scalable k-means++ algorithm which is called Oversampling and Refining (OnR) in the situation of big data by virtue of MapReduce. The main idea of OnR is to use only one MapReduce job, instead of two jobs, to complete the task of choosing new centers and computing clustering cost. For lack of communication in Mapper phase, we could not compute the