An improved map reduced framework to handle the Distributed Data and their Editions

S.G.Raja
Research Scholar, Vels University, Chennai, India

Abstract : This paper deals with find the improved map reduced framework to handle the large amount of grouped data. When we are deals with large amount of unorganized data we are end with cluster approach to organize them to serve its clusters. Version Control System (VCS) is the most commonly used approach to find the editions of the data. However before pushing the data into Data Lake, if we used improved map reduce framework easily retrieve the history of the data. A history always used to avoid the future mistakes and helps to improve or predict the upcoming business needs. The experiments performed on different types of data with different delimiters.

Keywords :- Distributed Data Clustering, Big Data, MapReduce, Hadoop, HDFS, Version Control Systems, Data lake loader

1. Introduction

Version control is software used to track modifications to a collection of files over time. The goal is not only to be able to determine exactly who made which modifications, but to also be able to have access to the full history of the project at any time [1]. Currently there are lot of approaches are existing to retrieve the history of the files. Basic needs of the business are history of the large amount of data. Storage is one of the most difficult challenges for a version control system. For every file, we must store every version that has ever existed. The logical size of a version control repository never shrinks. It just keeps growing and growing, and every old version needs to remain available. Even data compression and mining helps small size of clusters. But large size of clusters or its data sets we have to approach any one of the framework. Here we deals Map reduced framework which supports versioning to keep up the history of large amount of data.

- Firstly, I propose an Improved MapReduce framework that supports the flat file versions which are contains distributed data.
- Secondly, my experimental analysis proves that the Improved MapReduce framework helps to reduce the storage size of the history of the large amount of data.
- Finally, displays the comparison between Improved MapReduce framework and MapReduce framework.

2. Background of Hadoop and MapReduce

According to Hadoop documentation [3], Hadoop is an Apache open source framework inspired by Google File System [4]. It allows parallel processing on distributed data sets across a cluster of multiple nodes connected under a master-slaves architecture. Hadoop consists of two main components: HDFS [4] and MapReduce [5, 6].

The first component is the Hadoop Distributed File System (HDFS). HDFS is designed to support very large file of data sets. It is also distributed, scalable and fault-tolerant. The Big Data file uploaded into the HDFS is split into block file with specific size defined by the client and replicated across the cluster nodes. The master node (NameNode) manages the distributed file system, namespace and metadata. While the slave nodes (DataNode) manage the storage of block files and periodically report the status to NameNode.

The second one is the MapReduce programming model for intensive computation on large data sets in parallel way. To ensure good parallelism, the data input/output needs to be uploaded into the HDFS. In MapReduce framework, the master node works as JobTracker and the slave nodes as TaskTracker. The JobTracker assumes the responsibility and coordinates the job execution. The TaskTracker runs all tasks submitted by the JobTracker. As shown in Fig. 1, the MapReduce job runs in two main stages:

1. In the Map stage, the mappers (map tasks) are assigned to slave nodes that host the blocks data. Each mapper takes line-by-line the records of its input and transforms them into <key, value> pairs. Next, the map function defined by the user is called to produces another intermediate <key, value> pairs. The intermediate results are sorted locally by keys and sent to the reduce stage when all map tasks are completed.
2. In the Reduce stage, the reducers (reduce tasks) read the map stage outputs and group all values which share the same key to produce for each key an iterate values <key, iterable[value]>. Next the reduce function defined by the user is applied over the sorted intermediate data sets to produce a set of smaller <key, value> pairs and write finally its result into the HDFS.

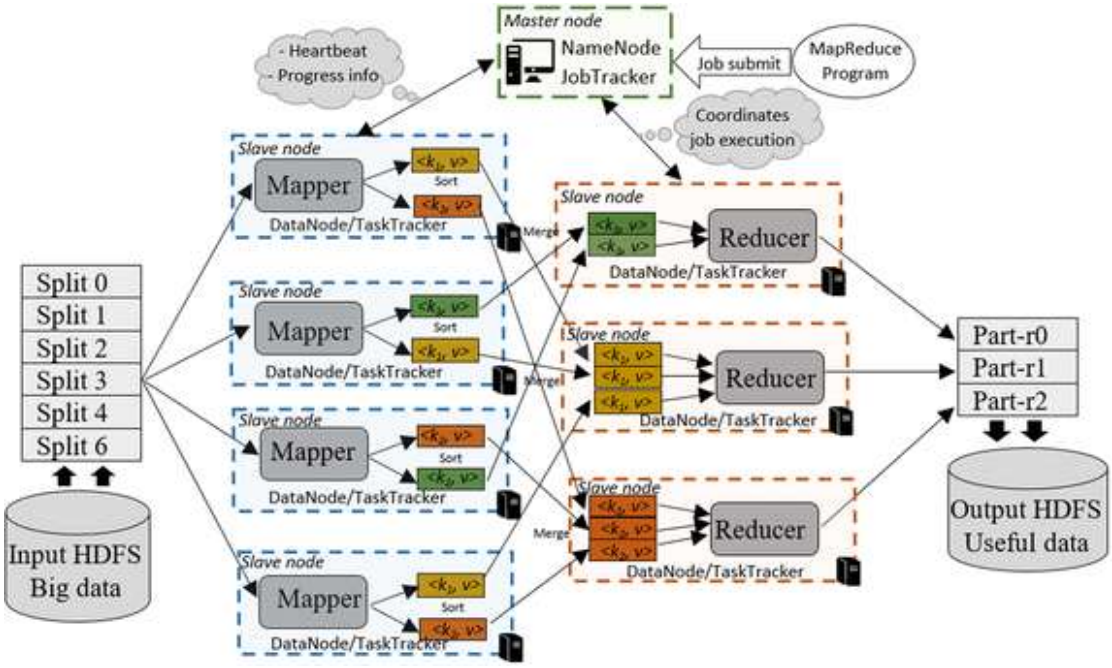

Fig1: The Job execution workflow

3. Improved MapReduce Job Execution workflow to keep its editions

In the Reduce stage, the new key value comparison layer checks whether the [art is exits in Output HDFS useful data or Not. If the Reducer already exists in the Output HDFS Useful data then keep the Reducer into History of Output HDFS Useful data. Using this approach in future all repeated transactions are reduced. Also the repeated key values pair details are tracked to produce various results like future endeavours, data analysis, and so on. Any one of the below approach helps to achieve the History of the Output HDFS Useful Data [7].

- A textual encoding of generic trees that enables their precise version control within standard line-based VCSs such as Git.
- A novel algorithm for computing the difference between two versions of a tree based on the diff reported by a line-based VCS.
- A novel algorithm for a three-way merge of trees, which allows the customization of conflict detection and resolution.

Fig 2, listed the new way of improved map reduced framework to handle the Distributed Data and their Editions

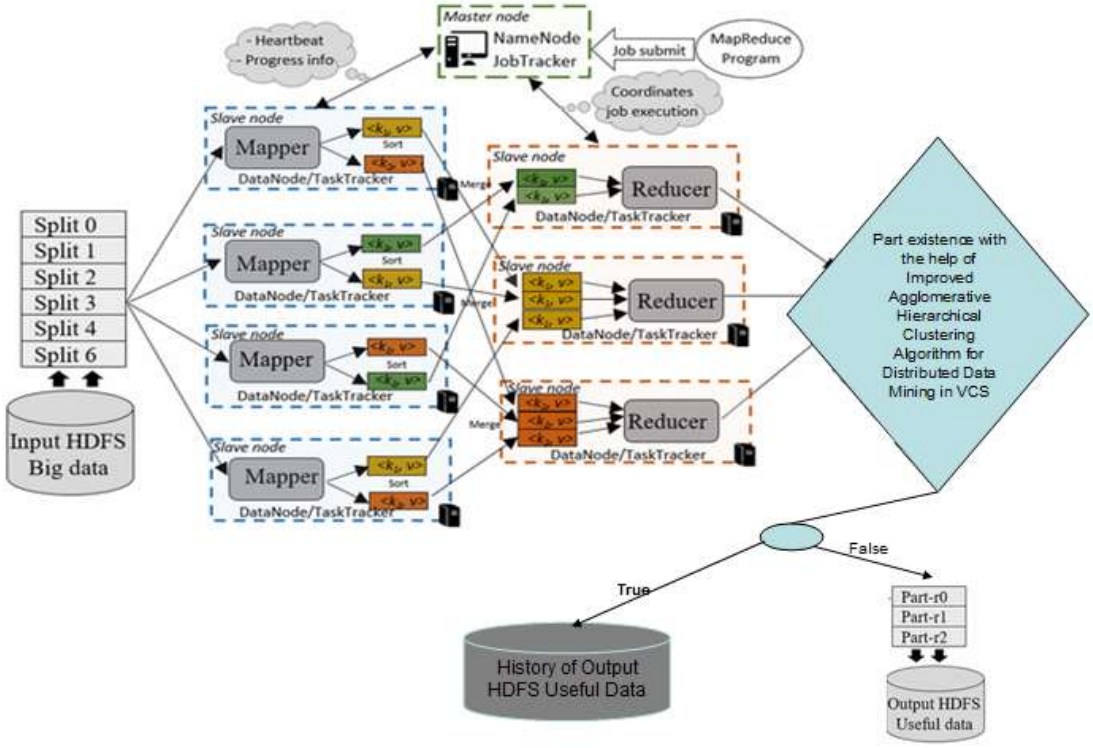

Fig2: Improved MapReduce Job Execution workflow The Job execution workflow

4. Conclusion and Further Works

To sum up in this paper, proposed the improved MapReduce framework to handle the grouped or combined key value pair histories. The experiment carried out in the paper 'Analysis of Improved Agglomerative Hierarchical Clustering Algorithm for Distributed Data Mining in Version Control Systems (VCS)' proves less storage space when we are retrieving the history reducer data in History of Output HDFS Useful Data. In addition to that, it is new way of flat file handle to avoid huge space in hard disk. For future work interested in:

Proposing the Improved MapReduce framework approach to Audio, Video and other grouped files.

Adapt the global approach to handle all the different type of files into one layer

References

[1] https://people.orie.cornell.edu/prs233/orie-5270/vc/vc.html

[2] The MapReduce-based approach to improve the shortest path computation in large-scale road networks: the case of A* algorithm - Wilfried Yves Hamilton Adoni, Tarik Nahhal, Brahim AghezzafEmail and Abdeltif Elbyed Journal of Big Data 3rd May 2018

[3] Hadoop, A. Welcome to Apache Hadoop. http://hadoop.apache.org/. Accessed 10 Mar 2017

[4] Ghemawat S, Gobioff H, Leung ST. The google file system. In: ACM SIGOPS operating systems review, vol. 37. New York: ACM; 2003. p. 29–43. https://doi.org/10.1145/945445.945450.

[5] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Commun ACM. 2008;51(1):107–13.

[6] Vavilapalli VK, Seth S, Saha B, Curino C, O'Malley O, Radia S, Reed B, Baldeschwieler E, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H. Apache hadoop YARN: yet another resource negotiator. In: Proceedings of the 4th annual symposium on cloud computing. Santa Clara: ACM Press; 2013. p. 1– 16.

[7] Precise Version Control of Trees with Line-based Version Control Systems Dimitar Asenov1 , Balz Guenat1 Peter M¨uller1 , and Martin Otth2