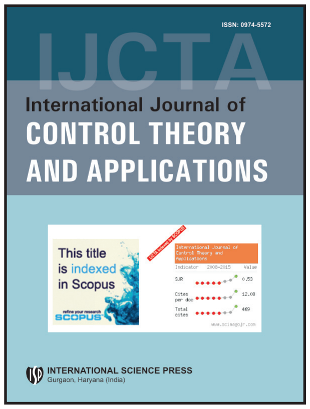


International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 9 • Number 44 • 2016

Simplified Data Analysis of Big Data in Map Reduce

Kartik Subramanian[a], Akshatha Prabhu[b] and Deepthy Ashok M[c]

[a-c]Department of Computer Science, Amrita School of Arts and Sciences, Amrita Vishwa Vidyapeetham, Amrita University, Mysuru Campus, Karnataka, India. Email: [a]iyerkartik123@gmail.com; [b]akshathaprabhu06@gmail.com; [c]deepthyashokm@gmail.com

Abstract: MapReduce is acclaimed worldwide for the enormous data insight handling in distributed computing. The workload consists of sequence of jobs, each following set of map tasks followed continuously with reduced jobs. The execution of the jobs is such that map tasks are executed before reduced tasks since map slots are occupied by map slots and reduce tasks are executed in reduce slots only. Single node is setup to execute the Map Reduce Word Count. Effects on the execution time on setting the number of reduced tasks and size in input files are studied. This paper aims at comparing the execution time of Word Count under varying conditions.

Keywords: Map reduce, Hadoop, word count, reduced tasks, key pair.

1. INTRODUCTION

Big Data describes a large volume of data. The data is classified as structured, unstructured and semi-structured which can be mined for meaningful information. To extract the value of the data, Big Data is applied on data analytic methods. The basic characteristics of Big Data are 3v that is velocity, volume and variety. In some cases, there are 4v that includes veracity. The key enablers of big data are increase in storage capacity, efficient processing powers and availability of data. The quickness in data arrival, storage and rate of retrieval is referred under velocity. The rate of growth of big data has elevated from gigabytes in 2005 to Exabyte in 2015. Maintaing a huge amount of data in old database is difficult and requires new mechanisms. The variety of data include traditional, raw, structured and semistructred type. The sources include emails, log files, sensors, social media forums, search indices and so on. The data in the form of text, audio, video and images are available in the form of text and numbers arriving mainly from the banking and insurance services insurance services followed by manufacturing sector and social network sites. It acts as the most popular, open-source software framework that is used by major portals like Yahoo and Facebook [3][8].

The life cycle of big data encompasses acquisition of log data from the application end user and aggregated data is obtained from the scattered data upon which analysis is made on the integrated data providing knowledge as the output to the end user. The data can be visualized as a general data access method or as a part of data analytics based on data integration. Big data is majorly used in log analytics, which cannot store enough logs and analyse them in a cost effective manner.

Data summarization and querying requires Hive that is used for batch processing through tables only. Apache pig utilises user defined functions that can be implemented using Java, and Python to perform transformations [1]. For real time random read and write, HBase is an open source, distributed, fault tolerant, scalable, multi-dimensional, fault tolerant database which is built on Google File System (GFS). All the data of HBase is organized by means of tables where random time stamp and unique number is auto generated for each value by HBase. SQOOP component is used to interact with the RDBMS in cases of importing and exporting data to RDBMS. Java based application that utilizes Graphical User Interface is the OOZIE that uses Hadoop stack where completion of present task depends on the completion of foregoing task. Online real data is captured through FLUME. A NO SQL database MONGODB stores the data in [key, value] pair architecture. Co-coordinating and managing a service in a distributed environment is a complicated process which is handled by zookeeper. The basic components of Hadoop are HDFS (it is used to store the large collection of data in chunks) and Map-Reduce (it is used to process the data that is stored in HDFS in a parallel manner). These Huge data can be accumulated through varied sources through HDFS that handles continuous updates [2-3]. It is a distributed file system that stores huge collection of files across multiple machines in huge cluster. Each file is stored as chain of blocks [4-5].

2. HADOOP FILE SYSTEM

Hadoop Architecture

Files and directories are stored in interconnected clusters that reside in HDFS. The architecture includes namenode-managing the file systems and data node stores data as blocks within the file.

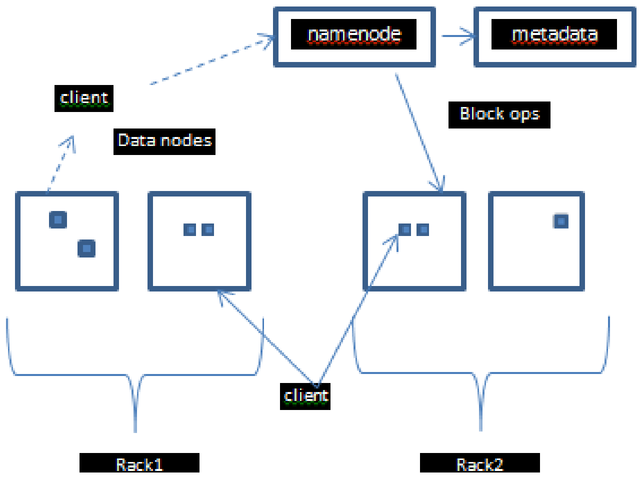


Figure 1: HDFS Architecture

Figure 1 depicts the hadoop file System, where namenode acts as a master server that performs the task of managing and performing operations on file system along with co-ordinating client access to files. Block operations on files and read/write operations on files are determined by the datanode. The file is divided into individual segments and stored in data nodes. Each file segment is referred as block. HDFS manages voluminous data sets and identifies quick and automatic fault detection.

2.1. Map Reduce

Large scale data intensive computations employs Map Reduce that has become a widely employed programming model. Dynamic routing of tasks that provides an elementary mechanism for fault tolerance and load. As long as the application obeys certain restrictions imposed by the programming model, the MapReduce engine ensures fault-tolerant and scalable execution. A MapReduce program may thus be developed and tested on a single node and be deployed with a high degree of confidence on a much larger scale. This makes MapReduce a particularly good match for data-intensive applications executing in the cloud.

In MapReduce programming model, data sets are modelled as collections of key/value Pairs. A MapReduce program processes one such data set and produces another. Two functions are needed to be specified by the programmer and they are the map and the reduce function. The map function is used for each input key/value pair producing a bunch of intermediate key/value pairs. The MapReduce engine groups the set of intermediate pairs by key, and the reduce function is invoked once for each unique intermediate key [7]. The reduce function may access all of the values held together with the given intermediate key using an iterator, and emits the key/value pairs that constitute the final output. Reduce merges various information arriving from the Map, computing result sets and achieving the reduce answer [3-4].

Figure 1 shows the process flow of map reduce. The input format fetches a file from HDFS and then call split method that splits each file. Each file is then read by record reader generating key/value pairs. Association of result of map phase to the input of the reduce phase is equally distributed throughout the Map Reduce library. The map phase runs a user defined mapper function on a set of key-value pairs [kj, vj] taken as input, and generates a set of intermediate key-value pairs. Each major (reduce) step reduces the number of data objects (key-value pairs) by an order of magnitude or more [1][2][5]. The key value pair from reduce is given to output file that maintains writer record. Each reducer provides a separate file in the output directory. Shuffling phase greatly affects the execution of data that is moved from map tasks to reduce tasks and is the reason for separating split into (key, value) phase to be accepted by mapper phase [4][11].

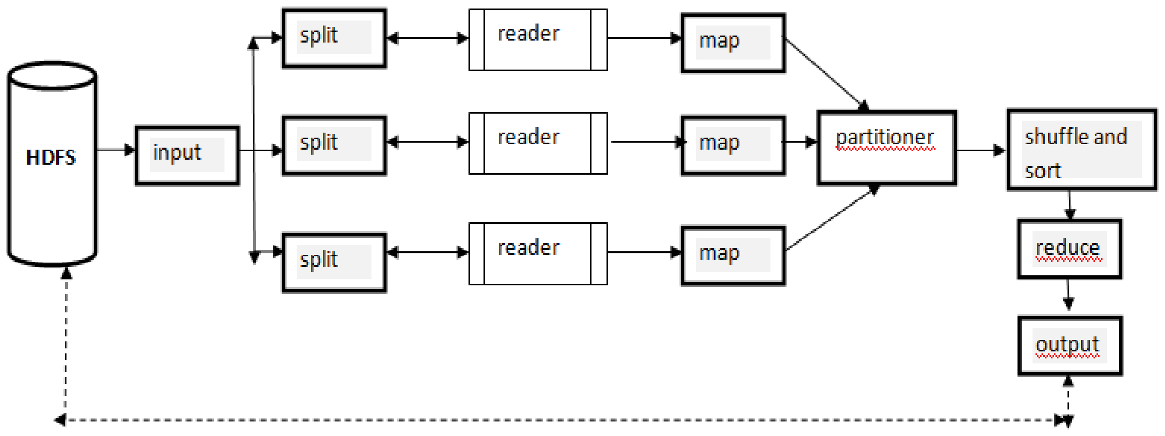


Figure 2: Process flow of map reduce

2.2. Name Node

Namenode performs all file handling operations such as opening, closing and replacing. It stores the blocks in a file and makes all the decisions regarding the blocks. It is directory trees of all files and the actual data is not stored [7].

2.3. Secondary Namenode

Snapshots of namenode's memory structure is stored in secondary namenode. It performs periodic file system read, changes the log files and apply them into the fsimage, thus updating the file system. Hence allows to start up faster next time [6].

2.4. Data Node

All the data of the application is stored in data node and waits for the request from Name node. Name node provides he location of the data and the data node contacts the client directly [7][9].

2.5. Job-Tracker

It schedules and monitors the map and reduce operations to clearly defined machines and supervises the success and failure of the task. In Hadoop Map-Reduce service job tracker is its reason for failure. If the Job tracker crashes, then all the jobs that are running will come to an abrupt stop [8].

2.6. Task Tracker

It takes the map, reduce and shuffle tasks from Job Tracker, it calculates the number of tasks it can accept by setting up the slots. The JVM processes the tasks in this tracker, regularly sending the heartbeat messages to Job Tracker informing it that it is still functioning.

3. MAP REDUCE WORD COUNT

Word Count is the basic Hadoop program that accepts a text file and counts the occurrence of each word. The result is a text file, each line contains a word and the frequency of occurrence of each word separated by tab. Three classes are used - the main class that triggers two methods mapper () and reducer (). In the mapper function the text is tokenized into words that form key value pair where key is the word itself with a value 1. The reducer class, sum of values of similar keys are calculated for the group of similar keys. The data types given are Hadoop specific which is convenient for massive parallel and fast read write operations.

The map and reduce function is defined as follows:

Map (Text T, Content C)

```
1.  Calculate number of lines n
    T[]=n
2.  LCOUNT[] = $\sum_{i=0}^{n} lci$
3.  for each $lci$ in T
4.  Calculate number of words wc
5.  WCOUNT[] = $\sum_{i=0}^{N} wci$
    for end
6.  for each word $wc_i$ in Line L
7.  output ($wc_i$, 1)
```

Reduce (String wc, Partial Counts PC)

PC pc is a list of aggregated partial counts

sw is the count of similar words

```
1.  sum = 0
2.  sum $wc_i = \sum_{i=0}^{sw} \text{sum} + pci$
    for end
3.  output ($wc_i$, sum)
```

4. EXECUTION OF MAP REDUCE

4.1. Steps to run Word Count

1. Start Hadoop Daemon

   start-all.sh

2. Open a text file

   gedit input

3. Fill the contents in text file input
4. Place the file input to hadoop directory

   Hadoop fs–put input /

5. Add the jar file and store in output file

   Hadoop jar/home/usr/wordcount.jar WordCount /input /output

6. Output is displayed in part-r-00000

   Hadoop fs–cat/output/part-r-00000

4.2. Changing the Number of Reduce Tasks

Variations in the number of reduce tasks alters the framework overhead, increasing the load balancing and reducing the chances of failure. [i]. Reduce tasks can be set using one of the following

1. Using configuration File "mapred-site.xml"

   mapped.reduce.tasks

2. By specifying in the driver class

   job.setNumReduceTasks(4)

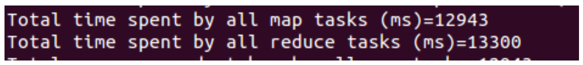
3. By specifying at the command line using Tool interface:

   Dmapreduce.job.reduces=2The input files

The input files used were the same as used in previous case. Here, comparison was made between programs when the number of reduce tasks were altered from 10 to 100 using the following function in the driver class job.setNumReduceTasks(10)

Reduced Task set to 10.

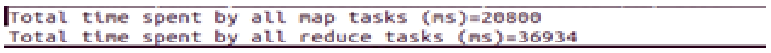
The result of the total time spent by reduced tasks when set to 10 is shown below:



```
Total time spent by all map tasks (ms)=12943
Total time spent by all reduce tasks (ms)=13300
```

Reduced Task set to 100.

The result of the total time spent by reduced tasks when set to 100 is shown below:



```
Total time spent by all map tasks (ms)=20800
Total time spent by all reduce tasks (ms)=36934
```

4.3 Changing the file Size

Input files of different data sizes have been considered and their execution time has been compared. Three files of 200MB, 600MB and 1GB were considered.

The table is as shown:

Table 1
Represents three files of 200MB, 600MB and 1GB

FILE SIZE	*TIME in seconds*
200MB	250
600MB	410
1GB	555

Acknowledgment

We would like to express our heart-felt gratitude to the institution and the people who have constantly guided and supported us.

5. CONCLUSION

The time taken by the word count increases as the file size increases. Variations in the number of reduced task also modifies the execution time of the program. As the number of reduced tasks increases so does the execution time.

The setup used is the single node cluster and the inputs given are static. Dynamic approach can be used to handle multiple task by the node and allocating the resources by calculating the number of free slots through various approaches like enhanced dynamic slot allocation, speculative execution and slot pre-allocation which aims at increasing the performance and resource utilization. Thereby reducing the complexities that arise due to starvation when tasks are aligned in queue.

REFERENCES

[1] Plantenga, T., Choe, Y., and Yoshimura, A. (2012). Using Performance Measurements to Improve Map Reduce Algorithms. Procedia Computer Science 9, pp 1920 – 1929.

[2] Du, D., Li, A., Zhang, L., and Li, H. (2015). Review on the Applications and the Handling Techniques of Big Data in Chinese Realty Enterprises. Ann. Data. Sci., DOI 10.1007/s40745-014- 0025-5.

[3] Lee, C-W., Hsieh, K-Y., Hsieha, S –Y., and Hsiao, H-C. (2014). A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments. Big Data Research 1, 14–22.

[4] http://wiki.apache.org/hadoop/ProjectDescription.

[5] The Hadoop Distributed Filesystem: Balancing Portability and Performance Jeffrey Shafer, Scott Rixner, and Alan L. Cox Rice University Houston, TX Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium, 28-30. March 2010

[6] http://en.wikipedia.org/wiki/Apache_Hadoop

[7] http://wiki.apache.org/hadoop/DataNode

[8] http://wiki.apache.org/hadoop/MapReduce

[9] Hadoop Wiki, https://wiki.apache.org/hadoop/HowManyMapsAndReduces, Retrieved: 16-Oct-2015

[10] https://www.bamu.net/library/theses%20database.html

[11] A. G. Hari Narayanan, Krishnakumar, U., and Judy, M. V., "An enhanced MapReduce framework for solving protein folding problem using a parallel genetic algorithm", in ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I, 2014, Vol. 248 VOLUME I, pp. 241-250.