The MapReduce Framework : Analysis and The Research Perspectives

Abstract –

MapReduce is a simple and powerful programming model which enables development of scalable parallel applications to process large amount of data which is scattered on a cluster of machines. The original implementations of Map Reduce framework had some limitations which have been faced by many research follow up work after its introduction. It is gaining a lot of attraction in both research and industrial community as it has the capacity of processing large data. In this review paper, we are going to discuss the map reduce framework used in different applications and for different purposes. This is a analysis done for implementing the architecture of Map Reduce from different research perspectives.

1. Introduction

As there is the continuous and tremendous increase in the computational power, the overwhelming flow of data is produced. For the same, the large scale data processing mechanism is required. Map Reduce is a simple and powerful programming model that enables easy development of scalable parallel applications to process vast amount of data on large clusters.[1] Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users have to specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with intermediate key. [2] The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.[3]

The Map Reduce model is applied to large batch-oriented computation connected primarily with time to job completion. The Map Reduce framework by Google and open-source Hadoop‟s system emphasize the usage through a batch-processing implementation strategy: the whole output of each map and reduce stage is materialized to stable storage before it can be consumed by the next stage. This materialization allows for a simple that is critical in large deployment, which have a high probability of slowdowns or failures at worker nodes.[4] MapReduce proposed by Google is a programming model and an associated implementation for large-scale data processing in distributed cluster.

2. Hadoop Distributed File System

HDFS is the file system component of Hadoop (Refer Figure 1). While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favor of improved performance for the applications at hand.

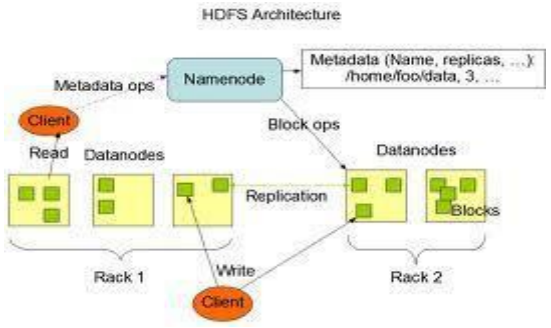

Fig 1 : HDFS Architecture Source: Apache Hadoop Website

Hadoop is composed of Hadoop Map Reduce, an implementation of Map Reduce designed for large clusters, and the Hadoop Distributed File System(HDFS), a file system optimized for batch oriented workloads such as Map Reduce. Hadoop installation consists of a single master node and many Worker Nodes. The master is called as the Job Tracker, is responsible for accepting jobs from clients, dividing those jobs into tasks, and assigning those tasks to be executed by worker nodes. Each worker node runs a Task Tracker process that manages the execution of the tasks currently assigned to that node. [4] Key-value pair forms the basic data structure in Map Reduce. The map function takes the input record and then generates intermediate key and value pairs. The reduce function takes an intermediate key and a set of values to form a smaller set of values

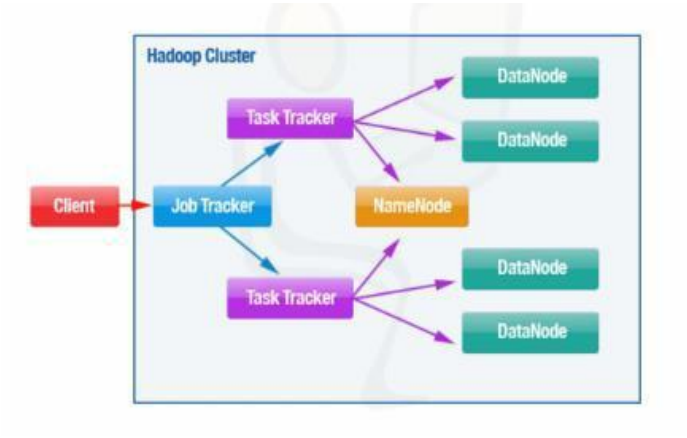

Fig 2 : Abstract View of Hadoop Cluster

3.Writing Files To HDFS :

1. HDFS stores file system metadata and application data separately
2. HDFS stores metadata on a dedicated server, called the Name Node.
3. Application data are stored on other servers called Data Nodes.
4. All servers are fully connected and communicate with each other using TCP-based protocols (Refer Fig 2)

4.Map Reduce Framework –

MapReduce is simple and efficient for computing aggregate. It is compared with "filtering then group by aggregation" query processing in a DBMS. The major advantages of it is ,

- Simple and Easy to use – the MapReduce model is simple but expressive. With MapReduce,a programmer defines job with only Map and Reduce functions, without

specifying physical distribution of the jobs across nodes.

- Flexible- MapReduce does not have any dependency on data model and schema. With MapReduce a programmer can deal with unstructured data easily, this facility is not given by DBMS.
- Independent of the storage – MapReduce is basically independent from underlying storage layers. It can work on Big Table and others[5]. Many projects at Google store data in Big Table which have different demands from Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products such as Google Earth, Google Finance.
- Fault Tolerance – MapReduce is highly fault tolerant, continues working in spite of failures per analysis job at Google.[6]

Instead of saying the disadvantages of MapReduce , it can be said that the MapReduce is **not** a suitable choice when it is :

- Real-time processing.
- It's not always very easy to implement each and everything as a MapReduce program.
    - No high-level language – it does not support any high level language like SQL in DBMS and any query optimization technique.
- When the intermediate processes need to talk to each other (jobs run in isolation).
- When the processing requires lot of data to be shuffled over the network.
- When there is a need to handle streaming data. MapReduce is best suited to batch process huge amounts of data which already is there.
- When you have On Line Transaction Processing is to be needed then MapReduce is not suitable for a large number of short on-line transactions.[7]

5.Research on Framework of Map Reduce –

Chronologically [1] is on the Google File Systems from 2003, which is a distributed file system. Basically, files are split into chunks which are stored in a redundant fashion on a cluster of commodity machines. In this, the GFS serves as a solid base for managing data. The role of Map Reduce is elaborated here, which allows development and execution of large scale data processing jobs. The Map Reduce is designed to be resilient to failures such as machine crashes. Google uses Map Reduce to process data sets up to multiple terabytes in size for purposes such as indexing web content. Next [2] is the MapReduce paper from 2004. MapReduce has become synonymous with Big Data. Legend has it that Google used it to compute their search indices.[4,8] The researchers tried to

improve the performance of MapReduce by decoupling HDFS from Hadoop using pipelining streaming and incremental writes to HDFS. Since Map Reduce depends on large number of user-defined parameters, researchers suggested profiling the user‟s jobs and suggest parameter tunings according the application behavior. [9]

The [10] discusses data access optimizations over Hadoop file system (HDFS) which included using indices, column store and improving data locality using file co-location strategies. On the other hand, other researchers try to improve the performance of MapReduce by decoupling HDFS form Hadoop using pipelining streaming and incremental writes to HDFS. Since MapReduce depends on large number of user-defined parameters, researchers suggested profiling the user‟s jobs and suggest parameter tunings according the the application behaviour. It also included the systems that tries to optimize failure recovery and redundancy of MapReduce through optimizing job placement to improve the performance of Hadoop. After reviewing the work done on MapReduce optimizations, it includes systems that are built over Hadoop to provide reliable and scalable high level operations by expressing them using a series of MapReduce jobs. Finally, it gets finished with reviewing systems that are built based on the architecture of MapReduce but provides more flexible operations and support wider range of applications.[10]

MapReduce is a parallel data processing framework designed for a very specific task: scanning large amounts of textual data to create a web search index. It was essentially designed with GFS. As a result, it boils down computation into just two phases: map, followed by reduce. Programmers have to write just two functions, one for each phase. Regarding the programming model, it‟s something that can be taught in a matter of days. Map and reduce are familiar from functional programming languages, and really small amounts of code can do very powerful things. It is quite but when dealing with big data, the operations basically have to be data-parallel to complete in any reasonable time. Google used MapReduce to do a wide variety of tasks, so the proof of utility is in the pudding.

The distributed, fault tolerant framework is what really is there. A single master manages all the workers (which potentially means a single point of failure), but this is way less likely than a worker failure.[12] But this single point of failure was present in Hadoop‟s Version 1. Prior to Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine.

This impacted the total availability of the HDFS cluster in two major ways:

- In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
- Planned maintenance events such as software or hardware upgrades on

the NameNode machine would result in windows of cluster downtime.

The HDFS High Availability feature addresses the above problems by providing the option of running two redundant Name Nodes in the same cluster in an Active/Passive configuration with a hot standby.[11]

Failure of workers is handled transparently, by restarting the worker. This is possible because the output from the map stage is durably written to disk storage, and then read by the reducers. Mapper input, of course, is durably stored as well, so they can also be easily restarted. Another feature is chopping off the latency tail caused by "straggler" workers by starting duplicate tasks towards the end of the job.

As a whole it can be said that the MapReduce is a powerful and simple framework that saw a lot of use at Google for a variety of tasks.[12]

The basic implementation of the Map Reduce is useful for handling large data processing and data loading in heterogeneous systems. It provides flexible framework for the executions of more complicated functions which can be supported in SQL. MapReduce is highly effective and efficient tool for large scale fault tolerant systems. It provides fine-grain fault tolerance for large jobs; failure in the middle of a multihouse execution does not require restarting the job from the beginning. Some improvements are suggested in the paper, MapReduce: A Flexible Data Processing Tool and they are

a. MapReduce should take advantage of natural indices
b. MapReduce users should avoid using inefficient textual formats
c. Most MapReduce output should be left unmerged, since there is no benefit to merging if the next consumer is another MapReduce program[13]

The use of MapReduce is also done for the analysis of logs. Nowadays service providers generate large amounts of logs from all kinds of services, and the benefits of analyzing them are to be found when processing them. For instance, if a provider is interested in tracking the behavior of a client during long periods of time, reconstructing user sessions, it is much more convenient to operate over all the logs. Logs are a perfect fit for MapReduce for other reasons too. First, logs usually follow a certain pattern, but they aren't entirely structured, so RDBMS is not useful at such times to process them and may require changes to the structure of the database to compute something new. Secondly, logs represent a use case where scalability not only matters but is a key to keep the system sustainable. As services grow,the amount of log increase from time to time. Companies such as Facebook and Rackspace [14] use MapReduce to examine log files on a daily basis as it generates statistics and analyzes them.

Data in a Hadoop cluster is broken down into smaller pieces (called blocks) and distributed throughout the cluster. In

this way, the map and reduce functions can be executed on smaller subsets of the larger data sets, and it provides the scalability that is needed for Big Data Processing.

The goal of Hadoop is to use commonly available servers in a very large cluster, where each server has a set of inexpensive internal disk drives. For the higher performance, Map Reduce tries to assign workloads to these servers where the large data is to be processed is stored. This is known as data locality. [15]

The comparison between the Hadoop implementation of Map Reduce framework and parallel SQL database management systems in terms of performance and development complexity is described here[16], the result of it is, parallel databases displayed a significant performance over Map Reduce in executing a variety of data intensive analysis tasks. MapReduce allows programmers with no experience with parallel and distributed systems to easily utilize the resources of a large distributed system. A MapReduce computation can process many terabytes of data on hundreds or thousands of machines.[16,13]

On the other hand, the Hadoop implementation is very easier and more straightforward to set up and use in comparison with the prior one. Map Reduce have shown the superior performance in minimizing the amount of work that is lost when the hardware failure occurs. [14] [17]

The HadoopDB Project is a hybrid system which tries to combine the scalability advantages of MapReduce with the performance and efficiency of parallel databases. The basic idea behind this is, to connect multiple single node database systems using Hadoop as the task co-ordinator and network communication layer. Queries are expressed in SQL but their execution is parallelized across nodes using MR framework. HadoopDB tries to achieve the fault tolerance and the ability to operate in heterogeneous environment. The MapReduce framework follows simple master-slave architecture. Queries are expressed in SQL, translated into MapReduce by extending existing tools, and as much work as possible is pushed into the higher performing single node databases.[15][18]

The above listed are the different perspectives from which the study of the application of MapReduce is done. Certainly, there are some more viewpoints which we came across in some more research papers for which it‟s necessary to take a note of them, without that the analysis would be incomplete.

In cloud systems, a service provider delivers virtually computer nodes to a number of users. This paradigm will be useful for both the academic researchers as well as industry practitioners as it allows to scale up and down in a pay-as-you-go manner. A cloud data processing system should provide high degree of elasticity, scalability, efficiency and fault tolerance. MapReduce is documented as

a possible means to perform elastic data processing in the cloud. This is for the reason, it has following characteristics –

- MapReduce programming model is simple but expressive and flexible. It is able to access the various types of data, structured or unstructured and data analysis done for traditional query processing, data mining, machine learning and graph processing.

- MapReduce is scalable in nature and installation of it can be run on over thousands of nodes where only tasks on failed nodes have to be restarted.[19,13]

Graphs are popular data structures which are used to model structural relationship between objects. They are used in a wide variety of high impact applications such as social networks, computer networks, telecommunication networks and the World Wide Web.

Eventually, Google also had to start mining graph data like the social graph in an online social network, so Pregel was published in 2010.[20]

The underlying computational model is much more complex than in MapReduce: Basically, you have worker threads for each node which are run in parallel iteratively.

In Surfer[21] a small set of high level building blocks that use the two primitives,

they are Map and Reduce. It provides a GUI using which developers can visually create large graph processing tasks. Surfer transforms a task into an execution plan composed of MapReduce. It automatically applies various optimizations to improve the efficiency of distributed execution. In this paper, the authors demonstrated the ease of programming features of the system and its efficiency on a social network. The authors have assured it is simple and highly efficient for large graph-based tasks.

The DisCo(Distributed Co-clustering) introduces an approach for distributed data pre-processing and co-clustering from raw data to end clusters using data pre-processing and co-clustering from the raw data to end cluster using MapReduce framework. [22]

Conclusion –

We discussed the analytical research perspectives and challenges to MapReduce and mentioned its progress according to the different researcher‟s perspectives. It is simple but provides good scalability and fault tolerance for large data processing. MapReduce can be used as complementary software with the DBMS when used for Data Warehousing. When the data is unstructured in nature then MapReduce is successfully used for analysis while Parallel DBMS can be used when the queries are executed in parallel manner from different nodes.