Analysis of Big Data Platform with OpenStack and Hadoop

Xiaoyan Li[1], Zhihui Lu[1(✉)], Nini Wang[2], Jie Wu[2], and Shalin Huang[3]

[1] School of Computer Science, Fudan University, Shanghai 200433, China
{xylil4,lzh}@fudan.edu.cn
[2] Engineering Research Center of Cyber Security, Auditing and Monitoring, Ministry of Education, Shanghai 200433, China
{14210240052,jwu}@fudan.edu.cn
[3] Wangsu Science & Technology Co., Ltd., Shanghai 200433, China
sallyhuang@chinanetcenter.com

Abstract. In the era of big data, the cloud infrastructure needs to strongly support big data. As a distributed computational framework, Hadoop is one of the de facto leading software tools for solving big data problems. The cloud infrastructure has been proven to be a good support for three-tier architecture applications. In this paper, we construct a Hadoop big data platform based on OpenStack cloud. At the same time, we design three experimental scenarios, carry out a set of experiments using the standard Hadoop benchmarks TestDFSIO, TeraSort and PI, and examine the performance. Our experiments reveal that the disk read operation of physical servers can be a bottleneck for TestDFSIO and TeraSort. Wider allocation of VMs over physical servers achieves better performance for read jobs of TestDFSIO and TeraSort. For CPU-intensive job PI, the best practice is to centralize the allocation of VMs over physical machines.

Keywords: Hadoop · Benchmarks · Big data · HDFS · Cluster · Openstack · Cloud

1 Introduction

Big data [1] era is coming. Many definitions of big data are given by researchers such as big data [2] is the data that is massive, too fast or too hard for existing tools to handle and process. Here massive means the size of data can range from petabytes (PB) to exabytes (EB) or to zettabytes (ZB) [3]. It can be generated through many sources like business processes, transactions, social networking sites, web servers, etc. and remains in structured as well as unstructured form [4].

Big data refers to the technologies and architectures, which were developed to capture, store, process and run better quality volumes of data in lesser amount of time or even in real time [5]. It is a system that lets digitize massive amounts of information and amalgamating it with on hand databases.

Hadoop [6] (Apache) is one of these big data technologies and is one of the de facto leading software tools for solving big data problems. Hadoop provides a distributed

G. Wang et al. (Eds.): APSCC 2016, LNCS 10065, pp. 375–390, 2016.
DOI: 10.1007/978-3-319-49178-3_29

computational framework Map-Reduce and a reliable scalable distributed file system HDFS(Hadoop Distributed File System) for the analysis and transformation of large amounts of data [7].

MapReduce [8] provides a computational framework for data processing. An MR program only consists of two functions, called Map and Reduce, which are supplied by the user and depend on the user's purposes.

A map function is used to process input key/value pairs and generate intermediate key/values, and a reduce function is used to merge all intermediate pairs associated with the same key and then generate outputs [9].

In recent years, the cloud infrastructure has been proven to be a good support for three-tier architecture applications, such as websites. But in the era of big data, the cloud infrastructure needs to strongly support big data application platform, such as Hadoop and Spark. In this paper, we propose a cloud-based framework based on OpenStack using Hadoop as a big data platform. At the same time, we design three experimental scenarios, carry out a set of experiments using the standard Hadoop benchmarks, namely TestDFSIO, TeraSort and PI, and examine the performance.

The rest of this paper is organized as follows: we review related work in Sect. 2. In Sect. 3, experimental setup is given. In Sect. 4, we design three experimental scenarios. Section 5 shows the results and the conclusions. Finally, we summarize the considerations and propose our future work in Sect. 6.

2 Related Work

To reduce the machine management difficulties, virtualization is used as a key technique for easy deployment, configuration, scheduling and efficient resource utilization. Xen [10], Kernel-based Virtual Machine (KVM) [11] and VMware [12] are well-known virtualization softwares. In [13], Jack Li et al. found that KVM was better for disk reading. So we examine the performance of VM Hadoop clusters on KVM from VM number, configuration and allocation.

Ishii M et al. built in [14] a Hadoop performance model and examined how the performance was affected by changing VM configuration, allocation of VMs over physical machines, and multiplicity of jobs. They found that performance of the I/O-intensive jobs was more sensitive to the virtualization overhead than that of CPU-intensive jobs. In our paper, we choose I/O-intensive job and CPU-intensive job to figure out the performance differences among different VM placements over physical servers and the influence of the number of VMs when the number of total VCPUs and total memory are fixed.

In [15], the authors proposed a simple big data workload differentiation, their results show that CPU intensive workloads consume more power and memory bandwidth while disk intensive workloads usually require more memory. But they didn't find out the bottlenecks of CPU intensive or disk intensive workloads.

In [16], Fan et al. designed a heuristic performance diagnostic tool which evaluates the validity and correctness of virtualized Hadoop by analyzing the job traces of popular big data benchmarks. With this tool, users could quickly identify the bottleneck

according to hints provided by this tool. We find the bottlenecks of TestDFSIO and TeraSort in our paper.

The main contributions we make are listed below:

- Our experiments reveal that the disk read operation of physical servers can be a bottleneck for TestDFSIO and TeraSort.
- We find that if there is enough resource, the best practice is to increase the number of VMs and not to increase the number of VCPUs in a VM for I/O intensive job.
- Our experiments show that wider allocation over physical servers achieves better performance for read jobs of TestDFSIO and TeraSort. For CPU-intensive job PI, the best practice is to centralize the allocation of VMs over physical machines.

3 Experimental Setup

In this section, we describe the environment for our experiments. In Sects. 3.1 and 3.2, we give an overview of our experimental scenario and physical configurations, and in Sect. 3.3, we provide a brief overview of the benchmarks we adopt.

3.1 OpenStack Cloud-Based Hadoop

Figure 1 provides an overview of the environment for our experiments.

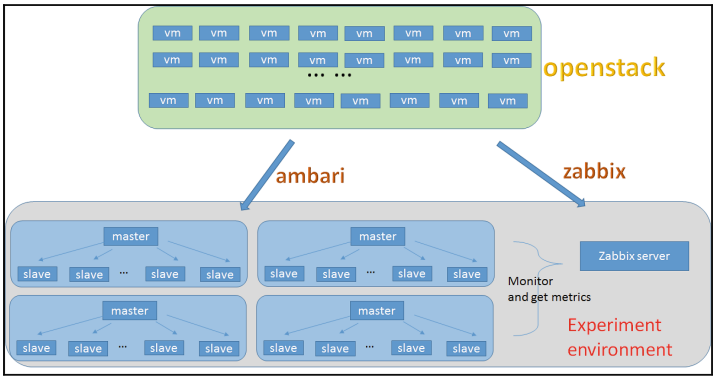

Fig. 1. Big data platform design

We use OpenStack as our cloud platform to launch the VM resource pool we need in the experiment, and we use Ambari to deploy Hadoop cluster into VM resource pool. And through installing one Zabbix server, we build a monitor system to retrieve metrics from the Hadoop clusters.

3.2 Physical Configurations

Table 1 provides an overview of physical configurations we use for our experiments.

Table 1. Hardware configuration

Physical machine	
Processor	Xeon E5-2603 v3
Memory	64 G
Operating System	CentOS 7.1
Disk	dell-10 k-2 TB

3.3 Benchmark

Several benchmarks are tested from Hadoop example applications: PI, TestDFSIO, and TeraSort.

- PI
  PI is a map/reduce program that estimates pi using a quasi-Monte Carlo method [17]. The map tasks are all independent and the single reduce task gathers very little data from the map tasks.
- TestDFSIO
  TestDFSIO is a map/reduce program that reads/writes random data from/to large files. It is mainly used to test the I/O speed of the cluster.
- TeraSort
  TeraSort is a standard benchmark created by Jim Gray. TeraSort is a two-phase Hadoop workload that performs in-place sort of all the words of a given data file.

4 Scenario Design

We design three experimental scenarios to examine the performance of Hadoop in this paper. We run the I/O intensive and CPU intensive experiments to see if there is difference between them. TestDFSIO is the I/O-intensive job, TeraSort and PI are the CPU-intensive jobs.

We get metrics such as memory usage, CPU utilization, read/write speed of disk, network input/output throughput along with other metrics by Zabbix. We select typical metrics to analyze their characteristics and achieve the purpose of each scenario.

4.1 Cluster Scalability Test

Target for Cluster Scalability Test: The purpose of this scenario is to figure out the bottleneck resource of VM/PM by adding VMs of same spec to the cluster.

Experiment Assumption for Cluster Scalability Test: The disk I/O of physical servers can be the bottleneck for I/O-intensive job, while CPU can be the bottleneck for CPU-intensive job. That is, in this scenario, disk write or read operation is the bottleneck for TestDFSIO, and CPU is the bottleneck for TeraSort.

Environment Brief Diagram: This environment makes some specific designs. All the VMs are in one cluster, which includes one master and several slaves. The slaves are all configured in the same way. But we use the cluster in three kinds of situations. Every job runs separately in one situation and the metrics are different. This allows us to compare results and draw conclusions. The total resources of three scenarios increase proportionately.

All the slaves are divided into three kinds of situations to do the test job. We launched VMs with proportional configuration and quantity. We rationally use the resources.

The Configuration and Distribution: There are totally 9 physical machines to deploy this environment. Every physical machine runs centOS 7.1 as operating system and all the virtual machines are Ubuntu 12.04. The Hadoop version is 2.6. We separate managing and monitoring nodes with job nodes. One master node, one Ambari server node and one Zabbix server node are on one physical machine. All the slave nodes are on the other 8 physical machines according to the arrangement as Fig. 2 shows. The configurations are shown in Table 2.

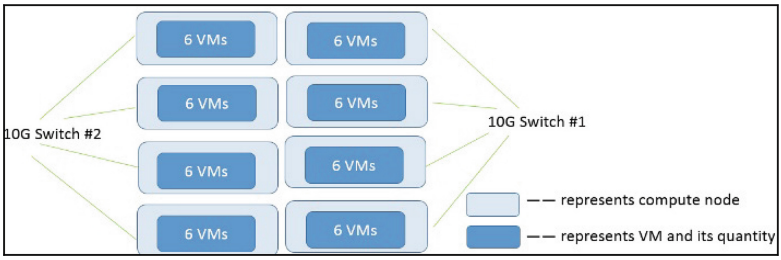

Fig. 2. Cluster scalability test environment diagram

Table 2. Configuration table of cluster scalability test

Node	CPU	Memory	Disk
Master	2 cores	8 G	100 G
Slave	1 cores	5 G	80 G
Ambari server	1 core	2 G	100 G
Zabbix server	8 cores	16 G	100 G

4.2 VM Specification Test

Target for VM Specification Test: The purpose of this scenario is to figure out the influence of the number of VMs when the number of total VCPUs and total memory are fixed.

Experiment Assumption for VM Specification Test: The disk I/O of physical servers can be bottlenecks for I/O-intensive job which we will prove in the cluster scalability scenario, while CPU intensive job is less sensitive to disk I/O if the number of VCPUs does not exceed the number of physical cores. That is, in this scenario, for I/O-intensive job, the cluster with more VMs will perform better. And for a CPU intensive job, the number of VMs in a cluster does not make a difference since the total number of VCPUs, in addition to being enough, are equal to or less than the total physical cores.

Environment Brief Diagram: For this environment, all the VMs are divided into two clusters as Fig. 3 shows, each including one master and several slaves. The slaves in the same cluster use the same configuration and differ from the slaves in the other cluster. Every job runs separately in two clusters and the metrics are different. This allows us to compare the results and draw conclusions. The total resources of two clusters are the same.

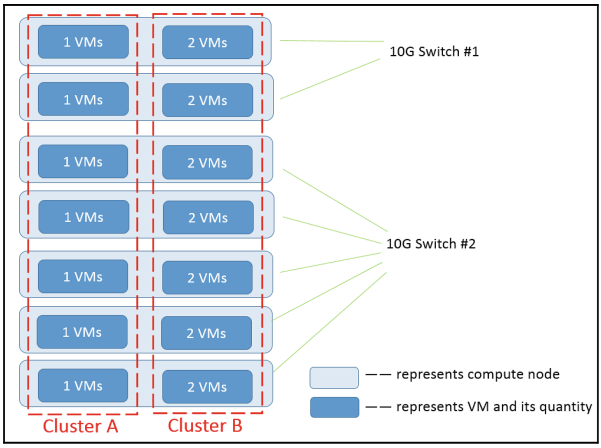

Fig. 3. VM Specification Test environment diagram

All the slaves are divided into two clusters to do the test job. We launch VMs with proportional configuration and quantity. We rationally use the resources.

The Configuration and Distribution: We use the same Hadoop version, operating systems and configurations for specification test as the ones for scalability test.

The configurations are shown in Table 3.

Table 3. Configuration table of VM specification test

Node	CPU	Memory	Disk
Master	2 cores	8 G	100 G
Slaves in cluster A	4 cores	10 G	100 G
Slaves in cluster B	2 cores	5 G	100 G
Ambari server	1 core	2 G	100 G
Zabbix server	8 cores	16 G	100 G

4.3 VM Placement Test

Target for VM Placement Test: The purpose of this scenario is to figure out the performance differences among different VM placements over physical servers with homogeneous VMs.

Experiment Assumption for VM Placement Test: In this scenario, for the I/O-intensive job, when the number of VMs in a physical server is changed, the centralized allocation needs more disk read and writes. Thus, the wider allocation over physical servers achieves a better performance.

Since the number of VCPUs used in VMs is less than the number of physical cores so that no context switch occurs, the VM allocation over physical servers does not affect the CPU-intensive job case.

Environment Brief Diagram: All the slaves are divided into three clusters to do the test job as Fig. 4 shows. We launched VMs which functioned as slaves with the same configuration.

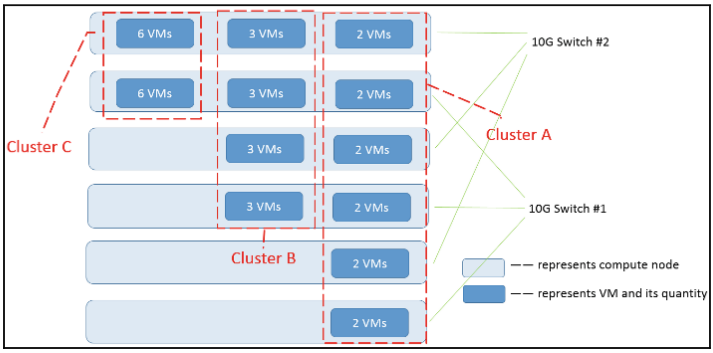

Fig. 4. VM placement test environment diagram

The Configuration and Distribution: We use the same Hadoop version, operating systems and configurations for VM placement test as the ones for the scalability test.

The configurations are shown in Table 4.

Table 4. Configuration table of VM placement test

Node	CPU	Memory	Disk
Master	2 cores	8 G	100 G
Slaves	1 core	5 G	80 G
Ambari server	1 core	2 G	100 G
Zabbix server	8 cores	16 G	100 G

5 Results and Discussion

5.1 Cluster Scalability Test

I/O-intensive Job-TestDFSIO: We set the VMs to 12, 24, 36, 48, set the map number to 8, 80, 800, 1600, 4000, set the reduce number to 1 and the file size to 128 M. The map number is equal to the total data size divided by the file size. As the reduce phase only need to collect and summarize all the statistical information of map tasks, we set the reduce number to 1. After we finish the write job of TestDFSIO, we record the total time of job printed in the Hadoop running console, the data size and calculate the total throughput as Fig. 5 shows.

$$<\text{Total Throughput}> \; = \; <\text{Total data size}> / <\text{Job execution time}>$$

When the data scale is set to 500 G, the hard disks volume of each task node is not enough during the process when VMs are 12 and 24, so the experiment of 500 G data can't be performed.

From Fig. 5 we can see that in most cases, when the data size is fixed, the total throughput is higher with a larger VM number. When the data size is more than 100 G with VMs at 48, the increment speed of total throughput is lower as the write job reaches the bottleneck.

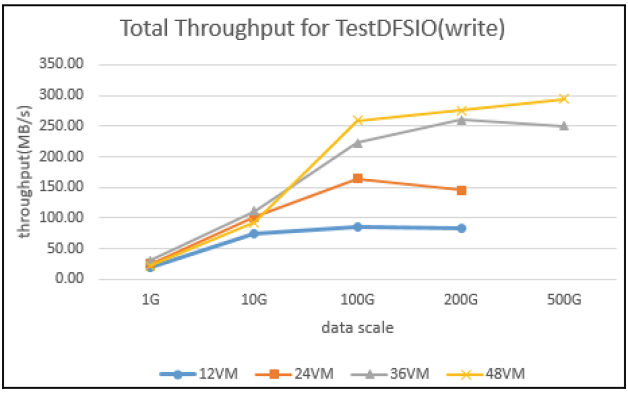

Fig. 5. Total throughput for TestDFSIO write job

So we get data from Zabbix when the data scale is 100 GB and draw the figures to find the bottleneck of TestDFSIO write job.

For the TestDFSIO write job, only map tasks execute actual workload. From Fig. 6 we can see 4 points as below:

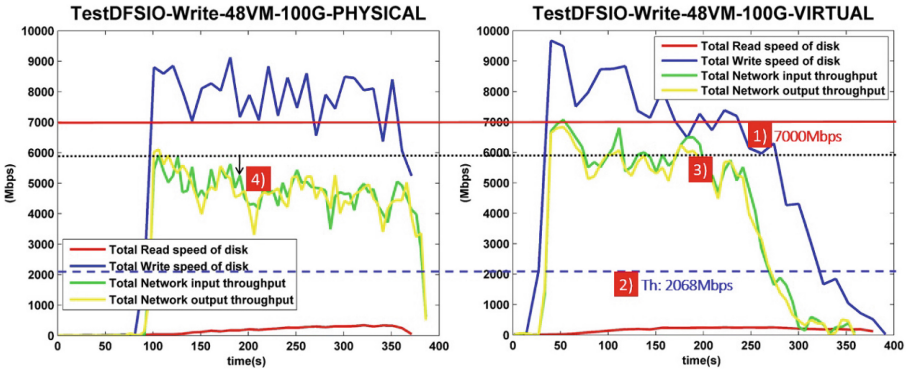

Fig. 6. Metrics for TestDFSIO write job

(1) *Disk throughput reaches above* 7000 Mbps/8 PM *(both in VM / PM layer.)*
(2) *Actual total throughput is* 2068 Mbps. *The disk throughput is much more than the actual total throughput as HDFS replica working during write job.*
(3) *Virtual environment: The network throughput is almost three times as much as the actual total throughput as HDFS replica working during write job and the HDFS replica is 3.*
(4) *Physical environment: Network transfer is less than virtual environment as local VM transfer within one PM is hidden from PM view.*

From Fig. 7 we can see CPU isn't fully utilized either in physical machines or in virtual machines, so CPU isn't the bottleneck.

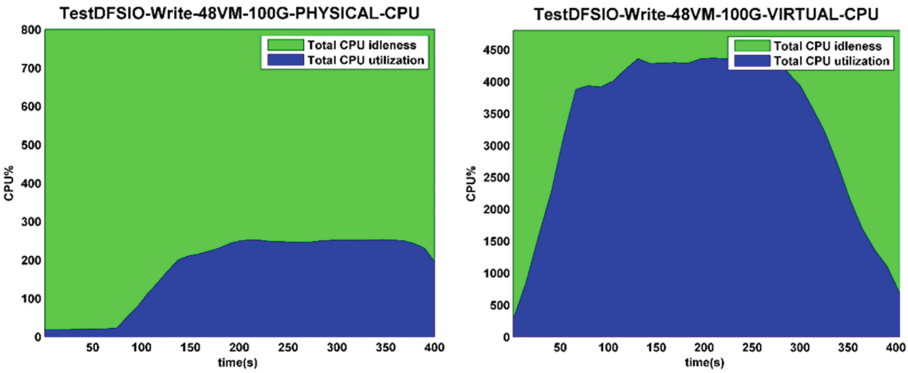

Fig. 7. Physical and Virtual CPU for TestDFSIO write job

We use the same configuration for read jobs as the one for writes. After we finish the read job of TestDFSIO, we record the relevant data as Fig. 8 shows.

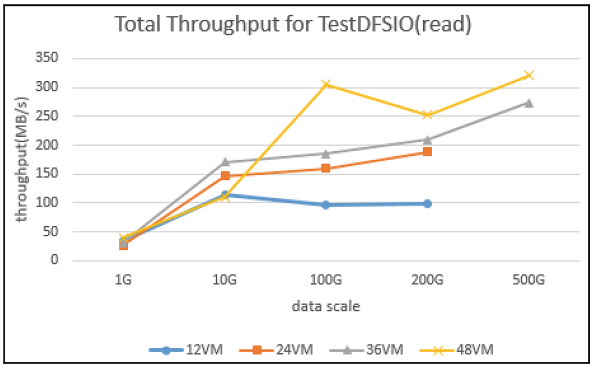

Fig. 8. Total throughput for TestDFSIO read job

When data scale is set to 500 G, the reason why the experiments can't be performed when VMs are 12 and 24 is the same as that for the write job.

From the Fig. 8 we can draw the same conclusion for read jobs as the one for writes.

So we get data from Zabbix and draw the figures to find the bottleneck of TestDFSIO read job. As the CPU utilization of read job is almost the same as that of the write job and CPU isn't fully utilized, we can conclude that CPU isn't the bottleneck.

For TestDFSIO read job, only map tasks execute actual workload. From Fig. 9 we can see 3 points as below:

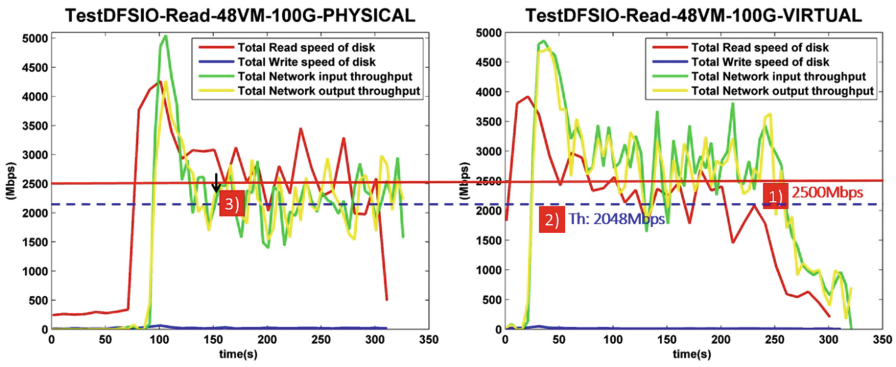

Fig. 9. Metrics for TestDFSIO read job

(1) *Disk throughput is* 2500–3000 Mbps/8 PM*, which is much lower than that of write.*
(2) *Actual Total Throughput is* 2048 Mbps*, which is almost the same as the disk read speed.*
(3) *Physical environment: The number of network transfer is less than the one in virtual environment as local VM transfer within one PM is hidden from PM view.*

Analysis and Conclusion of Cluster Scalability Test: For TestDFSIO write job, disk write is over 7000 Mbps/8 PM in TestDFSIO write. For the read job, disk read throughput is 1/3 of the disk write. Since network traffic throughput is higher in write test, disk read operation seems to be the bottleneck.

Using the same method we find that disk read operation, not the CPU is also the bottleneck of TeraSort, which doesn't bear out our assumption. The reason may be that TeraSort is not a particularly CPU-intensive job, and that it also needs a lot of writing and reading. So it is consistent with the assumption of the I/O-intensive job.

For the scalability test, we conclude that for TestDFSIO and TeraSort, the bottleneck is disk read operation.

5.2 VM Specification Test

I/O-intensive Job-TestDFSIO: We set the map number to 28, the reduce number to 1 and the file size to 1000 M. After we finish the write job of TestDFSIO, we record the relevant data as Fig. 10 shows.

Fig. 10. Execution time for TestDFSIO write job

From Fig. 10 we can see that cluster B have better performance than cluster A. The total time of cluster B is 46 % less than cluster A.

CPU-intensive Job-PI: We change the map number to 28, set the reduce number to 1 and the execute number per map to $5*10^9$. By changing the number of map and the execute number per map, we can indicate the desired PI accuracy. The larger of the

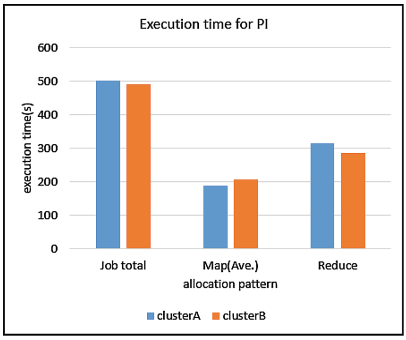

Fig. 11. Execution time for PI

multiplication of these two, the higher the accuracy. As the reduce phase only need to collect the statistical information of map tasks, we set the reduce number to 1. After we finish the job of PI, we record the relevant data as Fig. 11 shows.

VMs in cluster A and B are fully used when map number is 100. From Fig. 11 we can see that the performance of cluster B doesn't show much difference from that of cluster A. The performance degradation caused by the VM configuration change is 2 % for PI.

Analysis and Conclusion of VM Specification Test: We first set the map number to 28 for the write job of TestDFSIO. From the total execution time of the job we can see that for write, cluster B out-performs cluster A by 46 %. So we speculate that for I/O-intensive jobs, the best practice is to increase the number of VMs and not to increase the number of VCPUs in a VM.

Then we set the map number to 100 for PI, the performance of cluster B doesn't show much difference from that of cluster A. The performance degradation caused by the VM configuration change is 2 % for PI. So we conclude that for CPU-intensive job, the number of VMs in a cluster does not make much difference since the total number of VCPUs is enough and I/O utilization is small enough.

5.3 VM Placement Test

I/O-intensive Job-TestDFSIO: We set the map number to 48, the reduce number to 1 and the file size to 500 M. After we finish the read job of TestDFSIO, we record the relevant data as Fig. 12 shows.

VMs in cluster A, B and C are fully used when map number is 48. From Fig. 12 we can see that the total time of cluster A, B and C is increasing. The average time of map is increasing, too. We conclude that the wider allocation over physical servers achieves better performance for TestDFSIO.

CPU-intensive Job-TeraSort: We set the map number to 400, the reduce number to 12 and the sort size to 50 G. The map number is equal to the total data size divided by

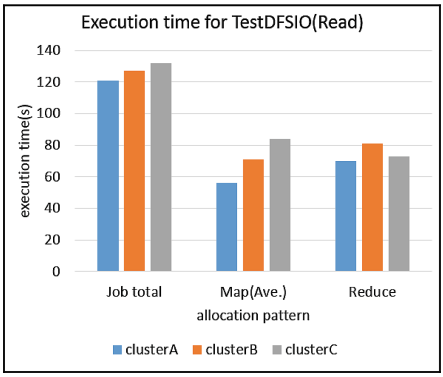

Fig. 12. Execution time for TestDFSIO read job

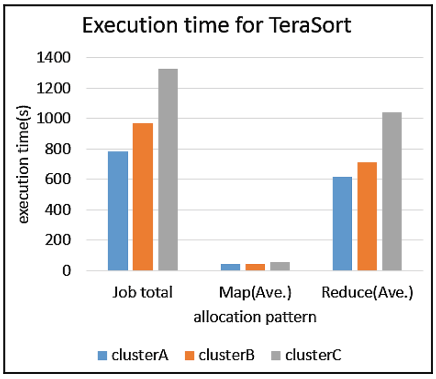

Fig. 13. Execution time for TeraSort

the file size. The reduce number is equal to the number of CPU cores. After we finish TeraSort job, we record the relevant data as Fig. 13 shows.

VMs in cluster A, B and C are fully utilized. From the Fig. 13 we can see that the total time difference among the three patterns is mainly caused by the cost of the reduce phase, and the total time of cluster A, B and C is increasing. We conclude that the wider allocation over physical servers achieves better performance for TeraSort.

CPU-intensive Job-PI: We set the map number to 48, the reduce number to 1 and the execute number per map to $1*10^9$. After we finish the job of PI, we record related data as Fig. 14 shows.

VMs in cluster A, B and C are fully utilized. From Fig. 14 we can see, the performance of cluster A and B doesn't show much difference. Cluster C achieves 21 % better performance than cluster A, achieves 26 % better performance than cluster B.

We conclude that the centralized allocation of VMs over physical machines can improve the efficiency of the CPU-intensive job-PI.

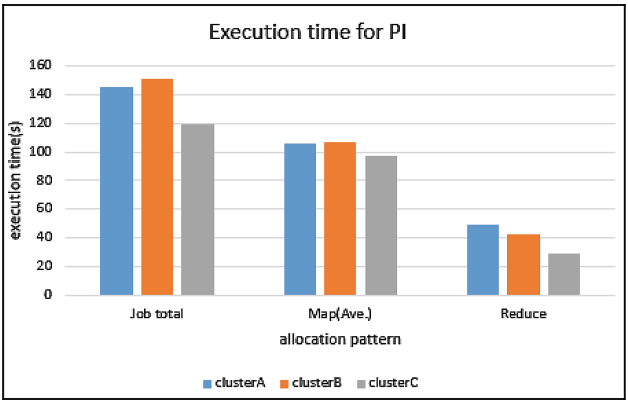

Fig. 14. Execution time for PI

Analysis and Conclusion of VM Placement Test: We set the map number to 48 for read jobs of TestDFSIO, set the map number to 400 for TeraSort jobs and we make full use of the VMs in 3 clusters for both of them. For TestDFSIO and TeraSort jobs, the total time of cluster A, B and C is increasing. So we conclude that the wider allocation over physical servers achieves better performance for TestDFSIO and TeraSort.

We set the map number to 48 for CPU-intensive job PI and we make full use of the VMs in this scenario in 3 clusters. Cluster C achieves 21 % better performance than cluster A, achieves 26 % better performance than cluster B. That is, the most centralized allocation of VMs over physical machines has the best performance.

For this scenario, we conclude that the wider allocation over physical servers achieves the better performance for TestDFSIO and TeraSort. The centralized allocation of VMs over physical machines can improve the efficiency of a CPU-intensive job-PI.

5.4 Summary

For scalability test, we conclude that for both TestDFSIO and TeraSort, the bottleneck is disk read operation.

For VM specification test, we conclude that for the I/O intensive job, the best practice is to increase the number of VMs and not to increase the number of VCPUs in a VM, for CPU intensive job-PI, the number of VMs in a cluster does not make big difference. The conclusion is the same as [14].

For the VM placement test, we conclude that the wider allocation over physical servers achieves the better performance for read job of TestDFSIO and TeraSort. TeraSort is not a typical CPU intensive job and it conforms to the assumption of the I/O intensive job. Centralized allocation of VMs over physical machines can improve the efficiency of a CPU-intensive job PI, which is not consistent with the conclusion of [14], possibly due to some context switch.

6 Conclusion and Prospect

In this paper, we have designed three experimental scenarios and compared their performances. Our experiments revealed that the disk read operation of physical servers can be bottlenecks for TestDFSIO and TeraSort. If the resource is enough, the best practice is to increase the number of VMs and not to increase the number of VCPUs in a VM for I/O intensive job. Wider allocation over physical servers achieves better performance for read job of TestDFSIO and TeraSort. For CPU-intensive job PI, the best practice is to centralize allocation of VMs over physical machines.

In our future work, the model of Hadoop will be studied and docker will be integrated into the big data system. We will test Hadoop performance by comparing HDFS and Ceph. Moreover, the rationality of parameter selection such as the number of map and reduce tasks and the influence of data size need to be further investigated.

Acknowledgments. This work is supported by Shanghai 2016 Innovation Action Project under Grant 16DZ1100200-Data-trade-supporting Big data Testbed. This work is also supported by 2016–2019 National Natural Science Foundation of China under Grant No. 61572137-Multiple Clouds based CDN as a Service Key Technology Research, Shanghai 2015 Innovation Action Project under Grant No. 1551110700 - New media-oriented Big data analysis and content delivery key technology and application, and Fudan-Hitachi Innovative Software Technology Joint Project-Cloud Platform Design for Big data.

References

1. Snijders, C., Matzat, U., Reips, U.D.: "Big Data": big gaps of knowledge in the field of internet science. Int. J. Internet Sci. **7**(1), 1–5 (2012)
2. Madden, S.: From databases to big data. IEEE Internet Comput. **16**(3), 4–6 (2012)
3. Kotiyal, B., Kumar, A., Pant, B., et al.: Big data: mining of log file through Hadoop. In: 2013 International Conference on Human Computer Interactions (ICHCI), pp. 1–7. IEEE (2013)
4. Patel, A.B., Birla, M., Nair, U.: Addressing big data problem using Hadoop and Map Reduce. In: 2012 Nirma University International Conference on Engineering (NUiCONE), pp. 1–5. IEEE (2012)
5. Nandimath, J., Banerjee, E., Patil, A., et al.: Big data analysis using Apache Hadoop. In: 2013 IEEE 14th International Conference on Information Reuse and Integration (IRI), pp. 700–703. IEEE (2013)
6. Hadoop. http://Hadoop.apache.org/Introduction
7. Song, G., Meng, Z., Huet, F., et al.: A Hadoop MapReduce performance prediction method. In: High Performance Computing and Communications, pp. 820–825. IEEE (2013)
8. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
9. Yang, H., Dasdan, A., Hsiao, R.L., et al.: Map-reduce-merge: simplified relational data processing on large clusters. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 1029–1040. ACM (2007)
10. Ko, B.M., Lee, J., Jo, H.: Toward enhancing block I/O performance for virtualized Hadoop cluster. In: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 481–482. IEEE Computer Society (2014)

11. Vasconcelos, P.R.M., de Araújo Freitas, G.A.: Performance analysis of Hadoop MapReduce on an OpenNebula cloud with KVM and OpenVZ virtualizations. In: 2014 9th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 471–476. IEEE (2014)
12. Kontagora, M., Gonzalez-Velez, H.: Benchmarking a MapReduce environment on a full virtualisation platform. In: 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 433–438. IEEE (2010)
13. Li, J., Wang Q, Jayasinghe D, et al.: Performance overhead among three hypervisors: an experimental study using Hadoop benchmarks. In: 2013 IEEE International Congress on Big Data, pp. 9–16. IEEE (2013)
14. Ishii, M., Han, J., Makino, H.: Design and performance evaluation for Hadoop clusters on virtualized environment. In: The International Conference on Information Networking 2013 (ICOIN), pp. 244–249. IEEE (2013)
15. Aggarwal, S., Phadke, S., Bhandarkar, M.: Characterization of Hadoop jobs using unsupervised learning. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 748–753. IEEE (2010)
16. Bortnikov, E., Frank, A., Hillel, E., et al.: Predicting execution bottlenecks in map-reduce clusters. Presented as part of the, p. 18 (2012)
17. Yin, J., Qiao, Y.: Performance modeling and optimization of MapReduce programs. In: 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, pp. 180–186. IEEE (2014)