ISSN (e): 2250 – 3005 || Vol, 04 || Issue, 5 || May – 2014 ||
International Journal of Computational Engineering Research (IJCER)

Mapreduce Performance Evaluation through Benchmarking and Stress Testing On Multi-Node Hadoop Cluster

[1,]Urvashi Chaudhary , [2,]Harshit Singh
[1,]*Indian Institute of Technology*
[2,]*ABV-Indian Institute of Information Technology and Management*

ABSTRACT
Over the past few years, many data based applications have been developed using the powerful tools provided by the World Wide Web. In today's world an organization may generate petabytes of data each second. To gain insights from this vast pool of data, applications based on data-mining and web indexing such as artificial neural networks, decision trees etc., play an important role. Such tools are tremendously helping social networking sites such as Facebook and Twitter to get meaningful information from their data. Hadoop is an open source distributed batch processing infrastructure. It works by distributing the amount of processing required by a task across a number of machines, thereby balancing the load depending on the node. Hadoop, in a very short time has gained huge popularity amongst many IT firms. Infact, many big players such as IBM, Google, Quora and Yahoo are using numerous applications based on Hadoop infrastructure.

KEYWORDS: *TestDFSIO performance benchmark,MapReduce, Apache Hadoop*

I. INTRODUCTION

MapReduce[7] has proven to be an efficient programming model for data processing. MapReduce programs coded in several different languages can be run on Hadoop. Since MapReduce programs run parallel, it is possible to perform analysis on large data sets simultaneously by deploying a number of machines. Although, the hadoop model appears complex initially but familiarity with its tools makes it easy to use. In the MapReduce [2] architecture there exist one master JobTracker and one slave TaskTracker in each clusternode. The master schedules and monitors the tasks of each job on the slaves. In case of a task failure the master re-executes the task. The slave simply performs the tasks assigned to it by the master.

II. RELATED WORKS

GridMix, [1] the name assigned to denote the past works on Hadoop performance benchmarking has been upgraded many times over the years. GridMix aims to symbolize real application assignments on Hadoop clusters and has been used to validate and measure optimizations across dissimilar Hadoop releases. Hadoop is a successful execution of the MapReduce model. The Hadoop framework consists of two main elements: MapReduce and Hadoop Distributed File System (HDFS).

MapReduce

It is a type of programming model and is usedto process and generate huge amounts of data. Their main functions like map and reduce are supplied by the user and depend on user's intention [4].Map: Map operates by processing a series of key/value pairs and generates zero or more key/value pairs. It is possible that the Map's input and output type are different from each other [2].Reduce: For each unique key in the sorted order, the application's Reduce function is called. Reduce generates zero or more output by iterating through thevalues that are linked with that key [2].

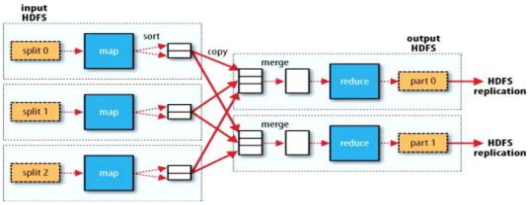

Figure 1.MapReduce Programming Model[6]

2.2 NameNode

The hierarchy of files and directories make the HDFS namespace. On the Name Node these are represented by inodes.The contents of the files are split into larger blocks (128MB standard but user selectable). Then each block of the file is replicated at multiple data nodes [3].

2.3 DataNode

It handles the task of storing the data on theHadoop File System and responding to the requests of the file system operations. On starting the cluster the data node is connected to the name node. On providing the location of the data to the client, the client directly communicates to the data node similar to the working of the MapReduceTaskTracker [3].

2.4 Secondary NameNode

It stores the log of the corrections made to the file system additional to a native file system. The HDFS reads the state from an image file called *fsimage*on
starting the name node. It then applies changes from the log file [3].

2.5 TaskTracker

It receives the tasks - Map, Reduce and Shuffle operations from a Job Tracker. It can accept a limited number of tasks configured by the set of slots. The Job-Tracker during scheduling a task looks for an empty slot on the same server that hosts the Data Node containing the data. If it is not found, as empty slot on the machine in the same rack is chosen [9].

2.6 JobTracker

It is used to track the particular nodes in the cluster i.e. the tasks of MapReduce. These nodes are mostly the nodes with the data in them or are in the same rack [8].

[1] JobTracker receives jobs from client applications.
[2] Name Node is communicated by JobTracker to find the location of the data.
[3] It then finds the TaskTracker nodes with empty slots at or cleans the data.
[4] It finally sends the work to the chosen TaskTracker node.

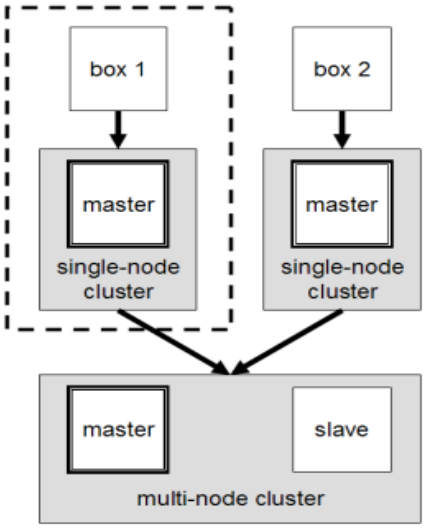

Figure 2. Multi Cluster Node

III. DESIGN AND IMPLEMENTATION

In this paper, we are deploying multi-node cluster Hadoop on two machines in which master machine with core i5 2.3 GHz processor, 2GB of RAM and slave machine with Pentium 4 3.4GHz, 1GB of RAM .Both machine has LAN network of 100 Mbps and setting up of cluster we use Hadoop 1.0.4 and UBUNTU 10.04
Before deploying Hadoop we have prerequisite as follows
Sun Java: It is requires working Java1.6.x or aboveinstallation.

Table 1. Writing Speed of Several files

Number of files	**Total Megabytes processed**	**Throughput MB/sec**	**Average I/O Rate**	I/O rate standard deviation	Test execution time (second)
1	1000	62.5273557181	62.52735519	0.0042950560	47.91
2	2000	37.1848583256	39.71986389	1.3875991805	65.98
3	3000	34.7567375935	37.71351623	12.865245652	65.98
5	5000	32.7102039481	33.51700210	4.9086852221	120.09
9	9000	31.2761720745	32.76895523	10.966624853	211.20
10	10000	19.5372027414	20.89837455	11.742043454	329.33

Table 2. Reading Speed of Several files

Number of files	**Total Megabytes processed**	**Throughput MB/sec**	**Average I/O Rate**	I/O rate standard deviation	Test execution time (second)
1	1000	62.5273557181	62.52735519	0.0042950560	47.91
2	2000	37.1848583256	39.71986389	1.3875991805	65.98
3	3000	34.7567375935	37.71351623	12.865245652	65.98
5	5000	32.7102039481	33.51700210	4.9086852221	120.09
9	9000	31.2761720745	32.76895523	10.966624853	211.20
10	10000	19.5372027414	20.89837455	11.742043454	329.33

ssh: It requires to manage its node i.e. remote machine .

Hadoop Implementation

We have implemented Hadoop on Ubuntu 10.04.

As the hadoop is written in java, we needed Sun jdk to run hadoop. This was not available on the official server of sun jdk. Now we have to install jdk to run hadoop.

- Installing the jdk 1.6 version for Ubuntu 10.04.
- apt-get install sun-java6-jdk
- IPv6 should be disabled.

For performance evaluation benchmarking and stress testing on Hadoop we use TestDFSIO tool.

IV. EXPERIMENT AND PERFORMANCE

4.1 Interpreting TestDFSIO Results

In this paper, TestDFSIO has been used to benchmark the read and write test for HDFS. This test proved to be extremely useful in determining performance blockage in the network and for conducting stress test on HDFS.This setup proved to be extremely economical in terms of hardware, OS and setup costs (specifically the Name Node and the Data Nodes) and to give us a first influence of how fast our cluster is in terms of I/O.

4.2 Data Throughput

From the table 1, 2 one can infer that throughput MB/sec and average I/O rate MB/sec emerge as the most important parameters. Both the parameters depend on the file size reading and writing speed obtained by the individual map tasks and the elapsed time to do so. Throughput MB/sec for a TestDFSIO job using N map tasks can be determined in the following manner. The index 1 <= i<= N denotes the individual map tasks:

$$\text{Throughput}(N) = \sum_{i=0}^{N} \text{filesize} / \sum_{i=0}^{N} \text{time} \quad (1)$$

Average I/O rate MB/sec is defined as

$$Average\ Rate\ I/O\ Rate(N) = \sum_{i=0}^{N} rate\ /N \quad (2)$$

Here the metrics “concurrent” throughput and average I/O rate in cluster’s capacity prove to be significant. Let us assume that TestDFSIO create 10 files and our cluster size is limited to 2 map slots. Hence, the number of MapReduce waves to write the full test data will be 10 (5 * 2) since the cluster will be able to run only 200 map tasks at an instant.

In this example, we multiplied the throughput and average I/O rate by the minimum of the number of files and the number of available map slots in the cluster. Here:
Concurrent throughput = 31.3 * 2 = 62.6 MB/s
Concurrent average I/O rate at 31.49 * 2 = 62.98 MB/s
As given in the figure 3 and figure 4 we can see that the throughput during reading is more than the throughput during writing. As the number of files is increased, the throughput of both reading and writing starts decreasing but in the reading case, throughput decreases more rapidly as compared to writing.

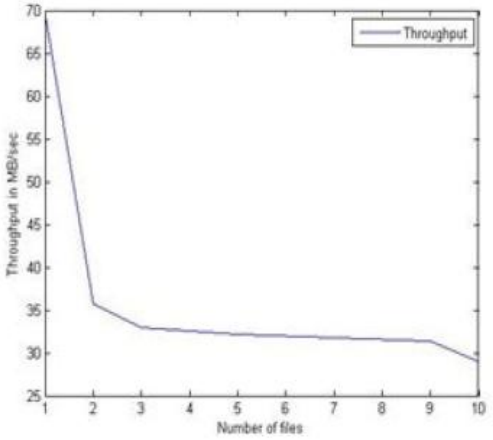

Figure 3 .Write throughput performance

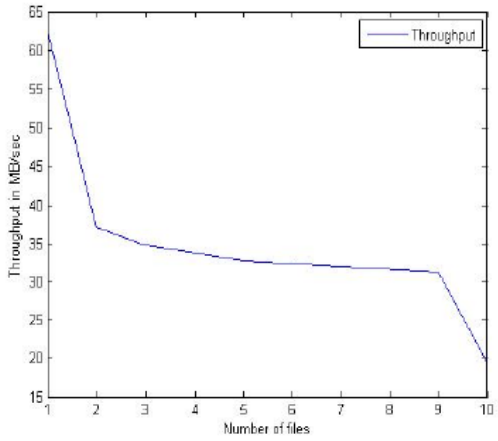

Figure 4. Read throughput performance

V. CONCLUSIONS

The HDFS reading performance of files is much faster than writing. It also shows that when number of files increases performance decreases. The evaluation of reading and writing Throughput runs on multi cluster in a particular machine. Time execution of files writing is more than reading as number of files increases it decreases. In the above figure we can easily see the throughput performance of file.Writing is more costly than reading according to the
scenario.

REFERENCES

[1] H. T. C. Douglas, "GridMix Emulating Production Workload for Apache Hadoop.",http://developer.yahoo.com/blogs/hadoop/posts/2010/04/gridmix3_emulating_production , 2010.
[2] T. White, HadoopThe Definitive Guide 3rdEdition, O'Reilly, 2012.
[3] M. Doug Cuttling, "Hadoop Home Page HadoopArchitecture.",http://hadoop.apache.org/docs/r1.0.4/hdfs_design.h tml,2012.
[4] Yahoo Inc., http://hadoop.apache.org/docs/stable/hdfs_user_guide.html,2008.
[5] M. Noll, "Michael Noll.",http://www.michael-noll.com.
[6] Zhang, "MapReduce Sharing.", 2011, pp. 8-15.
[7] J. Dean and S. Ghemawat, "MapReduce:Simplified data processing on large clusters." in*OSDI*, 2008.
[8] H. Apache, "JobTracker.", Apache,http://wiki.apache.org/hadoop/JobTracker, 2008.
[9] H. Apache, "TaskTracker.", Apache,http://wiki.apache.org/hadoop/JobTracker, 2008.