Efficient k-means++ Approximation with MapReduce

Yujie Xu, Wenyu Qu*, Zhiyang Li, Geyong Min, Keqiu Li, and Zhaobin Liu

Abstract—k-means is undoubtedly one of the most popular clustering algorithms owing to its simplicity and efficiency. However, this algorithm is highly sensitive to the chosen initial centers and a proper initialization is crucial for obtaining an ideal solution. To overcome this problem, k-means++ is proposed to sequentially choose the centers so as to achieve a solution that is provably close to the optimal one. However, due to its weak scalability, k-means++ becomes inefficient as the size of data increases. To improve its scalability and efficiency, this paper presents MapReduce k-means++ method which can drastically reduce the number of MapReduce jobs by using only one MapReduce job to obtain k centers. The k-means++ initialization algorithm is executed in the Mapper phase and the weighted k-means++ initialization algorithm is run in the Reducer phase. As this new MapReduce k-means++ method replaces the iterations among multiple machines with a single machine, it can reduce the communication and I/O costs significantly. We also prove that the proposed MapReduce k-means++ method obtains $O(\alpha^2)$ approximation to the optimal solution of k-means. To reduce the expensive distance computation of the proposed method, we further propose a pruning strategy that can greatly avoid a large number of redundant distance computations. Extensive experiments on real and synthetic data are conducted and the performance results indicate that the proposed MapReduce k-means++ method is much more efficient and can reach a good approximation.

Index Terms—k-means; k-means++; MapReduce; Approximation

1 Introduction

Clustering, which is one of the most fundamental problems in data analysis and management, has been applied in many areas of computer science and related fields, such as data mining, pattern recognition and machine learning [1] [2] [3] [4]. k-means is undoubtedly one of the most popular clustering algorithms owing to its simplicity and efficiency and has received significant research efforts. However, for the characteristic of gradient descent, k-means often converges to a local optimum and has no accuracy guarantees. Furthermore, the final solution is often far away from the global optimum. The fundamental reason is that k-means is highly sensitive to the chosen initial centers. Thus, many recent studies have focused on improving the initialization method [5] [6]. An important piece of work in this direction is the k-means++ [7] algorithm which consists of the initialization step and k-means step. In the initialization step, except that the first center is chosen randomly, each subsequent center is orderly chosen according to its squared distance from the closet center already chosen. More importantly, k-means++ has a provable approximation guarantee to the optimal solution.

- *J. Xu, W. Qu, Y. Li, and B. Zhao are with the Department of Information Science and Technology, Dalian Maritime University, Dalian, China, 116026.*
  E-mail: {yujiex, wenyu, lizy0205, zbliu}@dlmu.edu.cn
- *Y. Min is with the Department of Computing, University of Bradford, BD7 1DP, United Kingdom.*
  Email: G.Min@bradford.ac.uk
- *Q. Li is with Department of Computer Science and Technology, Dalian University of Technology, Dalian, China, 116024.*
  Email: likeqiu@gmail.com

However, k-means++ becomes inefficient as the size of data increases for the reasons below: (1) Large data (like Terabyte or Petabyte) means that there are a large number of clusters, leading to a huge number of distance computations. So, k-means++ initialization becomes inefficient and even impossible to process large data; (2) Large data cannot be stored on a single machine and need to be split to store on multiple machines. Thus, parallel processing of the large data is so far the best solution. Nevertheless, an obstacle of k-means++ initialization is its sequential nature when choosing initial centers. The probability that a point is chosen to be a center strongly depends on the previous center. Due to the lack of scalability, it is difficult to parallel k-means++ initialization efficiently; (3) k-means++ initialization algorithm chooses one center in each pass and its parallel implementation makes k passes over the data to produce the initial centers. Even though scalable k-means++ presented in [8] chooses more than one centers in each pass and is proven as a good approximation of the original k-means, it still needs too many passes in practice, which incurs huge communication and I/O costs.

MapReduce [9] is considered to be an efficient tool in situations where the amount of data is prohibitively large. However, for the second and the third obstacles mentioned above, MapReduce-based systems are still inefficient. To generate k centers, the MapReduce implementation of k-means++ initialization needs k rounds and $2k$ MapReduce jobs. In addition, a large number of data need to be transferred between multiple machines.

This paper is concerned with the k-means++ initialization algorithm in very large data situation and develops it with MapReduce. The major research challenges addressed are: (1) how to efficiently implement the k-

means++ initialization algorithm with MapReduce. The main idea behind our method is that, instead of using $2k$ MapReduce jobs to choose k centers, our method uses only one MapReduce job. Both Mapper phase and Reducer phase in our method execute the k-means++ initialization algorithm. The Mapper phase runs the standard k-means++ initialization algorithm and the weighted k-means++ initialization algorithm is executed on the Reducer phase. (2) Although k-means++ is $O(\alpha)$ approximation to the optimal k-means, we prove that our method is $O(\alpha^2)$ approximation to the optimal of k-means.

The major contributions of this paper are:

- We propose an efficient MapReduce implementation of k-means++ initialization which uses only one MapReduce job to choose k centers, avoiding multiple rounds of MapReduce jobs on many machines and thus reducing the communication and I/O costs significantly.
- We reach a theoretical guarantee of our method and prove that the proposed MapReduce k-means++ method obtains $O(\alpha^2)$ approximation to the optimal solution of k-means.
- To reduce the expensive distance computation of the proposed method, we further propose a pruning strategy can dramatically reduce the redundant distance computation.
- We conduct experiments on real and synthetic data. Experimental results indicate that our MapReduce k-means++ algorithm is much efficient and has a good approximation. Meanwhile, the pruning strategy can further improve the initialization method.

The rest of this paper is organized as follows. Section 2 reviews the previous related work. Section 3 presents the useful preliminaries. We describe MapReduce k-means++ algorithm and its theoretical analysis in Section 4. Section 5 presents the improved MapReduce k-means++ initialization algorithm. Section 6 reports the experimental results. Finally, Section 7 concludes the paper.

2 Related Work

Clustering problem has a long history and there have been a large number of studies on this topic concerned with many areas. Clustering has been considered in a parallel fashion, like MapReduce. The MapReduce implementation of k-means was first proposed in [10]. Papadimitriou et al. presented the distributed co-clustering framework which introduced practical approaches for distributed data preprocessing and co-clustering [11]. Ene et al. proposed the fast clustering using MapReduce [12] by adopting a MapReduce sampling technique to decrease the data size. The result of this method was applied to k-center and k-median algorithm. Moreover, they have sufficient flexibility for practical use because they run in a constant number of MapReduce rounds. Cordeiro et al. solved the problem of how to cluster a very large moderate-to-high dimensionality dataset [13] and proposed Parallel Clustering and Sample-and-Ignore clustering algorithm with MapReduce which could reduce communication and I/O cost significantly through sampling.

The clustering algorithms have been also extensively studied in terms of the MapReduce framework. Ekanayake et al. [14] presented a programming model and architecture-Twister which is a distributed in-memory MapReduce runtime that supports iterative MapReduce computation efficiently, such as k-means, and Deterministic Annealing Clustering. Twister performs and scales well form many iterative MapReduce computations. HaLoop [15] was also designed to support for iterative programs. HaLoop not only extends MapReduce with programming support for iterative applications, but also dramatically improves their efficiency by making the task scheduler loop-aware and by adding various caching mechanisms.

There are a lot of initialization methods for k-means, such as [5] [6] [16] [17]. Celebi et al. [18] proposed a comprehensive review of the initialization methods of k-means. Lloyd's [19] iteration is the popular method for finding a locally optimum solution to the k-means problem. In the beginning, it chooses a set of k centers at random. In each iteration, it generates a clustering result from the current set of centers. Then, the centroids of these derived clusters then become the centers for the next iteration. The iteration is then repeated until a stable set of centers is obtained. However, Lloyd's algorithm could not obtain a good result in terms of efficiency and quality for its random initialization method. Its running time may be exponential in the worst case and α is sometimes unbounded even when n and k are fixed in practice.

k-means++ [7] initialization is an important initialization method of k-means and we will give a detailed introduction in the following section. Scalable k-means++ [8] is an improved version of k-means++. It avoids the major downside of the k-means++ of its sequential nature and select ℓ centers in each iteration. This improvement drastically reduces the number of passes needed to obtain a good initialization and make k-means++ easy to parallel. It also gives a theoretical guarantee of the initialization algorithm. However, scalable k-means++ algorithm still needs too many MapReduce jobs to choose k centers. Generally speaking, it requires two MapReduce jobs in each iteration. Furthermore, MapReduce does not directly support the iteration operation and the more MapReduce jobs the more cost. Both k-means++ algorithm and scalable k-means++ algorithm are inefficient with MapReduce. In the following section, we propose our method which uses only one MapReduce job to choose k centers.

There are also many research studies on k-means from theoretical and algorithmic points of view. k-means++ [7] provides an $O(\alpha)$ approximation with the optimal clustering. Bahman Bahmani et al. [8] proposed an initialization algorithm obtaining a nearly optimal solution after a logarithmic number of passes and then showed that in practice a constant number of passes suffices. We have discussed above, this algorithm needs too many MapReduce jobs. The research in [12] proposed a fast clustering scheme which uses the sapling technology. This paper also proved that the

MapReduce-KCenter was $4\alpha + 2$ for the k-center problem and MapReduce-KMedian was $10\alpha + 3$ approximation for the k-median problem. Kanungo et al. proposed a local improvement heuristic for k-means++ based on swapping centers in and out. It also proved that this algorithm yielded a $(9 + \epsilon)$ approximation to the optimal [20]. Kumar et al. [21] obtained the first liner time $(1 + \epsilon)$ approximation for the k-means problem for fixed k and ε. Song et al. [22] presented three $O(1)$-approximation algorithms for the k-means clustering problem. Ostrovsky et al. [23] presented a novel and efficient sampling process for choosing initial centers for Lloyd's iteration. This method leads to provably near-optimal clustering solutions when applied to clustering instances.

3 PRELIMINARIES

To provide a technical context for the discussion in this paper, we begin with presentation of useful preliminaries. First, we give a problem definition of k-means and set up some notions (Section 3.1). Next, we discuss the algorithm of k-means++ in more detail (Section 3.2). Finally, a more precise overview of the MapReduce mode is given in Section 3.3.

3.1 Problem and Notation

Given a data set $X = \{x_1, x_2, ..., x_n\}$ in d-dimensional space, k-means algorithm divides X into k exhaustive clusters $Y = \{Y_1, Y_2, ..., Y_k\}$, $\cup_{i=1}^{k} Y_i = X, Y_i \cap Y_j = \emptyset$ for $1 \leq i \neq j \leq k$. For a cluster, its centroid is given by

$$centroid(Y_i) = \frac{1}{|Y_i|} \sum_{y \in Y_i} y$$

Let $C = \{c_1, ..., c_k\}$ be a set of centers and $||x_i - x_j||$ denotes the Euclidean distance between x_i and x_j. k-means algorithm usually generates k centers by optimizing the criterion of Sum of Squared Error (SSE), which is given by:

$$SSE(C) = \sum_{x \in X} \min_{c \in C} ||x - c||^2$$

The goal of k-means clustering is to find an optimal C and minimize the $SSE(C)$.

Let C_{OPT} denote the optimal clustering and SSE_{OPT} is the corresponding SSE. We refer to a solution C of k centers as an α approximation to be optimal if it satisfies $SSE(C) \leq \alpha SSE_{OPT}$.

Finding an optimal solution to this problem is NP-hard, various heuristics have been developed to provide approximate solutions. Lloyd's algorithm is a widely used heuristics, we have discussed in Section 2.

3.2 K-means++

A good clustering result satisfies the condition that the distance between arbitrary two clusters should be as far as possible. Intuitively, it is a wise choice to choose the initial centers that are far away from each other in the beginning. k-means++ initialization algorithm follows it, but the farthest point is not always chosen to be a center. k-means++ is extremely simple and runs very fast in practice. Actually, except that the first center is chosen uniformly and randomly from the data points, each subsequent center is chosen from the remaining data points with the probability proportional to its squared distance from the existing cluster center closet to the point. k-means++ addresses the obstacle of unbounded α and guarantees to find a solution that is $O(\alpha)$ approximation to the optimal k-means. Let $D(x)$ be the Euclidean distance between x and the nearest center that has already been chosen. The k-means++ initialization algorithm is presented as follows.

Algorithm 1: k-means++

Input: k, the number of clusters.
$X = \{x_1, x_2, \ldots, x_n\}$, a set of data points.
Output: $C = \{c_1, c_2, ..., c_k\}$.

$C \leftarrow \emptyset$
Choose one center x uniformly at random from X, $C = C \cup \{x\}$.
Repeat
  Choose $x \in X$ with probability $D(x)^2 / \sum_{x \in X} D(x)^2$
  $C = C \cup \{x\}$
Until k centers are chosen ;
Proceed as with the standard k-means algorithm

3.3 The MapReduce Model of Hadoop

MapReduce is a parallel programming framework for processing the large scale dataset with numerous machines. Succinctly, a typical MapReduce job consists of three sequential phases: map phase, shuffle phase and reduce phase. For a MapReduce job, the shuffle phase usually incurs considerable performance overhead and is a bottleneck in practice. MapReduce model interleaves sequential and parallel computation, that is the reduce tasks cannot start until all map tasks finish, while in map and reduce phase, tasks are parallel. Furthermore, all map tasks are independent from each other and there is no communication among them, so do reduce tasks.

For the iteration nature of k-means and k-means++ algorithm, it requires multiple MapReduce rounds to implement them. In each round, the data is distributed among all machines and each machine processes its own input separately. The output of these machines is either the final result or becomes the input of another round. However, the MapReduce-based systems lack built-in support for iterative programs. In MapReduce, the number of MapReduce rounds is a critical metrics to be optimized, since each additional job incurs significant running time overhead because of synchronization and congestion issues, and communication per round.

The parallel implementation of k-means with MapReduce has been available. The detailed information of

MapReduce k-means is shown in [10]. We will present the detailed descriptions of our parallel implementation of k-means++ and related problems in the following sections.

4 MapReduce k-means++ and Theoretical Analysis

In this section we present MapReduce k-means++ algorithm, our parallel version for initializing the centers and clustering data with MapReduce. The initialization phase of our method uses only one MapReduce job to choose k centers quickly and it is $O(\alpha^2)$ approximation to the optimal solution of k-means. We then give the theoretical guarantee of our method.

4.1 MapReduce k-means++

As mentioned before, Lloyd's iteration can be easily parallelized with MapReduce. Therefore, we only focus on k-means++ initialization algorithm with MapReduce in this section. Combining this initialization algorithm with MapReduce-k-means generates our MapReduce k-means++ algorithm. The k-means++ initialization method shown in Algorithm 1 includes two dependent phases: choosing one point as a center based on the probability and updating the sum of distances that points to their nearest centers. In a real MapReduce environment, each machine cannot see the entire input and cannot see the data on other machines either. Furthermore, there is no communication between machines during the Mapper phase and Reducer phase. Consequently, the MapReduce implementation of k-means++ initialization needs two sequential MapReduce jobs to choose one point as a center. The first job is responsible for choosing a center and the second one updates the sum of distances. However, it is a very expensive parallelized implementation of k-means++ initialization since it has to run at least $2k$ MapReduce jobs for k clusters, incurring high communication cost and I/O cost. Therefore, this is a poor solution when processing massive data although it is an $O(\alpha)$ approximation to the optimal of k-means.

In this section, we propose a fast k-means++ initialization algorithm with MapReduce. It uses only one MapReduce job and avoids the high cost of the above solution. Both Mapper phase and Reducer phase of our method run a k-means++ initialization algorithm. But their difference is that the algorithm in Mapper phase is a standard k-means++ initialization while in Reducer phase is a weighted k-means++ initialization. If both Mapper and Reducer phase run a standard k-means++ initialization, it incurs a large SSE from the optimal solution. Therefore, in Mapper phase of our method, except for choosing k centers, another important work is to compute the number of points that a center represents, this value is used to weigh the probability used in Reducer phase. In Reducer phase, we use the weighted probability to choose the initial k centers for k-means. The time complexity of our method is $O(knd)$ which is the same as the single iteration of k-means. Our method guarantees that the clustering result is an $O(\alpha^2)$ approximation to k-means and the detailed analysis is shown in the following section. Algorithm 2 and Algorithm 3 show the details of the proposed MapReduce k-means++ initialization algorithm.

Algorithm 2: Mapper phase of k-means++ initialization

Input: k, the number of clusters,
$X = \{x_1, x_2, \ldots, x_n\}$, a set of data points.
Output: $\langle num[i], c_i \rangle$, $i = 1, 2, ..., k$.
$num[i]$ denotes the number of points that center c_i represents.

$C \leftarrow \emptyset$
Choose one center x uniformly at random from X, $C = C \cup \{x\}$.
for $i = 1; i \leq k; i++$ **do**
    $num[i] = 0$;
while $|C| < k$ **do**
    Compute $D(x)$ between $x \in X$ and its nearest center that has already been chosen
    Choose x with probability $D(x) / \sum_{x \in X} D(x)^2$
    $C \leftarrow C \cup \{x\}$
for $i = 1; i \leq n; i++$ **do**
    find the nearest center $c_i \in C$ for x_i
    $num[i]++$
output $\langle num[i], c_i \rangle$

Algorithm 3: Reducer phase of k-means++ initialization

Input: k, the number of clusters,
X, the set of $\langle num, c \rangle$
Output: $C = \{c_1, c_2, ..., c_k\}$

$C \leftarrow \emptyset$
Choose one center x uniformly at random from X, $C = C \cup \{x\}$.
while $|C| < k$ **do**
    Compute $D(x)$ between $x \in X$ and its nearest center that has already been chosen
    Choose x with probability $numD(x) / \sum_{x \in X} D(x)^2$
    $C \leftarrow C \cup \{x\}$
Output C

System Issues. At a high level, we implement the MapReduce k-means++ initialization algorithm in Hadoop. There are two technical issues that must be handled during its implementation. Firstly, it requires the global information communication between map and reduce operation. To ensure that we can utilize two Hadoop features: Job Configuration and Distributed Cache. Job Configuration is a small piece of information communicated to every map and reduces task during task initialization. If a large amount of data must be communicated, Distributed Cache is the best choice. Second, both in map function and reduce function, data is processed in the form of tuple $\langle k, v \rangle$ and it flows into

map function and reduces function one by one. Therefore, it is impossible to implement k-means++ initialization algorithm in map function and reduce function. Fortunately, Hadoop provides the cleanup function for this purpose.

However, this algorithm incurs a lot of unnecessary distance computations. Thus, we will conduct a detailed analysis and propose an improved version in Section 5.

4.2 Theoretical Analysis

Although our algorithm is simple, it is high efficiency and has a good approximation of k-means. This section focuses on the analysis of our MapReduce k-means++ algorithm and achieves a theoretical guarantee. We firstly introduce some symbols and definitions used in this section which are shown in Table 1.

TABLE 1: Symbols and Definitions

Symbols	Definitions
X	the set of all data points
X_i	the set of data points processed by map task i
Y	the set of centers from all map tasks
$Y^{'}$	the multiset in Definition 2
$Y^{''}$	the set of centers chosen from X using k-means++
Y_i	the set of centers from map task i
C	the set of centers from reduce task
$\hat{C}$	the optimal set of centers chosen from X
$\hat{C}^*$	the optimal set of centers chosen from $Y^{'}$

Definition 1: Map $\varphi : X \to Y$. In our method, each map task runs a k-means++ algorithm and produces k centers. Thus, for any point $x \in X$, there always exists a center $y \in Y$ which is able to represent x best, that is the Euclidean distance between x and y is the smallest among all points. Because $|Y| \ll |X|$, a center in Y generally represents more than one point in X.

Definition 2: Multiset $Y^{'} = \{\varphi(x)|x \in X\}$. The element in $Y^{'}$ is allowed duplicate and the size of $Y^{'}$equals to the size of X. Y is a copy of $Y^{'}$ without duplicate elements.

Lemma 1: If clustering results are constructed with standard k-means++, then $SSE \leq \alpha SSE_{OPT}$, $\alpha = O(lnk)$, $SSE_{OPT} = \sum_{x \in X} \min_{\hat{c} \in \hat{C}} ||x - \hat{c}||^2$.

Proof: According to [7]. □

In fact, this lemma holds after only k-means++ initialization and k-means algorithm further decreases SSE.

Lemma 2: If the weighted k-means++ initialization algorithm is run on the Reducer phase, then

$$\sum_{x \in X} \min_{c \in C} ||\varphi(x) - c||^2 \leq \alpha \sum_{x \in X} \min_{\hat{c} \in \hat{C}} ||\varphi(x) - \hat{c}||^2$$

Proof: Note that $\varphi(x) \in Y^{'}$. After the Reducer phase, weighted k-means++ initialization algorithm generates k centers. If using the standard k-means++ algorithm, its solution is the α approximation to the optimal solution on Y, but not the α approximation to the optimal solution on $Y^{'}$. While, using weighted k-means++ initialization algorithm, although it works on Y, it considers the number of points in X that is presented by the center in Y. In fact, it works on $Y^{'}$. Therefore, based on Lemma 1, we have:

$$\sum_{x \in X} \min_{c \in C} ||\varphi(x) - c||^2 \leq \alpha \sum_{x \in X} \min_{\hat{c}^* \in \hat{C}^*} ||\varphi(x) - \hat{c}^*||^2 \quad (1)$$

Because $\hat{C}^*$ is the optimal solution to $Y^{'}$, $\hat{C}$ is the optimal solution to X and Y is the set of nk centers from X. Therefore,

$$\sum_{x \in X} \min_{\hat{c}^* \in \hat{C}^*} ||\varphi(x) - \hat{c}^*||^2 \leq \sum_{x \in X} \min_{\hat{c} \in \hat{C}} ||\varphi(x) - \hat{c}||^2 \quad (2)$$

That is, for the same number of centers, the centers chosen from $Y^{'}$ represent $Y^{'}$ better than the centers chosen from X. Combining (1) (2) and then the result follows. □

Lemma 3: If there are n map tasks that run k-means++ initialization algorithm and generate nk centers in total, then

$$\sum_{x \in X} \min_{c \in Y} ||x - c||^2 = \sum_{i=1}^{n} \sum_{x \in X_i} \min_{c \in Y_i} ||x - c||^2$$

$$\leq \sum_{x \in X} \min_{c \in Y^{''}} ||x - c||^2$$

Proof: Since the whole data X is divided into n splits (each split is 64MB by default), these splits are denoted by $X_1, ..., X_n$. Then each split is processed by one map task with k-means++ initialization algorithm to choose a set of k centers denoted by $Y_1, ..., Y_n$. Therefore,

$$\begin{aligned} &\sum_{i=1}^{n} \sum_{x \in X_i} \min_{c \in Y_i} ||x - c||^2 \\ &= \sum_{x \in X_1} \min_{c \in Y_1} ||x - c||^2 + ... + \sum_{x \in X_n} \min_{c \in Y_n} ||x - c||^2 \end{aligned} \quad (3)$$

If we use one machine to run k-means++ initialization algorithm on the whole data X and choose k centers $Y^{''}$. Then, we have,

$$\begin{aligned} &\sum_{x \in X} \min_{c \in Y^{''}} ||x - c||^2 \\ &= \sum_{i=1}^{n} \sum_{x \in X_i} \min_{c \in Y^{''}} ||x - c||^2 \\ &= \sum_{x \in X_1} \min_{c \in Y^{''}} ||x - c||^2 + ... + \sum_{x \in X_n} \min_{c \in Y^{''}} ||x - c||^2. \end{aligned} \quad (4)$$

Because k centers chosen from a small data set X_i represents X_i better than k centers from the whole data X. That is,

$$\sum_{x \in X_i} \min_{c \in Y_i} ||x - c||^2 \leq \sum_{x \in X_i} \min_{c \in Y^{''}} ||x - c||^2 \quad (5)$$

Combining (3) (4) (5) and then the result follows. □

Theorem 1: The MapReduce k-means++ initialization algorithm obtains a solution that is an $O(\alpha^2)$ approximation

to the optimal solution of k-means, that is.

$$\sum_{x\in X}\min_{c\in C}||x-c||^2 \le (\alpha^2+2\alpha)\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2$$

Proof: According to triangle inequality, we have

$$\begin{aligned}
&\sum_{x\in X}\min_{c\in C}||x-c||^2 \\
&\le \sum_{x\in X}\min_{c\in C}||\varphi(x)-c||^2+\sum_{x\in X}\min_{c\in Y}||x-c||^2 \\
&\le \alpha\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||\varphi(x)-\hat{c}||^2+\sum_{x\in X}\min_{c\in Y}||x-c||^2 \\
&\qquad\qquad (by\ Lemma2) \\
&\le \alpha(\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2+\sum_{x\in X}\min_{c\in Y}||x-c||^2) \\
&+\sum_{x\in X}\min_{c\in Y}||x-c||^2 \qquad (by\ triangle\ inequality) \\
&= \alpha\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2+(\alpha+1)\sum_{x\in X}\min_{c\in Y}||x-c||^2 \\
&\le \alpha\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2+(\alpha+1)\sum_{x\in X}\min_{c\in Y''}||x-c||^2 \\
&\qquad\qquad (by\ Lemma3) \\
&\le \alpha\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2+(\alpha+1)\alpha\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2 \\
&\qquad\qquad (by\ Lemma1) \\
&= (\alpha^2+2\alpha)\sum_{x\in X}\min_{\hat{c}\in\hat{C}}||x-\hat{c}||^2
\end{aligned} \tag{6}$$

For our MapReduce k-means++ initialization algorithm is an $O(\alpha^2)$ approximation to the k-means and the MapReduce k-means further decreases the squared distance error. Thus, our MapReduce k-means++ algorithm is an $O(\alpha^2)$ approximation to the optimal of k-means. □

Our proof is based on the triangle inequality. It should satisfy: for any three points x, y and z, $d(x,z) \le d(x,y)+d(y,z)$.

The proof of Theorem 1 uses triangle inequality twice. Three elements of the first one are: (1) After the weighted k-means++ initialization algorithm in Reducer phase, we can use the centers to represent all centers, i.e., $\sum_{x\in X}\min_{c\in C}||x-c||^2$; (2) Because these centers are generated from the map task output, we can also use them to represent the points of map output, i.e., $\sum_{x\in Y}\min_{c\in C}||x-c||^2$. However, the cardinality of X and Y is different and the triangle inequality is not satisfied on this condition. So, we Definition 1 and Definition 2 to replace $\sum_{x\in Y}\min_{c\in C}||x-c||^2$ with $\sum_{x\in X}\min_{c\in C}||\varphi(x)-c||^2$; (3) After the Mapper phase, we use the nk centers to represent all points, i.e., $\sum_{x\in X}\min_{c\in Y}||x-c||^2$. Combining (1) (2) (3), the triangle inequality is satisfied. The second triangle inequality is used in line 4; its analysis is the same as the first one and we cannot give it due to space constraints.

For the triangle inequality, Theorem 1 enlarges the upper bound of approximation. In fact, this approximation is much less than $\alpha^2+2\alpha$, especially after the post processing of k-means algorithm with the centers from k-means++ initialization.

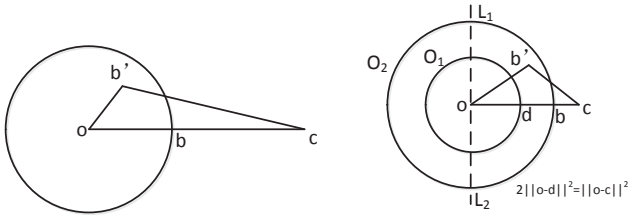
5 Improved MapReduce k-means++ Initialization

We observe that the proposed MapReduce k-means++ initialization algorithm in Hadoop is very expensive. Although each map task only processes 64MB data and the reduce task processes the data that is much smaller than the whole data, it is still expensive due to the following reasons: (1) the inherent iteration nature of k-means++ initialization algorithm. Although our algorithm puts iteration operation on a single machine and avoids a lot of communication and I/O cost suffering from k rounds of MapReduce jobs on multiple machines, the number of iterations cannot be reduced. (2) Most importantly, once a new center is chosen, it needs to determine whether the points belonging to other clusters should be assigned to the new center or not. Usually, for each iteration, both map task and reduce task have to scan all the data to compute the distance between the point and the new center, and then compare the new distance with old distance to determine the assignment of this point. For example, assuming a machine processes n points to choose k centers, it needs to compute $O(nk^2)$ times in the worst case. When n and k are very large, this operation is very expensive. This motivates us to explore an improved algorithm which can choose k centers quickly and inexpensively.

As mentioned above, k-means++ initialization algorithm need iterate k times to generate k centers and its iteration time cannot be reduced. However, it is unnecessary to calculate the distances of all points when a new center is determined. For example, if a point is far away from the new center, that is this point is represented better by the old center than by the new center, i.e., not necessary to calculate the distance between this point and the new center. In this section, we propose an improved MapReduce k-means++ initialization algorithm which can prune a lot of redundant distance computations based on the triangle inequality. We have the following theorem and corollary.

Theorem 2: Let A be the set of centers that have been chosen, $\forall o \in A$, B is the set of points that are represented by o, c is a new center just determined, $b \in B$ denotes the point that is represented worst by o, i.e., $\forall b' \in B$, $||o-b'||^2 \le ||o-b||^2$. If $||o-c||^2 \ge 2||o-b||^2$, then the distances of the points in B to the new center c need not be calculated, i.e., they still belong to the old center o, not the new center c.

Proof: As shown in Fig. 1(a), o is an arbitrary center in A, b and b' are two arbitrary points in B, b is also the worst point that is represented by o, c is a new center. The purpose of this theorem is to avoid the distance computation between b' and c, i.e. not to compare $||b'-o||^2$ with $||b'-c||^2$ and determine the assignment of b'. Because of $||b'-$



(a) Theorem 2 (b) Corollary 1

Fig. 1: The Schematic Diagram of Theorem2 and Corollary1

$c||^2 > ||o-c||^2 - ||o-b^{'}||^2$ and $||o-b^{'}||^2 < ||o-b||^2$, hence $||b^{'}-c||^2 > ||o-c||^2 - ||o-b||^2$. And because the condition $||o-c||^2 \geq 2||o-b||^2$, clearly, we can derive $||b^{'}-c||^2 > ||o-c||^2 - ||o-b||^2 > ||o-b||^2 > ||o-b^{'}||^2$. □

Note that the distance between b and o gives the upper bound. Any point whose distance to o is smaller than it need not be calculated and they all must retain assigned to center o. From the aspect of geometry, the old center o and the distance $||o-b||^2$ determine a circle, the new center c and the smallest distance $||o-b||^2$ determine another circle, if there is no intersection between two circles, i.e. $||o-c||^2 \geq 2||o-b||^2$, the points belong to one circle still belong to it.

Corollary 1: If $||o-c||^2 < 2||o-b||^2$, there exists a point $d \in B$ whose distance to o satisfies $||o-d||^2 \leq ||o-b||^2$. Any points in B whose distance to o are less than or equal to $||o-d||^2$ construct a set $C \subseteq B$. If $||o-c||^2 \geq 2||o-d||^2$, then the distances of the points in C to the new center c need not be calculated, that is, they still belong to the old center o, not the new center c.

Corollary 1 is the complement of Theorem 2. As shown in Fig. 1(b), we use Corollary 1 as follows. When a new center c is determined and not all points in B are still represented by the center o, i.e., $||o-c||^2 < 2||o-b||^2$. We can use Corollary 1 to determine a point d (it may be a real point or a virtual point) that satisfies $||o-c||^2 = 2||o-d||^2$, then the points within circle O_1 are still represented by center o and these distances are not necessary to calculate.

With Theorem 2 and Corollary 1, the improved initialization method avoids a lot of redundant distance computations. However, the triangle inequality has its own drawback in reducing the distance computation. For example, as shown in Fig. 1(b), using the triangle inequality, the improved initialization method only avoid the distance computation of points within circle O_1. While the points within circle O_1 and circle O_2 need to be calculated. In fact, the points within circle O_1, O_2 and at the left of line L_1L_2 need not to be calculated either. If using a more accurate angle, the pruning power will be strengthened. In the future, we will consider using the triangle to further optimize our algorithm.

6 Experimental Results

In this section, the experimental results of our proposed algorithms are presented. Noticing that the main merits of MapReduce k-means++ initialization are: (1) efficiency, it is much more efficient than the parallel implementation of k-means++ initialization since it only requires one MapReduce job to obtain k centers; (2) good approximation, although MapReduce k-means++ obtains an $O(\alpha^2)$ approximation to the k-means which is a little worse than k-means++ ($O(\alpha)$ approximaition), it is still much better than MapReduce k-means with random initialization method. No matter MapReduce random initialization or MapReduce k-means++ initialization, the ultimate purpose is to serve the k-means. Thus, we evaluate MapReduce k-means algorithm with different initialization methods. At last, for the expensive distance computation of MapReduce k-means++ initialization, we have proposed an improved MapReduce initialization algorithm, which will also be evaluated in this section.

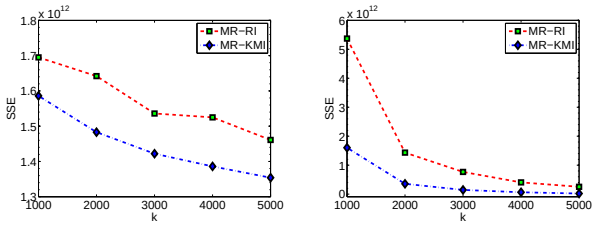
All experiments are performed on a homogeneous Hadoop cluster running the latest stable version of Hadoop 0.20.2. The cluster consists of 12 machines with 1 master node and 11 slave nodes. Each node has 2 AMD Opteron 2212 2.00GHz CPUs, 8GB of RAM, 80GB SCSI HDD, Intel 82551 10/100Mbps Ethernet Controller. The operating system of each node is Ubuntu 10.10 server 64bit and per Hadoop daemon is allocated 1GB memory. This cluster has 1 TaskTracker and 1 DataNode daemon running on each slave, and a single NameNode and JobTracker daemon on the master. All machines are directly connected to a 100Mbps switch. We configure 2 map slots and 2 reduce slots on each node. The DFS chunk size is 64MB.

We conduct the experiments on the following datasets:

- Oxford Buildings Dataset: This is a real dataset consists of 5062 images collected from Filckr by searching for particular Oxford landmarks. A large number of 128-dimension SIFT features is extracted from each image and there are more than 17 million features in total. Its size is 5.67GB.
- Synthetic Dataset: We also generate a random dataset of points and each point is also 128-dimension. It consists of 5000 centers and the points are randomly generated around the centers. Each cluster contains about 4000 points. This dataset includes more than 20 million points and its size is about 15GB.

The following approaches are evaluated in the experiments.

- Efficiency of MapReduce k-means++ initialization and scalable k-means++ initialization.
- Approximation of MapReduce k-means++ initialization and MapReduce random initialization.
- Approximation of MapReduce k-means++ and MapReduce k-means.
- Efficiency of MapReduce k-means++ initialization and improved MapReduce k-means++ initialization.



(a) Oxford Dataset (b) Synthetic Dataset

Fig. 2: SSE Comparison of MR-RI and MR-KMI on Different Datasets

6.1 Efficiency of MapReduce k-means++ Initialization and scalable k-means++ Initialization

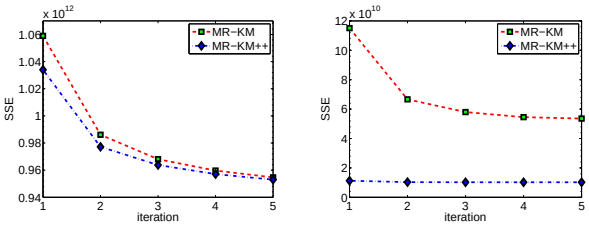
As mentioned above, our MapReduce k-means++ initialization algorithm only uses one MapReduce job to obtain k centers, while the scalable k-means++ needs too many MapReduce jobs. However, our algorithm does not avoid the iteration nature of k-means++ initialization algorithm. In fact, it replaces the iteration on multiple machines with the iteration on a single machine. Therefore, it saves a lot of communication and I/O cost and usually these are the bottlenecks in the cloud computing environment. From this perspective, our MapReduce k-means++ initialization algorithm is much more efficient than the scalable k-means++ initialization.

6.2 Approximation of MapReduce k-means++ Initialization and MapReduce Random Initialization

In this experiment, we compare the approximation of MapReduce Random Initialization (MR-RI) algorithm and MapReduce k-means++ Initialization (MR-KMI) algorithm on different datasets. The results are summarized in Fig. 2. To show the results more clearly, we also summarized the results of synthetic dataset in Table 2. From Fig. 2 we can see, no matter real dataset and synthetic dataset, the approximation of both MR-RI and MR-KMI become better with the increase of centers. But comparing MR-KMI with MR-RI, the former can achieve a much better approximation than the latter. Especially on Oxford dataset (Fig. 2(a)), although the approximation of both MR-KMI and MR-RI is at the same magnitude (10^{12}), the SSE of MR-KMI is about 10^{11} less than the MR-RI at the same number of centers. And for the random nature of MR-RI, its approximation shows the obvious fluctuation, while MR-KMI shows a stable approximation trend with the increase of centers. For synthetic data (Fig. 2(b) and Table 2), the approximation of MR-KMI is much better than MR-RI when k varies from 1000 to 5000 and SSE is over two orders of magnitude (from 10^{10} to 10^{12}). Especially, when $k = 5000$, SSE of MR-KMI is about $1/13$ of the MR-RI. The reasons are that the points of the synthetic dataset are generated around the centers to create 5000 clusters and MR-RI is difficult to capture the above characteristics for its random nature.

TABLE 2: SSE of MR-RI and MR-KMI on Synthetic Dataset

k	1000	2000	3000	4000	5000
MR-RI	5.37e12	1.43e12	7.67e11	4.05e11	2.54e11
MR-KMI	1.60e12	3.56e11	1.45e11	6.44e10	2.00e10



(a) Oxford Dataset (b) Synthetic Dataset

Fig. 3: SSE Comparison of MR-KM and MR-KM++ on Different Datasets

6.3 Approximation of MapReduce k-means++ and MapReduce k-means

This section compares the approximation of MapReduce k-means (MR-KM) and MapReduce k-means++ (MR-KM++) on Oxford dataset and synthetic dataset. k is set to 5000 and these centers are from the previous experiment. Both MR-KM and MR-KM++ iterate 5 times in this experiment. The results are summarized in Fig. 3. We also show the results of the synthetic data in Table 3. As shown in Fig. 3, SSE of both MR-KM and MR-KM++ decreases as the iteration varies from 1 to 5. It also follows the gradient descent nature of k-means. But our algorithm MR-KM++ has a better approximation than MR-KM. For Oxford dataset (Fig. 3(a)), the maximum SSE difference between MR-KM and MR-KM++ occurs when they iterate once. It is about $3e10$ ($1.03e12$ vs $1.06e12$). With the increase of iteration, this difference becomes small and tends to be stable. For example, when iteration is 5, the difference between them is the smallest ($1.7e9$). For synthetic dataset (seeing Fig. 3(b) and Table 3), compared to MR-KM, MR-KM++ has a huge advantage in approximation to k-means and SSE of MR-KM is about 10 times bigger than that of MR-KM++ (iteration=1). When iteration is from 3 to 5, the SSE of MR-KM++ is almost constant (about $1.02e10$). From the above analysis, our MapReduce k-means++ has a fast convergence speed.

TABLE 3: SSE of MR-KM and MR-KM++ on Synthetic Dataset

Iteration	1	2	3	4	5
MR-KM	1.15e11	6.66e10	5.80e10	5.45e10	5.36e10
MR-KM++	1.13e10	1.03e10	1.024e10	1.023859e10	1.023856e10

With 5 times of iterations, SSE of MR-KM++ is stable, not only on real dataset but also on synthetic dataset. We use $SSE_{initialization}/SSE_{iteration=5}$ to evaluate the approximation of the initialization method to the final

TABLE 4: The Number of Distance Calculations on Oxford Dataset

	$k = 1000$		$k = 3000$		$k = 5000$	
Algorithm	map	reduce	map	reduce	map	reduce
MR-KMI	1.63e10	9.39e7	4.90e10	8.46e8	8.17e10	2.35e9
IMR-KMI	1.56e10	8.24e7	4.23e10	6.89e8	6.64e10	1.82e9

TABLE 5: The Number of Distance Calculations on Synthetic Dataset

	$k = 1000$		$k = 3000$		$k = 5000$	
Algorithm	map	reduce	map	reduce	map	reduce
MR-KMI	2.00e10	2.24e8	6.00e10	2.02e9	1.00e11	5.60e9
IMR-KMI	9.17e8	1.82e6	2.66e9	6.13e6	4.35e9	1.06e7

solution. They are 1.53 (MR-RI) and 1.41 (MR-KMI) on Oxford dataset, 47.39 (MR-RI) and 1.96 (MR-KMI) on synthetic data. So our initialization method has a good approximation to the final solution.

6.4 Efficiency of MapReduce k-means++ Initialization and Improved MapReduce k-means++ Initialization

Because the point in the dataset we use in our experiments is 128-dimension and the size of the dataset is very large, MapReduce k-means++ initialization takes a long time to complete, even k is small. Therefore, we compare the number of distance calculations in the MapReduce k-means++ initialization (MR-KMI) and the improved MapReduce k-means++ initialization (IMR-KMI). We consider the number of distance calculations in Maper phase and Reducer phase. Table 4 and Table 5 summarize the results of the experiments on Oxford dataset and synthetic dataset respectively.

The experimental results indicate that with the Theorem 2 and Corrollary 1 IMR-KMI reduces a lot of distance computations over MR-KMI on both dataset. It achieves 99% in Reducer phase and 96% in Mapper phase on synthetic dataset. And this reduction is stable with the increase of k. While for Oxford dataset, this reduction is slighter than synthetic dataset, but the maximum reduction is up to 18.6% ($k = 5000$) in Mapper phase and 22.4% ($k = 5000$) in Reducer phase. More importantly, this reduction becomes more significant with the increase of k.

7 Conclusions

This paper investigates the important problem of clustering and studies k-means++ algorithm in a very large data situations. We develop k-means++ initialization with MapReduce efficiently and propose the MapReduce k-means++ algorithm. The MapReduce initialization algorithm uses only one MapReduce job to choose k centers. The standard k-means++ initialization and weighted k-means++ initialization are applied to the Mapper phase and Reducer phase, respectively. For the reduction of MapReduce jobs, our algorithm saves a lot of communication and I/O cost. Furthermore, the proposed algorithm provides an $O(\alpha^2)$ approximation to the optimal solution of k-means. Considering the expensive distance computation of our initialization method, we also propose a pruning strategy using triangle inequality. Extensive experiments on real and synthetic data have been conducted. The results indicate that the proposed MapReduce k-means++ algorithm is much efficient and has a good approximation. Meanwhile, the pruning strategy can further improve the initialization method.

Acknowledgment

This work is supported by the the National Science Foundation for Distinguished Young Scholars of China under grant No. of 61225010, National Nature Science Foundation of China (Nos. 61173162, 61173165, 61370199, 61300187, 61300189 and 61370198), New Century Excellent Talents (No. NCET-10-0095), the Fundamental Research Funds for the Central Universities(Nos. 2013QN044 and 2012TD008).

References

[1] E. Chandra and V. P. Anuradha, "A survery on clustering algorithms for data in spatial database management systems," *Computer Applications*, vol. 24, no. 9, pp. 19–26, 2011.

[2] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 505–516.

[3] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1–58, 2009.

[4] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg, "Indexing and searching 100m images with map-reduce," in *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, 2013, pp. 17–24.

[5] J. F. Lu, J. B. Tang, Z. M. Tang, and J. Y. Yang, "Hierarchical initialization approach for k-means clustering," *Pattern Recogn. Lett.*, vol. 29, no. 6, pp. 787–795, 2008.

[6] S. Y. T. Onoda, M. Sakai, "Careful seeding method based on independent components analysis for k-means clustering," *Emerging Technologies in Web Intelligence*, vol. 4, no. 1, pp. 51–59, 2012.

[7] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[8] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *PVLDB*, vol. 5, no. 7, pp. 622–633, 2012.

[9] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, 2004, pp. 10–10.

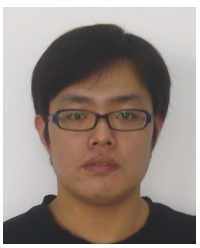
[10] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Proceedings of the 1st International Conference on Cloud Computing*, 2009, pp. 674–679.

[11] S. Papadimitriou and J. Sun, "Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 512–521.

[12] A. Ene, S. Im, and B. Moseley, "Fast clustering using mapreduce," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 681–689.

[13] R. L. Ferreira Cordeiro, C. Traina, Junior, A. J. Machado Traina, J. López, U. Kang, and C. Faloutsos, "Clustering very large multi-dimensional datasets with mapreduce," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 690–698.

[14] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 810–818.
[15] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 285–296, Sep. 2010.
[16] N. Ailon, R. Jaiswal, and C. Monteleoni, "Streaming k-means approximation," in *Advances in Neural Information Processing Systems*, 2009, pp. 10–18.
[17] M. A. Hasan, V. Chaoji, S. Salem, and M. J. Zaki, "Robust partitional clustering by outlier and density insensitive seeding," *Pattern Recogn. Lett.*, vol. 30, no. 11, pp. 994–1002, 2009.
[18] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm." *Expert Syst. Appl.*, vol. 40, pp. 200–210, 2013.
[19] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.
[20] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "A local search approximation algorithm for k-means clustering," *Comput. Geom. Theory Appl.*, vol. 28, no. 2-3, pp. 89–112, 2004.
[21] A. Amit Kumar, Y. Sabharwal, and S. Sen, "A simple linear time $(1 + \epsilon)$-approximation algorithm for k-means clustering in any dimensions," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004, pp. 454–462.
[22] M. Song and S. Rajasekaran, "Fast k-means algorithms with constant approximation," in *Algorithms and Computation*, 2005, pp. 1029–1038.
[23] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy, "The effectiveness of lloyd-type methods for the k-means problem," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006, pp. 165–176.



Yujie Xu received the master's degree from Dalian Maritime Universion, China in 2010. He is currently completing a Ph.D. in the department of Information Science and Technology at the University of Dalian Maritime, China. His main research interests include large-scale data processing and Cloud Computing.

Wenyu Qu received the bachelor's and master's degrees from Dalian University of Technology, China in 1994 and 1997, and the doctorate degree from Japan Advanced Institute of Science and Technology in 2006. She is a professor at the School of Information and Technology, Dalian Maritime University, China. She was a lecturer in Dalian University of Technology from 1997 to 2003. Her research interests include cloud computing, computer networks and information retrieval. She has published more than 80 technical papers in international journals and conferences. She is on the committee board for a couple of international conferences.

Zhiyang Li received the bachelor's and doctoral degrees from Dalian University of Technology in China in 2004 and 2011. He is a assistant professor in School of Information and Technology, Dalian Maritime University, China. His research interests include cloud computing, information retrieval and computer vision.

Geyong Min is a Professor of Computer Science in the Department of Computing at the University of Bradford, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Next Generation Internet, Wireless Communications, Multimedia Systems, Information Security, Ubiquitous Computing, Modelling and Performance Engineering. His recent research has been supported by UK EPSRC, Royal Society, Nuffield Foundation, and European FP. Prof. Min has published over 200 research papers in prestigious international journals, including IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Communications, IEEE Transactions on Wireless Communications, and IEEE Transactions on Multimedia, and in reputable international conferences, such as ICDCS, IPDPS, GLOBECOM, and ICC. He is an Editorial Board member of 9 international journals. He served as the Guest Editor for 17 International Journals and was the Chair or Vice-Chair of 30 international conferences/workshops. He was awarded the Outstanding Leadership Awards from IEEE International conferences CIT'2010/ScalCom'2010/ICESS'2010, ScalCom'2009, HPCC'2008

Keqiu Li received the bachelor's and master's degrees from the Department of Applied Mathematics at the Dalian University of Technology in 1994 and 1997, respectively. He received the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology in 2005. He also has two-year postdoctoral experience in the University of Tokyo, Japan. He is currently a professor in the School of Computer Science and Technology, Dalian University of Technology, China. He has published more than 100 technical papers, such as IEEE TPDS, ACM TOIT, and ACM TOMCCAP. He is an Associate Editor of IEEE TPDS and IEEE TC. He is a senior member of IEEE. His research interests include internet technology, data center networks, cloud computing and wireless networks.

Zhaobin Liu received his Ph.D. in Computer Science from Huazhong University of Science and Technology in China in 2004. He is an Associate Professor in School of Information Science and Technology, Dalian Maritime University, China. His research interests include cloud computing, cloud storage and I/O systems. He has more than 50 publications in journals or international conferences. international conferences.