Spring 2-17-2020

Analysis of Cloud Bursting on Openstack Infrastructure to AWS

Bao Pham
bhpham@my.harrisburgu.edu

Ronald C. Jones
Harrisburg University of Science and Technology

Majid Shaalan
Harrisburg University of Science and Technology

Analysis of Cloud Bursting on Openstack Infrastructure to AWS

Bao Pham
Openstack & Student
Computer Information Sciences
Harrisburg University of Science and Technology
Harrisburg, Pennsylvania, USA
BHPham@my.harrisburgu.edu

Ronald C. Jones
Faculty
Computer Information Sciences
Harrisburg University of Science and Technology
Harrisburg, Pennsylvania, USA
rcjones@harrisburgu.edu

Dr. Majid Shaalan
Professor & Program Lead
Computer Information Sciences
Harrisburg University of Science and Technology
Harrisburg, Pennsylvania, USA
mshaalan@harrisburgu.edu

***Abstract*—Cloud computing is the development of distributed and parallel computing that seeks to provide a new model of business computing by automating services and efficiently storing proprietary data. Cloud bursting is one of the cloud computing techniques that adopts the hybrid cloud model which seeks to expand the resources of a private cloud through the integration with a public cloud infrastructure. In this paper, the viability of cloud bursting is experimented and an attempt to integrate AWS EC2 onto an Openstack cloud environment using the Openstack OMNI driver is conducted.**

Keywords— (OpenStack, VM, infrastructure, private cloud, public cloud, cloud burst, VM, container)

I. Introduction

Cloud computing is the development of distributed and parallel computing that seeks to provide a new model of business computing by automating services and efficiently storing proprietary data [1]. In simpler terms, cloud computing is a scalable on-demand configurable resources computation model. It provides many types of infrastructure in an ad hoc system where everything provided to the end-users exists as a utility service over the Internet. The term cloud is an analogy to describes the web as a place where applications are pre-installed and exist as a service [1]. A service can be data, virtual machine (VM), storage, or software that is ready to be shared on the web [1].

Cloud bursting is a cloud computing technique that seeks the expansion of a private cloud (internal data centers) infrastructure through the integration with a public cloud infrastructure [3][4]. The public cloud resources are provisioned when the local resources have reached a certain threshold to meet their demand. The extra workloads are transferred to a public cloud where the enterprise is renting. There are issues that hinder cloud bursting from being adopted widely as a solution for high availability and scalability [4]. One issue is the delay time in the synchronization of an application and its data being offload to the public cloud when the threshold has been reached in the private cloud. The duration of copying the disk image of VM or its volume can be long. In addition, issues arise when moving VM to the public infrastructure that utilizes a different hypervisor than that of the private cloud [5].

II. Background

Cloud computing enables businesses with the ability to provide instantaneous services to the end-users with a fraction of the cost [3]. With such benefits, many enterprises host their products as cloud services by renting on a public cloud platform such as Amazon Web Service (AWS) and Microsoft Azure. Depending on the configuration of their application, enterprises can either deploy their application as a container or host it on a VM. A container is a lightweight machine that operates on top of a physical server and its host operating system (OS) kernel whereas a VM exists an emulation of a computer system and requires its own OS [6]. For hosting applications that require the entire resource of the OS and the functionality of many other applications, VMs are a better choice. Meanwhile, containers are a great choice for deploying the same application many times due its ability of self-replication [6]. However, in recent years, containers have become a better choice of application deployment as it is quicker to be redeployed than a VM [6].

In recent years, it has become evident that outsourcing the entire IT infrastructure to third parties won't be applicable in many cases [3]. Enterprise applications are often faced strict requirements in terms of performance, delay, and uptime. In addition, legal issues can arise since public clouds are distributed anywhere on the planet making it difficult

to rely solely on a virtual public interface. Despite that, the great computation resources provided by a public cloud platform is appealing to the enterprises. Furthermore, renting on a public cloud infrastructure exclude enterprises from capital expenditure on hosting their own infrastructure which allows them to focus solely on the maintenance of their application [3]. However, if an enterprise encountered issues on the public platform, it won't be the first to received help as there are many other enterprises on the platform. Hence, hosting one's own infrastructure and having the ability to harness additional resources from a public infrastructure during workload peak is beneficial. This allows the enterprise to avoid legal issues and have on-time maintenance while utilizing the resources from a third party as a last measure [3].

III. RELATED WORK

There have been recent efforts in analyzing the effectiveness of cloud bursting and the hybrid cloud model.

Bharti [1] et al. provided a list of cloud computing platforms with hybrid cloud integration capability. In addition, they discussed ongoing issues with cloud computing such as privacy, legality, reliability and security.

In their newly proposed bursting method that exploits nested virtualization and advanced networking, technologies, Acs [2] et al. discussed the requirement for different API-s for services that IaaS clouds provide. Their experiment shown that seamless cloud bursting increases deployment time by 5-10% when migrating a collection of VMs.

Buyya [3] et al. presented the integration of the AWS environment and Aneka Cloud. Their analysis on how the hybrid model handles the sporadic demand on IT infrastructure in an enterprise. Their comprehensive evaluation concluded that by leasing the public cloud environment concluded that leasing a public cloud environment could bring more economical benefits if compared to buying and maintaining a single new server. They found out that smaller tasks size lead to more wastage when trying to maintain the queue time comparable to average task duration, when the workload trace is scaled down by different factors and explore the behavior of the policies as the average task size changes. However, this can only be achievable through proper provisioning policy and scheduling algorithm.

Fishman [5] et al. developed a virtualization platform for IaaS clouds to deploy existing VMs without any modifications to the mobility between private and public clouds, and easy duplication throughout the entire deployment.

Celest [6] explored the performance of containerization on IoT devices on IoT cloud. The overhead produced from hypervisor meditation is eliminated as containerization virtualizes on top of the OS-level instead of requiring a hypervisor. This enables application to run near-native performance. As result, they saw great response time between multiple Raspberry Pi to a targeted server.

IV. SETUP

The Openstack platform is a set of software tools for building and managing cloud computing environments. These tools are united under one MySQL Database for communication and storing metadata. In simpler terms, Openstack serves as an API that relay these tools to create the cloud environment.

An Openstack environment was configured on one Supermicro blade that contains 140 GB SSD, 48 GB of RAM, and an Intel ® Xeon ® E5640 2.67 GHz CPU with four cores. The Openstack OMNI driver is built for Openstack Liberty, which is deprecated, Ubuntu 14.04.06 LTS was installed on the server to configure the deprecated version of Openstack. The Openstack environment contains only the necessary components required by the driver which included Nova, Glance, and Neutron. Nova is the component that provides the provisioning and the management of VMs. Neutron is the component that delivers networking-as-a-service in the compute environment (VMs networks). Glance is the imaging service that allows the discovering, registering, and retrieving of VM images. The OMNI driver utilizes these three components to integrate the Openstack environment with the public cloud environment. The modified Nova component is responsible for snapshotting the VM onto the public cloud when the Openstack environment has reached its threshold. The threshold is set at when either CPU, RAM, or HD capacity has reached 25% and it is also dependent on the requirement of the unexpected load.

The EC2 component of the driver was moved into the folder of each Openstack component. Each component configuration file was modified to calling the OMNI EC2 driver. In addition, the AWS secret key and access key were passed onto each config files to allow the driver to establish the connection with AWS. A neutron network with a subnet of 16 was created as it is the largest mask that is allowed by AWS. The ip-address allocation pool for VMs was set from 11.11.1.4 – 11.11.1.254 due to AWS has reserved ips from x.x.x.0 to x.x.x.3. CentOS 7 was used as the main image for the VMs since it is supported on AWS.

V. Approach

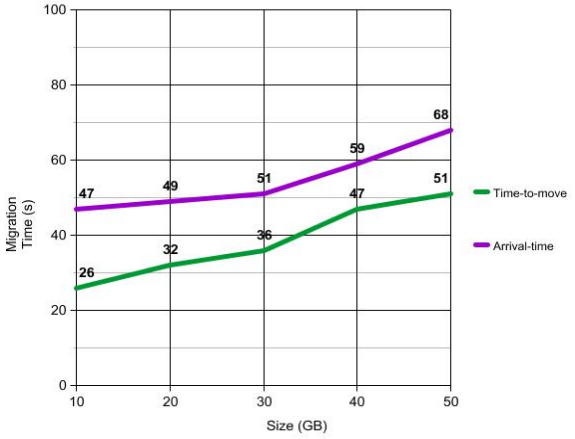
Three methods are used to measure the time to snapshot the VMs onto the AWS infrastructure. The first method measures the migration time of one VM with at different storage sizes during the peak of the cloud. The first method seeks to measure the correlation between the size of VM and its migration time. The second method measures the migration time of moving multiple VMs at 20 GB during the peak of the cloud. The second method focuses on the synchronization of the VM network and its effects on the migration of multiple VMs. Lastly, the third method experimented on the migration of four VM at various size simultaneously. Each VM is running a simple neural network calculating over one million data entries to simulate a working VM.

Two primarily attributes, time to move (TTM) and Arrived Time (ART), are recorded to measure the time of the migration. TTM describes the amount of time it takes for the VM to be prepared for the migration. This time incorporates the time of copying the content of the VM, pausing it, and saving its state. The ART measures the time it takes for the VM to be deployed on the public infrastructure and the time it takes to become active again. It is the total time of TTM and the deployment time.

VI. Finding

VM	SIZE (GB)	TTM	ART
1	10	26 s	47 s
2	20	32 s	49 s
3	30	36 s	51 s
4	40	47 s	59 s
5	50	51 s	68 s

Table 1. First Method

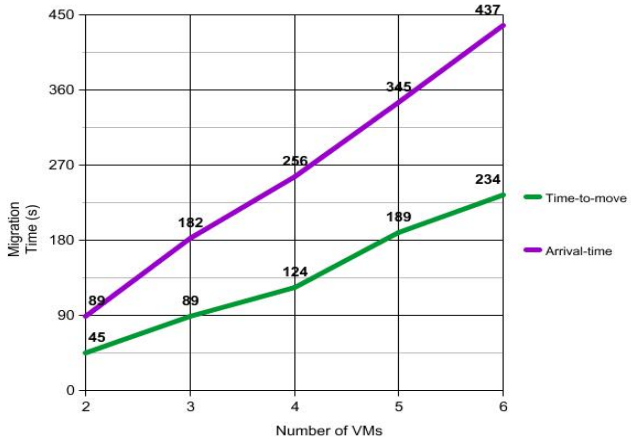


Graph 1. Migration of One VM at Various Size

Table (1) shown that in the first method, as the size of the VM increases the longer it takes to move the VM. In addition to the overall size, the contents within the VM can heavily affect the TTM as applications running inside of the VM required for the VM to temporarily pause the applications and store their metadata. The VM and its contents resume their operations after they're moved to the external site (AWS). The metadata must be properly stored and transferred along with the VM to the new site. This process is crucial for the VM to resume its operations at the new designated site from where it was paused. Our experiments showed that as the number of applications running inside increases, the more preparation is needed to package the metadata and the VM variables, the more overhead latency time adds up, and TTM and ART rise significantly.

# of VM	SIZE (GB)	TTM	ART	Avg. TTM	Avg. ART
2	20	45 s	89 s	22.5 s	44.5 s
3	20	89 s	182 s	29.67 s	60.67 s
4	20	124 s	256 s	31 s	64 s
5	20	189 s	345 s	37.8 s	69 s
6	20	234 s	437 s	39 s	72.83 s

Table 2. Second Method



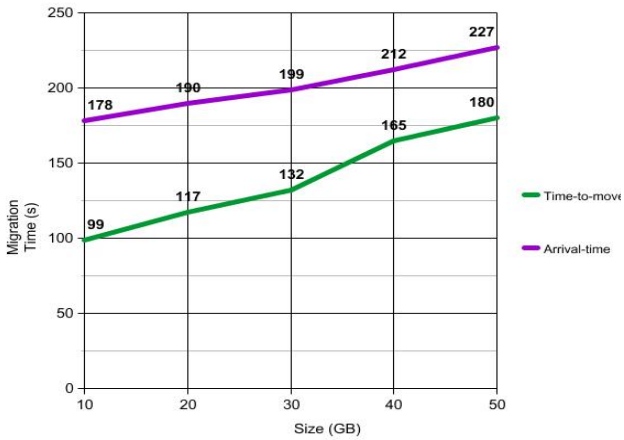
Graph 2. Multiple VMs Migration at Fixed Size

The second method seeks to analyze the effects of network synchronization during the migration phase. The average TTM and average ART are calculated to depict the TTM and ART of one VM during the migration of multiple VMs. As the number of VMs increases during the migration, the data shown that the ART of one VM is greater contrast to the migration of just one VM. This increase in time complexity is due to the need for applications within the VMs to

reconfigure and synchronize the networking parameters on both the internal and external cloud environments to facilitate an efficient migration.

# of VM	SIZE (GB)	TTM	ART	Avg. TTM	Avg. ART
4	10	99 s	178 s	24.75 s	44.5 s
4	20	117	190 s	29.25 s	47.5 s
4	30	132 s	199 s	33 s	49.75 s
4	40	165 s	212 s	41.25 s	53 s
4	50	180 s	227 s	45 s	56.75 s

Table 3. Third Method



Graph 3. Migration of 4 VMs at Various Size

The third method affirms the effects of migrating multiple VMs at different sizes contrast to the migration of different number of VMs at one fixed size. Graph (3) shown that the ART and TTM are significantly lower when moving four VMs at different sizes. Meanwhile, graph (2) shown that the ART is affected greatly as the number of VMs increases during the snapshot. In addition, the TTM rate of change in graph (2) is significantly greater than graph (3).

These findings depict the synchronization issue of both networking environments and its local running applications. In addition, the migration of multiple complex VMs adds another level of complexity resulting in a longer delay in the resuming the VMs' operations. We believe that, in general these latency issues will remain problematic to any hybrid cloud environment, unless there is a direct, high-performance, low-latency interconnection infrastructure between the two cloud models involved in the migration process. Both graph (2) and graph (3) depict these findings.

VII. Result

We report results of the experiment conducted on a hybrid environment built from the integration of Openstack and Amazon EC2 environments.

In particular, graph (1) plots the TTM and ART of migration of one VM at various sizes to show impact of varying the size of a VM. Graph (2) contrasts the data in graph (1) by illustrating that there is greater impact on the TTM and ART when migrating large number of VMs. The average ART shown in graph (2) affirms that the ART of migrating one VM during the migration of a cluster of VMs increases heavily.

In addition, the last method confirmed that the variation in the size of individual VM does not heavily impact during the moving of multiple VMs. The synchronization of the VMs is affected greatly when migrating multiple VMs. The delay is introduced when the Openstack environment tries to establish its connection to the Amazon API. Once the request is made, the Openstack prepare its targeted VMs for migration. When the VMs are ready to be moved, the private environment make another request to the EC2 API while forwarding the VMs files. The OMNI module is responsible for reconfiguring the networking of the VM in its files. Once the files are forwarded, the module requests the EC2 to create the VMs. When the VMs are created, the bursting module requests EC2 to forward their information back to the private environment, which allow it to create entries in the nova console allowing the user to know the VMs are operational.

Furthermore, we should note that the performance of the bursting module is not reliable. The instance of Openstack is outdated along with the OMNI module while EC2 is heavily updated. We updated the code in the module and replaced outdated code libraries with newer Openstack libraries. In addition, we updated the Neuron and Nova components of Openstack to use the last updated version in Liberty. The experiments were conducted many times to collect substantial data because the snapshotting of VMs was often prone to failure. This is due to the bursting module losing its established connection to the EC2 API when EC2 failed to build the VMs because of bad network configuration of its files.

VIII. Conclusion

We have presented the findings on the synchronization issue of cloud bursting. When migrating many VMs, the time to synchronize is heavily impacted in contrast to the migration of a small number of VMs at various sizes. Despite that, cloud bursting is a great application for smaller infrastructure to scale out by integrating with public

infrastructures. However, it should be used as the last resort when dealing with high peaks. The cloud bursting model can be enhanced through optimal scheduling algorithms which could result in better synchronization time and lower delay between each separate migration. Thus, during high peaks, different migrations will not overlap each other and not result in a drop in the connection with EC2 API.

IX. Future Work

These findings are still not substantial to fully depict the issue. The collected data are only on VMs running small scale applications to mimic a working infrastructure. Furthermore, the private environment and its bursting module are outdated, where the chance of migration failure is very high. A newer bursting module is required for the newer release of Openstack environment which can properly establish communication with EC2 API and minimize the migration failure.

References

[1] Bharti, Drsantosh & Goudar, R. (2012). Cloud Computing–Research Issues, Challenges, Architecture, Platforms and Applications: A Survey. International Journal of Future Computer and Communication. 10.7763/IJFCC. 2012.V1.95.

[2] Acs, S., Kozlovszky, M., & Kacsuk, P. (2014). A Novel Cloud Bursting Technique. *9th IEEE International Symposium on Applied Computational Intelligence and Informatics*. doi: 10.1109/SACI.2014.6840050

[3] Buyya, R., Garg, S. K., Mattess, M., & Vecchiola, C. (2011). Cloud Bursting: Managing Peak Loads by Leasing Public Cloud Services. Retrieved from http://www.buyya.com/~raj/papers/CloudBurst-BC-2011.pdf

[4] Cloud definitions. NIST Special Publication 800-146. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf

[5] Fishman, A., Rapoport, M., Budilovsky, E., & Eidus, I. (2013). HVX: Virtualizing the Cloud. Retrieved from https://www.usenix.org/system/files/conference/hotcloud13/hotcloud13-fishman.pdf

[6] Celest, A., Mulfari, D., Fazio, M., Villari, M., & Puliafito, A. (2016). Exploring Container Virtualization in IoT Clouds. 2016 IEEE International Conference on Smart Computing (SMARTCOMP). doi: 10.1109/SMARTCOMP.2016.7501691