IJITKM Special Issue (ICFTEM-2014) May 2014 pp. 70-73 (ISSN 0973-4414)

HADOOP: A Framework for Distributed Computing

Sneha Singh
Department of Computer Science And Applications, Kurukshetra University, India
sneha8034@gmail.com

Abstract- With data growing so rapidly and the rise of unstructured data accounting for about 90 % of the data today, the time has come for the enterprises to re-evaluate their approach to data storage, management and its analysis. This enormously growing data has been given the name Big Data. Hadoop platform has been designed to tackle the problems associated with handling such an enormous data-that doesn't fit nicely into tables. Hadoop framework implements Map-Reduce as its programming model and its own distributed file system HDFS in a scalable manner to store and manage terabytes of data. The basic concept used in Map-Reduce algorithm is to divide the task in hand into subtasks and clubbing together the intermediate result sets to generate the most precise result while HDFS has evolved as a better distributed file system than its predecessors in terms of reliability, availability and throughput. The paper illustrates the overall architecture of Hadoop. Given a data intensive application running on a Map Reduce cluster, the placement of data in Hadoop architecture and the workflow of MapReduce process is exemplified.
Keywords: Big data , Hadoop , HDFS , Map Reduce

1. Introduction

With the evolution of technology, there has been a steep decrease in the hardware cost enabling the data scientists store as much information as they need but limitation was imposed by the data traffic that can be put on the network. Moreover, the scalability of computation of petabytes of data and loss of the raw data that has to be converted into relational format before storage were other major limitations that arose with traditional database systems storing Big Data. Beyond business, we don't have to look very far back in the fields of medicine and agriculture to find the time when we were data poor just because we didn't have tools to capture, analyze and store the real data that we were generating[1]. Hadoop has emerged as the ultimate solution to these limitations.

The underlying technology was invented by Google to efficiently index their data. Over a decade, companies like Facebook, Amazon, Yahoo have been using Hadoop as its base technology for storing and processing their vast data repositories. It is an open source technology under Apache License. There are mainly two layers that work in collaboration to form the Hadoop operating system-Execution engine (MapReduce) and Distributed file system(HDFS). Programs using MapReduce as the programming model are automatically parallelized and executed on a large cluster of commodity machines. While MapReduce is taking care of computation, HDFS provides high throughput. In this paper, there is a brief overview of all the facets of Hadoop. The fundamentals of HDFS and MapReduce algorithm has been exemplified. Finally, Hadoop's fault tolerance is discussed and has been compared with traditional distributed database systems.

2. HADOOP Distributed File System

As shown in the Fig. HDFS is one of the two prominent layers of Hadoop. HDFS is the fault tolerant distributed file system with quick and automatic recovery as its chief architectural goals. It has a master/slave architecture. An HDFS cluster consists of one NameNode,a Secondary NameNode and several DataNodes

(i)NameNode

A NameNode acts as a master node that acts as a repository of the file system metadata and regulates access to files by clients. It executes the file system namespace operations. An additional task of NameNode is the reassignment of blocks to the DataNodes in case one of their crashes.

(ii)DataNode

The DataNodes manage storage attached to the nodes that they run on. Internally, a file is split into blocks (by default 64 Kb) and these blocks are stored in a set of DataNodes. The blocks of a file are replicated to provide fault tolerance. The NameNode maps these blocks to DataNodes and manages these replicas through heartbeats sent by DataNodes indicating its normal functionality. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, their deletion and replication upon instruction from the NameNode[2]. DataNodes are arranged in racks. Two network switches are put in the architecture. One switch provides the intra-rack communication while the another provides the inter-rack communication.

(iii)Secondary NameNode

Secondary NameNode comes into action when primary Namenode fails. As status of work is contained in the NameNode so crash of Namenode is accountable. Secondary NameNode is just a copy of NameNode. The concept of checkpoints is used to keep the two NameNodes consistent.

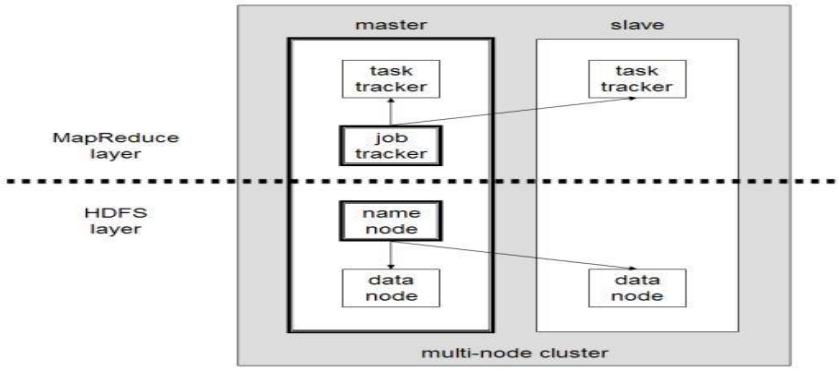

Fig.1 Hadoop Overview

3. Map Reduce

Hadoop is a rapidly evolving ecosystem that implements Google MapReduce Algorithms in a scalable fashion. MapReduce is a programming model used for processing huge data sets. Under the MapReduce model, the data processing primitives are called mappers and reducers. Users specify a map function that processes a key/value pair to produce a set of intermediate key/value pairs, and a reduce function that merges all the intermediate values associated with the corresponding intermediate key. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over.

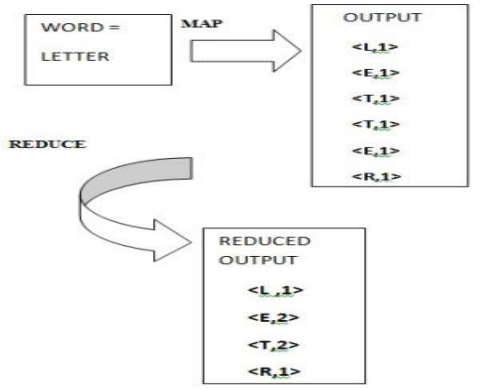

Fig.2 Example Map Reduce

Map function requires the user to handle the input of a pair of key/value and produces a group of intermediate key and value pairs. <key,value> consists of two parts, value stands for the data related to the task, key stands for the "group number " of the value . MapReduce combine the intermediate values with same key and then send them to reduce function. Reduce function is also provided by the user, that handles the intermediate key pairs and the value set relevant to the intermediate key value. Reduce function merges these values to get a smaller set of values. In MapReduce framework, the programmer does not need to care about the details of data communication, so <key,value> is the communication interface for the programmer in MapReduce model [3].

The programs are automatically executed parallely on commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines,handling any machine failures, and managing the required inter-machine communication[4]. The idea of Map Reduce is to hide the complex details of parallelisation, fault tolerance, data distribution and load balancing in a simple library. In addition to the computational problem, the programmer only needs to define the parameters for controlling data distribution and parallelism[5].

The diagram here depicts a MapReduce example. Here the counting of the alphabets of word LETTER is executed on parallel machines to give intermediate result sets of map phase. After mapping , reduce operation is performed to give the final result,i.e. count of each alphabet of the string.

4. Overview Of Workflow

A brief overview of how the Hadoop environment works can be described in following words:

1)The MapReduce library in the user program first splits the input files into M blocks of generally 16 MB to 64 MB. It then starts up many copies of the program on a cluster of machines.

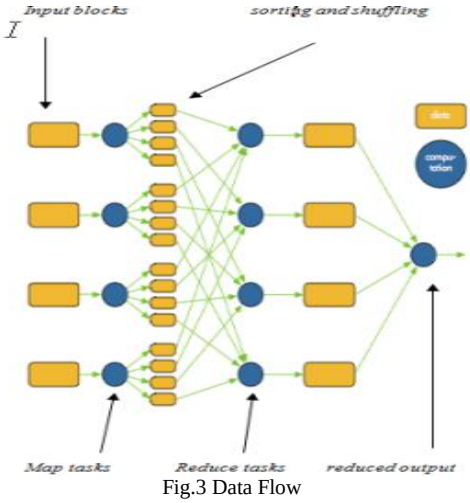

Fig.3 Data Flow

2) One of the copies of the program is the master. Others are workers that are assigned work by the master. The master picks idle workers and assigns each one a map or reduce task.

3) A worker who is assigned a map task reads the contents of the assigned input block, parses key/value pairs out of it and finally passes each pair to the user-defined Map function.

4) The intermediate key/value pairs produced by the Map function are buffered in the memory. The locations of these buffered results on the local disk are passed back to the master, who forwards these locations to the reduce workers.

5)When a reduce worker is notified by the master about these locations, it uses RPC (remote procedure calls) to read the buffered data from the local disks of the map workers. After reading all intermediate data, the reduce worker sorts it by the intermediate keys so that all occurrences of the same key are grouped together .

6)The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the Reduce function. The output of the Reduce function is appended to a final output file for this reduce partition.

5. Fault Tolerance In HADOOP

The very primary objective that makes HDFS superior to its predecessors is its capacity to store the data reliably even in the prescence of failures. The failures can be NameNode failures, DataNode failures or failures due to network partitions.

(i) A network partition may cause a DataNode to get disconnected from the NameNode. This connectivity failure can be detected by the absence of heartbeats from the DataNode that sends heartbeat regularly to the Namenode under normal circumstances. The node is considered dead and refrained from any further input output requests to it. It may result in fall of replication factor of some blocks which is thus, monitored by NameNode.

(ii) Sometimes fault in storage device or network may cause the data fetched from a DataNode to get corrupted. This corruption is controlled by the HDFS client software using checksums on the contents of the HDFS files. At the time of creation of HDFS files, the checksum on each block is computed by the client. This checksum is stored in a separate hidden file.Whenever the client receives data from datanode, it matches the checksum in the corresponding checksum file. If the data is found to be corrupted,the data can be fetched from another replica of that block.

(iii) The central data structures of HDFS are FsImage and EditLog. Any corruption to these files may cause failure to any HDFS instance. So, multiple copies of these data structures is necessary and hence their synchronization in case of any update.

6. Comparisonwith Traditional Systems

A brief comparison between Hadoop and traditional distributed databases can be summarized as follows:

(1) Computing Mode

Hadoop works on the notion of jobs where job is the basic unit of work whereas transaction is the basic unit of work in traditional databases. However, no concurrency control is provided by Hadoop in contrast to traditional distributed database systems that work on ACID properties**.**

(2) Data Model

Traditional database is based on structured data with known schema whereas Hadoop accommodates any kind of (un)structural data.

(3) Cost Model

Traditional distributed database system uses expensive servers while Hadoop uses low cost commodity machines.

(4) Fault Tolerance

Traditional distributed database system suffers from rare failures while having recovery mechanisms. On the other hand, failures are common over thousands of machines on hadoop framework. But fault tolerance is quite advanced.

7. Conclusion

This paper illustrates the basics of the hadoop architecture. The two major layers of the hadoop data operating system-HDFS and MapReduce have been explained. The working of Map Reduce algorithm in Hadoop Distributed File System has been exemplified with the help of an example for finding the occurrences of a character when a string is entered . Hadoop is a constantly growing, complex ecosystem of software and provides no guidance to the best platform for it to run on. The Hadoop community leaves the platform decisions to end users. Because of its architecture to keep up with a single file system name space, if large numbers of small objects and files are being created in the HDFS, the single NameNode would get overwhelmed with large numbers of small I/O requests and the network would be slowed down working to keep up with the large numbers of small packets being sent across the network and processed. Thus, Hadoop has come out as the ultimate platform for huge data sets but with large sized files and objects.

References

[1] Mike Olson, Amr Awadallah, Jeff Hammerbacher and Doug Cutting, Ask Bigger Questions whitepaper, Round Table Conferenc, Cloudera.

[2] Dhruba Borthakur, "The Hadoop Distributed File System: Architecture And Design"Ping Zhou,Jingsheng Lei and Wenjun Ye," Large-Scale Data Sets Clustering Based on MapReduce and Hadoop", Journal of Computational Information Systems 7: 16 (2011) 5956-5963.

[3] Jeffrey Dean and Sanjay Ghemawat," Mapreduce: simplified data processing on large clusters".Commun. ACM, 51(1):107–113, 2008.

[4] Madhavi Vaidya, "Parallel Processing of cluster by Map Reduce". International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.1, January 2012.

[5] Mohammad Y. Eltabakh, " Hadoop:A framework for data intensive distributed computing", WPI, CS561-Springer,2012

[6] Erik Riedel, Christos Faloutsos, Grath A. Gibson, and David Nagle. "Active disks for large scale data processing", IEEE Computer,pages 68-74, June 2001

[7] Ralf Lammel, "Google's MapReduce programming model-revisited", Microsoft corp. WA, USA

[8] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, Yuan Yuan YU, Gary Bradski,Andrew Ng and Kunle Olukotun."Map-Reduce for Machine Learning on multicore". NIPS,2006

[9] http://hadoop.apache.org/mapreduce