Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 705 – 716

doi:10.3906/elk-1501-43

Hadoop framework implementation and performance analysis on a cloud

Göksu Zekiye ÖZEN[1,*], Mehmet TEKEREK[2], Rayimbek SULTANOV[1]
[1]Department of Computer Engineering, Kyrgyz Turkish Manas University, Bishkek, Kyrgyzstan
[2]Department of Computer Education and Instructional Technology, Kahramanmaraş Sütçü İmam University, Kahramanmaraş, Turkey

Received: 06.01.2015 • **Accepted/Published Online:** 19.02.2016 • **Final Version:** 10.04.2017

Abstract: The Hadoop framework uses the MapReduce programming paradigm to process big data by distributing data across a cluster and aggregating. MapReduce is one of the methods used to process big data hosted on large clusters. In this method, jobs are processed by dividing into small pieces and distributing over nodes. Parameters such as distributing method over nodes, the number of jobs held in a parallel fashion, and the number of nodes in the cluster affect the execution time of jobs. The aim of this paper is to determine how the numbers of nodes, maps, and reduces affect the performance of the Hadoop framework in a cloud environment. For this purpose, tests were carried out on a Hadoop cluster with 10 nodes hosted in a cloud environment by running PiEstimator, Grep, Teragen, and Terasort benchmarking tools on it. These benchmarking tools available under the Hadoop framework are classified as CPU-intensive and CPU-light applications as a result of tests. In CPU-light applications, increasing the numbers of nodes, maps, and reduces does not improve the efficiency of these applications; they even cause an increase in time spent on jobs by using system resources unnecessarily. Therefore, in CPU-light applications, selecting the numbers of nodes, maps, and reduces as minimum is found as the optimization of time spent on a process. In CPU-intensive applications, according to the phase that small job pieces is processed, it is found that selecting the number of maps or reduces equal to total number of CPUs on a cluster is the optimization of time spent on a process.

Key words: Big data, Hadoop, cloud computing, MapReduce, KVM virtualization environment, benchmarking tools, Ganglia

1. Introduction

Unstructured and structured data obtained from different types of media (websites, blogs, smart phones, embedded systems, etc.) can be referred to as big data. Big data are defined as any kind of data source that has at least three characteristics: volume, velocity, and variety. Variety describes different formats of data and different data sources. The data can be integrated and transformed into each other. Velocity is related to the retrieval and storing speed of data. The volume of data grows exponentially. As the data size grows, the applications and architectures need to be reevaluated quickly [1].

One of the most widely used methods for processing, analyzing, storing, and retrieving big data is the MapReduce method [2]. The MapReduce method, developed by Google in 2004, has become an accepted standard in the field of big data analysis [3].

The Hadoop framework is one of the common frameworks that processes distributed big data by applying the MapReduce method [4]. The Hadoop framework uses a distributed, scalable, and portable Hadoop

*Correspondence: goksu.ozen@manas.edu.kg

Distributed File System (HDFS), which is written in the Java programming language in order to perform these jobs [5]. HDFS allows big data to form blocks, distributed to nodes, processed and retrieved successfully [6].

On a Hadoop cluster, while the master node can serve as NameNode and DataNode, slave nodes can serve only as DataNode. NameNode is responsible for creating data blocks, distributing the data to other nodes, managing distribution operation, and recreating a data block in the case of failure on a data block. On the other hand, a DataNode is responsible for saving data blocks and backing up data held in other DataNodes [7].

Multiple clusters can be used for big data processing and each cluster can have multiple nodes. While increasing the number of nodes in a cluster makes it difficult to manage, virtualization solutions have been developed for installation, deployment, scheduling, and efficient resource management. Kernel based virtual machine (KVM) is one of the widely used open-source virtual machine solutions in this area [8].

Benchmarking tools for big data processing such as PiEstimator, Teragen, Terasort, and Grep are presented under the Hadoop framework [9]. Data sets can be generated by these benchmarking tools and also prepared data sets can be used. Big data works frequently used for searching, sorting, counting words, and verifying can be performed on these data sets.

There are a few studies about performance of the Hadoop framework in the literature. Tan et al. reported in their study that adjusting the configuration parameters of Hadoop framework provides the possibility of obtaining higher operating performance and they introduced an analytical model for job execution time [10]. Extensible MapReduce (XMR) is a model consisting of determining the number of maps and reduces module and creating a routing scheduler module in the shuffling stage [11]. XMR has focused on stages directly affecting the workflow of the MapReduce method. It has made changes to the Hadoop code. Starfish is an automatic adjustment application developed for big data analysis. It automatically adjusts various settings to improve the performance of the Hadoop system. It collects monitoring data about works at runtime. Starfish is a complicated system adjusting several Hadoop parameters automatically based on the collected data. The application itself also brings the system some extra load. However, the application needs to work on Hadoop for a period of time to make these adjustments [12]. Automatic Resource Inference and Allocation (ARIA) is a framework creating job profiles starting from recorded information on the system about the works that Hadoop has done before. It tries to estimate the execution time of a new job using this information [13]. Parameterized framework developed by Zhang et al. creates a platform model by examining past works of the Hadoop system. At the same time it creates a MapReduce performance model of existing work by monitoring it for a while. It tries to predict the job execution time by using these two models. In Zhang's proposal several counters are placed in Hadoop code to reduce the impact of monitoring the framework on the system operating performance. It is a complex method because it requires recompilation of Hadoop code [14]. Babu et al. revealed the impact of Hadoop settings on big datasets [15]. Fadika et al. performed tests to realize which framework shows high operating performance on which works by comparing two different MapReduce frameworks with Hadoop. In their study, tested applications are classified as CPU-intensive, memory-intensive, or data-intensive applications [16].

In the present study, tests were carried out on a Hadoop cluster with 10 nodes hosted on DigitalOcean cloud infrastructure by running PiEstimator, Grep, Teragen, and Terasort benchmarking tools on it. DigitalOcean is a cloud service provider. Cloud service providers introduce customizable infrastructures on demand. The cost is calculated according to the time of use. A distributed big data work's cost is related to the size of the cluster and completion time of the work.

The main goal of the present study was to investigate how the number of maps and reduces according to

the number of working nodes on a cluster deployed on a cloud environment affects Hadoop framework working performance depending on CPU usage.

In this context, answers to the following questions have been sought:

How does the number of maps, reduces, and nodes for distributed big data processing on Hadoop framework structured with default parameter values affect working performance?

How are the values of parameters such as number of maps and reduces determined on a cluster?

What are CPU utilization rates of PiEstimator, Grep, Terasort, or Teragen benchmarking tools?

For optimization, is it possible to estimate the number of nodes, maps, and reduces of the Hadoop system by executing PiEstimator, Grep, Terasort, or Teragen benchmarking tools?

2. Materials and methods

In this study, PiEstimator, Teragen, Grep, and Terasort benchmarking tools were tested using the same test setup on different numbers of nodes with different numbers of maps and reduces with the default settings of Hadoop.

2.1. Experimental setup

All tests in this study were implemented on DigitalOcean cloud infrastructure by creating 10 virtual machines (nodes). Each node was configured with an Intel Hex-Core 2.6 GHz single-core processor, 1 GB of memory, and 30 GB of solid-state drive (SSD). Ubuntu Linux 12.04 LTS 64 Bits Server Edition was preferred as the operating system and KVM was selected as virtualization environment. On each node, Java Development Kit (JDK) version jdk 1.6.0_45 was installed and its settings were adjusted. Then Hadoop 1.0.3 was installed and necessary adjustments were made.

One of the nodes was selected as master and the other nodes were selected as slaves. Nodes were accessed via Secure Shell (SSH). To increase the performance of the work, parameter settings can be performed in the Hadoop framework depending on the type of data and CPU usage [12]. In this study, no extra parameter adjustment was made to cause performance improvement; only basic parameter adjustments were done for Hadoop to work on multiple nodes.

2.2. Realization of tests

In this study, tests were classified as CPU-light and CPU-intensive applications. CPU-intensive applications tend to utilize CPU 100% but CPU-light applications do not [16]. Memory usage, disk usage, network traffic usage etc. performances of nodes running on a cluster can be monitored and recorded by the Ganglia monitoring tool [17].

CPU utilizations of benchmarking tools were measured and saved as average CPU utilization of nodes with Ganglia. Before deciding to use 1 GB memory on nodes, some tests were carried out to monitor memory usage of preferred benchmarking tools. 10 GB of data was sorted with the Terasort benchmarking tool on nodes with 256 MB, 512 MB, 1 GB, and 2 GB memory. Figure 1 shows the memory usage of tests with 2 GB and 1 GB memory, and Figure 2 shows the results with 512 MB and 256 MB memory. According to the test results presented, job execution time did not change on nodes with 2 GB, 1 GB, and 512 MB memory, but it changed on nodes with 256 MB memory. Job execution times are shown in Table 1. 512 Mb and 256 MB memory usage caused insufficient memory and swap area usage on nodes. 1 GB memory was selected as optimum amount when the results were taken into consideration.

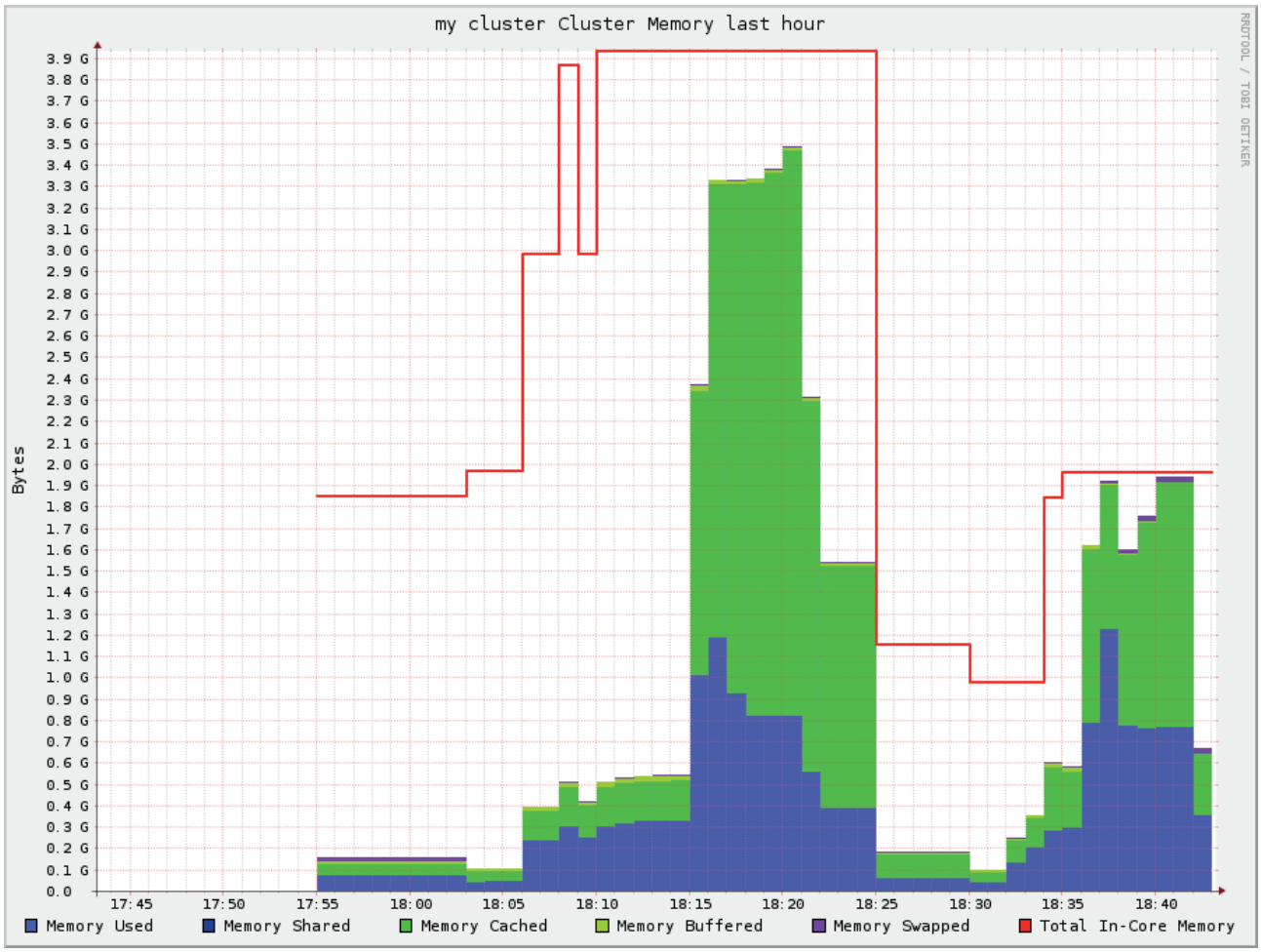

Figure 1. 10 GB Terasort test on 2 nodes with 2 GB and 1 GB memory.

Table 1. 10 GB Terasort test on 2 nodes with different amounts of memory.

Memory amount	Job execution time
2 GB	6m47.616s
1 GB	6m23.646s
512 MB	7m43.556s
256 MB	16m1.737s

In this study, equations were derived by using CPU core counts. CPU core count was taken into consideration instead of node counts. Nodes were configured with 1 CPU core. Terasort tests with 10 GB data were carried out on 2 nodes with 2 CPU cores each and 4 nodes with 1 CPU core each to decide on this configuration. Job execution times are shown in Table 2. According to the results presented there, job execution times for these different configurations are nearly the same.

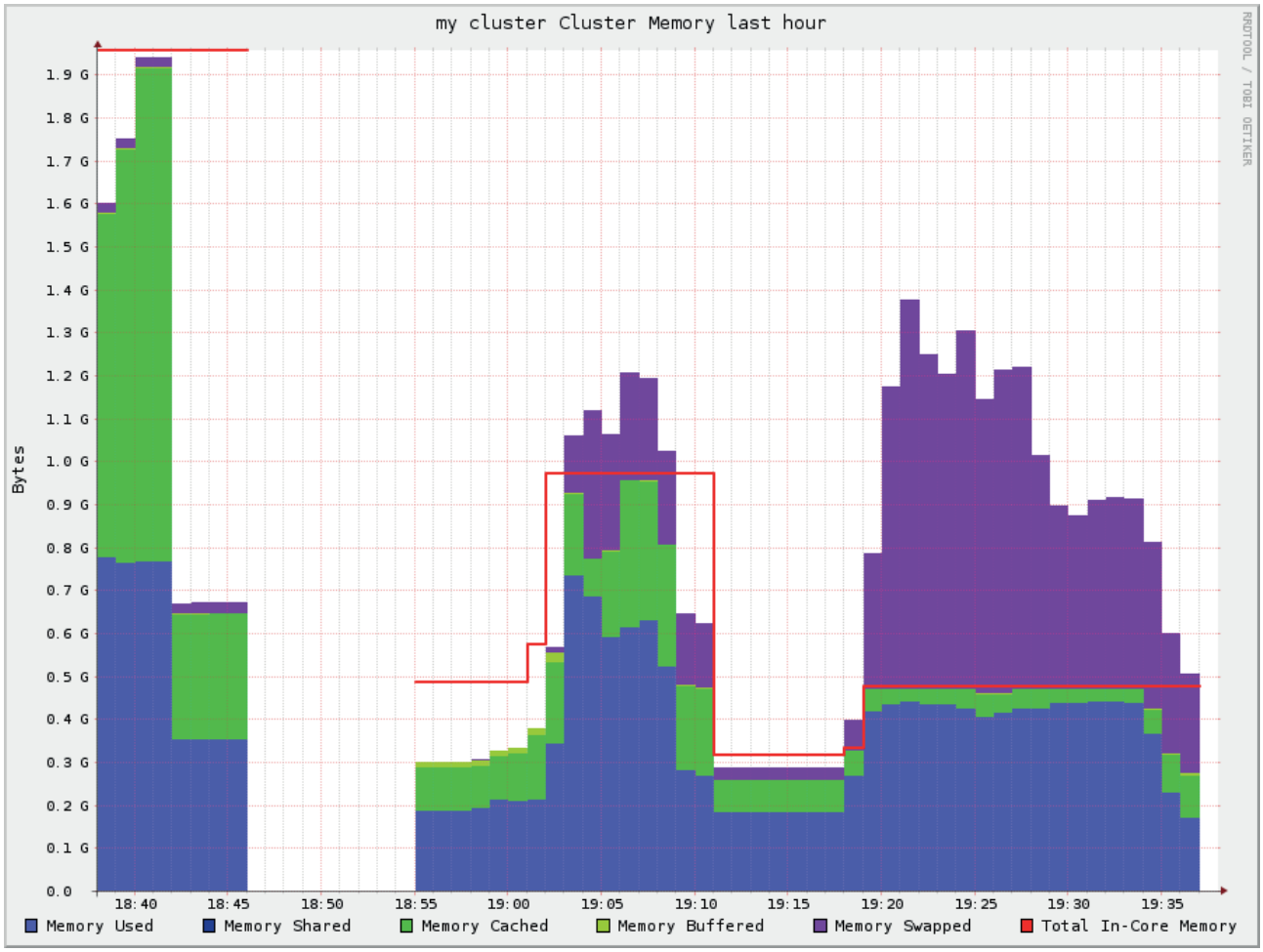

Figure 2. 10 GB Terasort test on 2 nodes with 512 MB and 256 MB memory.

Table 2. 10 GB Terasort tests job execution results on 2 nodes with 2 CPU cores each and 4 nodes with 1 CPU core each.

Number of nodes	Number of reduces	Total CPU core count/ memory amount	Job execution times
2	4	4 / 4 GB	13m38.001s
4	4	4 / 4 GB	13m.47.987s

CPU utilization percentages are shown in Figure 3. CPU utilization of the Teragen test with 20 maps on 2 nodes is shown between the time 10:00 and 10:05. CPU utilization of the Terasort test with 20 reduces on 5 nodes is shown between the time 10:06 and 10:14. CPU utilization of the Grep test with 1 map on 5 nodes is shown between the time 10:16 and 10:19. CPU utilization of the PiEstimator test with 1 map on 5 nodes is shown between the time 10:20:30 and 10:21:20. These graphical representations are selected among minimum CPU utilizations. It is seen that Terasort and Teragen tests are intended to utilize CPU nearly 100% but Grep and PiEstimator are intended to utilize nearly 15%.

Teragen and Terasort are described as CPU-intensive applications but Grep and PiEstimator are described as CPU-light applications according to the results presented in Figure 3.

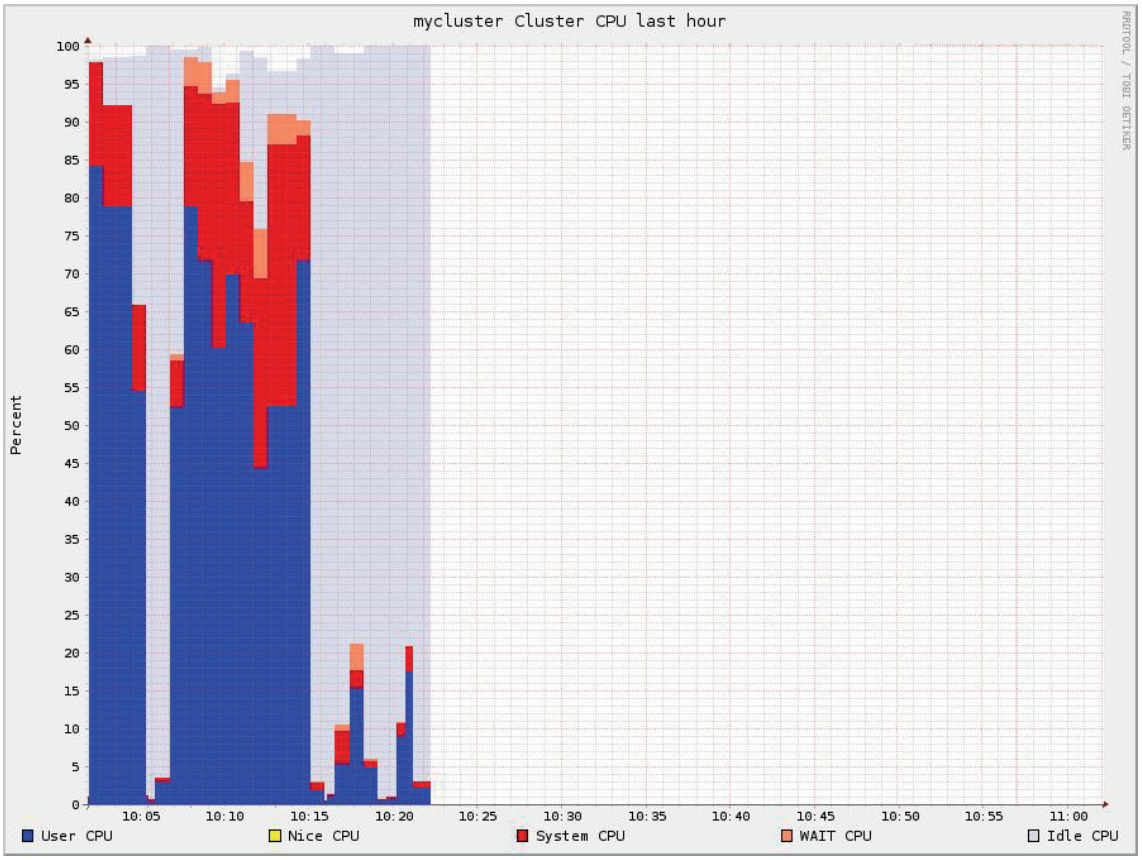

Figure 3. CPU utilization graphics (as percentage) of Teragen, Terasort, Grep, and PiEstimator benchmarking tools obtained with Ganglia monitoring tool.

Several tests for all benchmarking tools were performed on different number of nodes using different numbers of maps and reduces. All tests were carried out 10 times and arithmetic averages of the results were calculated. Benchmarking tools used in the tests and their details are presented below.

2.2.1. PiEstimator benchmarking tool

The PiEstimator benchmarking tool estimates pi using the Monte Carlo method [18]. The input number used in the test was derived from multiplication of map number and sample number. Differentiation of input number affects job execution time. At the map phase, inputs are divided into smaller parts and a specified number of samples is used for each map. In this case, number of inputs becomes unchanged but number of job pieces changes.

PiEstimator tests were performed on 1, 2, 5, and 10 nodes by using 1, 10, and 100 maps, respectively. Sample number was chosen as 10000000, 1000000, and 100000, respectively, being careful not to change the input number.

2.2.2. Grep benchmarking tool

The Grep benchmarking tool searches for the desired word or pattern in text files in an input directory and finds the count [19]. It was executed with a data set consisting of a 1 GB text file that was the combination and replication of a variety of files obtained from the Gutenberg project. Tests were performed on a 10-node

cluster using 1, 2, 5, and 10 nodes, consecutively. On each node 1, 10, and 100 maps were used. For each map 1, 2, 5, 10, 20, 50, and 100 reduces were used.

Reduce number is given as a parameter while Grep is running. Map number also can be given as a parameter but it is not used in the Grep benchmarking tool. Map number depends on size of the text file in the input directory and the block size of the HDFS [10]. Map number was adjusted by changing HDFS block size with copy parameters while copying files to the HDFS.

2.2.3. Teragen benchmarking tool

The Teragen benchmarking tool is used to generate random data to desired size. It can be specified how the generated data will be written to the HDFS [20]. Teragen does not contain a reduce phase. Map number was given as a parameter when executing. Teragen tests were performed on a 10-node cluster by using 1, 2, 5, and 10 nodes, and 1, 2, 5, 10, 20, 50, and 100 maps were used on these nodes. 10 GB of random data was generated in each test.

2.2.4. Terasort benchmarking tool

The Terasort benchmarking tool sorts the data in files located in an input directory and writes sorted data to files in an output directory [2]. Reduce number was given as a parameter when executing Terasort. 10 GB of random data generated by Teragen was written to files located in an output directory by sorting with Terasort; 1, 2, 5, 10, 20, 50, and 100 reduces were used on a 10-node cluster when executing Terasort, and 2, 5, 8, 20, 50, and 150 maps were used for each reduce value.

2.2.5. Evaluation of results

The test results are presented according to CPU usage of benchmarking tools.

2.3. CPU-light applications

PiEstimator and Grep benchmarking tools are classified as CPU-light applications according to test results provided by Ganglia. The results of PiEstimator and Grep tests given as graphics are presented below.

2.3.1. PiEstimator benchmarking tool

For PiEstimator tests, job execution times on different number of nodes with different number of maps and samples are presented in Figure 4 and CPU utilizations are presented in Figure 5.

According to Figure 4 values, it is seen that job execution time is minimum when number of nodes is 10 and number of maps is 1 (33.018 s). When system resources are increased 10 times, job execution time is decreased only 15%. It is seen from the test results that CPU utilization is a maximum of 58% on 1 node with 100 maps and a minimum of 5% on 10 nodes with 1 map.

CPU utilization increases if number of maps increases and it decreases if number of maps decreases as seen in Figure 5. Thus, the parameter said to affect the performance of a job is number of maps while number of inputs is constant. According to the test results, job execution time increases if CPU utilization increases. However, this increase is not stable.

In PiEstimator tests, using only 1 node instead of a cluster with 10 nodes provides 90% decrease in cost with 15% increase in job completion time.

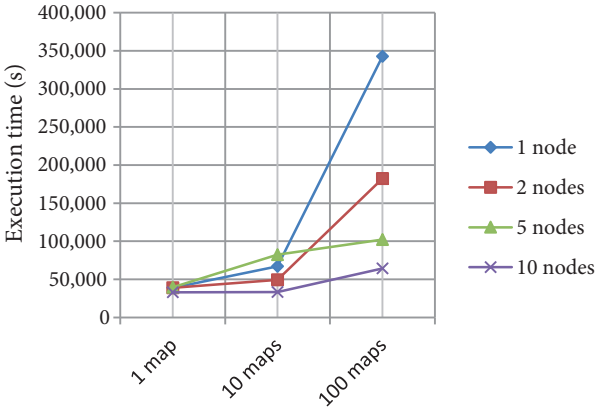

Figure 4. PiEstimator test job execution times.

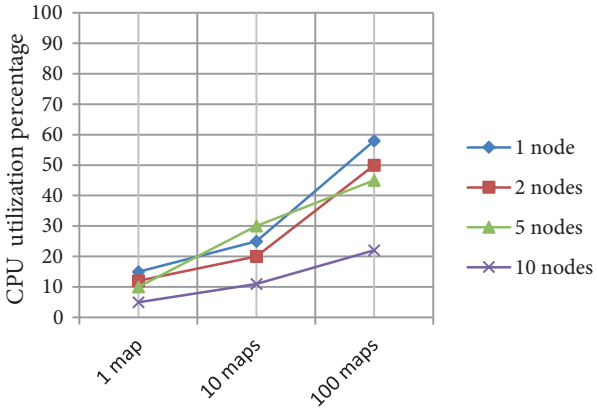

Figure 5. PiEstimator test CPU utilizations.

2.3.2. Grep benchmarking tool

Grep tests were conducted on 10 nodes with using 10 maps and different reduces (1, 2, 5, 10, 20, 50, and 100 reduces). Increasing number of reduces caused increment of job execution time and CPU utilization. Therefore, in all Grep tests, while keeping the number of reduces constant at the best value (1 reduce), tests were conducted with different numbers of maps (1, 10, and 100 maps) on different number, of nodes (1, 2, 5, and 10 nodes) as shown in Figure 6. CPU utilizations are shown in Figure 7.

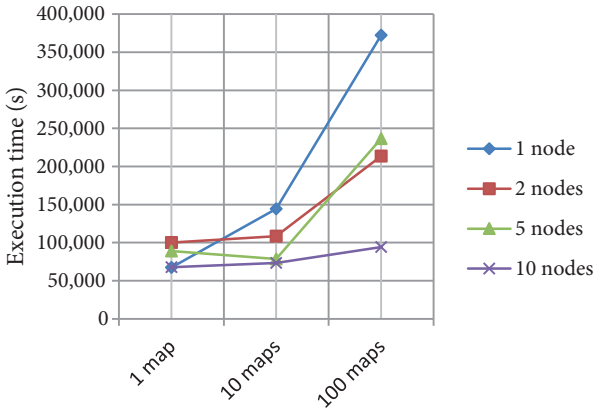

Figure 6. Grep test job execution times with constant reduce number (1 reduce).

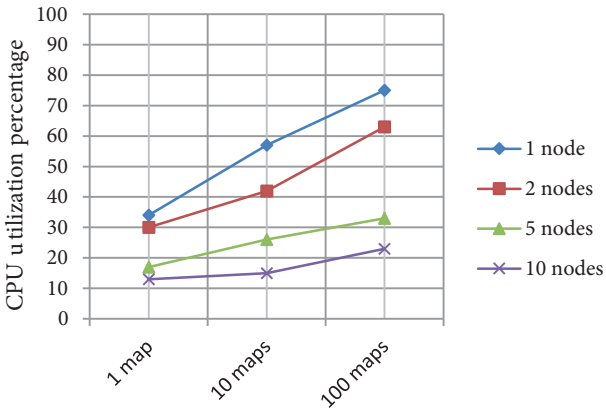

Figure 7. Grep test CPU utilizations with constant reduce number (1 reduce).

According to results presented in Figure 6, job execution time is minimum (67.523 s) with 1 map on 1 node. Change in job execution times with number of maps is shown in Figure 6. CPU utilization is minimum (13%) with 1 map on 10 nodes and maximum (75%) with 100 maps on 1 node. As seen in Figure 7, increasing number of maps causes an increment in CPU utilization and decreasing number of maps causes a decrease in CPU utilization.

According to the test results, job execution time increases if CPU utilization increases. However, this increase is not stable. Increasing number of maps does not decrease job execution time. It can be said that the parameter affecting performance of a job execution on the same number of nodes is number of maps.

In Grep tests, using only 1 node instead of a cluster with 10 nodes provides 90% decrease in cost with also 1% decrease in job completion time.

2.4. CPU-intensive applications

Teragen and Terasort benchmarking tools are classified as CPU-intensive applications according to test results provided by Ganglia. The results of Teragen and Terasort tests given as graphics are presented below.

2.4.1. Teragen benchmarking tool

For Teragen tests, CPU utilizations and job execution times on different number of nodes with different numbers of maps are presented in Figures 8 and 9.

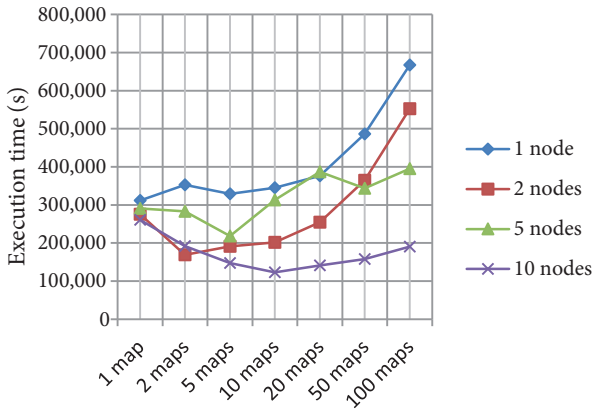

Figure 8. Teragen test job execution times.

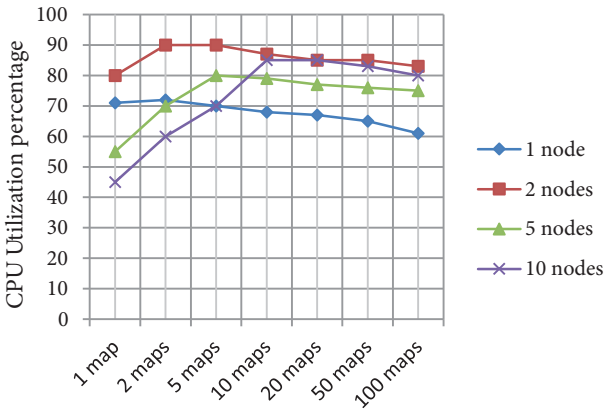

Figure 9. Teragen test CPU utilizations.

According to the results presented in Figure 8, job execution time is minimum when number of maps is selected as the same number of nodes. Minimum job execution times as shown in Figure 8 are 311.755 s with 1 node and 1 map, 168.923 s with 2 nodes and 2 maps, 218.111 s with 5 nodes and 5 maps, and 123.035 s with 10 nodes and 10 maps. Maximum CPU utilizations are 71% with 1 node and 1 map, 90% with 2 nodes and 2 maps, 80% with 5 nodes and 5 maps, and 85% with 10 nodes and 10 maps.

According to Figures 8 and 9, job execution time is minimum when CPU utilization is maximum. Increasing number of nodes improves job performance. Job execution time is 311.755 s on 1 node with 1 map and 123.035 s (with 60% performance improvement) on 10 nodes with 10 maps.

The relationship between number of maps and number of nodes for system optimization is presented in Eq. (1). Mn corresponds to number of maps, Nn is for number of nodes, and $CPUn$ corresponds to existing number of CPU cores on the node in Eq. (1).

$$Mn = Nn \times CPUn \tag{1}$$

In Teragen tests, job execution time is minimum when number of maps is 10 on a cluster with 10 nodes. Using only 1 node with 1 map instead of a cluster with 10 nodes provides 90% decrease in cost with 60% increase in job completion time. Using a cluster with 2 nodes with 2 maps provides 80% decrease in cost with 28% increase in job completion time. Using a cluster with 5 nodes with 5 maps provides 50% decrease in cost with 44% increase in job completion time.

2.4.2. Terasort benchmarking tool

Terasort tests were conducted on constant number of nodes (10 nodes) with constant number of reduces (10 reduces) and different number of maps (2, 5, 8, 20, 50, and 150 maps).

According to the test results on constant number of nodes with constant number of reduces, number of maps is not an effective parameter on job execution time and CPU utilization performance. Number of maps was selected as the default value (150 maps) according to HDFS file system block size (64 MB). CPU utilizations and job execution times of Terasort tests on different number of nodes with using different number of reduces and constant number of maps (150 maps) are shown in Figures 10 and 11. Job execution times are shown in Figure 10 and CPU utilizations are shown in Figure 11.

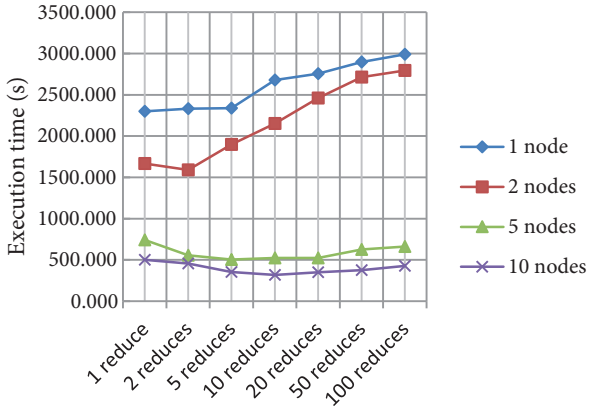

Figure 10. Terasort test job execution times with constant number of maps (150 maps).

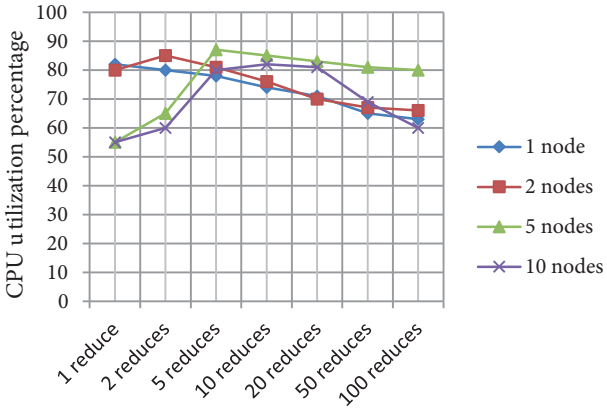

Figure 11. Terasort test CPU utilizations with constant number of maps (150 maps).

According to results presented in Figure 10, minimum job execution times are 2299.372 s on 1 node with 1 reduces, 1590.587 s on 2 nodes with 2 reduces, 504.177 s on 5 nodes with 5 reduces, and 318.537 s on 10 nodes with 10 reduces. These results can be seen in Figure 10. Maximum CPU usages are 82% on 1 map with 1 reduce, 85% on 2 nodes with 2 reduces, 87% on 5 nodes with 5 reduces, and 82% on 10 nodes with 10 reduces.

Job execution time performance is best when number of reduces is selected as the same number of nodes. As seen in Figures 10 and 11, job execution time is minimum when CPU utilization is maximum.

While job execution time is a minimum of 2299.372 s on 1 node with 1 reduce, it is a minimum of 318.537 s (with 87% performance improvement) on 10 nodes with 10 reduces.

The relationship between number of reduces and number of nodes for optimization of system is presented in Eq. (2). Rn corresponds to number of reduces, Nn is for number of nodes, and $CPUn$ corresponds to existing number of CPU cores on the node in Eq. (2).

$$\pi = \frac{m}{n} \times 4Rn = Nn \times CPUn \tag{2}$$

In Terasort tests, job execution time is minimum when the number of reduces is 10 on a cluster with 10 nodes. Using only 1 node with 1 reduce instead of a cluster with 10 nodes provides 90% decrease in cost with 87% increase in job completion time. Using a cluster with 2 nodes with 2 reduces provides 80% decrease in cost with 80% increase in job completion time. Using a cluster with 5 nodes with 5 reduces provides 50% decrease in cost with 37% increase in job completion time.

3. Conclusion

In this study, on a cloud system, a 10-node cluster is created by using KVM virtualization software. On these nodes, tests are carried out on prepared data sets with different numbers of maps and reduces by using benchmarking tools available under the Hadoop framework. In these tests, it is investigated how numbers of maps and reduces affect the performance of the Hadoop framework.

According to the test results, PiEstimator and Grep benchmarking tools are classified as CPU-light applications. It is not necessary to use a cluster for CPU-light applications to perform distributed processing. It is seen that in a single node with minimum number of maps, job execution time becomes minimum. If a CPU-light application is performed on a cluster, job execution time does not decrease; it even increases, because splitting, shuffling, and fetching times are added to total time. Thus, numbers of nodes, maps, and reduces of CPU-light applications should be minimum for system optimization. Number of nodes in a cluster is prepared on demand by cloud service providers and charged according to the usage time. In this case, decreasing number of nodes affects the cost.

Benchmarking tools Teragen and Terasort are classified as CPU-intensive applications according to the test results. In CPU-intensive applications, it is seen that increasing the number of nodes and increasing CPU utilization improve the job execution performance. Thus, according to the phase where the small job pieces is processed, numbers of maps or reduces of CPU-intensive applications should be the same as number of CPU cores in the cluster. In addition, increasing number of nodes of the cluster decreases the job completion time. The number of nodes in the cluster would be chosen according to the budget.

Hadoop has more parameters to improve job execution performance. These parameters could be adjusted in addition to number of nodes to improve job execution performance in future studies. In this paper, the data set is generated by the selected benchmarking tools. In future studies, different data sets could be used to measure job execution performance of the Hadoop framework.

References

[1] Hurtwitz J, Nugent A, Halper F, Kaufman M. Big Data for Dummies. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[2] Slagter K, Hsu CH, Chung YC, Zhang D. An improved partitioning mechanism for optimizing massive data analysis using mapreduce. The Journal of Supercomputing 2013; 66: 539-555.

[3] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. Commun ACM 2008; 51: 107-113.

[4] Perera S, Gunarathne T. Hadoop MapReduce Cookbook. Birmingham, UK: Packt Publishing, 2013.

[5] Aditya BP, Manashvi B, Ushma N. Addressing big data problem using hadoop and map reduce. In: 2012 Nirma University International Conference on Engineering; 6–8 December 2012; Ahmedabad, India. New York, NY, USA: IEEE. pp. 1-5.

[6] Wu Y, Ye F, Chen K, Zheng W. Modeling of distributed file systems for practical performance analysis. IEEE T Parall Distr 2014; 25: 156-166.

[7] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies; 3–7 May 2010; Incline Village, NV. New York, NY, USA: IEEE. pp. 1-10.

[8] Lee SW, Yu F. Securing KVM-based cloud systems via virtualization introspection. In: 47th Hawaii International Conference on System Sciences; 6–9 January 2014; Waikoloa, HI. New York, NY, USA: IEEE. pp. 5028-5037.

[9] Guo S. Hadoop Operations and Cluster Management Cookbook. Birmingham, UK: Packt Publishing, 2013.

[10] Tan YS, Tan J, Chng ES. Hadoop framework: impact of data organization on performance. Wiley Online Library 2011; 43: 1241-1260.

[11] Premchaiswadi W, Romsaiyud W. Optimizing and tuning MapReduce jobs to improve the large-scale data analysis process. Int J Intell Syst 2013; 28: 185-200.

[12] Herodotou H, Lim H, Luo G, Borisov N, Dong L, Cetin FB, Babu S. Starfish: A self-tuning system for big data analytics. In: 5th Biennial Conference on Innovative Data Systems Research; 9–12 January 2011; Asimolar, CA. pp. 261-272.

[13] Verma A, Cherkasova L, Campbell RH. ARIA: Automatic resource inference and allocation for MapReduce environments. In: Proceedings of the 8th ACM International Conference on Autonomic Computing; 14–18 June 2011; Karlsruhe, Germany. New York, NY, USA: ACM. pp. 235-244.

[14] Zhang Z, Cherkasova L, Loo BT. Benchmarking approach for designing a MapReduce performance model. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering; 21–24 April 2013; Prague, Czech Republic. New York, NY, USA: ACM. pp. 253-258.

[15] Babu S. Towards automatic optimization of MapReduce programs. In: Proceedings of the 1st ACM Symposium on Cloud Computing; 10–11 June 2010; Indianapolis, IN, USA. New York, NY, USA: ACM. pp. 137-142.

[16] Fadika Z, Dede E, Govindaraju M, Ramakrishnan L. Benchmarking MapReduce implementations for application usage scenarios. In: 12th IEEE/ACM International Conference, Grid Computing; 21–23 September 2011; Lyon, France. New York, NY, USA: IEEE. pp. 90-97.

[17] Massie ML, Chun BN, Culler DE. The ganglia distributed monitoring system: design, implementation, and experience. Parallel Comput 2004; 30: 817-840.

[18] Venner J. Pro Hadoop. New York, NY, USA: Apress, 2009.

[19] Maurya M, Mahajan S. Performance analysis of MapReduce programs on hadoop cluster. In: 2012 World Congress on Information and Communication Technologies; 30 October 2012–02 November 2012; Trivandrum, India. New York, NY, USA: IEEE. pp. 505-510.

[20] White T. Hadoop Definitive Guide. 2nd Ed. Sebastopol, CA, USA: O'Reilly Media, 2010.