International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

An Adaptive Partitioning Technique to Improve the Performance of Bigdata

R. Siva Kumar[1], K. Nageswara Rao[2]

[1]M. Tech, Department of Computer Science and Systems Engineering, Andhra University, Visakhapatnam

[2]Assistant professor, Department of Computer Science and Systems Engineering, Andhra University, Visakhapatnam

Abstract: ***The performance of parallel computing purely depends on the data partitions. Hadoop is the framework to perform the computations on the partitioned data across many systems and produce the results. It also suffers with the statistic partitioning concept present in the framework over large datasets. In this paper we propose a dynamic partitioning algorithm which improves the data analytics over large datasets. Our algorithm provides user friendly reports on the given dataset and reduces the cost of the project. This new algorithm improvises the effective utilisation of the nodes in the cluster and reduces the execution time.***

Keywords: BigData, HDFS, MapReduce, Dynamic DataPartition, Distributed computing

1. Introduction

Day-by-day the amount of data increased is very rapid as it is collected from various sources. It is very difficult to handle this huge volume of data with a single machine so it is distributed among many systems. Hadoop[4] is a framework which is used for distributed computing[3][5] in which Map Reduce programming model is used. The data is stored in Hadoop Distributed File System and is common for all the nodes in the cluster. HDFS[4] as the name implies stores the data in a distributed manner and provides high throughput. It is also more reliable as it stores multiple copies of data in many computer nodes.

MapReduce has become more popular due to its inbuilt fault tolerant nature. It is easily scalable over multiple computing systems. It combines the results of multiple systems and returns a single result. The performance of the Map Reduce depends on the number of Data Partitions. The data is splitted into number of chunks and gives as a input to the mapper to produce the key value pairs as a output. This output is given as input to the reducer and obtains the final result. Before starting the reducer task the Hadoop framework performs data partitioning to sort and shuffle the mapper output according to the key objects. The sorted data is sent to the reducer phase such that the values of same key will be sent to the single reducer. This type of partitioning produces poor results as the data has keys which are partitioned and have non uniform number of values. Here few nodes have completed their task in quick amount of time and has to wait for other nodes to complete their work. This scenario leads to improper utilisation of the nodes and increases the time of execution. To imporvize the utilisation of nodes and to reduce the execution time we are introducing dynamic data partitioning technique.

2. Related Work

Map-Reduce:
zMapReduce [2] is a programming model and is the heart of hadoop. It is designed for processing large volumes of data by dividing the work into a set of parallel independent tasks. The main feature of Hadoop framework is data locality. Data locality means moving the algorithm to the data nodes instead of moving complete data to the node where the algorithm is present. When the computation is over on the data the algorithm is moved across all Data Nodes rather than moving complete data to the Node where the algorithm is written. Moving code to the Nodes is Cheaper than Moving Data in between the nodes.

Map Reduce [7] has an inbuilt fault tolerance mechanism due to the replication factor. By default the replication factor is 3 in hadoop and if we want we can change it by modifying the configuration files data. Hadoop is a set of 5 deamons in which job-tracker and task tracker are the map reducing daemons.

Hadoop has a Master-Slave Architecture[4] in which the jobtracker executes in the master node and the task tracker runs in the slave system. Map Reduce jobs are assigned to the Master node. The Job Tracker checks out the availability of the task trackers and assigns the work to the nearest node. The node which responds to the RPC fast is considered as the Nearest node. Job Tracker checks the status of the all Task Trackers by sending a heart beat for every few minutes whether the Task Tracker is dead or alive. If any of the Task Tracker is dead it assigns the work to the next nearest node.

Data Flow of Map Reduce:
In MapReduce the mapper takes input as a key value pairs and generates the list of key value pairs. The output of the map method is taken as input for the partitioning and it generates the output as a (key, list of values). The output of the partitioning method is taken as a input for the reduce method and generates the output as a (key, value) pairs. The general form of the functions are represented as follows.

mapper: (K1, V1) → list(K2, V2)
partitioner: list(K2, V2) → (K3, list(V3)
reducer: (K3, list(V3)) → list(K4, V4)

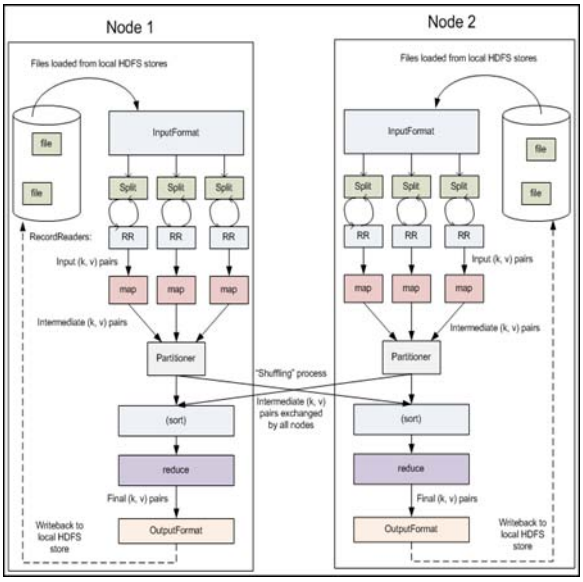

Figure 1: Data flow of map-reduce

Dynamic Data Partitioning

Map Reduce job takes input from the HDFS and process the data and produces the output onto HDFS. The input dataset is splitted as chunks according to the block/split size where each map task process the chunks and produces the output as a (key, Value) pairs. The output from the map is taken as a input for the reduce method and process according to the methods present in it. Before starting the reducer phase partitioning takes place which maps the outputs based on the keys of the mapper result i.e., the records with the same key are sent to the same reducer. Data partitioning[1] can be done in two ways

- Framework Driven Partitioning
- User Driven Partitioning

By default hadoop uses the Hash partitioning[6] function to partition the data. Hash partitioning uses the hash code method to partition the data into their appropriate reducers i.e., it chooses which Key Value(KV) pair is to be sent to which reducer by calculating the (number of key objects) modulo (total number of partitions).

In the User Driven Partitioning [1] the user can perform the partitions according to the data. The user must have an idea about the data on which he wants to perform partitions. The user needs to overwrite the predefined function in his own code using get Partition method. By this we will reduce the cost of the project.

3. Methodology

Case of Poor Partitioning

Suppose if the dataset having a particular key appears more times than the other keys and if you want to send the values of your key to single partition, and then the other key values to remaining partitions according to their hash mappings are done in two ways.

Firstly, the key which has more values is sent to one partition. The remaining keys are sent to the partitions according to their hashCode mapping. So the data is not uniformly partitioned. This partitioned data is given as input to the reducers. So the partition which contains more data will take more time and the remaining are waiting for this reducer task to complete. So here we should arrange in such way that the data may be shared across different reducers. The dataset considered is sensex data which is in the form described below

Example
sensexid
sensex reportname
sensexloc
sensexturnoverrate
sensex flurate
sensex closingbal
sense impact on market

Example Data:
1000 NSE-DAILY-report delhi 25000 15 26170 positive
1001 bse-weekly-report Mumbai 6500 09 7500 Negative

The daily report of the sensex will be calculated based on the fluctuation rate so we can dynamically partition the data according to the values present in the attribute sensex flurate. If the Fluctuation rate is 25 and above consider it as Highest Record of the day and if it is in the range of 18-25 consider it as Average Record of the day if the value is in the range of 10-17 then it is normal record. If it is less than 10 consider it as poor record.

Data should be populated with the fields Sensex Id, Sensex Loc, Sensex Turnverrate, Sensex FluRate, ClosingBal and Market Impact.

The weekly report of the sensex will be calculated based on the fluctuation rate so we can dynamically partition the data according to the values present in the attribute sensex flurate.

If closing balance is greater than 35000 consider it as a GoodTurnover and perform it with a reducer and if it is in the range of 20000-35000 consider it as an AvgTurnover and send the related data to one reducer and if it is less than 20000 consider it as a BadTurnover. So by this dynamic partitioning the user can easily get the reports and also provide the effective utilisation of the resources. The below figure shows the execution time of the Sensex data with normal partitioning and dynamic partitioning by varying the total number of splits of the input data.

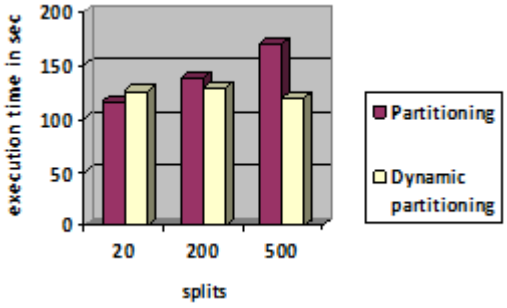

Figure 2: Execution Times in Milliseconds

From the above figure it clearly proves that if the number of splits are increased the ddynamic data partitioning produces the results very fast.

4. Conclusion

Hadoop Map Reduce programming is capable of processing very large amount of data in parallel. The effectiveness of the model depends on the number of partitions of the data. Hadoop uses Hash partitioning method to make the partitions after the map method is done. It produces effective results but consumes more time. In this paper we used dynamic data partitioning method to produce the faster results. Our method will make use of the resources effectively and produces the results. By the results we prove that the Dynamic Data Partitioning will perform well with the increase in the number of splits when the data is huge in volume.

References

[1] Kenn Slagter, Ching-Hsien, HsuYeh-Ching Chung Dynamic Data Partitioning and Virtual Machine Mapping: Efficient Data Intensive Computation 2013 IEEE International Conference on Cloud Computing Technology and Science
[2] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters”, Proceedings of the 6thOSDI Symposium, 2004.
[3] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: a survey," ACM SIGMOD Record, vol. 40, 2012, pp. 11-20
[4] http://hadoop.apache.org
[5] J. Dittrich and J.-A. Quiané-Ruiz, "Efficient big data processing in Hadoop MapReduce," Proceedings of the VLDB Endowment, vol. 5, 2012, pp. 2014-2015.
[6] F. Chang, J. Dean, S. Ghemawat, WC Hsieh, DA. Wallach, M. Burrows, T. Chandra, A. Fikes and RE. Gruber, “Bigtable: A Distributed Storage System for Structured Data”, ACM Transactions on Computer Systems, Vol. 26, 2008.
[7] Ekanayake, J., Pallickara, S. and Fox, G. (2008) „MapReduce fordata intensive scientific analyses“, in eScience ‘08, IEEEFourth International Conference on, pp.277–284.