Performance Evaluation of MapReduce applications on Cloud Computing Environment, FutureGrid

Yunhee Kang [1], Geoffrey C. Fox [2]

[1] Division of Information and Communication
Baekseok University
115 Anseo Dong, Cheonan, Korea 330-704
yunh.kang@gmail.com
[2] Pervasive Technology Institute
Indiana University
2719 E 10th St. Bloomington, Indiana, IN 47408, USA
gcf@indiana.edu

Abstract. This paper describes the result of performance evaluation of two kinds of MapReduce applications running in the FutureGrid: a data intensive application and a computation intensive application. For this work, we construct a virtualized cluster system made of a set of VM instances. We observe that the overall performance of a data intensive application is strongly affected by the configuration of the VMs. It can be used to identify the bottleneck of the MapReduce application running on the virtualized cluster system with various VM instances.

Keywords*:* MapReduce application, Cloud computing, FutureGrid

1 Introduction

Cloud computing is one of the most explosive technologies as natural evolution in the IT industry. It is designed to provide on demand resources or services over the Internet and with the reliability level of a data center [1]. Virtualization provides a way to abstract the hardware and system resources from an operation system, which is used to reduce the actual number of physical servers and to improve scalability and workloads in the cloud environment. In cloud computing environment, a virtual machine (VM) is a computing platform that creates a virtualized layer between the computing hardware and the application. FutureGrid based on cloud computing plays a role as a resource provider, which has a cloud stack including IaaS, PaaS and SaaS [2].

The demanding requirements in the FutureGrid have led to the development of a new programming model like MapReduce. A MapReduce application is deployed at the provider's computing infrastructure and the application can be composed as a service from other cloud services. A runtime supplies the developers with a programming language level environment with a set of well-defined APIs. It is referred to as PaaS.

This paper describes the result of performance evaluation of two kinds of MapReduce applications: one is a data intensive application and the other is a computational intensive application. For this work, we construct a virtualized cluster system made of a set of nodes of which are VM instances in the FutureGrid [2]. To monitor and measure the performance of nodes in the virtualized cluster system while the MapReduce applications are running on nodes, we use a tool, `top` command. To take a system snapshot of the cluster system, we write a shell script that extracts information including load average and memory/swap usage from the results of `top` command.

The paper is organized as follows: Section 2 describes the related works including brief overview of MapReduce, Twister and the FutureGrid. We report the results from our experimental study and the observations from the result of our experiments in Section 3. Conclusions are presented in Section 4.

2 Related Works

2.1 MapReduce

MapReduce programs are designed to compute large volumes of data in a parallel fashion. It is a kind of data parallel languages aimed at loosely coupled computations that execute over given data sets [3]. This requires dividing the workload across a large number of machines. The degree of parallelism depends on the input data size. MapReduce, introduced by Dean and Ghemawat at Google, is the most dominant programming model for developing applications in cloud computing environment [3-5].

In a MapReduce application supported by a MapReduce library, all map operations can be executed independently. Each reduce operation may depend on the outputs generated by any number of map operations. All reduce operations can also be executed independently. The following describes the MapReduce programming model:

- The computation takes a set of input (key, value) pairs, and produces a set of output (key, value) pairs. The computation is expressed as two functions: Map and Reduce.
- The Map takes an input pair and produces a set of intermediate (key, value) pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key key' and passes them to the Reduce function.
- The Reduce accepts an intermediate key key' and a set of values of that key. It merges together these values to form a possible smaller set of values.

Traditional parallel applications are based on a runtime library for message passing such as MPI [10] and PVM [11] that have some programming features of communication and synchronization. The feature provided by a runtime library is a low-level primitive. In MapReduce, a programmer is able to focus on the problem that

needs to be solved since only the map and reduce functions need to be implemented, and the framework takes care of computing the programmer has to deal with lower-level mechanisms to control the data flow [2,4].

2.2 Twister

There are some existing implementations of MapReduce such as Hadoop [6] and Sphere [7]. Twister is one of MapReduce implementations, which is an enhanced MapReduce runtime with an extended programming model that supports an iterative MapReduce computing efficiently [8]. In addition it provides programming extensions to MapReduce with broadcast and scatter type for transferring data. These improvements allow Twister to support iterative MapReduce computations highly efficiently compared to other MapReduce runtimes. It reads data from local disks of the worker nodes and handles the intermediate data in distributed memory of the worker nodes.

All communication and data transfers are performed via a pub/sub messaging system NaradaBrokering that is an open-source, distributed messaging infrastructure [9]. Twister uses a publish/subscribe messaging infrastructure to handle four types of communication needs; (i) sending/receiving control events, (ii) send data from the client side driver to the Twister daemons, (iii) intermediate data transfer between map and reduce tasks, and (iv) send the outputs of the reduce tasks back to the client side driver to invoke the combine operation.

2.3 FutureGrid

FutureGrid is a distributed testbed for developing research applications and middleware, which employs virtualization technology to allow the testbed to support a wide range of operating systems. This project provides a capability that makes it possible for researchers to tackle complex research challenges aimed at minimizing overhead and maximizing performance in computer science related to the use and security of grids and clouds. It has been offering a flexible reconfigurable testbed based on dynamically provisioning software to support deploying a specific image to variety of environments composed of virtual machines [2].

A machine image is used as a template that is an abstraction of software stack including such as operating system, middleware and end-user access solutions. Hence a machine instance is an actual instantiation of the template. One of the goals of this project is to understand the behavior and utility of cloud computing approaches. FutureGrid dynamically provides diverse configurations that have different operating systems and middleware configurations. The goal of dynamic provisioning is to partition a set of resources in an intelligent way to provide a user-defined environment to any user that makes such a request.

3 Experimental Results

3.1 Experiment Environment

In this experiment, a virtualized cluster system consists of a set of nodes that are allocated from a cluster named `India`, which is one of FuturGrid environments. The image of an instance basically contains Linux 2.6.27 and Java VM 1.6. Each instance provides a predictable amount of dedicated computing capacity that is defined in FutureGrid. Table 1 shows an overview of the types of VM instances to be used in the experiments. Hereafter CPU is used to represent core.

Table 1. Main specification of VM instance type

Type of VM instance	Main HW Features		
	CPU	Memory (Mbyte)	Disk
c1-medium	1	1,024	7
m1-large	2	6,000	10
m1-xlarge	2	12,000	10

We make a configuration, which is based on the type of VM instance described in Table 1, for a virtualized cluster system as testbed and use various configurations that are used to evaluate performance of the aforementioned MapReduce applications. A configuration has various middleware setups. It is used to represent a specific workload. For example, `Type-2` represents an unbalanced load allocation and `Type-4` represents a balanced load allocation. Table 2 shows the list of configurations to be used in our experiments except `gf14-gf15` and `India`. The `gf14-gf15` is composed of two Linux machines. `India` is a multi-core machine having 1,024 cores in 128 nodes in the FutureGrid.

Table 2. Configuration of virtual cluster systems

Configuration	Main Features			Virtual Clusters
	CPU	Memory (Mbyte)	Location of NaradaBroker	
Type-1	1	1,024	c1-medium	1*c1-medium
Type-2	3	7,181	c1-medium	1*c1-medium 1*m1-large
Type-3	3	7,181	m1-large	1*c1-medium 1*m1-large
Type-4	4	12,000	m1-large	2*m1-large
Type-5	2	12,000	m1-xlarge	1*m1-xlarge

To set up the virtualized cluster systems, we deploy images and run the instances in `India`. A MapReduce application is implemented on a system using:

- Twister 0.8
- NaradaBroker 4.2.2
- Linux 2.6.x running on Xen

We gather and analyze OS-level performance metrics, without requiring any modifications to Twister, its applications or the OS to collect these metrics. For data collection, we choose a Linux command to be used as a system-monitoring tool, `top` that provides a dynamic real-time view of a running system, including information about system resource usage and a constantly updated list of the processes that are consuming the most resources.

3.2 Experiment: Data Intensive Application

In this experiment, two different computing environments are evaluated, which are running a data intensive application with various configurations: one is a cluster system composed of physical machines and the other is a virtualized cluster computing system. For this work, we construct a MapReduce application that is used to transform a data set collected from a music radio site, Last.fm(http://www.last.fm/) that provides the metadata for artists include biography by API, on the Internet. The goal program is to histogram the counts referred by musicians and to construct a bi-directed graph based on similarity value between musicians in the data set.

We compare both environments with application's performance metrics in terms of elapse time and standard variation. The graph in Figure 1 plots the results using the MapReduce application. In `Type-1`, there is starvation problem as the application run that is caused by no enough computing resources including CPU and memory capability. In the part of the graph, `Type-2` to `Type-5`, we see that as the resources of VMs including CPU and memory increase, the elapse time of the application and the value of its standard variation decrease.

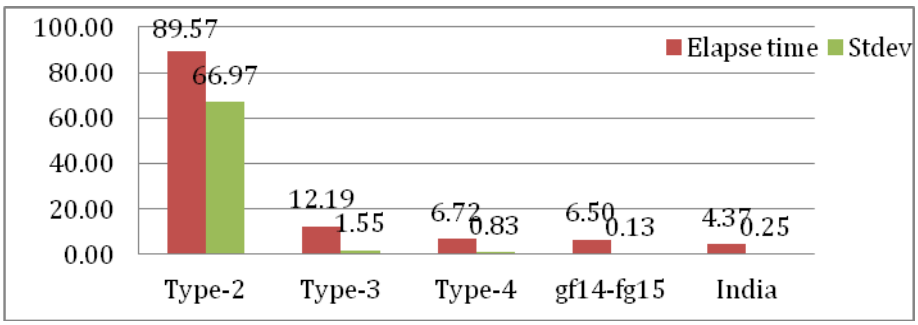

Fig. 1. Elapse time of similarity: 5 configurations virtualized cluster systems (4 configurations) and a physical cluster system (1 configuration)

What we observed is that the number of CPUs has less impact on the elapse time in comparison with the results of `Type-3` and `Type-4`. Though performance degrades

as the application runs in the virtualization environment, the performance of `Type-3` still provides 80.9% of the average performance of `gf14-gf15` and `India` when running the physical computing environment. However, the elapse time of type `Type-4` is 98.6 % of the elapse time of `gf14-gf15`.

As shown in Figure 2, the average memory usage is almost 100% of real memory and swap area in the node having a resource shortage problem during the most of its running time. Hereafter NB stands for NaradaBroker that plays a role as message. Hence the application is delayed due to the heavy disk I/O caused by high swap value.

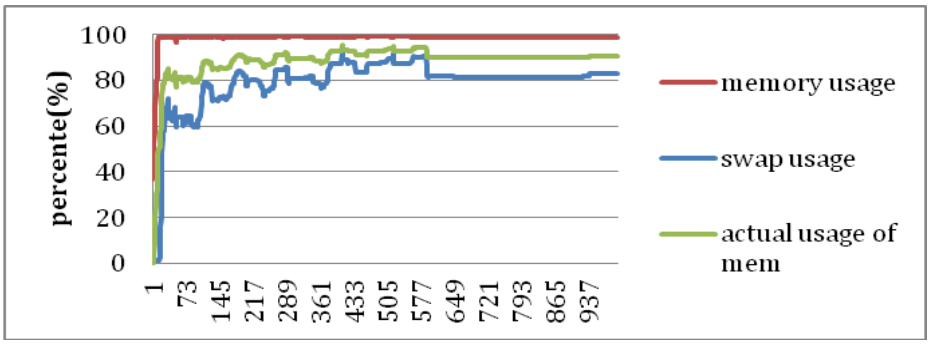

Fig. 2. Memory/swap area usage of Type-2 (NB running on the node typed with c1-medium)

As shown in Figure 3, the average memory usage is around 80% in the node during its running time. But the average swap area usage is less than 1% in running time. As a result, it can properly handle the I/O requests from the application.

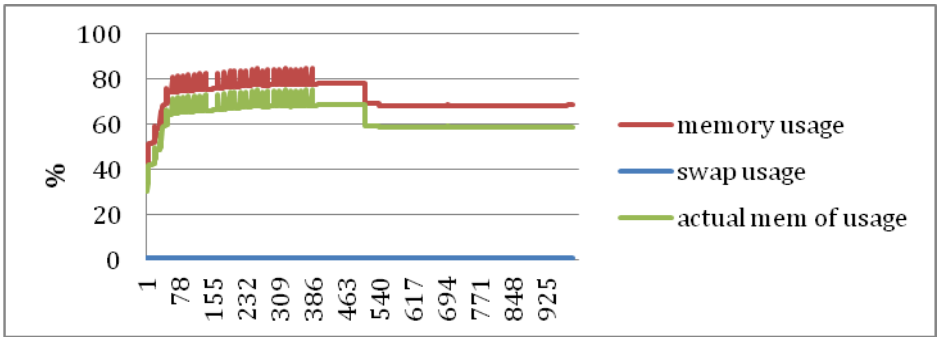

Fig. 3. Memory/swap area usage of Type-3 (NB running on the node typed with m1-medium)

Based on the performance evaluation we may choose the configuration of a virtualized cluster system to provide 80% of performance of a real cluster system. This observation induces that load balancing is helpful in spreading the load equally across the free nodes when a node is loaded above its threshold level:

- The performance of the application running on the Twister strongly depends on the throughput of a message broker, Naradabroker.
- The pending of the application is caused by broken pipe between a Twister daemon and a Naradabroker server when Naradabroker has a threshold of the limitation to accept a connection from Twister due to its QoS requirement.
- The capability of Naradabroker in the middleware configuration affects the performance of an application as the application runs in the same configuration computing resource.

3.3 Experiment: Computation Intensive Application

To do performance evaluation of a MapReduce application typed computation intensive, one configuration, `type-5`, is added to the configurations of this experiment. In this experiment, we use a parametric k-means algorithm with 100,000 data points, which is to organize these points into k clusters. We compare environments, a virtual cluster system and a physical cluster system, with application's performance metrics in terms of elapse time and standard variation.

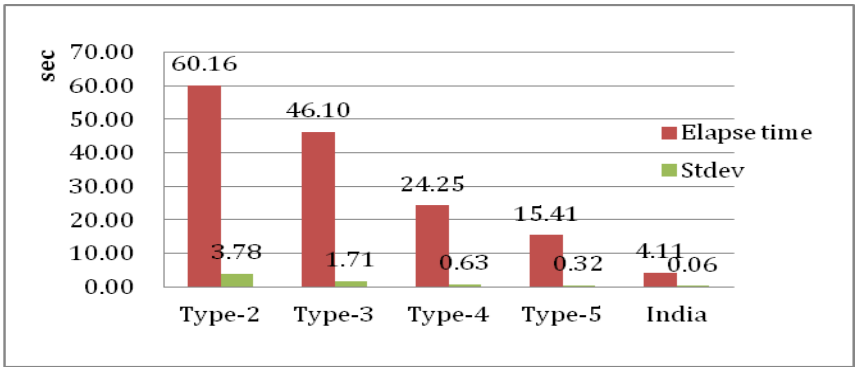

Fig. 4. Elapse time of k-means: 5 configurations - virtualized cluster systems (4 configurations) and a physical cluster system (1 configuration)

As shown in Figure 4, our experiments indicate that the average of elapse time increases over 375.5% in the virtualized cluster computing system, in comparison with a physical cluster system, represented by India. In Type-1, there is also the same starvation problem that is occurred as the data intensive application runs. Besides, the elapse time decreases proportionally as VM's CPU capability is added to the virtualized cluster computing system. Furthermore, the standard deviation is less affected by configuration change and the size of input data. In the physical cluster system, the value remains very low at about 1-2% of the variation of elapse time due to the capability of system mainly related with CPU power. In addition, the standard variation in the three configurations of the virtualized cluster computing system remains low at about 2.0-3.78%. A similar trend is observed in the values of standard deviation of all configurations. Hence we can expect that as the number of available VMs increases, there is a proportional improvement of elapse time.

3.4 Summary of the experiments

In summary, performance evaluation based on the metrics, load average and memory/swap area usage, according to the type of specific application is essential to choose properly a configuration that consists of a set of instances in the FutureGrid.

This observation induces that load balancing is helpful in spreading the load equally across the free nodes when a node is loaded above its threshold level. Though load balancing is not so significant in execution of a MapReduce algorithm, it becomes essential to handle large files in the case of having limited computing resources in the FutureGrid. As a highlight, it enhances hardware utilization in resource-critical situations with a slight improvement in performance. It is important to determine a configuration of a virtual cluster system in order to run a MapReduce application efficiently in the FutureGrid. Eventually we conclude that the appropriate selection of a set of VM instance types increases the overall utilization of resources in the FutureGrid. This approach is the way to identify the relationship between the type of applications and resources allocated for running them.

The result obtained from the experiments leads that it is important to determine a configuration of a virtual cluster system in order to run a MapReduce application efficiently in cloud computing. This approach is the way to identify the relationship between the type of applications and resources allocated for running them.

3.5 Observation of System Behavior with Anomaly

This section is mainly focused on the resource starvation. Figure 5 and Figure 6 show the snapshot of a virtualized system that has a resource shortage problem. A buffer value is used to measure the throughput of network I/O between a MapReduce application and a middleware to be used as message broker.

Figure 5 shows the buffer value is so low. The buffer value indicates how much of the memory in use. The buffer is currently being used for I/O buffering. Basically an I/O request happens when a Map task is finished and a Reduce task is started. Simultaneously there is no available cache memory in 7 seconds, while the memory usage is sharply high by up to 100 % as shown in the Figure 5. Hence system is still pending caused by little or no memory available. It can exacerbate failures.

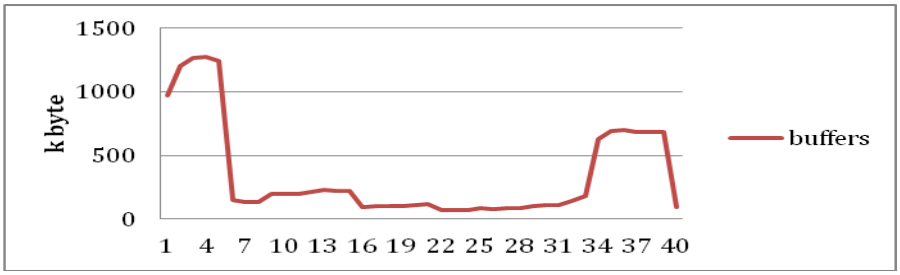

Fig. 5. Buffer variation in VM

We also observed that the variation of swap area is happened. Figure 6 shows the usage of swap area is sharply increased. It triggers a write burst to a disk and affects the response time of a MapReduce application. Because of the swapping, the system has slowed down, and heavy disk drive activity can be happened. There is still a small amount of free memory.

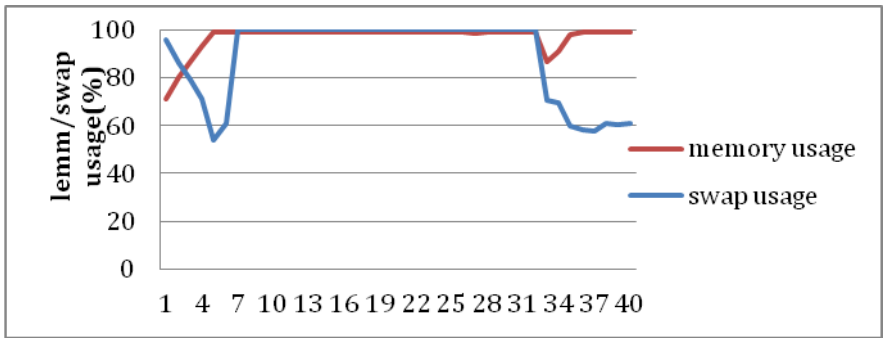

Fig. 6. Memory and swap variations in VM

We have observed an anomaly of system behavior as a data intensive MapReduce application runs in a virtualized cluster system that consists of VMs in the FutureGrid. It is caused by having no enough computing resources including CPU and memory capability to run the application and by having an inappropriate configuration of a set of VMs associated with a middleware setting.

4 Conclusion

Cloud computing is designed to provide on demand resources or services over the Internet. What we have observed in the experiments is that the overall performance of data intensive application is strongly affected by the throughput of the messaging middleware since it requires to transfer data when a map task sends the intermediate result to a reduce task. When it is close to limit of available memory as a data intensive MapReduce application runs on the specific configuration of nodes, the elapse time sharply increases and its standard deviation is high. The performance of the MapReduce application is so strongly affected by the configuration of VM. However the performance of computational intensive application is associated with CPU throughput. It is less affected by the configuration of VMs having the same CPU power. This work represents a first step towards understanding the relationship between the configuration of VMs and performance effects associated with the type of applications. The result of the experiments can be used for selecting the proper configuration based on the proposed guideline in cloud computing. It can be used to identify the bottleneck of a MapReduce application running on the resource given

VM configuration. It will be used to extend the information service system associated with the middleware for cloud computing.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. EECS Department, University of California, Berkeley (2009)
2. https://portal.futuregrid.org/
3. Dean, J., Ghemawat, S.: MapReduce: A Flexible Data Processing Tool. CACM 53, 72-77 (2010)
4. Morton, K., Friesen, A., Balazinska, M., Grossman, D.: Estimating the Progress of MapReduce Pipelines. IEEE 26th International Conference on Data Engineering (ICDE), 2010, pp. 681 - 684, Long Beach, CA (2010)
5. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. CACM 51, 107-113 (2008)
6. Wang, F., Qiu, J., Yang, J., Dong, B., Li, X., Li, Y.: Hadoop high availability through metadata replication. Proceeding of the first international workshop on Cloud data management, pp. 37-44, Hong Kong, China (2009)
7. Grossman, R., Gu, Y.: Data mining using high performance data clouds: experimental studies using sector and sphere. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 920-927. ACM, Las Vegas, Nevada, USA (2008)
8. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G.: Twister: A Runtime for Iterative MapReduce. The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010, (2010)
9. Geoffrey Fox and Shrideep Pallickara. Deploying the NaradaBrokering Substrate in Aiding Efficient Web & Grid Service Interactions. Invited paper for Special Issue of the Proceedings of the IEEE on Grid Computing. Vol 93, No 3, 564-577(2005).
10. MPI. *MPI(Message Passing Interface)*. Available: http://www-unix.mcs.anl.gov/mpi/
11. PVM. *PVM(Parallel Virtual Machine)*. Available: http://www.csm.ornl.gov/pvm/