International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

MapReduce: Architecture and Internals

V. Sajwan[1], V. Yadav[2]

[1, 2]MGM's College of Engineering and Technology, Noida Sector-62, India

Guided By: **Dr. S. Srivastava**
MGM's College of Engineering and Technology, Noida Sector-62, India

Abstract: ***MapReduce is programming model and implementation for generating and processing large data sets. Users specify map function that processes key function to generate key pair and reduce function that merges all the resulted value with the sane intermediate value. Programs written in this function style are automatically executed parallelly on the large cluster. The run-time system take care of the complexity in partitioning the data input, scheduling the program's execution across the machine cluster and inter-machine communication. This allows programmer to easily utilize the resources of the whole distributed system. In this paper, we describe the architecture of the mapReduce and its working .We also describe the components of the MapReduce and give its brief description .We have discussed the internals of the MapReduce and explain the working of MapReduce with an example.***

Keywords: MapReduce, Job Client, Task Tracker, Job Tracker, Name Node

1. Introduction

In our world the data are measured in tera- and petabytes. Over the past five years, many organization and social media process large amounts of raw data such as text document, crawled document,web request logs etc. Most such computation is straightforward. In such case, input data is usually large and computation has been distributed across the hundred of machines in order to finish the task in reasonable amount of the time .The issues of how to parallelize computation the whole data set.

As a reaction to this complexity we design a new abstraction that allows simple computation parallelly. MapReduce framework was originally developed by Google MapReduce is the programming model for processing and generating large data sets with parallel, distributed algorithm on a cluster. The MapReduce function is consist of map () and reduce () function. Map() performs filtering and sorting and a Reduce() performs a summary operation .The "MapReduce System" processing by distributed servers , running the various tasks in parallel, managing all communication and data transfers between the various parts of the system and providing for redundancy and fault tolerance. The model is inspired by the mapper and the reducer function commonly used in the functional programming. MapReduce libraries has been written in many programming languages with different level of the optimization. A popular open source implementation that has support for distributed shuffles is the part of Apache Hadoop. It is programmable framework pulling data in parallel out of our cluster. MapReduce is the low level programming and that why Hive and pig evolved. Hive and Pig are simple as compare to the MapReduce and more user friendly.

The main reasons of this work are simple and powerful interface that allows automatic parallelization and distribution on the large data sets.

Section 2 describes the architecture of the MapReduce and its components. Section 3 describes the working of the MapReduce. Section 4 describes the Internals of the MapReduce.

Over 70 companies use Hadoop including Yahoo!, Facebook, Adobe and IBM. The programmer may abstract from the issues of distributed and parallel programming because it is the MapReduce implementation that takes care of the network performance, fault tolerance, load balancing etc. Map written by the user takes the input pair and produces a set of intermediate key/values pairs. The reduce function is written by the user and it merges the values from the set of the values for that key. MapReduce is used in weather forecasting. We have so much prediction based on MapReduce. It is also used in Health Care like patient report, ECG monitoring etc.

1.1 Key Features of MapReduce

1.1.1 Scale-out Architecture :- We can add server to increase its processing power. If we want to add more data all we have to do is add one more node.
1.1.2 Security Authentication :- Working with HDFS make sure that only approved users can work with the data in the system. It gives security to the user.
1.1.3 Flexibility :- We can write the map function and reduce() in any programming language like java and python.
1.1.4 Resiliency and High availability :- Multiple job trackers and Task trackers monitors that jobs fail independently and restart automatically.
1.1.5 Optimized Scheduling :- MapReduce performed scheduling according to prioritization.

2. MapReduce Architecture

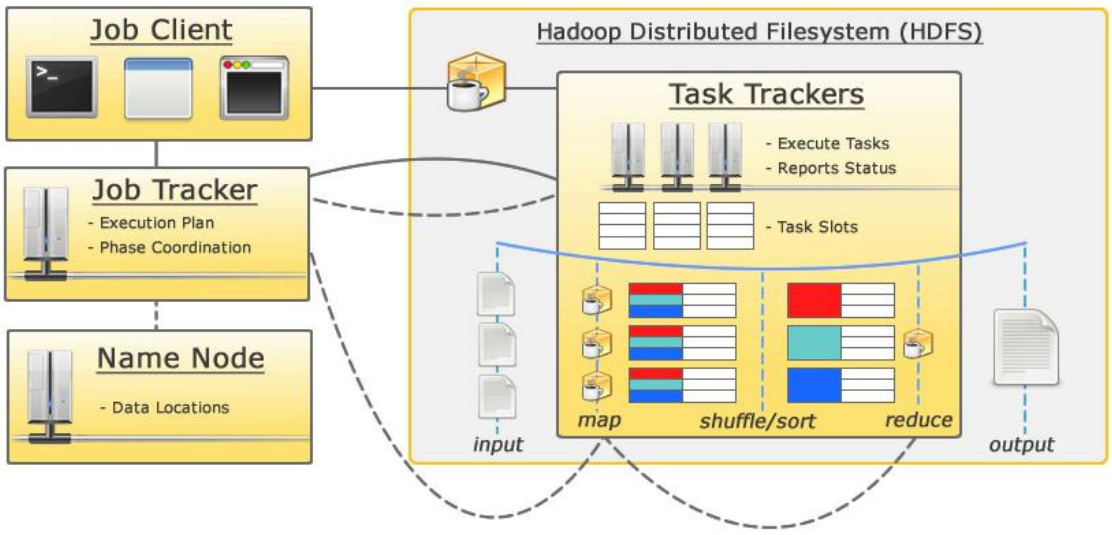

Figure 1: Internals of MapReduce

2.1 Job Clients

The job client the one who submits the job.A job contains the mapper function and reducer function and some configuration function that will drive a job.

2.2 Job Tracker

The job tracker is the master of task trackers which are the slaves work on data nodes. The job tracker responsibilities to come up with the execution plan and it is coordinate and schedule the plan cross the test trackers. It also can do phase coordination.

2.3 Test Tracker

Test tracker is the one who gonna break down the job into tasks that is map and reduce task .Every task tracker has slots on it. the job tracker take map and reduce function all of them compile binary and throw them into the task slots which actually do the execution over the map and reduce functions. They also report the their report back to the job tracker. If something fails the job tracker will know about it. It just reschedule that task on the another task tracker.

3. Working of MapReduce

So lets say we sit in our command shell and Hadoop is the jar file user go execute this job in the cluster.

Step1: Job Client submit their job to the job tracker also in the background its gonna copy the binary that contains the mapper and reducer function implementation along with configuration information those input and output pass its gonna put those all inside the HDFS. It is centralized area that closed to the test trackers because once they need use that node they gonna download it locally on the data node by using test tracker's command

Step 2: Job tracker query the name node .Job trackers want the location in the names of the data nodes that contains the data it is working for.

Step 3: On that information its gonna go to the test tracker and job tracker forms the execution plan is by communicating to the first of the task tracker closes to the data. It first communicates with task tracker which is closes to the data if they have available slots. If they don't have slots then it goes to the data locality find out the any task tracker in the same rack have available slots and if it can't do that it just pick any rack in the cluster it has task tracker have available slots. It can be data local which is optimal, rack local or cross rack. So it's gonna figure all that up based on what is available task slots closest to the data . So ones it figure all that stuff out it's gonna create its execution plan

Step 4: From their task tracker take over and do the heavy lifting. So the task tracker will execute the mapper and reduce code and send that data through the mapReduce pipeline

Step5: It executing all all the information its sending statistic report heartbeats back to the job tracker. It sends information about task and what phase currently in so in that way we can perform phase coordination all the maps need to be finish before it can hit to the reduce phase and also information .It also report slot availability so if slot open up we can schedule more maps on the task tracker and evens from other jobs

Step 6: Job tracker manages those phases when all those phases are done all over task tracker is finished up we get final output .The job tracker is gonna update statistic success which will notify the client and all the while the client can tap and get the information. If we feel we can run the job we can personages our mapper and reducer phases.

4. Internals of MapReduce

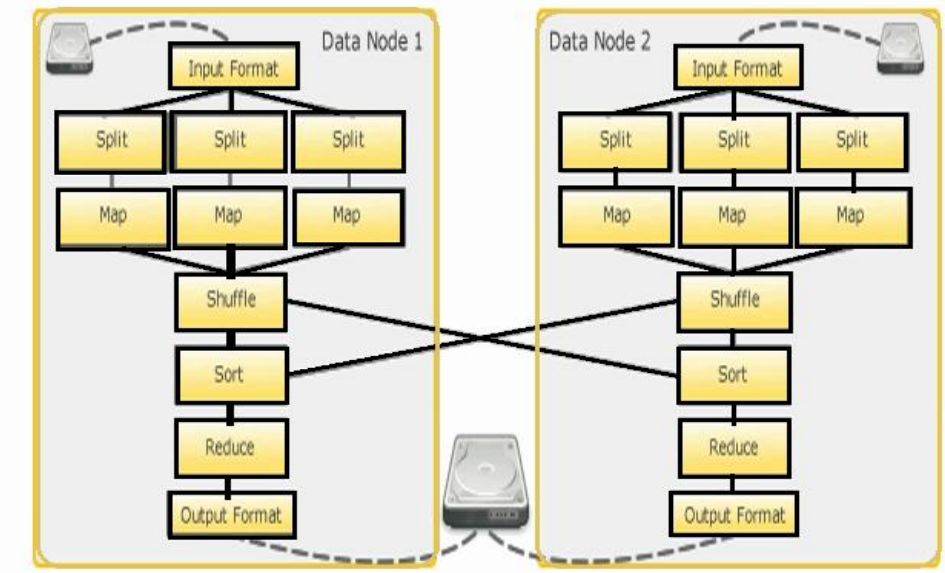

Figure 2: Internals of the MapReduce

In Figure 2, we can see the internals of the MapReduce. These are the phases of the MapReduce. The flows through these phases and gives us output

4.1 Spilt Phase

The split phase uses the input format to bring data off the disk out of HDFS and split it up. The default input format is the text input format which breaks up the data line by line. Each line is split and send to the mapper. So, if we have large data we could have thousands and thousands of mappers running (depending on the cluster setup) simultaneously against this data .There are variety of input format depending on the kind of the data which we storing the HDFS. If we storing the image data there is binary input format. If we storing database there is database input format .

4.2 Map

The mapper just goes and gets the data we want. It runs on the key that we pairs. It transforms the input split into the pairs based on user-defined code. Mapper is based on keys get the values we want.

4.3 Shuffle & Sort

It gonna takes all the data nodes that are part of this job, shuffle which is portioning and grouping the data and sorting it and send it to the reducers.

4.4 Reducers

Reducers work on the sorted data and aggregated all the results.

4.5 Output Format

Output Format sends the data to the HDFS.

5. Programming Model

To Use MapReduce, the programmer expresses their desired computation as a series of job. The input to a job is an input specification that will yield key-value pairs. Each job consists of two stages : first a user defined map function is applied to each input records to produce a list of intermediate key-value pairs .Second a user-defined reduce function is called once for each distinct key in map output and passed the list of intermediate values associated with key. The MapReduce framework automatically parallelizes the execution of these functions and ensures fault tolerance.

Optionally, the user can supply a combiner function. Combiners are similar to reduce functions, except that they are not passed all the values for a given key: instead, a combiner emits an output value that summarizes the input value is passed.

```
public interface Mapper<K1, V1, K2, V2> {
void map(K1 key, V1 value,
OutputCollector<K2, V2> output);
void close();
}
```

6. Execution in MapReduce

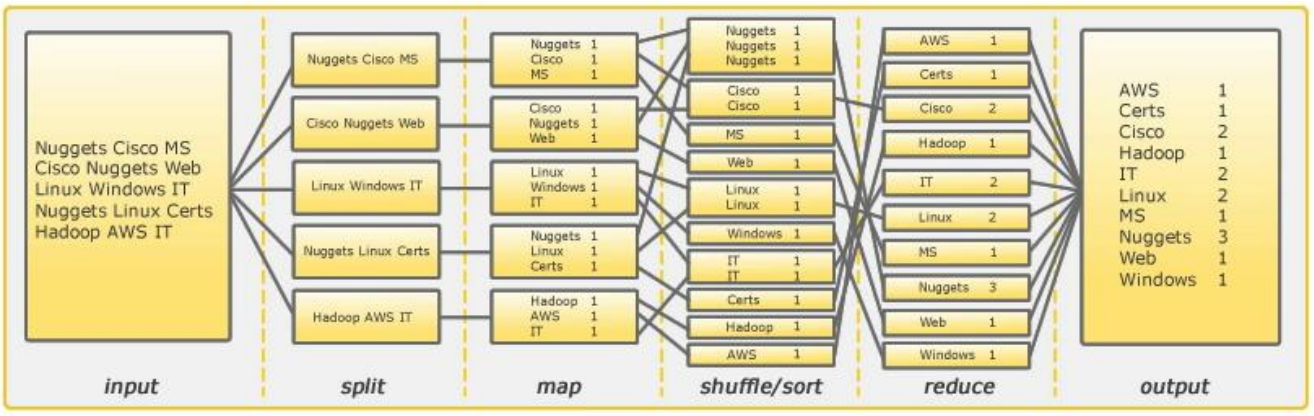

Figure 3: Data Through the MapReduce Internals

In figure 3. We have a file in input containing 5 lines .We split our input. We split our file line by line and then we will send it to the mapper. In mapper the line is broken down into words with count next to them. Then shuffle and sort phase takes all the data to the mapper, brings all the data together and shuffle it or sort it and sends it to the reduce .The reducer have simple job of aggregating all the results and give us final output

7. Conclusion

The MapReduce programming model has been successfully used at Google and many companies for different purposes .We assign this success to several reasons. First the model is easy to use. Without programmer's parallel and distributed system, it hides the details of parallelization, fault tolerance, locality optimization and load balancing. Second, very large problems are easily expressible using MapReduce as MapReduce computations. Third, the MaprReduce implementation makes efficient use of machine resources and hence suitable for many large companies like Google.

We have learnt several things from this work. First, what is really MapReduce is and application and key features of the MapReduce. Second, we get the internals of the MapReduce which give us deep knowledge of MapReduce. Third, we get how data is implemented in the MapReduce with the help of its internals and also learnt about its components and their workings.

8. Acknowledgement

The author is grateful to the participants who contributed to research and our guide Dr. Sanjay Srivastava.

References

[1] Shipa, Manjit kaur, "*Big Data and Methodology",* 10 Oct, 2013
[2] Pareedpa**,** A.; Dr.Antony Selvadoss, "*Significant Trends of Big Data*", 8 Aug, 2013
[3] Gurpeet Singh Bedi, Ashima, "*Big Data Analysis with Dataset Scaling in Yet another Resource Negotiator (YARN)",* 5April, 2013
[4] *Hadoop-The Definitive Guide*, Tom White, Edition-3, 27Jan, 2012
[5] Mrigank Mridul, Akashdeep Khajuria, Snehasish Dutta,Kumar N, "*Analysis of Big data using Apache Hadoop and MapReduce",* Volume 4, May 2014
[6] IBM 2012, What is big data: Bring big data to the enterprise,http://www.01.ibm.com/software/data/bigdata/, IBM
[7] Sam Madden, *"From Databases to Big Data",* IEEE computer society,2012
[8] *"Data Mining with BigData"* ,Xindong Wu, Xingquan Zhu, Gong-Qing Wu, Wei Ding , 1041-4347/13/$31.00 © 2013 IEEE
[9] Russom, *"Big Data Analytics"* , TDWI Research,2011
[10] An Oracle White Paper, *"Hadoop and NoSQL Technologies and the Oracle DataBase",* February 2012
[11] "*MapReduce Online*", Tyson Condie ,Neil Conway ,Peter Alvaro ,Joseph Hellerstein