Solving Mean-Shift Clustering Using MapReduce Hadoop

Maksat N. Kalimoldayev
Institute of Information and Computational Technologies
Almaty, Kazakhstan
mnk@ipic.kz

Vladimir Siladi
Faculty of Natural Sciences
Matej Bel University
Banska Bystrica, Slovakia
vladimir.siladi@umb.sk

Maksat N. Satymbekov
Institute of Information and Computational Technologies
Almaty, Kazakhstan
m.n.satymbekov@gmail.com

Lyazat Naizabayeva
Institute of Information and Computational Technologies
Almaty, Kazakhstan
naizabayeva@gmail.com

***Abstract*—Paper presents results of a practical experiment that was conducted in order to pursuit main objective - design and implement novel iterative MapReduce framework based on Hadoop technology. To study the effects of implementation parallel scientific applications, we deployed. Despite the fact that they did not show better performance than the same algorithm implemented in MPI, we conducted experiments to solve problems iterative computations using the Hadoop architecture. As a result of the experiments, the fail-safe behavior of Hadoop technology was revealed in the solutions of complex hadacies.**

***Index Terms*—Hadoop, MapReduce, Mean-Shift, MPI, HDFS**

I. Introduction

Paper presents results of a practical experiment that was conducted in order to pursuit main objective - design and implement novel iterative MapReduce framework [1] based on Hadoop technology [2], [3]. To study the effects of implementation parallel scientific applications, we deployed algorithm of mean shift clustering on Hadoop platform. The reminder of the article is structured as follows. The first section presents introduction. In second section we present matematical model mean shift clustering. In third section we describe Hadoop, MapReduce and HDFS. In fourth section we comprised parallel algorithm for solving the mean shift clustering. In fifth section "Implementation and evaluation" presents MapReduce implementations performance. Finally sections discussion and conclusions. Our work is based upon modifying the programming model and using Hadoop standard features. Cluster is a collection of data members having similar characteristics. The process of establishing a relation or deriving information from raw data by performing some operations on the data set like clustering is known as data mining. Data collected in practical scenarios is more often than not completely random and unstructured. Hence, there is always a need for analysis of unstructured data sets to derive meaningful information. This is where unsupervised algorithms come in to picture to process unstructured or even semi structured data sets by resultant. Mean Shift Clustering is one such technique used to provide a structure to unstructured data so that valuable information can be extracted. This paper discusses the implementation of the Mean Shift Clustering Algorithm over a distributed environment using ApacheTM Hadoop. Research result is the comparison of the result with different cluster settings and MPI model.

II. Mean shift clustering

The mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters. Given n data points $x_i, i = 1, 2, ..., n,$ on a d-dimensional space R_d, the multivariate kernel density estimate obtained with kernel $K(x)$ and window radius h is

$$f(x) = \frac{1}{nh^d} + K(\frac{x - x_i}{h}) \tag{1}$$

For radially symmetric kernels, it suffices to define the profile of the kernel $k(x)$ satisfying

$$K(x) = c_{k,d}k(||x||^2) \tag{2}$$

Where $c_{k,d}$ is a normalization constant which assures $K(x)$ integrates to 1. The modes of the density function are located at the zeros of the gradient function $\nabla f(x) = 0$.

The gradient of the density estimator (1) is

$$\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1} (x_i x) g(||\frac{x - x_i}{h}||^2) \tag{3}$$

where $g(s) = -k^1(s)$. The first term is proportional to the density estimate at x computed with kernel $G(x) = c_{g,d}g([||x||^2])$ and the second term

$$m_h(x) = \frac{\sum\limits_{i-1}^{n} x_i g(||\frac{x-x_i}{h}||^2)}{\sum\limits_{i-1}^{n} g(||\frac{x-x_i}{h}||^2)} - x \tag{4}$$

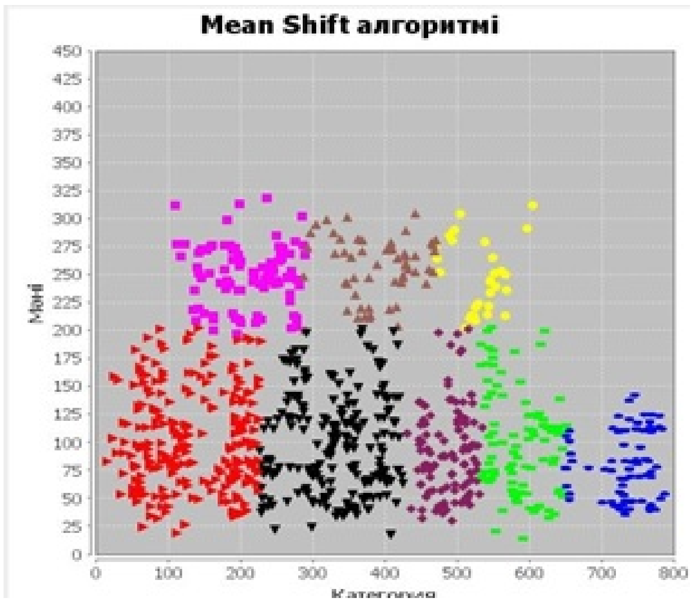

Fig. 1. Mean Shift.

is the mean shift. The mean shift vector always points toward the direction of the maximum increase in the density. The mean shift procedure, obtained by successive

- computation of the mean shift vector $m_h(x^t)$
- translation of the window $x^{t+1} = x^t + m_h(x^t)$

is guaranteed to converge to a point where the gradient of density function is zero. Mean shift mode finding process is illustrated in Figure 1. The mean shift clustering algorithm is a practical application of the mode finding procedure:

III. Hadoop, MapReduce and HDFS: A Developers Perspective

Hadoop is an open-source project overseen by the Apache Software Foundation Hadoop is mostly used for reliable, scalable and distributed computing, but can also be used as a general purpose file store capable of hosting petabytes of data. Many companies use Hadoop for both research and production purposes. Initially, Hadoop was primarily a tool for storing data and launching MapReduce tasks, but now Hadoop is a large stack of technologies that are somehow related to processing large data (not just with MapReduce). The core components of Hadoop are:

-Hadoop Distributed File System (HDFS) is a distributed file system that allows you to store information of almost unlimited volume.

- Hadoop YARN - a framework for managing cluster resources and task management, including the MapReduce framework.

Many of the features and configurations supported by Hadoop provide usability, power and wide-ranging application possibilities for this platform. For example, Yahoo! and countless other organizations are using Hadoop to analyze tons of bits and bytes of information. On the other hand, Hadoop

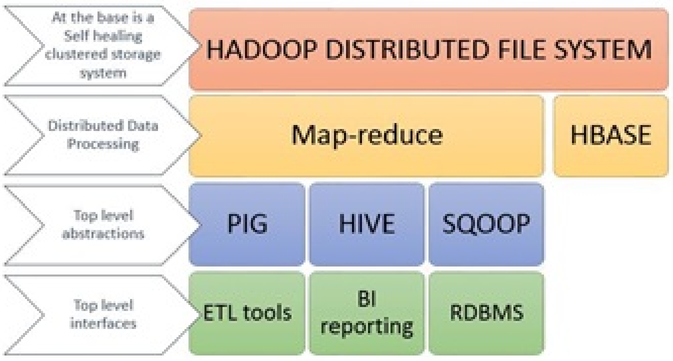
At the base is a Self healing clustered storage system
HADOOP DISTRIBUTED FILE SYSTEM
Distributed Data Processing
Map-reduce
HBASE
Top level abstractions
PIG
HIVE
SQOOP
Top level interfaces
ETL tools
BI reporting
RDBMS

Fig. 2. Hadoop Architecture.

can be easily launched on one node. All you need for this is the ability to program in Java (including generics).

In the Hadoop infrastructure, the map and reduce functions are implemented as an extension of the base classes. Specific implementations of functions are linked into a single module in accordance with the selected configuration, which, in addition, defines the input and output formats. Hadoop is well suited for processing huge files containing structured data. One of the main advantages of this platform is that it organizes the initial parsing of the input file, so you just have to process each line. Thus, the task of the map function is to determine which data is to be extracted from the incoming text line.

Hadoop handles large number of data from different system like Images, videos, Audios, Folders, Files, Software, Sensor Records, Communication data, Structured Query, unstructured data, Email conversations, and anything which we can't think in any format[5]. All these resources can be stored in a Hadoop cluster without any schema representation instead of collecting from different systems. There are many components involved in Hadoop like Avro, Chukwa, Flume, HBase, Hive, Lucene, Oozie, Pig, Sqoop and Zookeeper. The Hadoop Package also provides Documentation, source code, location awareness, Work scheduling. A Hadoop cluster contains one Master node and Many Slave nodes. The master node consists of Data node, Name node, Job Tracker and Task Tracker where slave node acts as both a Task Tracker and Data node which holds compute only and data only worker node. The Job Tracker manages the job scheduling. Basically Hadoop consists of two Parts. They are Hadoop Distributed File system (HDFS) and Map Reduce[5].HDFS provides Storage of data and Map Reduce provides Analysis of data in clustered environment. The Architecture of Hadoop is represented in figure 2

HDFS is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute sites throughout a cluster to enable reliable, extremely rapid computations:

- Data is distributed across many machines at load time.
- HDFS is optimized for large, streaming reads of files rather than for irregular random reads.

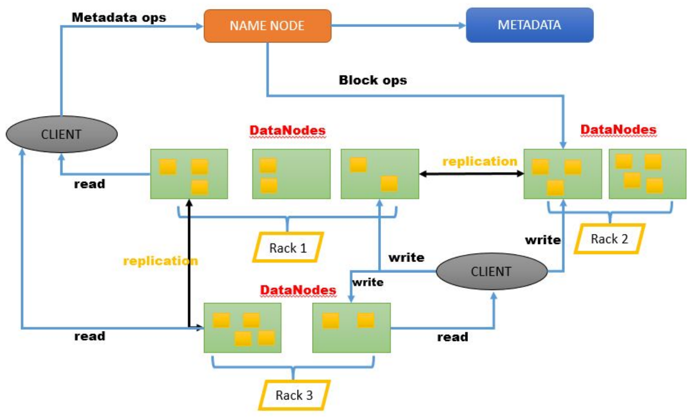

Fig. 3. HDFS Architecture.

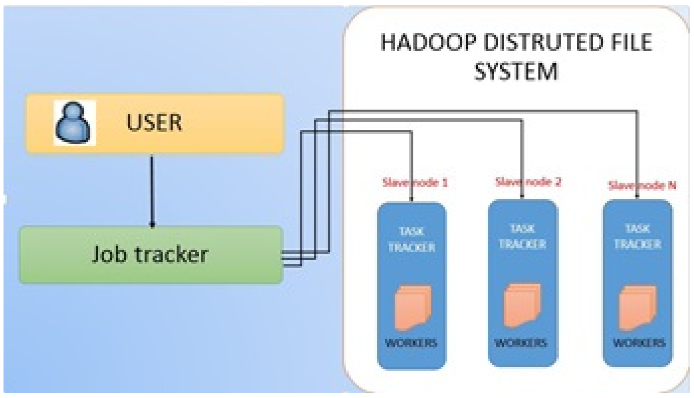

Fig. 4. Mapreduce Architecture.

- Files in HDFS are written once and no random writes to files are allowed.
- Applications can read and write HDFS files directly via the Java API.

The communication between the nodes occurs through Remote Procedure calls. Name node stores metadata like the name, replicas, file attributes, locations of each block address and the fast lookup of metadata is stored in Random Access Memory by Metadata. It also reduces the data loss and prevents corruption of the file system. Name node only monitors the number of blocks in data node and if any block lost or failed in the replica of a datanode, the name node creates another replica of the same block. Each block in the data node is maintained with timestamp to identify the current status. If any failure occurs in the node, it need not be repair immediately it can be repaired periodically. The architecture of HDFS is shown in Figure 3. Security Issues in HDFS The HDFS is the base layer of Hadoop Architecture contains different classifications of data and it is more sensitive to security issues.

MapReduce is a programming model and a framework for writing applications of large-scale computations on the basis of a large number of computations:

- Provides automatic parallelization and distribution of tasks.
- has built-in mechanisms for maintaining stability and operability in case of failure of individual elements.
- Gives status and monitoring tools.
- Provides a clean abstraction layer for programmers.

As an infrastructure designed to handle large data sets, the MapReduce model is optimized for distributed work across multiple computers. As the name implies, this infrastructure includes two main functions. The task of the map function is to split the incoming data stream into smaller sets and transfer these data sets to other processes for further processing. The reduce function analyzes the results of processing individual sets, obtained by the map function, and passes them to the output of the program.

The programmer designs a Map function that uses a (key,value) pair for computation. The Map function results in the creation of another set of data in form of (key,value) pair which is known as the intermediate data set. The programmer also designs a Reduce function that combines value elements of the (key,value) paired intermediate data set having the same intermediate key. [5] Map and Reduce steps are separate and distinct and complete freedom is given to the programmer to design them. Each of the Map and Reduce steps are performed in parallel on pairs of (key,value) data members. Thereby the program is segmented into two distinct and well defined stages namely Map and Reduce. The Map stage involves execution of a function on a given data set in the form of (key,value) and generates the intermediate data set. The generated intermediate data set is then organized for the implementation of the Reduce operation. Data transfer takes place between the Map and Reduce functions. The Reduce function compiles all the data sets bearing the particular key and this process is repeated for all the various key values. The final out put produced by the Reduce call is also a dataset of (key,value) pairs. An important thing to note is that the execution of the Reduce function is possible only after the Mapping process is complete. Each MapReduce Framework has a solo Job Tracker and multiple task trackers. Each node connected to the network has the right to behave as a slave Task Tracker. The issues like division of data to various nodes , task scheduling, node failures, task failure management, communication of nodes, monitoring the task progress is all taken care by the master node. The data used as input and output data is stored in the file-system.

IV. Parallel algorithm for solving the Mean-Shift Clustering

Algorithm 1 Mapper design for Mean-Shift Clusreing

```
1. procedure MeanShiftDesign
2. Load Cluster file
3. fp=Mapcluster file
4.  Create two list
5.  listnew=listold
6. CALL read(Mapcluster file)
7. newfp = MapCluster()
```

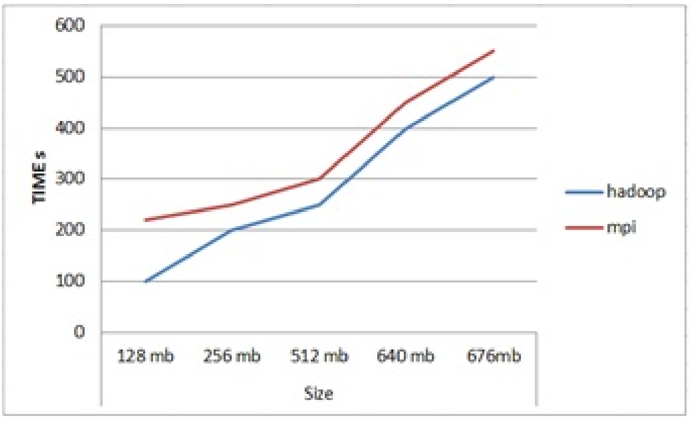

Fig. 5. Mapreduce Architecture.

```
8. dv = 0
9. Assign correct centeroid
10. read(div)
11. calculate centeroid
12. div = minCenter()
13. CALL MeanSHIFTReduce()
14. end procedure = 0
```

```
Algorithm 2 Reducer design for Mean Shift Clustering
1. procedure MeanShiftReduceClustering
2.NEW listofclusters
3. COMBINE result clusters from MAPCLASS.
4. if cluster size too high or too low then
RESIZE the cluster
5. $C_{Max}$ = findMaxSize(ListofClusters)
6. $C_{MIN}$ = findMinSize(ListofClusters)
7. if $C_{Max} > \frac{1}{20}$ totalSize then Resize(Cluster)
8. WRITE cluster FILE to output DIRECTORY
```

```
Algorithm 3 Implementing MeanShift Function
1. Procedure MeanShift Function
2. If Initial Iteration then LOAD cluster file from DORECTORY
else Read cluster file from previous iteration
3. Create new JOB
4. SET MAPPER to map class Defined
5. SET REDUCER to reduce class define
6. Paths for output directory
7. SUBMIT JOB
```

V. Implementation and Evaluation

In order to test the application we set up the infrastructure that included 8 Core i-5 Processor PCs with 16Gb RAM, HP Blade servers with 4 Core Intel Xeon Processors and Gigabit network connection Switch. All computing machines got 64-bit Ubuntu 12.10 operating system and Hadoop 1.2.1 platform installed.

VI. CONCLUSION

In this paper we studied the adaptation of scientific computing tasks to a cloud environment, such as MapReduce. The presented study presents a new iterative processing platform for Hadoop. Despite the fact that they did not show better performance than the same algorithm implemented in MPI, we conducted experiments to solve problems iterative computations using the Hadoop architecture. As a result of the experiments, the High-Availability behaviour of Hadoop technology was revealed in the solutions of complex problem.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[2] C. Lam, Hadoop in action. Manning Publications Co., 2010.
[3] T. White, Hadoop: The Definitive Guide. O'Reilly Press, 2009.
[4] Lammel, R.: Google's MapReduce Programming Model - Revisited. Science of Computer Programming 70, 1–30 (2008).
[5] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu,P. Wyckoff, R. Murthy, "Hive – A Warehousing Solution Over a Map-Reduce Framework," In Proc. of Very Large Data Bases, vol. 2 no. 2,August 2009, pp. 1626-1629
[6] M. Fashing and C. Tomasi. Mean shift is a bound optimization. IEEE Trans. Pattern Analysis and Machine Intelligence, 27(3):471–474, Mar. 2005.
[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence, 24(5):603–619, May 2002.
[8] Apache Hadoop. http://hadoop.apache.org/
[9] Y. Wong. Clustering data by melting. Neural Computation, 5(1):89–104, Jan. 1993
[10] X. Yuan, B.-G. Hu, and R. He. Agglomerative mean-shift clustering. IEEE Trans. Knowledge and Data Engineering, 24(2):209–219, Feb. 2010.