Implementation of K-Means Algorithm by Using Map Reduction and Bulk Synchronous Parallelism

[1]Ashish A. Golghate, [2]Shailendra W. Shende

Department of Information Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India
Email : [1]ashishgolghate@gmail.com, [2]shailendra.shende@gmail.com

Abstract.- K-Means algorithm to classify or to group objects based on attributes/features into K number of group. Grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. The purpose of K-mean clustering is classify the data. Map Reduce is a software framework for easily writing applications which process vast amounts of data. One of the major drawbacks of the Map-Reduce (MR) model is to simplify reliability and fault tolerance. It does not preserve data in memory across consecutive MR jobs. The Bulk Synchronous Parallelism (BSP) programming model, as an alternative to the MR model that does not suffer from this restriction and under certain circumstances allows complex repetitive algorithms to run entirely in the collective memory of a cluster.

I. INTRODUCTION

Clustering is one of the most important unsupervised learning methods [16]. A loose definition of clustering could be the process of organizing objects into groups whose members are similar in some way . The goal of clustering is to determine the intrinsic groups in a set of unlabeled data. A cluster is therefore a set of objects which are \similar. between them and are \dissimilar. to the objects belonging to other clusters. Clustering has many applications in many fields such as market research, image analysis, bioinformatics, data mining, pattern recognition.

K-means is a famous clustering algorithm, which is one of the simplest unsupervised learning algorithms, and it is usually very fast .It is also one of the top-10 algorithms in data mining However, there is no guarantee that it will converge to the global optimum, since it is heuristic algorithm. An important problem of wide concern is k-mean fs instability and sensitivity to outliers. And dealing with high dimensions and large data sets can be problematic because of time complexity.

The Map-Reduce (MR) programming model [1] has been emerged as a popular framework for large-scale data analysis. In particular, Hadoop [14],the most prevalent implementation of this framework, has been used extensively by many companies on a very large scale. One of the major drawbacks of the MR model is to simplify reliability and fault tolerance, it does not preserve data in memory between the map and reduce tasks of a MR job or across consecutive MR jobs. Consequently, using the original implementation of the MR model, to pass data to the next job in a MR work flow, a MR job must dump its data to the distributed file system (DFS), to be read by the next MR job. This restriction imposes a high overhead to complex MR work-flows. The Bulk Synchronous Parallelism (BSP) programming model [3], on the other hand, which precedes the MR model by more than a decade, has been recently advocated as an alternative to the MR model that does not suffer from this restriction and under certain circumstances allows complex repetitive algorithms to run entirely in the collective memory of a cluster.

A BSP computation consists of a sequence of supersteps. Each superstep is evaluated in parallel by every peer participating in the BSP computation. A superstep consists of three stages: a local computation, a process communication, and a barrier synchronization. Each peer maintains a local state in memory, which is accessible by this peer only across all supersteps. In addition, during the local computation stage of a superstep, each peer has access to the messages sent by other peers during the previous superstep. It can also send messages to other peers during the process communication stage, to be read at the next superstep. The barrier synchronization stage synchronizes all peers to make sure that they have received all the sent messages before the next superstep. To cope with peer failures, each peer may use checkpoint recovery, to flush the volatile part of its state to the DFS, thus allowing to rollback to the last checkpoint when a failure occurs. In many graph analysis problems a large part of the state is used for storing the graph, which is immutable and does not require checkpointing. Apache's Giraph is an open-source alternative to Pregel and Apache's Hama[11] is a

general BSP computation platform built on top of Hadoop that can process any kind of data.

The main contributions of this paper can be as follows:

- To overcome the instability and the sensitivity to outliers of k-means, and improve the stability and accuracy.
- The inefficiency problem in clustering on large data sets is solved by using a distributed computing framework MapReduce and Bulk Synchronous Parallelism.

II. RELATED WORK

The map-reduce (MR) model was first introduced by Google in 2004 [14]. Several large organizations have implemented this model, including Apache Hadoop and Pig, Apache/Facebook Hive, Google Sawzall , and Microsoft Dryad. The most popular MR implementation is Hadoop, an open-source project developed by Apache, which is used today by Yahoo! and many other companies to perform data analysis. There are also a number of higher- level languages that make MR programming easier, such as HiveQL, PigLatin, SCOPE , and Dryad/Linq .Hive is an open-source project by Facebook that provides a logical RDBMS environment on top of the MR engine,well suited for data warehousing. Using its high-level query language, HiveQL, users can write queries, which are optimized and translated into MR jobs that are executed using Hadoop

The BSP model was introduced by Leslie G. Valiant in 1990 and has been since improved and extended by many others. The best known implementations of the BSP model for data analysis on the cloud are Google's Pregel and Apache's Giraph and Hama. Although Pregel has already been used extensively by Google for large graph analysis, it has not reached the popularity of MR yet. Its open-source counterpart, Giraph, and the general BSP platform, Hama are still in their very early stages of development and are not used in the same degree as Pregel. The BSP model is compared with the MR model in which discusses how to map any MR computation to a BSP, and vice versa using Hama.

III. BACKGROUND

In this section, introductions about the algorithms and technologies adopted in this paper are given. An outline on k-means is given in section 1, MapReduce in section 2 and Bulk Synchronous Parallelism in section 3.

1.K-means:

James MacQueen used the term k-means in 1967, though the idea can be traced back to Hugo Steinhaus in 1956 . Stuart Lloyd first proposed the standard algorithm in 1957, though it was not published until 1982 .

K-means clustering is a cluster analysis method which aims to partition n objects into k clusters, in which each object belongs to the cluster with the nearest centroid. It attempts to find the centers of natural clusters in the data set [16].

Algorithm description as following:

Input: The number of clusters k and n documents
Output: k clusters

Step1. Randomly select k documents from n documents as the initial cluster centers.

Step2. Calculate the distances of the rest documents to the every center of the clusters, and assign each of the rest documents to the nearest cluster.

Step3. Calculate and adjust each cluster center.

Step4. Iterate Step2 ~ Step3 until the criterion function convege. Program ends.

2. MapReduce:

MapReduce is a programming model and an associated implementation for processing an generating large data sets, which was introduced by Google to solve certain kinds of distributable problems. Programs written in MapReduce's functional style can be automatically parallelized and executed on a high performance computing cluster of a large number of low-cost ordinary personal computers. It is inspired by the map and reduce functions, Users specify a map function that processes a (key, value) pair to generate a set of intermediate (key, value) pairs, and a reduce function that merges all intermediate values associated with the same intermediate key [1].

The main advantage is that the map and reduce functions of MapReduce are allowed for distributed processing. Each map operation is independent from others. Therefore, all map operations can be performed in parallel, and a set of reduce operations can be performed in parallel. Similarly, a set of reduce operations can perform the reduction phase independently. The map function, written by the user, takes one pair of data with a type in a data domain, and returns a list of pairs in the same or a different domain. It is applied to every (k1, v1) pair in parallel, and produces a list of intermediate (k2, v2) pairs per map invocation.

$$\text{Map}(k1, v1) \rightarrow \text{list}(k2, v2)$$

The MapReduce framework collects all intermediate (k2, v2) pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys, and passes them to the reduce function.

The reduce function, also written by the user, is then applied to each group in parallel, and produces a collection of values in the same domain. It accepts an intermediate key k2 and a list of values list (v2) for k2. It merges together these values to form a possibly smaller list of values list (v3). Each reduce call typically produces either one or an empty return. The returns of all calls are collected as the desired result list.

Reduce(k 2, list(v2)) → list(v3)

Thus the MapReduce transforms a list of (key, value) pairs into a list of values.

Apache Hadoop is a software platform that allows the user to write and run applications that process vast amounts of data easily . Hadoop implements MapReduce with the HDFS (Hadoop Distributed File System). HDFS is designed to run on commodity hardware. It has many similarities with distributed file systems.

3. Bulk Synchronous Parallelism:

A BSP computer consists of processors connected by a communication network. Every processor has a fast local memory, and may follow different threads of computation. A BSP computation a series of global supersteps. A superstep consists of 3 components:

- Concurrent computation: Several computations take place on every participating processor. Every process only uses values stored in the local memory of the processor. The computations are independent in the sense that they occur asynchronously of all the others.
- Communication: The processes exchange data between themselve. This exchange takes the form of one-sided put and get calls, rather than two-sided send and receive calls.
- Barrier synchronisation: When a process reaches this point it waits until all other processes have finished their communication actions.

The computation and communication actions do not have to be ordered in time. Barrier synchronization concludes the superstep: it has the function of ensuring that all one-sided communications are properly concluded. That global synchronization is not needed in models based on two-sided communication, since these synchronize processes implicitly.

HAMA is a distributed computing framework based on BSP (Bulk Synchronous Parallel) computing technique for massive scientific computations designed to run on massive data-sets stored in Hadoop Distributed File System(HDFS). It is currently an incubator project by the Apache Software Foundation. Apache Hama leverages BSP computing techniques to speed up iteration loops during the iterative process that requires several passes of messages before the final processed output is available. Hama provides an easy and flexible programming model, as compared with traditional models of Message Passing .It is compatible with any distributed storage (e.g., HDFS, HBase, …, etc), so you can use the Hama BSP on your existing Hadoop clusters.

Why Hama and BSP?

Today, many practical data processing applications require a more flexible programming abstraction model that is compatible to run on highly scalable and massive data systems (e.g., HDFS, HBase, etc). A message passing paradigm beyond Map-Reduce framework would increase its flexibility in its communication capability. Bulk Synchronous Parallel (BSP) model fills the bill appropriately. Some of its advantages over MapReduce and MPI are:

- Supports message passing paradigm style of application development
- Provides a flexible, simple, and easy-to-use small APIs
- Enables to perform better than MPI for communication-intensive applications
- Guarantees impossibility of deadlocks or collisions in the communication mechanisms

IV. EXPERIMENTAL ENVIRONMENT AND DATA SET

In this paper, experiments are based on a PC with the following hardware configuration: Intel (R) Core (TM) DuoCPU T6570 @ 2.10GHZ, 2.10GHZ, 2.00GB RAM and250GB hard disk. The software configuration uses Windows VistaTM Home Basic. Based on the platform mentioned above, VMware workstation 6.0.2 is used to construct a virtual cluster, with the same hardware platform: one CPU processor; 128MB RAM; and 8GB hard disk. The software environment uses the same configuration: inux operating system; the distributed computing platform of Hadoop and Hama; and Java development platform JDK 1.6. The data used in this experiment comes from text classification corpus of 20_newsgroups Data Set.

V. CONCLUSION

In this paper, work represents only a small first step in using the MapReduce programming technique in the process of large-scale data Sets. We take the advantage of the parallelIism of MapReduce to design a parallel K-Means clustering algorithm based on MapReduce. This algorithm can cluster the massive data, making full use of the Hadoop cluster performance. It can finish the text clustering in a relatively short period of time.But Bulk Synchronous Parallelism is much faster than MapReduce, because you do not have to submit a new job for a new computation step. In BSP the superstep is less costly than running a MapReduce job, therefore BSP is faster.

REFERENCES:

[1] Aditya B. Patel, Manashvi Birla, Ushma Nair, Addressing Big Data Problem Using Hadoop and Map Reduce , NIRMA UNIVERSITY

INTERNATIONAL CONFERENCE ON ENGINEERING, NUiCONE-2012,.

[2] Tomasz Kajdanowicz, Wojciech Indyk, Przemyslaw Kazienko and Jakub Kukul, Comparison of the Efficiency of MapReduce and Bulk Synchronous Parallel Approaches to Large Network Processing , IEEE 12th International Conference on Data Mining Workshops 2012

[3] Wadoud Bousdira, Frédéric Gavay, Louis Gesbertz, Frédéric Loulerguex, and Guillaume Petiot, Functional Parallel Programming with Revised Bulk Synchronous Parallel ML First International Conference on Networking and Computing 2010

[4] I. Gourlay, P. M. Dew, K. Djemame Bulk Synchronous Parallel Computing Using a High Bandwidth Optical Interconnect Proceedings of the International Parallel and Distributed Processing Symposium 2010

[5] Songchang Jin, Shuqiang Yang, Yan Jia Optimization of Task Assignment Strategy for Map-Reduce 2nd International Conference on Computer Science and Network Technology 2012

[6] Foto N. Afrati ,Dimitris Fotakis , Jeffrey D. Ullman Enumerating Subgraph Instances Using Map-Reduce in WSDM, 2012.

[7] Jiamin Lu, Ralf Hartmut G¨uting, Parallel Secondo: Boosting Database Engines with Hadoop IEEE 18th International Conference on Parallel and Distributed Systems 2012

[8] F. Afrati and J. Ullman, Optimizing multiway joins in a map-reduce environment, IEEE Transaction of Knowledge and Data Engineering, vol. 23, no. 9, pp. 1282–1298, 2011.

[9] Hadoop. http://hadoop.apache.org/.

[10] D. Battre, et al. Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing . In SOCC'10.

[11] Hama. http://incubator.apache.org/hama/.

[12] M. Zaharia, et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In Memory Cluster Computing . 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2012.

[13] A. Tiskin. The Bulk-Synchronous Parallel Random Access Machine . Theoretical Computer Science, 196(1,2), 1998.

[14] Hadoop Website, http://hadoop.apac

[15] Zhao, W., Ma, H., et al.: Parallel K-Means Clustering Based on MapReduce. In: CloudCom 2009. LNCS, vol. 5931, pp. 674–679 (2009)

[16] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, 2000.

[17] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D.Piatko, R. Silverman, and A. Y. Wu, \An Efficient Kmeans Clustering Algorithm: Analysis and Implementation., IEEE Transactions on Pattern Analysis and Machine Intelligence, July, 2002, Vol. 24, No. 7, pp. 881-892.

[18] J. Dean, S. Ghemawat, \MapReduce: Simplified Data Processing on Large Clusters., Communications of the ACM, January, 2008, Vol. 51 No. 1, pp. 107-113.

[19] D. Borthakur, \The Hadoop Distributed File System: Architecture and Design., http://hadoop.apache.org/.

[20] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters.In Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation, pages 137-149, 2004.