Parallel implementation of K-Means clustering algorithm based on mapReduce computing model of hadoop

Hongbo Xu, Nianmin Yao, Qilong Han, Haiwei Pan

College of Computer Science and Technology,
Harbin Engineering University, Harbin, China

Abstract
In recent years, data clustering has been studied extensively and a lot of methods and theories have been achieved. However, with the development of the database and the popularity of Internet, a lot of new challenges such as Big Data and Cloud Computing lie in the research on data clustering. The paper presents a parallel k-means clustering algorithm based on MapReduce computing model of Hadoop platform. The MapReduce computing model has two phases: a map phase and a reduce phase. The map phase calculates the distances between each point and each cluster and assigns each point to its nearest cluster. All the points which belong to the same cluster are sent to a single reduce phase. The reduce phase calculates the new cluster centers for the next MapReduce job. Experiments on different sizes of datasets demonstrate that the proposed algorithm shows good performance on the speedup, the scaleup and the sizeup. Thus it fits to data clustering on huge datasets.
Key words: DATA CLUSTERING, K-MEANS CLUSTERING ALGORITHM, MAPREDUCE, PARALLEL COMPUTING

Introduction

With mature database technologies and universal data applications, business, enterprises, research institutions or government departments have accumulated a large amount of data stored in different forms. How to store and handle these massive amounts of data, as well as further dig out useful knowledge which can guide the applications has become a thorny issue.

Data mining is also known as knowledge discovery in databases. Data mining extracts unknown potential valuable information or pattern from large, incomplete, noisy, blur, random data. With the rapid development of computer technology and the popularity of the network, people have more opportunities to use convenient way to exchange information with the outside world. However, the influx of large amounts of data increases the difficulty of obtaining useful information. How to obtain valuable information from large amounts of data brings problems of implementing data mining system. Due to the high complexity of processing these data, the computing power of the system is difficult to meet the requirements. At this point, the limited computing resources which traditional stand-alone server can provide often cannot meet the requirements. There need distributed computing technology to achieve large-scale parallel computing.

Data clustering is an important research topic in the field of data mining. Data clustering analyzes the data and finds useful information. Based on the thinking of “Like attracts like”, the so-called data clustering is a process which divides

the collection of physical or abstract objects into multiple classes or clusters. A cluster is a collection of data objects. Data objects in the same cluster are as similar as possible. However, data objects from different clusters are as different as possible. [1] By clustering, one can identify dense and sparse areas and find an interesting correlation between the overall distribution pattern and data attributes. The k-means algorithm belongs to a basic division method of clustering analysis.

In the face of massive data, existing clustering algorithms have encountered the bottleneck in time and space complexity. This is one of the questions needed to be solved urgently in the field of clustering algorithms. An idea to solve this problem is to apply the parallel processing technology to data clustering, design efficient parallel clustering algorithms, and improve the performance of data clustering algorithms handling massive amounts of data.

The abstraction of parallel programming models (PThread, MPI, PVM and OpenMP etc.) in traditional high-performance computing is not high. Developers need to be familiar with the underlying configuration and parallel implementation details.

Cloud computing has been gotten widespread attention as an emerging business model. [2] Hadoop is a cloud computing platform which can more easily develop and parallel process large data. [3] Its main features include strong expansion capacity, low cost, high efficiency and good reliability. Hadoop platform consists of two parts: Hadoop Distributed File System (HDFS) and MapReduce computing model. [4] On the basis of cloud computing platform Hadoop, the paper presents a parallel k-means clustering algorithm based on MapReduce computing model.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 introduces the MapReduce computing model. Section 4 presents a parallel k-means clustering algorithm based on MapReduce computing model. Section 5 shows experimental results and evaluates the parallel algorithm with respect to speedup, scaleup, and sizeup. Finally Section 6 gives the conclusions.

Related Work

Data set X={x_1, x_2, …, x_i, …, x_n} contains n d-dimensional data points. Each data point belongs to d-dimensional data space, which is defined as xi∈Rd. The variable k is the number of data subsets to be generated. The k-means clustering algorithm organizes the data objects into k divisions, which is defined as C={c_k, k=1, 2, …, K}. Each division represents a cluster c_k. Each cluster has a cluster center μk. Select the Euclidean distance as the similarity criterion, and calculate the square of the distance between the points in the cluster and the cluster center. The square of the distance is defined as

$$J(c_k) = \sum_{x_i \in C_k} || x_i - \mu_k ||^2$$

The goal of clustering makes the minimum of the total distance of all clusters. The total distance of all clusters is defined as

$$J(C) = \sum_{k=1}^{K} J(c_k)$$

$$J(C) = \sum_{k=1}^{K} J(c_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} || x_i - \mu_k ||^2 = \sum_{k=1}^{K} \sum_{i=1}^{n} d_{ki} || x_i - \mu_k ||^2$$

$$d_{ki} = \begin{cases} 1, if\ x_i \in c_i \\ 0, if\ x_i \notin c_i \end{cases}$$

Apparently according to the least squares method and the Lagrange principle, the cluster center μk is the average of data points in the cluster ck. The k-means clustering algorithm starts from initial k categories, and assigns each data point to each category in order to reduce the total sum of squared distances. With the increase of the number of the categories, the total sum of squared distances tends to reduce. When k is equal to n, J(C) is 0. Therefore, the total sum of squared distances obtains the minimum only under the determined number k.

Paper [5] presented a clustering algorithm based on MPI (Message Passing Interface). Because MPI uses the way of inter-process communication to coordinate parallel computing, this leads to lower parallel efficiency, memory overhead. Paper [6] presented the parallelization of a k-means algorithm based on PVM (Parallel Virtual Machine). However the algorithm is limited by the system and lacks flexibility. Paper [7] used multi-core CPU platform to improve the clustering speed. Paper [8] presented a fast clustering algorithm based on GPU (Graphics Processing Unit). Paper [9] presented an efficient parallel clustering algorithm in a high-performance cluster environment. These solutions which paper [7-9] presented are based on costly high-performance hardware. These solutions cannot make large-scale promotion. Paper [10] presented a clustering algorithm using data and task parallelism. The disadvantage is that communication overhead between nodes is high.

MapReduce Computing Model

Hadoop is a distributed basic framework developed by the Apache Foundation. Users can develop distributed applications without knowing

distributed underlying details. Hadoop takes full advantage of the power of high-speed computing and mass storage of clusters. Hadoop implements a distributed file system (HDFS, Hadoop Distributed File System). HDFS has a high fault tolerance feature, and is designed to be deployed in low-cost hardwares. And it provides a high transmission rate to access the application with large data sets. HDFS relaxes POSIX requirements, accesses data in the file system in the form of a file stream.

Hadoop consists of many elements. Its bottom is Hadoop Distributed File System (HDFS). HDFS stores all files on the storage nodes in the clusters. The above layer of HDFS is the MapReduce engine.

HDFS is a distributed file system with a high degree of fault tolerance, and suitable for the deployment on the low-cost machines. HDFS provides the high-throughput access to data, and is very suitable for the application of large-scale data sets.

The architecture of HDFS is shown in Figure 1. HDFS uses the Master / Slave architecture. HDFS mainly consists of the following components: Client, NameNode, Secondary NameNode and DataNode. These components are described separately.

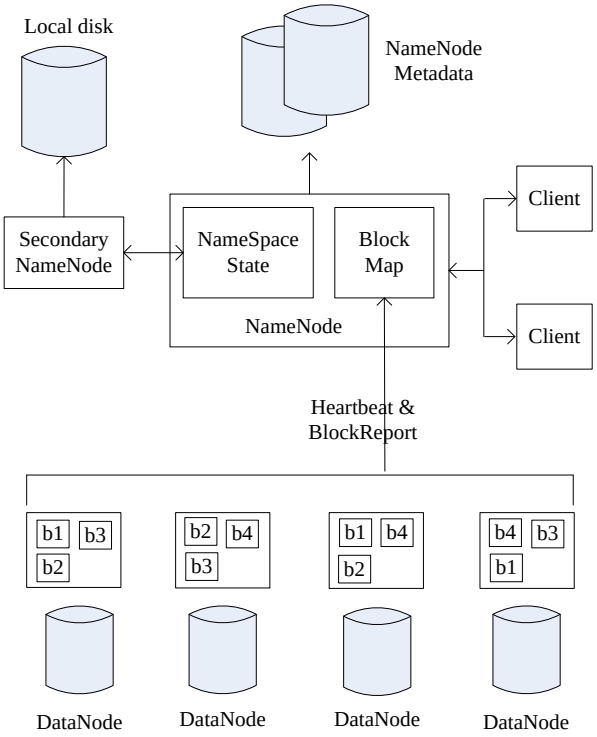

Figure 1. The architecture diagram of HDFS

1) Client

Client interacts with NameNode and DataNode to access the files on HDFS. Client provides a file system interface which is similar with POSIX for users to call. It is the manifold of the system.

2) Name Node

There is only one NameNode in the cluster of Hadoop. It is responsible for the management of the file directory tree and associated metadata information. The metadata information is stored on the local disk in the form of the file fsimage (HDFS metadata mirror file) and the file editlog (HDFS file change log). When HDFS restarts, two files are reconstructed out. In addition, NameNode is also responsible for monitoring the health status of each DataNode. Once some DataNode faild,

then HDFS removes the DataNode and backups the data on the DataNode.

3)Secondary NameNode

The most important task of Secondary NameNode isn't the Hot Backup of the metadata of NameNode. But it periodically merges the file fsimage and the file editlog, and transfers these files to NameNode. It should be noted that in order to reduce the pressure of NameNode, NameNode itself doesn't marge the file fsimage and the file editlog, but orders Secondary NameNode to complete.

4)DataNode

In general, DataNode is installed on each slave node. It is responsible for actual data storage. DataNode regularly reports data information to NameNode. DataNode organizes file contents into fixed size blocks. By default, the block size is 64MB. When users upload large files to HDFS, the files will be cut into several blocks which are stored on different DataNodes. Meanwhile, in order to ensure reliable data, the same block will be written into several different DataNodes. The stored procedures are transparent to the users after the files are cut.

Like HDFS, MapReduce of Hadoop also adopts Master / Slave architecture, specifically as shown in Figure 2.

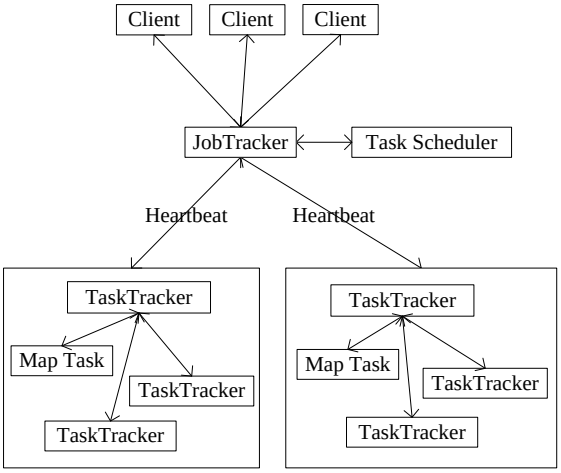

Figure 2. The architecture diagram of MapReduce of Hadoop

1) Client

User-written programs of MapReduce are submitted to JobTracker by Client. Meanwhile, users can view running states of jobs by the interfaces provided by Client. The jobs represent the programs of MapReduce in Hadoop. A program of MapReduce can correspond to several jobs, and each job will be broken into several Map / Reduce tasks.

2) JobTracker

JobTracker is mainly responsible for resource monitoring and job scheduling. JobTracker monitors the health of TaskTrackers and jobs. Once JobTracker finds failures, it will transfer the respective tasks to other nodes. Meanwhile, JobTracker will track the progress of the task, the usage of resource, and tell this information to Task Scheduler. When a resource is idle, Task Scheduler selects the appropriate task to use the resource. In Hadoop, Task Scheduler is a pluggable module. The user can design the appropriate scheduler according to the needs.

3) TaskTracker

TaskTracker will report periodically the usage of resources and the schedule of tasks to JobTracker through Heartbeat. Meanwhile, TaskTracker receives the commands sent from JobTracker, and performs the appropriate actions such as starting a new task, killing tasks and so on. TaskTracker divides resources on the node into Slots with the same amount. Slot represents computing resources such CPU, memory and so on. After a task gets a slot, it has the opportunity to run. The role of the scheduler is to assign idle slots on each TaskTracker to the tasks. The slot is divided into two kinds, map slot and resource slot which Reduce task and Map task use respectively. TaskTracker defines the degree of task concurrency by the number of slots . The number is a configurable parameter.

4) Task

Task is divided into two kinds, Map task and Reduce task. TaskTracker starts a task. HDFS uses the fixed-size block as the basic unit of data storage, and for MapReduce, its processing unit is the split. The correspondence between the split and the block is shown in Figure 3.

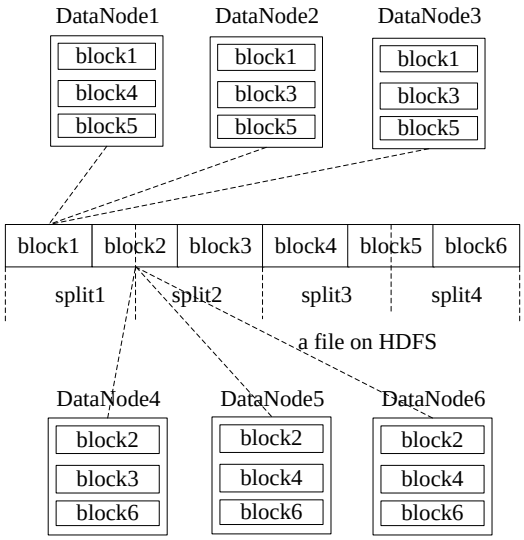

Figure 3. The correspondence between the split and the block

MapReduce is a programming paradigm that expresses a large distributed computation as a sequence of distributed operations on data sets of key/value pairs. The MapReduce framework of Hadoop harnesses a cluster of machines and executes user defined MapReduce jobs across the nodes in the cluster. MapReduce is composed by JobTrackers and TaskTrackers. A MapReduce computation has two phases, a map phase and a reduce phase. The input to the computation is a data set of key/value pairs.

In the map phase, the framework splits the input data set into a large number of fragments and assigns each fragment to a map task. The framework also distributes the many map tasks across the cluster of nodes on which it operates. Each map task consumes key/value pairs from its assigned fragment and produces a set of intermediate key/value pairs. For each input key/value pair (k, v), the map task invokes a user defined map function that transmutes the input into a different key/value pair (k', v').

Following the map phase the framework sorts the intermediate data set by key and produces a set of (k', v') tuples so that all the values associated with a particular key appear together. It also partitions the set of tuples into a number of fragments equal to the number of reduce tasks.

In the reduce phase, each reduce task consumes the fragment of (k', v') tuples assigned to it. For each such tuple it invokes a user-defined reduce function that transmutes the tuple into an output key/value pair (k, v). Once again, the framework distributes the many reduce tasks across the cluster of nodes and deals with shipping the appropriate fragment of intermediate data to each reduce task.

Tasks in each phase are executed in a fault-tolerant manner. If node(s) fail in the middle of a computation the tasks assigned to them are re-distributed among the remaining nodes. Having many map and reduce tasks enables good load balancing and allows failed tasks to be re-run with small runtime overhead. Figure 4 shows MapReduce computing model.

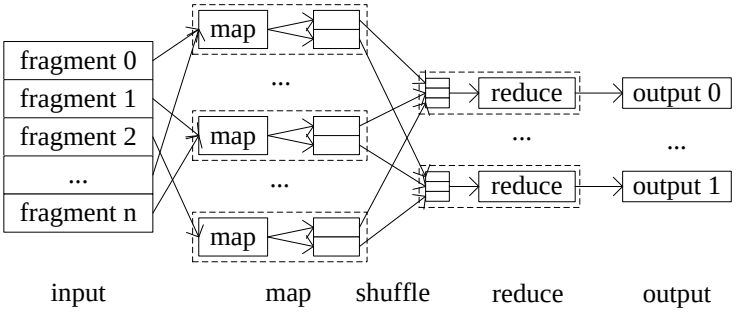

Figure 4. MapReduce computing model

Parallel k-means clustering algorithm

The k-means algorithm is a classical clustering algorithm widely used in scientific research and industrial applications. The k-means clustering algorithm is an indirect clustering method based on similarity measure between samples.

The k-means clustering algorithm is relatively simple. First, choose k objects from n data objects as the initial cluster centers. For the rest of the other data objects, according to their similarity (distance) with these cluster centers, respectively assign them to the most similar clusters which cluster centers represent. Then calculate the mean of all data objects in each cluster as the new cluster center. This process is repeated until the standard measure function begins to converge.

Generally use Euclidean distance to calculate the distance between data objects xi(xi1, xi2, …, xin) and xj(xj1, xj2, …, xjn). The specific formula follows:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + ... + (x_{in} - x_{jn})^2}$$

Figure 5 shows the serial calculation process of the k-means clustering algorithm.

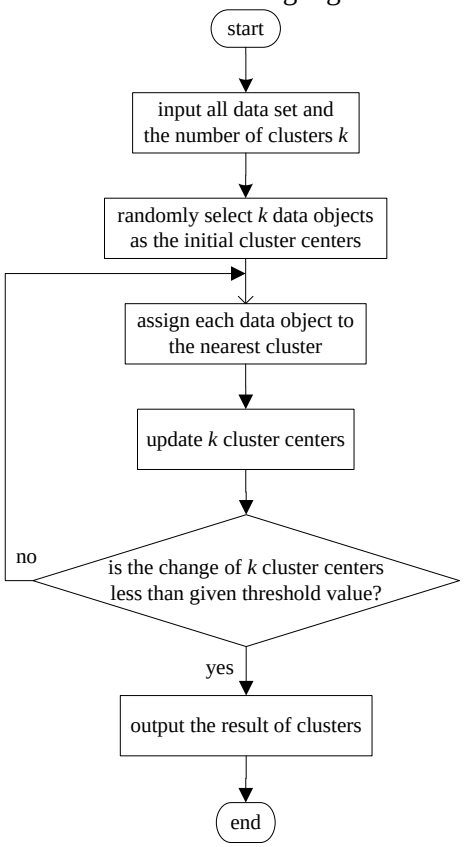

Figure 5. The serial calculation processes

The k-means algorithm has the advantage of handling large data sets. The k-means algorithm is scalable and efficient. The time complexity of the serial k-means clustering algorithm is relatively high. The time complexity is O(n*k*i), where n is the total number of data objects; k is the number of the resulting clusters; i is the number of the iterations of the algorithm. The basic operation is to calculate the distance from the data object to the center point. If 104 data objects are gathered into 100 clusters, then there need to calculate the distances from data objects to center points 106 times in one iteration. This is the most time-consuming part of the algorithm, also easily processed in parallel. A data object is compared with the cluster centers, meanwhile the other data objects can also be compared with the cluster centers.

The main computational work of the k-means cluster algorithm is to assign each data object to its nearest cluster, and the operations of assigning different data objects are independent. Therefore, the operations can be performed in parallel.

In each job, the parallel k-means algorithm performs the same operations of map and reduces separately. First data objects are randomly selected as the centers, and these k center points are stored in a file on HDFS as a global variable. Then, each job consists of three parts: map, combine and reduce. Figure 6 depicts the implementation process of the parallel k-means clustering algorithm based on MapReduce computing model. Mapreduce computing model requires input data stored in row. So that the input data can be processed by fragments, and fragmented data have no correlation. Fragmentation process is completed by MapReduce running environment without writing codes.

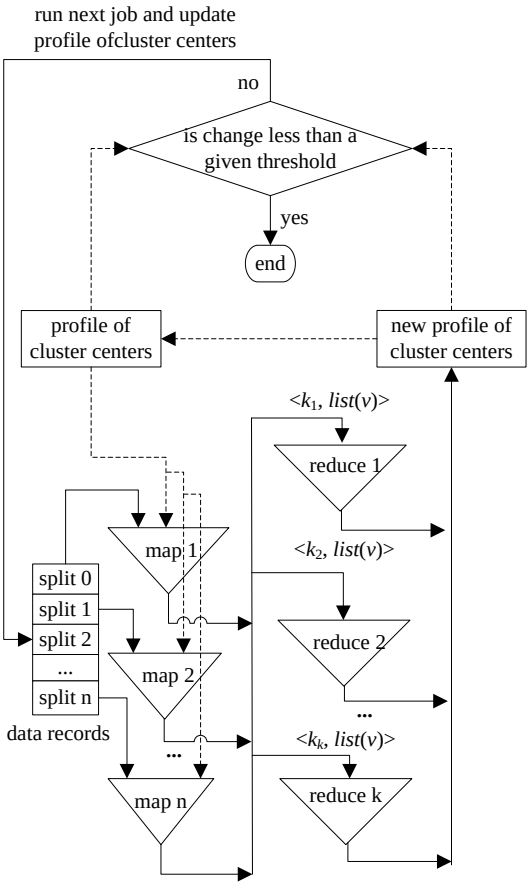

Figure 6. The parallel computing processes

Map function

The task of the map function is to compute the distance from the record to cluster center, and re-mark the new cluster it belongs to. Its input is all records to be clustered and the profile of cluster centers. The input pairs <key, value> of map function are the default format of MapReduce framework. The key is the offset of the current record relative to the start of input data file. The value is a string which is composed by d-dimensional coordinate values of current record. First parse each dimension from the value, and then calculate the distance between the point and k center points to find the nearest cluster. Finally output <key', value'>, key' is the index of the nearest cluster, value' is the string which is composed by d-dimensional coordinate values of current point. The pseudo-codes of map function follow:

```
void map(Writable key, Text point) {
minDistance = MAXDISTANCE;
for(int i = 0; i < k; i ++) {
if(distance(point, cluster[i]) < minDistance) {
minDistance = distance(point, cluster[i]);
currentClusterID = i;
}
}
emitIntermediate(currentClusterID, point);
}
```

Reduce function

The task of the reduce function is to calculate the new cluster centers according to the intermediate results of the map function for the use of the next round of MapReduce job. The intermediate result is data pairs <key, value> where key is cluster ID, the value is the coordinates of points. All the records with the same key are sent to a reduce task. The reduce task accumulates the number of the points and the total number of each coordinates of the points, calculates the mean of each coordinates, and gets a new profile of cluster centers. The pseudo-codes of reduce function follow:

```
void reduce(Writable key, Iterator<PointWritable>
points) {
num = 0;
while (points.hasNext()) {
PointWritable currentPoint = points.next();
 num ++;
 for(int i = 0; i < dimension; i ++) {
    sum[i] += currentPoint[i];
}
}
for(int i=0; i < dimension; i++)
mean[i] = sum[i] / num;
}
emit(key, mean);
}
```

Compute the distance between previous cluster centers and current cluster centers. If the variation is less than a given threshold, then the algorithm ends. Otherwise, replace previous profile of cluster centers with current profile of cluster centers and start a new round of MapReduce job.

Figure 4 shows implementation process of parallel k-means clustering algorithm. Before the reduce task starts, group and sort the intermediate results of map task in order to improve the efficiency.

There are six points in Figure 7. These points are p1(11, …, 11), p2(111, …, 111), p3(22, …, 22), p4(222, …, 222), p5(33, …,33), p5(33, …, 33) and p6(333, …, 33).

The map phase uses three TaskTrackers. Each TaskTracker processes one-third of the total data. The map function calculates the distances between each point and the centers of k clusters and chooses the nearest center.

In the first TaskTracker, the point p1 belongs to cluster C1 and the calculation result is <C1, <1, (11, …, 11)>>; The point p2 belongs to cluster C2 and the calculation result is <C2, <2, (11, …, 11)>>. In the second TaskTracker, the point p3 belongs to cluster C1 and the calculation result is <C1, <3, (22, …, 22)>>; The point p4 belongs to cluster C2 and the calculation result is <C2, <4, (222, …, 222)>>. In the third TaskTracker, the point p5 belongs to cluster C1 and the calculation result is <C1, <5, (33, …, 33)>>; The point p6 belongs to cluster C2 and the calculation result is <C6, <4, (333, …, 333)>>.

In the shuffle phase, the points which belong to the cluster C1 are transmitted to the first ReduceTracker; The points which belong to the cluster C2 are transmitted to the second ReduceTracker. Each ReduceTracker calculates the average coordinates of all points in each cluster, gets the center coordinates of the clusters and updates the center of each cluster. If the change of the new cluster center point with the original center point is less than a predetermined threshold value, the job is stopped, otherwise continue the next round of job.

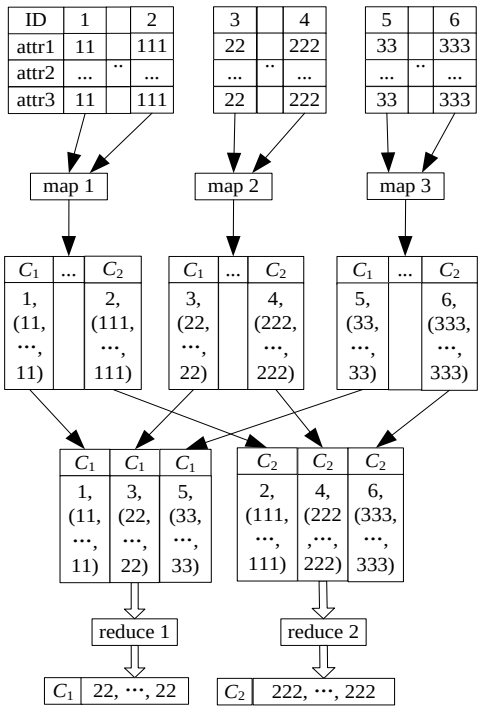

Figure 7. Data processing

In MapReduce computing model, the pending files are stored in the distributed file system platform of Hadoop. According to the size of the file to be processed, the data blocks are stored on different nodes. The size of each block is 64MB. In the map phase, a data block is assigned to a map task. The task completes the relevant calculation of the data block. So that the most time-consuming operation originally handled by a host will be distributed to multiple nodes in parallel processing. If each node on average completes the P map tasks, then the time complexity of parallel k-mean algorithm is O(n*k*i/P).

Experiments and Results Analysis

Figure 8 shows the structure of the cloud computing platform in the experiments. A machine acts as the Nameode and Jobtracker. The other ten machines act as Datanodes and Tasktrackers. The hardware configuration of each node is as follows. CPU model is Intel Xeon X3330. The memory is 8GB. The SATA hard drive is 2TB. Onboard network controller is Intel dual gigabit. According to the method which the official website of the Hadoop project introduces, configure the clusters based on Hadoop 2.2.0.

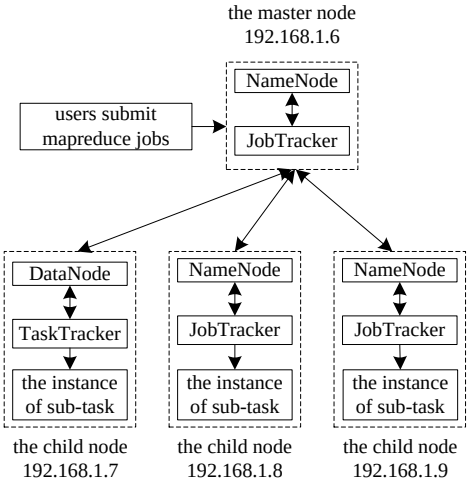

Figure 8. The cloud computing platform

Data are synthetic in the experiment. The dimension of data is 48. To test the performance of the algorithm, the experiment constructs 1G, 2G, 4G, 8G, 16G, 32G 6 different sizes of data sets. The experiment uses speedup, scaleup and sizeup as the evaluation indicators. Because parallel k-means algorithm initializes the centers randomly, the group of experiments is repeated 20 times. The average of the execution time is represented as the final result of the group of experiments.

Stand-alone performance

The experiment compares a computing node in the Hadoop cluster with serial k-means clustering algorithm in dealing with the same scale data and the same hardware configuration. In the experiment, k-means clustering algorithm uses a serial implementation Weka which is an open-source data mining tools University of Waikato

develops. Weka integrates most of the data mining algorithms, and is widely used in academia. But the papers [11] and [12] point out that some deficiencies exist in Weka system. When data is larger, some algorithms will be the cases, mining long time, CUP running 100%, insufficient memory, etc. For Weka system, set its maximum memory maxheep to 1000MB, correspondingly set JVM memory of TaskTracker mapred.child.java.opts to 1000MB. In contrast to these two experiments, the size of the data to be processed increases gradually. The results are shown in Table xxx. The variable t1 is the time Weka software uses. The variable t2 is the time one cluster node runs k-means clustering algorithm. There are 20 fields per record and 10 cluster categories. The initial cluster centers are randomly generated.

Table 1. The experimental results of stand-alone treatment

No.	size(KB)	number(105)	t1(s)	t2(s)
1	2687	20508	9	88
2	5370	41016	21	96
3	16100	123048	46	115
4	47079	369144	135	153
5	50729	398143	insufficient memory	178
6	78411	615240	Insufficient memory	223

These results show that as the input data growth, the consumption of memory and other resource increases on the machine running Weka. This result in performance degradation of the machine, until report insufficient memory, and the computing tasks cannot be completed. However the Hadoop cluster can complete the computing tasks. This reflects that parallel k-means clustering algorithm based on Hadoop platform has the ability to handle large data. When the amount of input data is very small, the processing efficiency of parallel k-means algorithm on Hadoop is much lower than processing efficiency of non-parallel algorithm on a stand-alone. This is because job start and the interaction of a large Hadoop cluster need to consume certain resources, and occupy a larger proportion of total consumption. The actual logic processing time of the computing tasks occupies a smaller proportion of total consumption.

Cluster performance

When parallel k-means algorithm processes data sets of different sizes, the speedup is recorded. The experimental results are shown in Figure 6.

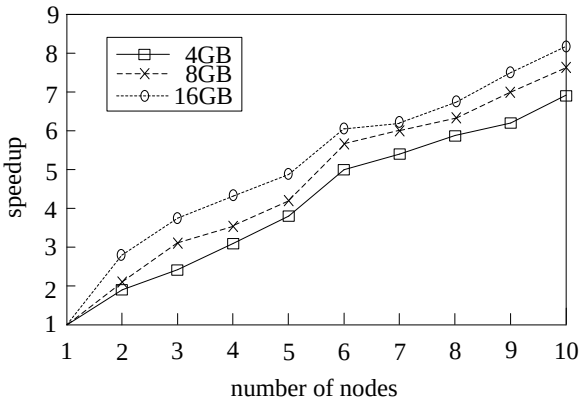

Figure 9. Ttest results of the speedup performance

As can be seen from Figure 9, the speedup of parallel k-means algorithm is nearly linear. And with the increasing size of the data set, the speedup will get better and better. There are two reasons: 1) The pairs <key, value> which the map and reduce process are designed reasonably, so that the parallel k-means algorithm can execute efficiently and quickly; 2) In the design of the k-means parallel algorithm, the combine operation greatly reduces the cost of the communication between the master node and the slave nodes. And the larger the size of the data set, the higher traffic reduction.

Therefore, the larger the size of the data set, the better the speedup of the algorithm.

Three groups of the experiments test the sacleup of the parallel k-means algorithm. The first group tests one node, two nodes, four nodes and eight nodes operating on 1G, 2G, 4G and 8G data. The second group tests one node, two nodes, four nodes and eight nodes operating on 2G, 4G, 8G and 16G data. The third group tests one node, two nodes, four nodes and eight nodes operating on 4G, 8G, 16G and 32G data. The results are shown in Figure 10.

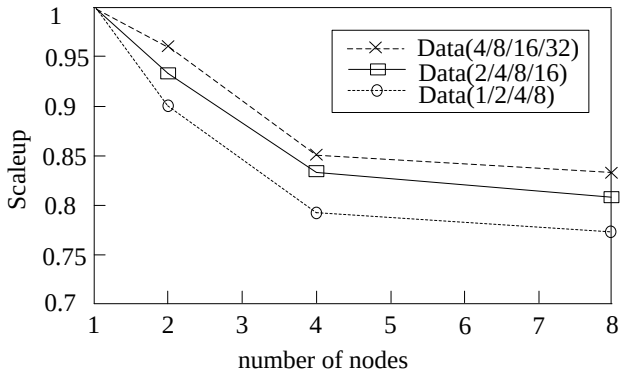

Figure 10. Test results of the scaleup performance

Figure 10 shows test results of the scaleup performance. When the number of platform nodes and the size of data set increase with the same proportion for the same group of data sets, the scaleup of the algorithm reduces gradually. Because the number of platform nodes increases, the communication cost between the nodes will gradually increase. When the number of platform nodes and the size of data set increase with the same proportion, the execution time of the algorithm increases. From the results, we can see that with the gradual increase of the size of the data set, the scaleup of parallel k-means algorithm is getting better. When the data size increases, it can bring all of the computing power of each node. Further in the analysis of the speedup, parallel k-means algorithm adds the combine process. The larger the data set, the highly the cost of the communication between the master node and the slave nodes reduces. Thus, the performance of the algorithm in the second group of data sets is better than the first data sets. The performance of the algorithm in the third group of data sets is better than the second data set.

Figure 8 shows the test results of the data expansion rate. As can be seen from the results, when the number of nodes is less than three, execution time of the algorithm increases with the same proportion of the size of the data set. With the number of nodes in the platform increasing, the efficiency of the algorithm handling large data sets increases highly. For example, in the three-node platform, the execution time of processing 32G data is 10 times 1G data. Experiment results show that the parallel k-means algorithm is suitable for running in large-scale cloud computing platform, and can be effectively applied to the analysis and mining of massive data.

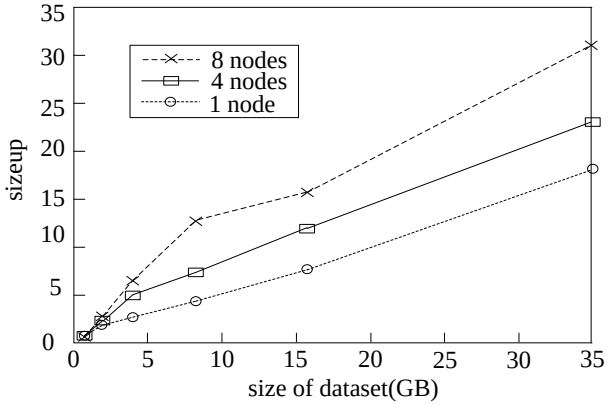

Figure 8. Test results of the sizeup performance

By experimental results, the parallel k-means clustering algorithm is deployed on Hadoop clusters. The algorithm has good speedup and scalability. When Morgan theorem begins to fail and the data to be processed grow explosively, using efficient MapReduce parallel computing method to implement data mining algorithms including k-means algorithm has practical significance in the cloud computing platform.

Conclusions

This paper conducts in-depth research on the parallel k-means algorithm based on MapReduce computing platform of Hadoop. First, briefly describe the basic components of Hadoop platform including structural relationships of HDFS framework and the workflow of all stages of MapReduce. Then, consider the main issues, the main processes in the design of the parallel k-means algorithm based on Hadoop. Finally, the experiments on the data set of different sizes show that the parallel clustering algorithm is suitable for running on the large-scale cloud computing platform, and can be effectively applied to the analysis and mining of massive data.

With the rise of cloud computing concepts, the research on data mining and clustering algorithms based on cloud computing platform gradually becomes a hot topic of scholars. Future research directions include the following. Study

general law of the parallelization of clustering algorithms. Find the relation between data scale and the number of nodes. Find influencing factors of speedup and scalability to design highly efficient parallel clustering algorithm. Study information security and privacy issues based on data mining applications in the cloud computing platform. Solving the problem will play a key role in cloud computing applications in the actual business.

Acknowledgements

This work was financially supported by National Natural Science Foundation of China (61073047), Fundamental Research Funds for the Central Universities (HEUCFT1202, HEUCFT100609), Harbin Special Funds for Technological Innovation Talents (2012RFLXG023).

References

1. Jiawei Han, Micheline Kamber, Jian Pei (2012) Data mining Concepts and Techniques Third Edition. *China Machine Press*, 72(11), p.p.46-53.
2. Armbrust M, Fox A.(2009) Above the clouds: a Berkeley view of cloud computing. *University of California at Berkeley*.
3. Tom White.(2012) Hadoop: The Definitive Guide Third Edition. *O'Reilly Media*.
4. Dean J, Ghemawat S.(2008) MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), p.p.07-113.
5. Zhao Zongtang, Sun Shenli, Fan Ji (2005) Parallel clustering algorithm based on MPI. *Journal of Zhengzhou Institute of Aeronautical Industry Management*, 24(3), p.p.160-171.
6. Mao Jiali (2003) *K-means Algorithm and Parallelization.* Chongqing: College of Computer Science and Engineering Chongqing University.
7. Li Jingbin, Yang Liu, Hua Bei (2011) Research on parallel K-Medoids algorithm based on multi-core platform. *Application Research of Computers*, 28(2), p.p.498-505.
8. Cao Feng, Zhou Aoying (2007) Fast Clustering of Data Streams Using Graphics Processors. *Journal of Software*, 18(2), p.p.291-302.
9. Zhou Bing, Feng Zhonghui, Wang Hexing (2007) The Study of Parallel Clustering Algorithm for Cluster System. *Computer Science*, 34(10),p.p.4-16.
10. Boutsinas B, Gnardellis T.(2002) On distributing the clustering process. *Pattern Recognition Letters*, 23(8), p.p. 999-1008.
11. Wegener D, Mock M, Adranale D.(2009) Toolkit-based high-performance data mining of large data on MapReduce clusters. *IEEE International Conference on Data Mining, Washington: IEEE*, p.p.296-301.
12. Chen Hui-ping, Lin Li-li, Wang Jian-dong, Miao Xinrui (2008) Data mining platform WEKA and secondary development on WEKA. *Computer Engineering and Applications*, 44(19), p.p.76-79.