Global Scientific JOURNALS

GSJ: Volume 6, Issue 7, July 2018, Online: ISSN 2320-9186
www.globalscientificjournal.com

A Study on BigData Management in an Apache Environment

1. **Prof.M.Suneetha**, Research Scholar, Computer Science & Eng,
Rayalaseema University, Kurnool, A.P(Inida).
2. **Dr. VSK Reddy**,PhD,(KGP), Principal, MRCET, Hyderabad,A.P(India)

Abstract

MapReduce is a processing technique and programming model done in lateral and scattered manner is the core component of BigData. It is an associated implementation for processing and generating large data sets. MapReduce is used to perform the task of filtering, aggregation and to maintain the efficient storage structure. The proposed method will process the huge volume of data in parallel as small chunks in distributed clusters.

Keywords: Hive, Hadoop, BigData, MapReduce, HDFS,Partition.

Introduction

BigData as the name suggests deals with the large amount of data. Big data has evolved from various stages starting from primitive and structured data to complex relational data and now very complex and unstructured data. Data sets grow rapidly. Data is most important as it helps organizations as well as persons to take out information and use it to make various decisions. To manage and retrieve Data it should be stocked in database so that easily manageable. Using Database Management System all the operations of data handling and maintenance are much monotonous task in growing data environment. One of the best solution is partitioning . Partitioning provides user-friendliness, maintenance and impulsive query performance to the database users.

MapReduce is associated implementation for processing and generating large data sets.

Architecture

In Hadoop Cluster MapReduce is mainly used in parallel processing to manage huge data To provide parallelism, Distribution of Data, and fault tolerance this has designed by Google. MapReduce handles the data in the form of Key value pairs. This pair is nothing but a mapping element between two data items which are linked. In the Key value pair , the key (Q) acts as an identifier for the value (A). So the key value pair (QA) is a pair in which Q is a node.

To process huge amount of data Mapreduce role is a major one. The Mapreduce follow different and the output is stored in Hadoop Distributed File Structure with replications. Job Tracker reduces jobs and it plays a main role in scheduling and tracking of jobs. Task Tracker responsibility is to map and reduce tasks.

Hadoop MapReduce Architecture

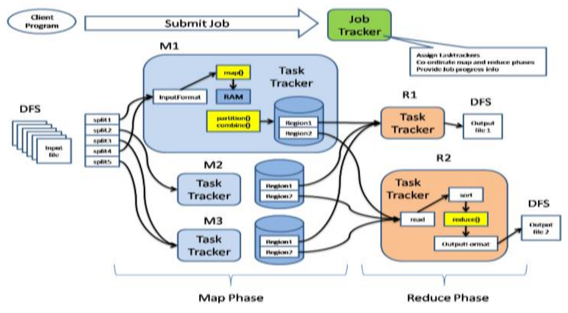

Fig 1. Hadoop MapReduce Architecture

The architechture of Mapreduce consists of two stages of processing one is Map stage and the other one is reduce stage. The Intermediate process stage is in between the map stage and reduce stage. The shuffle and sorting operations of the output data will be processed by the intermediate and then the data will store in the local file system of Apache environment.

Mapper Phase

In the mapper phase the input will split into two components. They are Key and Value pairs (QA). During the processing stage the key is writable and comparable the value is during the processing stage only writable. If input is given to Apache environment letus say Hadoop system then the Job tracker assigns tasks to task tracker. tasks to task tracker. The input data can be split into several splits. In general the input splits are the logical splits. The Record reader converts these input splits into the pairs of Key Value(QA). For different applications the input format may vary. The mini Reducer is a Combiner. If the output of the mapper is large then it requires high network bandwidth. For the better performance and to solve the bandwidth problem place the reduced code into mapper as combiner. Hash partition is used here by default. Partitioning aka bucketing plays an important role in dealing with huge data. Partitioning can be customized on any data on different conditions and different basis. Partitioning will split data into many folders with the help of reducers at the end of mapreduce phase. For query purpose partitioning is an efficient method. Based on the requirement of the Business the partition code can be designed.

Intermediate Process

The mapper output the data undergoes shuffle and sorting in intermediate process. The intermediate data is going to get stored in the local file system without having any replications in Hadoop nodes. The intermediate data after logical computations will be generated. Hadoop uses a Round-Robin algorithm to write the intermediate data to the local disk.

Reducer Phase

Shuffled and sorted data is going to pass a input to the reducer. In reducer phase, all incoming data is combine and same actual key value pairs are going to write into hdfs system. Record writer writes data from reducer to hdfs. The reducer is not mandatory for searching and mapping purpose.

Reducer logic is mainly used to start the operations on mapper data which is sorted and finally it gives the reducer outputs like part-r-0001etc. Options are provided to the set the number of reducers for each job that the user wanted to run. In the configuration file mapred-site.xml, we have to set some properties which will enable to set the number of reducers for the particular task.

Speculative Execution plays an vital role during job processing. If two or more mappers are working on the same data and if one mapper is running slowly then the Job tracker assigns tasks to the next mapper to run the program fast. The execution will be on FIFO (First In First Out).

MapReduce word count Example

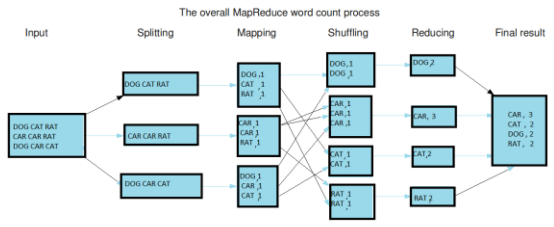

Fig.2: MapReduce word count example

Suppose the text file having the data like as shown in Input part in the above figure. Assume that, it is the input data for the MR task. We have to find out the word count at end of MR Job. The internal data flow can be shown in the above example diagram. The line splits in splitting phase and gives a key value pair to input by record reader.

Here, three mappers are running parallel and each mapper task is to going to generate output for each input row that comes as input to it. After mapper phase, the data is going to shuffle and sort. All the grouping will be done here and value is passed as a input to Reducer phase. The reducers then finally combine each key-value pair and pass those values to HDFS via record writer.

MapReduce Algorithm

Generally MapReduce paradigm is based on sending map-reduce programs to computers where the actual data resides.

- During a MapReduce job, Hadoop sends Map and Reduce tasks to appropriate servers in the cluster.
- The framework manages all the details of data-passing like issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on the nodes with data on local disks that reduces the network traffic.
- After completing a given task, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

Inputs and Outputs

The MapReduce framework operates on key-value pairs, that is, the framework views the input to the job as a set of key-value pairs and produces a set of key-value pair as the output of the job, conceivably of different types.

The key and value classes have to be serializable by the framework and hence, it is required to implement the Writable interface. Additionally, the key classes have to implement the Writable Comparable interface to facilitate sorting by the framework.

Both the input and output format of a MapReduce job are in the form of key-value pairs –

(Input) <Q1, A1> -> map -> <Q2, A2>-> reduce -> <Q3, A3> (Output).

	Input
Map	<Q1, A1>
Reduce	<Q2, list(A2)>

Big Data management Implementation

Many leading enterprises are using analytics to gain competitive advantage. They are investing in big data analytics and smartly outperforming their competitors. If your organization doesn't have BIG data analytics strategies in place, you are missing the BIG opportunity.

MapReduce is a framework that is used for writing applications to process huge volumes of data on large clusters of commodity hardware in a reliable manner. This chapter takes you through the operation of MapReduce in Hadoop framework using Java.

MapReduce Implementation

The table includes the monthly electrical consumption and the annual average for five consecutive years.

We need to write applications to process the input data in the given table to find the year of maximum usage, the year of minimum usage, and so on. This task is easy for programmers with finite amount of records, as they will simply write the logic to produce the required output, and pass the data to the written application.

Let us now raise the scale of the input data. Assume we have to analyze the electrical consumption of all the large-scale industries of a particular state. When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be heavy network traffic when we move data from the source to the network server.

To solve these problems, we have the MapReduce framework.

Conclusion

There are two approaches considered by the MapReduce according to the Hadoop distributed file system research that focuses on handling small data files. Those two are completion time on hadoop input cluster and the usage of memory. By the consideration of these two approaches, there is advanced in the algorithm proposed. Future work is in progress. Need to survey on this topic more.

Acknowledgement

This is a simply study and survey paper to know more about my research concepts.

REFERENCES

1. Bittencourt, L.F. and Madeira, E.R.M. "A Performance-Oriented Adaptive Scheduler for Dependent Tasks on Grids," Concurrency and Computation: Practice and Experience.

2. Caron, E. Chis, A. Desprez, F. And Su, A. "Design of Plug-in Schedulers for a GRIDRPC Environment," Future Generation Computer Systems, vol. 24, no. 1, pp. 46-57.

3. Dinda, P.A. And O'Hallaron, D.R. "Host Load Prediction Using Linear Models," Cluster Computing, vol. 3, no. 4, pp. 265-280.

4. Dinda, P.A. "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 2, b pp. 160-173.

5. Eddy Caron, Andreea Chis, Frederic Desprez, Alan Su (November 2011) "Plug-in Steering of Computational Grids," Int'l J. High Performance Computing Applications, vol. 14. no. 4, pp. 357-366.

6. Waheed et al., "An Infrastructure for Monitoring and Management in Computational Grids," Proc. Fifth Int'l Workshop Languages, Compilers and Run-Time Systems for Scalable Computers, vol. 1915, pp. 235-245.
7. Wolf, F. and Mohr, B. "Hardware-Counter Based Automatic Performance Analysis of Parallel Programs," Proc. Conf. Parallel Computing (ParCo '03), pp. 753-760.
8. Massie, M.L. Chun, B.N. And Culler, D.E. "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Parallel Computing, vol. 30, no. 7, pp. 817-840.
9. Peter Dinda, A. and David R. O'Halloran (July 2012) "AN Extensible Toolkit for Resource Prediction in Distributed Systems" School of Computer Science Carnegie Mellon University Pittsburgh, PA, 15213.
10. Sam Verboven, Peter Hellinckx, Frans Arickx and Jan Broeckhove (2011) "Runtime Prediction based Grid Scheduling of Parameter Sweep Jobs" University of Antwerp Antwerp, Belgium.