ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

MapReduce Framework for Data Cache in Big-Data

A.Karthik[1], Dr.T.Meyyapan[2]

[1] *M.Phil Research Scholar, Department of Computer Science and Engineering*
[2]*Professor, Department of Computer Science and Engineering*
Karaikudi,Tamilnadu.
[1]karthikindhiran@gmail.com,
[2]meyyappant@alagappauniversity.ac.in

Abstract—The term big data indicates to the large scale distributive data processing applications that operate on remarkably large amount of data. Hadoop is familiar open source software to implement the MapReduce framework systems for big data. MapReduce framework produces a huge amount of intermediate data. That intermediate information is discarded after the tasks finish, because MapReduce can't utilize them. In the proposed data cache framework method, every task submits their intermediate result to the cache manager. The intermediate results are managed by the cache manager. Tested experiment results demonstrate that cache considerably improves completion time of MapReduce job.

Key words: Big data cache manager, cache manager , map reduce cache.

I. INTRODUCTION

Google MapReduce is a programming and software framework for large scale Distributed computing on large amounts of data.MapReduce is mostly developed for the organization and processing of Big Data is very huge amounts of data that look for high level of analyzing capabilities.MapReduce concepts are using automatically parallel job scheduling system is Computation in terms of a Map and a Reduce function[1]. MapReduce is a programming concept that runs in the background of Hadoop to supply scalability and simple data processing solutions. The Hadoop consists of the Hadoop is ordinary, which provides access to the file systems are supported by Hadoop[2]. Mainly the (HDFS) Hadoop Distributed File System provides distributed file storage and is optimized for huge unchallengeable blobs of data[3]. The Hadoop group will consist of a single master and several workers nodes. The master node runs many processes, with a JobTracker and a Name Node. The JobTracker is incharge for managing operation jobs in the Hadoop cluster. The HDFS is managed by the NameNode[4]. The JobTracker and the Name Node is typically collocated on the similar physical machine. MapReduce jobs are separated into tasks.The TaskTracker is managing to the Tasks.The Map function works with the help of mapper class. The Reducer function works with the help of reducer class.

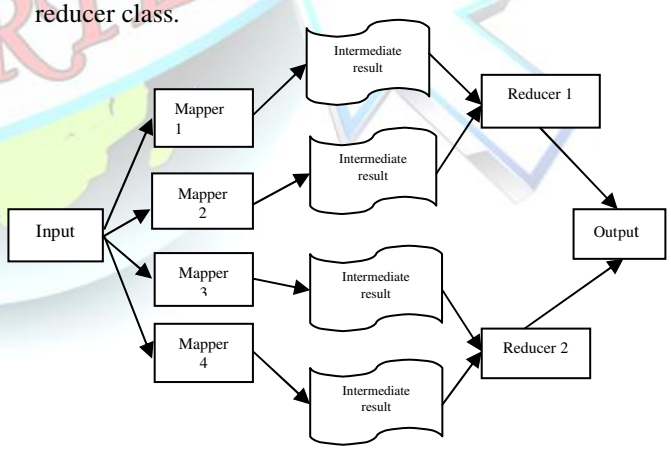

Fig. 1 A high-level illustration of the MapReduce programming model and the underlying implementation architecture.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

II. MAP PHASE

The Map phase is user defined function. In Map phase is input is loaded. First split and then feed mappers. Records are called each in individual data items. The MapReduce concepts are the input split the files to each worker and produces records. The size of the total intermediate data can be very large mapReduce framework generates large amount of intermediate data. The intermediate results are stored and manage in the cache manager.Cache mention to the intermediate data that is generate by worker nodes and processes during the implementation of a MapReduce task. A part of cached data is stored in a DFS-Distributed File System. Cache item is define by a Two tuple is a source and process. The source is the name of a file in the Distributed File System. The process is a linear list of available operations performed on the source file. The word count applications, each the mapper node and the process omit a list of word and count tuples that documentation the count of each word in the file that the mapper processes. The designed and implemented by application developers who are in charge for implementing their MapReduce tasks. The prototype, present several supported operations. The cache things are incorporated and managed by the cache manager in which answers the queries for mappers and reducers.

- **Item Count**: All occurrences of each item in a text file is counted where the items are separated by a user defined separator.
- **Sort**: Records of the files are sorted using sort operation.
- **Selection**: This process sorts the records of the file. The relationship operator is defined on two items and returns the arrange of priority This process choose an item that meets a given Condition. A particular selection process involves selecting the middle of a linear list of items.
- **Transform**: This process transforms all items in the input file into a various item. The transformation is described further by the additional information in the process descriptions.
- **Classification**: This process declares the items in the input files into several groups. The cache items are incorporated and managed by the cache manager which answers the querries for mappers and reducers.

III. REDUCE PHASE

The map phase cache description used on scheme, the original input and the functional processes are compulsory. The original input is received by storing the intermediate outcome of the map phase in the Distributed File System[5]. The solution is to apply a better explanation for the original input of the cache items in the reduce phase. The explanation should contain the original data files generated in the Map phase. For instance, two data files file A and file B, are companied and produce two middle results, input A and input B, for two reducers input A and input B should contain file A and file B as its shuffling source. Thus, resulting with a formation of new intermediate data files obtained from incremental processing of the Map phase. In the similar way the shuffling inputs are identified. The reducers can distinguish new inputs from the shuffling sources by the newly arrived intermediate result from the Map phase to form the final results. For instance, consider as another input called C is a newly obtained result from Map phase, the shuffling results file A and file B contain a new shuffling source, inputC.data. As the result of shuffling input A, input B, and input C a reducer recognizes the input, file A. The output of input A and input B are obtained by querying the cache manager to produce the final results. To get new results shuffling output of input C is added.

IV. IMPLEMENTATION

Input is loaded for mapper phase. Before the mapping process input is digesting work is done using the algorithm of Message Digest (MD8). Then the input is going to next process called mapping. Mapper phase split the files and assign the partial work to various mappers and produces the intermediate results. The Map phase is shuffle and sort the intermediate files. The Map phase Digested input data are intermediate files stored in then incorporate to cache manager. Then the intermediate output is given as the input for the Reducer phase.

The output is generated with the help of Reducers. If the work is required next time, after the input process the Cache manager checks in the input files. If the files are same in the input files are directly send to the Reduce phase. But not stored in the cache manager. Otherwise the intermediate files of the input files going to get place in cache manager. Then cache manager send the intermediate files to Reduce phase. Final results are computed by multiple Reducers then provide the output.

With the help of mapper input can be divided as many small parts. These dividing processes paralyze the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

work. So the speed of execution becomes higher. Then partial files are processed and generate the intermediate result. Intermediate results are placed into cache manager it the same work is required one more time.

Then directly cache manager produce the intermediate result to the reducer from this way the speed of execution is increases.

Step 1: Start
Step 2: Give the input to process
Step 3: Split the work using Mapper phase
Step 4: Digest the Message using MD8
Step 5: Checkout the Binary value in cache
Step 6: If the Binary value is already present
Directly give the intermediate result to the Reduce Phase
Step7: If the Binary value does not present do the Mapper Work and update the cache manager.
Step 8: Process the Reduce work
Step 9: Produce the result

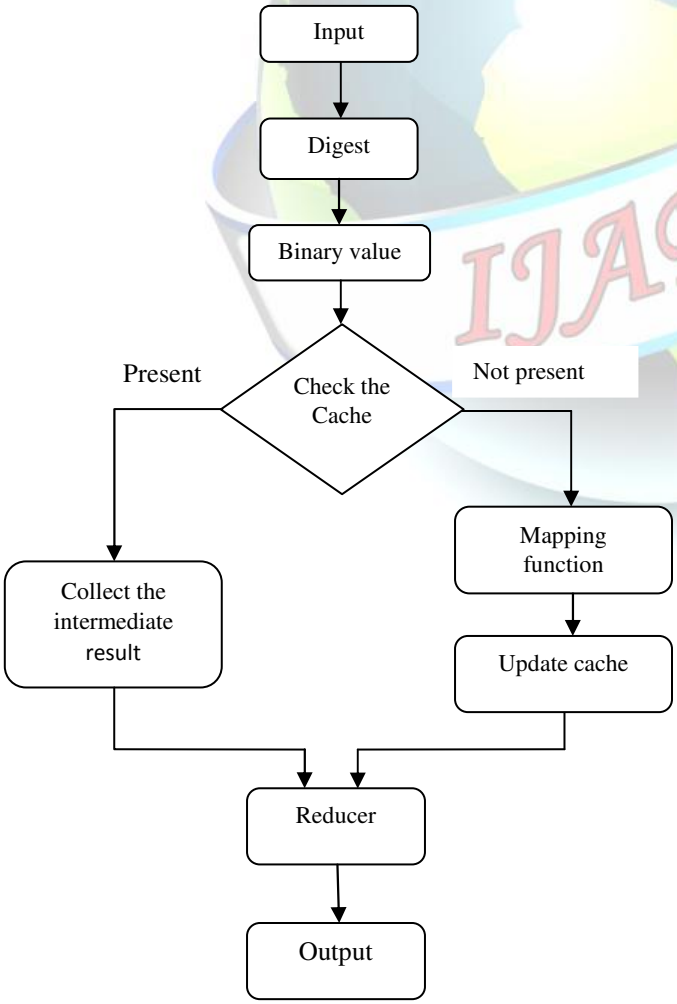

Fig.2 Implemented MapReduce Framework for Data Cache in Big-Data Architecture.

Figure 2 shows the process in step by step. Cache manager placed between map and reducer phase. It is used to avoid the duplication of work.

V. RESULT AND DISCUSSION

Existing method provides the speed in limited way. Proposed method execution process done in high speed. Existing method execution speed is limited and slow when the amount of data is increased. Map Reduce framework for data cache in big data method having the quality of standard speed of execution. When the amount of data input is increased then the speed of execution also increases.

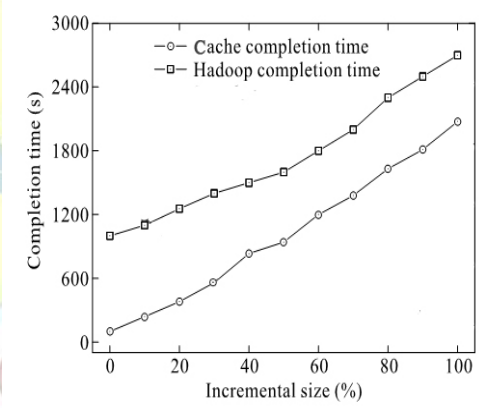

Fig. 3 The Cache over the Hadoop and their completion time of process

Figure 3 shows the speed of execution. Cache completion time is lesser than the hadoop completion time.

V. CONCLUSION

Map reduce process provides large amount of intermediate data and this framework is unable to use the intermediate data. In MapReduce, Framework for Data Cache stores the spilled work's results data to the cache manager. It uses the intermediate data in the cache manager before executing the actual computing work. It can eliminate all the duplicate tasks in incremental Map Reduce jobs. It also provides less execution time. When the intermediate data are stored in cache, the data will be collected shortly. In Future, the cache memory will be increased for storage and input files will also the increased. In this work the input file is a single. In the future work, multiple files will be used for input.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

REFERENCES

[1] Yaxiong Zhao_, Jie Wu, and Cong Liu, Dache: A Data Aware Caching for Big-Data Applications Using the MapReduce Framework,vol 19,no. 1,pp. 2014

[2] J. Dean and S. Ghemawat, Mapreduce: Siplified data processing on large clusters, Commun. of ACM, vol. 51,no. 1, pp. 107- 113, 2008.

[3] L. Popa, M. Budiu, Y. Yu, and M. Isard, Dryadinc: Reusing work in large-scale computations, in Proc. ofHotCloud'09, Berkeley, CA, USA, 2009.

[4] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi, Query optimization for massively parallel data processing, inProc. of SOCC'2011, New York, NY, USA,2011.

[5] M. Zaharia, A. Konwinski, A. D. Joseph, R.Katz,and I. Stoica, Improving mapreduce performance in Heterogeneous environments, in Proc. Of OSDI 2008,Berkeley, CA, USA, 2008.