Towards an Adaptive High-performance Execution of Scientific Applications in a Dynamic Cloud Environment

Mohamed-K HUSSEIN

Tabuk University, Saudia Arabia.

Faculty of Computers and Informatics, Suez Canal University, Egypt.

***Abstract*—** During the last decade, the needs for high-performance computing for distributed scientific applications have been addressed over multiple high-performance environments including clusters and Grid computing technologies. Recently, cloud computing technology offers cheap and large-scale high-performance computing environment. Infrastructure as a Service cloud (IaaS) offers instant access to large-scale computing resources. However, the performance of the resources can dynamically varies according to the changing load conditions on the resources. Further, scientific applications require complex communication/computation pattern, such as optimized MPI for communication. For these reasons, it is challenging to achieve high-performance in a cloud environment.

This paper presents an initial framework towards achieving adaptive high-performance execution for a distributed scientific application over a private dynamic cloud environment. The adaptation is achieved by migrating the distributed components of the benchmark application, which suffer performance degradation, to a promising different resource. The proposed framework contains a monitoring layer which monitors the execution times of the running application's components. A decision layer issues the migration decision considering the execution times and the cost of the migration. Finally, the paper presents the applicability of the proposed framework on a private IaaS cloud managed by Eucalyptus.

***Keywords*—** Scientific computing, Cloud computing, high-performance computing, MPI applications, Eucalyptus, Adaptive execution.

I. INTRODUCTION

Scientific applications are usually distributed large-scale compute intensive application. As a result, it has been executed using high-performance computing resources to reduce the computational complexity into a reasonable time, such as supercomputers [1]. However, such high-performance computing resources are expensive. As a result, the Grid computing technology has emerged to provide the scientists with cheap high-performance and large-scale computing resources through collaboration among multiple academic organizations [2-6]. However, there could be limitations between the hosting operating systems and the software requirements of the developed scientific applications. For example, scientific applications may require specific tools and APIs that have to be available during the runtime on the Grid's resources. For example, distributed scientific applications with complex communication/communication patterns during runtime require message-passing tools, such as parallel vector machines (PVM) and message passing interface (MPI), on the Grid's resources during the runtime. However, the required tools and APIs may not be available on the resources of the Grid where applications are scheduled for execution [7].

Cloud computing has emerged as the cutting edge IT technology to provide a flexible, on-demand elastic computing infrastructure [8, 9]. The cloud computing has gained its increasing popularity by the advancement of the virtualization technology. The virtualization creates different logical machines, called virtual machines (VM) images, to share the same hardware, and at the same time run isolated from each other. The isolated VMs may include different running operating systems (OS) and the user level software installed on an operating system. The Virtual Machine Monitor (VMM), called hypervisor, is a software layer mediates the access of the VMs to the physical resources, and allows the VMs to operate as if they were running on different machines independently [10]. As a result, a flexible control over the VM image can be obtained which is hard to have such feature with other high-performance infrastructure, such as the Grid and Clusters [11]. Popular virtualization hypervisors are VMware vSphere [12], XEN [13] and KVM [14].

The cloud management is a software layer on top of the virtual machine hypervisor VMM, in order to controls the VMs. This layer accesses the entire physical infrastructure, manages all the available virtual resources and delivers the virtual resources as a service over high-speed Internet. Cloud computing technology brings the potential for providing high-performance and the available supercomputing power to the public free or for small charges. There is a number of open source Cloud management layers, such as Eucalyptus [15, 16], Nimbus [17] and OpenNebula [18], which allow organizations and

individuals to build private clouds to have full control and to improve the utilization of the available computational resources.

The promising features offered by Infrastructure as a Service (IaaS) Cloud, such as on-demand access to large-scale computing resources and elasticity, has made a clear trend towards using the cloud in scientific computing [7, 19-21]. Many research has been conducted to study the performance of scientific application on cloud environments, especially in medical imaging, astronomy and physics [19]. However, to the best of our knowledge, none of these studies has considered the dynamic nature of the cloud where the executed application may suffer performance degradation during runtime because of the changing load conditions on the allocated resources.

This paper presents an initial design to a framework for an adaptive execution of distributed scientific applications on a dynamic cloud environment. The adaptive execution is achieved by terminating and checkpointing the distributed component of the application, which suffer performance degradation, and restarting a new instance on a promising available resource. A similar framework was designed on a Grid environment [22]. The framework is adopted for adaptive execution on a cloud environment, including cloud support for redistribution of the checkpointing files, resources monitoring, resource allocation and restarting the distributed components of the executed application on a different node.

The remainder of this paper is organised as follows. Section 2 presents the related work on cloud computing and experiences in executing scientific applications in cloud environments as well as adaptation in the cloud. Section 3 presents the proposed adaptive framework for executing a scientific application on a private cloud environment. Section 4 describes the proposed cloud environment, the experimental setup, and the experimental results. Finally, conclusions and future work are given in Section 5.

II. Background

Several research projects have been conducted to study the performance of scientific applications on cloud environments, especially in medical imaging [7], astronomy [20] and physics [21]. In [21], an analysis of the feasibility of executing a distributed multi-physics coupled models application on a private cloud environment is conducted. The communication pattern is employed using MPI and Open-MX as optimized runtime tools for communication. The performance analysis has shown that assigning multiple cores per VM can achieve a slight better execution times performance than execution times achieved using a standalone machine. Other research projects, such as Science Cloud [19] and Future Grid [20], have been conducted to execute scientific application on a cloud environment. Further, in [23], an evaluation of the impact of Xen on MPI distributed applications on a private cloud is conducted.

In [19], a framework has been developed for executing MPI scientific applications on a Cloud environment. The framework achieved high-performance by making the application elastic over the compute resources, i.e. increasing the number of instances of the components of the application whenever a performance degradation is discovered. This approach achieves the adaptation by load balancing the workload of the whole application over the available compute resources. Our proposed approach depends on discovering the compute resources which causes the degradation of the performance and migrating the components of the application to compute resources with less load conditions. The results have shown that the overhead of the communication is small and the adaptation is feasible. In [24], a mechanism has been presented for executing multithread OpenMP applications on a cloud environment. The proposed mechanism achieves the adaptation in terms of controlling the virtual machines (VMs), i.e increasing/decreasing the number of CPUs and amount of assigned memory to each VM, according to the number of threads in execution. This approach focuses on the dynamic changing requirement of the application during runtime. However, our proposed approach focuses on distributed iterative application. In this type of application, the response time for each iteration are approximately the same.

Many different approaches have targeted adaptation of server-based applications on cloud environment [25-27]. These mechanisms focus only on the unpredictable changing conditions of the workloads during runtime. They handle the degradation of the performance by increasing the number of the virtual machines which host the application components, and by load balancing the increasing workloads on the replicated virtual machines instances. Our approach is different by focusing in the adaptation of the long-running distributed scientific application where these applications are designed to execute on a specific number of compute resources, and cannot explore elasticity using replication mechanism.

III. The Proposed Adaptive Framework

This section starts with a detailed description of the benchmark application used in the study, then a presentation of the cloud architecture used for executing the application. Finally a detailed description of the proposed adaptive framework.

A. The Benchmark Application Description

The benchmark application used in this study is the Multi-physics coupled model application [21]. The importance of the coupled model application comes from the outstanding advantage of simulating complex scientific phenomena with an advanced accuracy in

different areas, such as climate, space weather, solid rockets, fluid structure interaction, heart disease and cancer studies [28]. Coupling together different models of individual systems, which affect the system of interest, provides an accurate simulation for phenomena under study [22]. These types of application are large-scale, long running which would require high-performance execution environment and computation time in order of weeks if not months to produce the results. In computational terms, coupled models can be viewed as distributed component-based applications in which each individual model becomes a software component. In the general coupling framework, reported in [1, 9], for example, each individual model embodies three phases of operation, namely initialization, iteration and termination, as shown in Figure 1. The models are synchronized via exchange of information in a series of put() and get() calls to the run-time architecture during each iteration. For each message transfer, the sender model executes put() with a suitable data structure, and the receiver model receives the communicated data by executing a corresponding get(). The data sent between the models is termed coupling data. Each individual model may act at a different length or time scale, or focus on a distinct underlying physical phenomenon.

```
Input: Total_timestep, curr_timestep
Output: solution
//Initialization phase
{start MPI communication }
//start iterations
Loop
        get_coupled_data()
        {main timestep computation}
        Put_coupled_data()
{terminate MPI communication}
//termination phase
```

Figure 1. The structure of distributed coupled model application.

Each model repeatedly perform one round of get()s, then do some work based on the received coupling data, then perform one round of put()s. A single cycle of this iteration is known as a minor cycle and is equivalent to one time-step in the underlying iterative model. But, different scientific models are allowed to iterate at different rates and transformer models are then required in order to reconcile these rates. For example, if there are two models coupled together, and one model executes n rounds of put()s for every round of get()s in the second model, an intermediate n-for-1 transformer model is required to perform the necessary reconciliation [21]. An n-for-1 transformer thus performs n minor cycles during which it performs n rounds of get()s before performing one round of put()s. A 1-for-n transformer performs n minor cycles during which it performs one get() group followed by n put() groups. The resulting cycle of the entire coupled model is termed a major cycle. For example, the HybridMD coupled model is an interesting modelling and simulating complex fluids that handles molecular dynamics (MD) coupled to computational fluid dynamics (CFD) within a single simulation. The main goal is to study fluid flow over a surface [6]. The interaction of fluid molecules with the surface is modelled atomistically using classical molecular dynamics, while the fluid bulk is modelled using a continuum method. Each model tackles a separate physical region of the system. The data exchange between models provides the boundary conditions for each region (e.g. temperature and pressure). Considering the interaction between the hydrodynamics and the specific molecular processes near the surface clarifies the properties of these physical systems and addresses key problems, such as how hydrodynamics effects influence molecular interactions at interfaces, at lipid bilayer membranes and within individual macromolecules or assemblies of them. The HybridMD coupled model consists of two models (CFD and MD) and two transformers, as shown in Figure 2Figure 2. MD iterates at three times the frequency of CFD. Transformer1 is a 1-for-3 transformer that is used to reconcile the communication between CFD and MD. Transformer2 is a 3-for-1 transformer that reconciles the communication between MD and CFD. One major cycle of the entire coupled model involves three minor cycles of the MD model and one minor cycle of the CFD model.

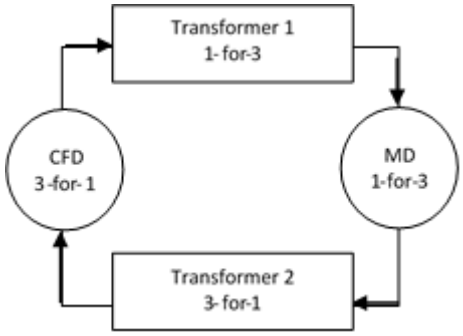


Figure 2. Data flow between the component models for the HybridMD coupled model.

The architecture of the benchmark application has two main advantages for adaptive execution in a cloud environment using migration. First, it is easy to monitor the execution performance through measuring the response time for each timestep. The average response time gives a good indication of performance degradation, it increases more than a specified threshold if the load on the compute node increases. Second, it is easy to checkpoint the execution progress at the end of each timestep. Hence, the distributed component can restart the execution on a different compute node from the checkpoint file.

B. The Private Cloud Infrastructure

Eucalyptus is an open source software which implements Infrastructure as a service (IaaS) Cloud [15]. The IaaS infrastructure allows the end user to flexibly execute distributed scientific applications over the allocated resources by employing parallel runtimes over the accessed VMs images. The main advantage of Eucalyptus is that it is compatible with commercial cloud products such as Amazon EC2 and S3 [15]. This compatibility enables to run a scientific application on a private cloud using Eucalyptus and a public cloud using Amazon without modification in execution framework or the application. Eucalyptus architecture consists of a number of components, namely Cloud Controller, Node Controller, Cluster Controller, Storage Controller, and Walrus, as shown in Figure 3.

The Cloud Controller (CLC) is responsible for managing the available virtual resources, such as Servers, network and storage. The Node Controller (NC) runs on each physical machine (PM), and controls the available virtual machines. The Cluster Controller (CC) collects information on the installed virtual machines and schedules the VMs for execution on the NC. For data storage service, the Storage Controller (SC) which manages storage block volumes by communicating between NC and CC. Also, Walrus is a file-based storage service for the VMs images and users data [16].

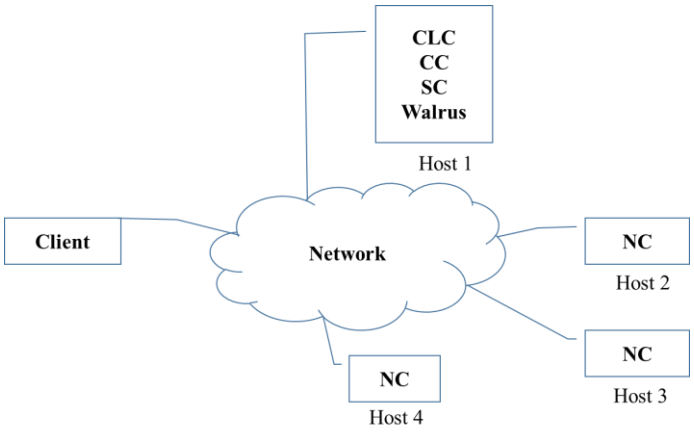


Figure 3. A private Cloud infrastructure using Eucalyptus.

On each physical machine Xen is used as a VMM hypervisor to create the virtual machine of a desired configuration. Xen is a popular open source software for VMM which is used to obtain high performance of multi-cores physical machines [13]. Several research has been conducted to study the performance impact of Xen on MPI applications [21, 23]. These research has proved that Xen can be used to bring higher performance especially when several cores are assigned to single virtual machine. OpenMPI is used as the MPI implementation. Further, in the proposed cloud infrastructure, Open-MX is used to reduce the high latency of MPI communication over Ethernet networks using TCP [29].

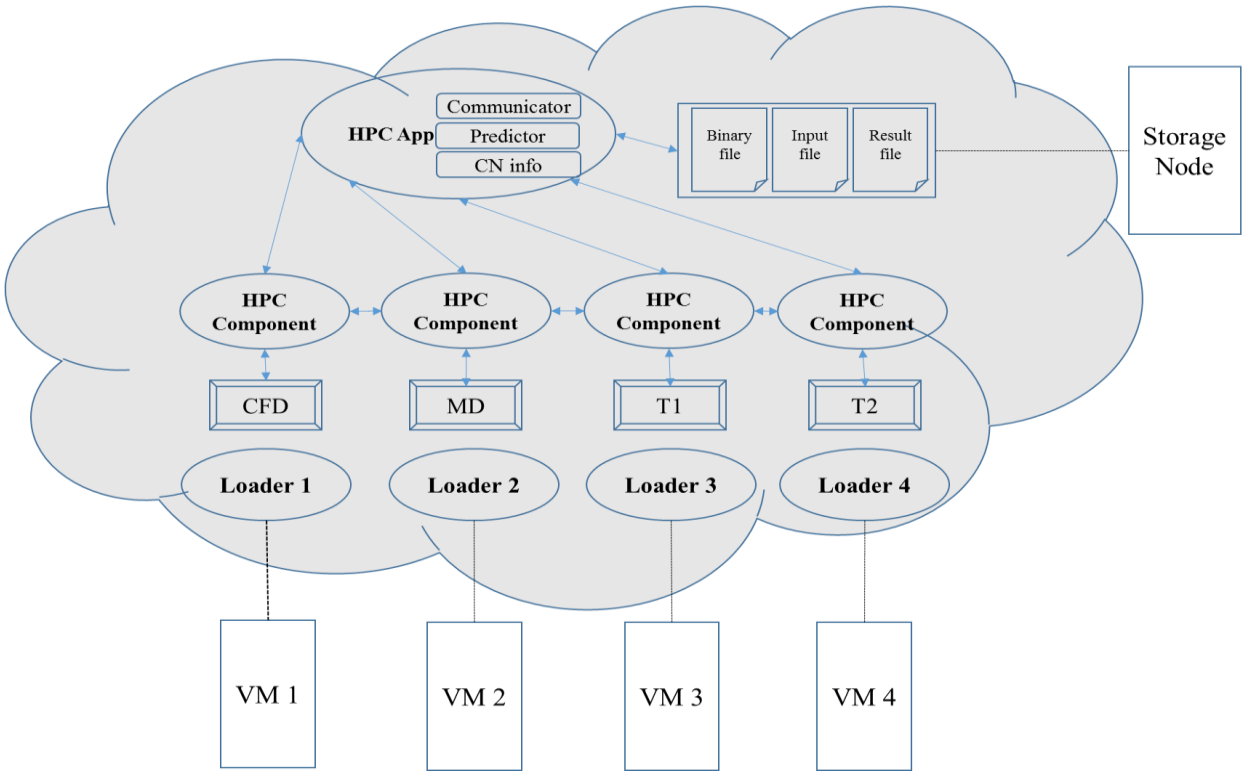


Figure 4. The Proposed adaptive high-performance Execution framework.

C. The Proposed adaptive High-performance Execution framework

Figure 4 shows the overall architecture of the proposed framework. The framework is adopted from an architecture for high performance execution of distributed scientific application on a grid environment [22]. The framework achieves adaptive performance through feedback control through three main layers.

The first layer is the high-performance control of the whole application (HPC App) which consists of three main components, namely communicator, predictor, and compute node information (CN Info). The communicator interacts with the HPC Components. The predictor is mainly a decision layer based on the average response time for each component of the application and the current load on the compute node as a measure of the execution progress. Once the predictor discover a performance degradation of the application, it issues a migration decision. The communicator sends a migration decision to HPC component to stop execution, checkpoint the progress and redeploy the distributed components of the application on the other available compute nodes.

The high-performance of each distributed components (HPC Component) is a software wrapper for each component. Its main responsibilities are checkpointing whenever it is necessary, especially when a migration decision is issued. Further, HPC component handles the MPI communication between the distributed components. Finally, it restarts the migrated component from the checkpoint file.

The loader are responsible for starting the components for run when the application starts, or restarting a component on a new compute node when a migration decision is issued. Further, the loader transfer the checkpoint files to the desired compute node.

The framework is deployed on the Eucalyptus. On each compute node a virtual machine where the each loader and a distributed component of the application along with the HPC Component are deployed. The HPC App is deployed on another compute node. The migration decision is based on the average response time for each component, $ave\_respone_i$, and the remaining number of iterations, $remain\_iter_i$. The migration decision is issued using the following Equation.

$$\max_i \; ave\_response_i * remain\_iter_i > thresold$$

The threshold value is used to control the migration decision. The threshold value is set taking into account the cost of the checkpointing, *checkpointcost*, and restarting the communication, *commcost*, as well as the expected response time on the new deployment configuration, as shown in the following Equation.

$$thresold = checkpoint_{cost} + comm_{cost} + response\_time(new\ deployment)$$

IV. Experimental Setup and Evaluation

This section evaluates applicability and feasibility of the proposed adaptive execution of distributed framework using the distributed benchmark application described in Section A.

The experiments are conducted using four physical machines, each machine has i7 core Intel 2.2 GHz processor and 32GB memory. The virtualization layer is based on Xen hypervisor version 4.3. The VMs are deployed using Eucalyptus version 4. Each node runs Ubuntu version 12 operating system. OpenMPI version 1.5 along with Open-MX version 1.4 are used as MPI. 1 Gigabit Ethernet network fabric is used for networking. On Each compute node, a virtual machine is deployed and assigned 4 CPUs and 16 GB memory.

Table 1.

The deployment of the adaptive framework on the Cloud resources.

Compute Node Number	deployment
1	HPC APP + HPC Components(transformers)
2	HPC Component(CFD model)
3	HPC Component(MD model)
4	No deployment

An initial deployment of each distributed component is set on the resources of the set private cloud infrastructure, as shown in Table 1. Figure 5 shows the execution performance of the MD model on the deployment configuration shown in Table 1. As shown in the figure, after timestep 120 of the execution of the MD model, the load on compute node 3 is increased by running several instances of a matrix multiplication program. After timestep 170, the distributed component of the MD model managed to stop its execution, checkpoint its progress and restart execution at compute node 4.

Figure 6 shows the overhead cost of the migration in seconds. The major overhead is in restarting the MPI communication over the new deployment. The overhead is relatively large. However, with the actual long-running scientific distributed application, this overhead can be tolerated for in the large scale execution time..

Figure 5. The response times of the MD model in milliseconds).

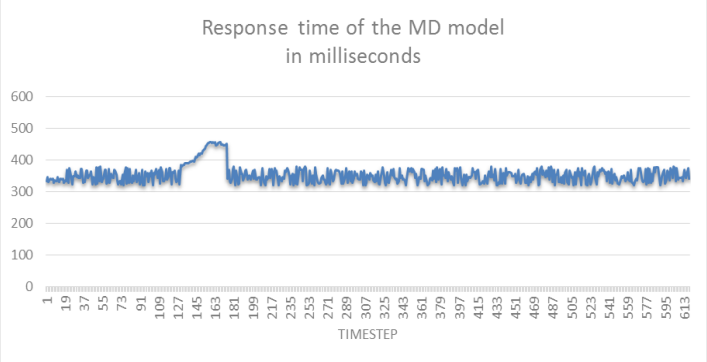
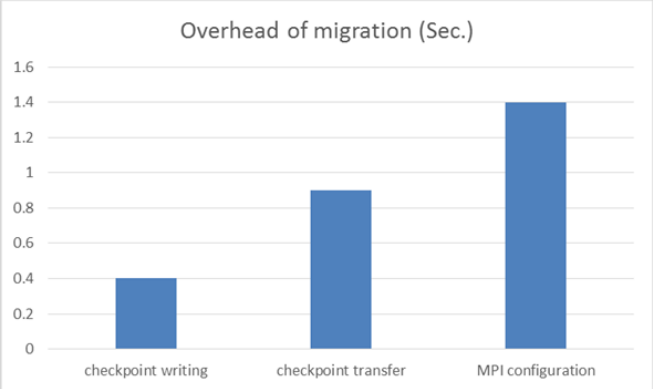


Figure 6. The overhead cost of the migration.



V. Conclusions and Future Work

This paper has presented the initial design of a framework towards achieving an adaptive performance execution for a distributed scientific MPI application over a private dynamic Cloud environment. The adaptation is based on migrating the distributed components of the application, which suffer performance degradation, to a promising different resource. The proposed framework contains a monitoring layer which monitors the execution times of the running application's components. A decision layer issues the migration decision considering the execution times and the cost of the migration. Finally, the evaluation using a distributed scientific coupled model application presents the feasibility of the proposed adaptive framework on a private Cloud by Eucalyptus Cloud.

In this paper, the threshold value is set statically based on the experience of the overhead cost of the migration and restarting the MPI communication. The future work will focus on the dynamic placement of the threshold value. Further, a simple and accurate predictor is required to predict the response times of the remaining timesteps based on load conditions of the allocated resources.

References

[1] 1. Jha, S., et al., Understanding Scientific Applications for Cloud Environments, in Cloud Computing. 2011, John Wiley & Sons, Inc. p. 345-371.

[2] 2. Foster, I. and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure. 2003: Morgan Kaufmann Publishers Inc.

[3] 3. Coveney, P.V., et al., Scientific Grid Computing: The First Generation. Computing in Science and Engg., 2005. 7(5): p. 24-32.

[4] 4. A grid-enabled web service for low-resolution crystal structure refinement. Acta Crystallographica Section D Biological Crystallography, 2012. 68(3): p. 261.

[5] 5. Pordes, R., et al., New science on the Open Science Grid. Journal of Physics: Conference Series, 2008. 125(1): p. 012070.

[6] 6. Pordes, R., t.O.S.G.E. Board, and J. Weichel, Analysis of the current use, benefit, and value of the Open Science Grid. Journal of Physics: Conference Series, 2010. 219(6): p. 062024.

[7] 7. Vecchiola, C., S. Pandey, and R. Buyya, High-Performance Cloud Computing: A View of Scientific Applications, in Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks. 2009, IEEE Computer Society. p. 4-16.

[8] 8. Armbrust, M., et al., A view of cloud computing. Commun. ACM, 2010. 53(4): p. 50-58.

[9] 9. Buyya, R., et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 2009. 25(6): p. 599-616.

[10] 10. Uhlig, R., et al., Intel Virtualization Technology. Computer, 2005. 38(5): p. 48-56.

[11] 11. Ekanayake, J. and G. Fox, High Performance Parallel Computing with Clouds and Cloud Technologies, in Cloud Computing, D. Avresky, et al., Editors. 2010, Springer Berlin Heidelberg. p. 20-38.

[12] 12. Burd, S.D., et al. Virtual Computing Laboratories Using VMware Lab Manager. in System Sciences (HICSS), 2011 44th Hawaii International Conference on. 2011.

[13] 13. Barham, P., et al., Xen and the art of virtualization. SIGOPS Oper. Syst. Rev., 2003. 37(5): p. 164-177.

[14] 14. Childers, B., Virtualization shootout: VMware server vs. VirutalBox vs. KVM. Linux J., 2009. 2009(187): p. 12.

[15] 15. Nurmi, D., et al., The Eucalyptus Open-Source Cloud-Computing System, in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. 2009, IEEE Computer Society. p. 124-131.

[16] 16. Kijsipongse, E. and S. Vannarat, Autonomic resource provisioning in rocks clusters using Eucalyptus cloud computing, in Proceedings of the International Conference on Management of Emergent Digital EcoSystems. 2010, ACM: Bangkok, Thailand. p. 61-66.

[17] 17. Tudoran, R., et al., A performance evaluation of Azure and Nimbus clouds for scientific applications, in Proceedings of the 2nd International Workshop on Cloud Computing Platforms. 2012, ACM: Bern, Switzerland. p. 1-6.

[18] 18. Sempolinski, P. and D. Thain, A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus, in Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science. 2010, IEEE Computer Society. p. 417-426.

[19] 19. Raveendran, A., T. Bicer, and G. Agrawal. A Framework for Elastic Execution of Existing MPI Programs. in Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on. 2011.

[20] 20. Deelman, E., et al., The cost of doing science on the cloud: the Montage example, in Proceedings of the 2008 ACM/IEEE conference on Supercomputing. 2008, IEEE Press: Austin, Texas. p. 1-12.

[21] 21. HUSSEIN, M.-K. and M.-H. MOUSA, High-performance Execution of Scientific Multi-Physics Coupled Applications in a Private Cloud. International Journal of

Advanced Research in Computer Science and Software Engineering, 2014. 4(2): p. 11-16.

[22] 22. Hussein, M., et al., Adaptive performance control for distributed scientific coupled models, in Proceedings of the 21st annual international conference on Supercomputing. 2007, ACM: Seattle, Washington. p. 274-283.

[23] 23. Youseff, L., et al., Evaluating the Performance Impact of Xen on MPI and Process Execution For HPC Systems, in Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing. 2006, IEEE Computer Society. p. 1.

[24] 24. Galante, G. and L.C.E. Bona, Supporting Elasticity in OpenMP Applications, in Proceedings of the 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2014, IEEE Computer Society. p. 188-195.

[25] 25. Chieu, T.C., et al., Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment, in Proceedings of the 2009 IEEE International Conference on e-Business Engineering. 2009, IEEE Computer Society. p. 281-286.

[26] 26. Galante, G. and L.C.E. de Bona. A Survey on Cloud Computing Elasticity. in Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on. 2012.

[27] 27. Roy, N., A. Dubey, and A. Gokhale, Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting, in Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing. 2011, IEEE Computer Society. p. 500-507.

[28] 28. Murugavel, S.S., S.S. Vadhiyar, and R.S. Nanjundiah, Adaptive Executions of Multi-Physics Coupled Applications on Batch Grids. J. Grid Comput., 2011. 9(4): p. 455-478.

[29] 29. Goglin, B., High-performance message-passing over generic Ethernet hardware with Open-MX. Parallel Comput., 2011. 37(2): p. 85-100.