Modeling the Virtual Machine Launching Overhead under Fermicloud

Hao Wu*§, Shangping Ren*, Gabriele Garzoglio†, Steven Timm†, Gerard Bernabeu†,Seo-Young Noh‡

***Abstract*—FermiCloud is a private cloud developed by the Fermi National Accelerator Laboratory for scientific workflows. The *Cloud Bursting* module of the FermiCloud enables the FermiCloud, when more computational resources are needed, to automatically launch virtual machines to available resources such as public clouds. One of the main challenges in developing the cloud bursting module is to decide *when* and *where* to launch a VM so that all resources are most effectively and efficiently utilized and the system performance is optimized.**

However, based on FermiCloud's system operational data, the VM launching overhead is not a constant. It varies with physical resource (CPU, memory, I/O device) utilization at the time when a VM is launched. Hence, to make judicious decisions as to *when* and *where* a VM should be launched, a VM launch overhead reference model is needed. The paper is to develop a VM launch overhead reference model based on operational data we have obtained on FermiCloud and uses the reference model to guide the cloud bursting process.

I. Introduction

Cloud technology has been benefiting general purpose computing for quite some years. The *pay-on-demand* model brought by cloud computing allows companies to avoid over provision at its early project development stage. As the cloud technology develops, many scientific research institutions have migrated their research from traditional grid and distributed computing platform to the cloud computing environment. These research areas include life science [13], astronomy [15] and earthquake research [10], to name a few.

One successful example of using cloud computing technology is the STAR project on Relativistic Heavy-Ion Collider at the Brookhaven National Laboratory [1], [2]. The STAR project studies the fundamental properties of nuclear matter which only exist in a high-density state called a Quark Gluon Plasma [1]. Because of resource shortage from the local grid service, the STAR team started to collaborate with the Nimbus team at Argonne National Laboratory to migrate its experiment to a computer cloud. The Nimbus tools enable virtual machines in private cloud to be deployed on Amazon EC2.

One of the advantages a computer cloud has over traditional grid computing is that the resource utilization of the underlying infrastructure can be significantly improved by deploying different tasks on the same physical computer node. In addition, computation power can also be dynamically allocated to tasks when more resources are needed by the tasks. The other benefit of using a computer cloud over a grid is that a cloud has "unlimited" resources – when the private cloud is fully occupied, cloud bursting techniques can temporarily acquire external resources from, for instance, public clouds to fulfill the need.

Fermi National Accelerator Laboratory (Fermilab), as a leading research institution in the high energy physics (HEP) field, started to build a private computer cloud, the FermiCloud, in 2010. The FermiCloud has successfully served the HEP experiments since its establishment. The cloud bursting tool vCluster [8] is developed to automatically allocate resources for scientific workflows from both FermiCloud and public clouds such as Amazon EC2. However, how to dynamically allocate resources for the scientific workflows that reduces the average response time of scientific workflows as well as entire system's operational cost is a research and also an engineering challenge yet to be overcomed.

The resource allocation problem in cloud computing started to draw more attention in the research community in recent years [11], [6]. However, most of the research in the resource allocation area assumes that the virtual machine launching overhead is negligible. As a result of this assumption, neither the launching overhead nor the dependency between the overhead and resource utilization are taken into consideration in designing their resource allocation algorithms. However, our production line operation data indicates that the virtual machine launching overhead can have significant variations.

The VM launching overhead has two aspects, i.e. (1) it consumes system resources while it is launched and (2) it takes time to complete the process of the launch. Both of these types of overhead can impact the system's performance. More specifically, VM launching consumes a significant amount of CPU and disk IO resources, leading to a high system CPU and disk IO utilizations at the time of launching. If each host computer in the private cloud happens to launch a new virtual machine at the same time, due to high system utilization caused by VM creations, the computer cloud may consider all the hosts fully occupied and decides to perform cloud bursting and create the virtual machine on an external public cloud. Such additional cost is unnecessarily rendered and could be prevented if we have a VM launch overhead reference model.

Furthermore, if a task requires an additional virtual machine in order to complete its work, but the virtual machine takes a much longer time to complete its launching than expected, it is possible that the task has already finished its

*Illinois Institute of Technology,10 W 31st street, 013, Chicago, IL, USA, {hwu28, ren}@iit.edu. The research is supported in part by NSF under grant number CAREER 0746643 and CNS 1018731.

†Fermi National Accelerator Laboratory, Batavia, IL, USA, {garzogli,timm,gerard1}@fnal.gov

‡National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, Korea, rsyoung@kisti.re.kr

§Hao Wu works as an intern in Fermi National Accelerator Laboratory, Batavia, IL, USA

work before the virtual machine is ready for executing the task. This again leads to resource waste and an added cost due to without a VM launching overhead reference model.

In this paper, we are to (1) study the VM launching overhead behavior based on real operational data obtained from FermiCloud; (2) develop a reference model for virtual machine launching overhead from both timing and utilization perspectives; and (3) evaluate the accuracy of the developed reference model.

The rest of the paper is organized as follows: Section II discusses related work. Section III analyzes the virtual machine launching overhead through a large amount of experiments on FermiCloud. Section IV presents a reference model for virtual machine launching overhead. Section V evaluates the accuracy of the proposed model. We conclude the work in section VI.

II. Related Work

Lots of researches have been done on evaluating the cloud performance and modeling cloud. One of the most influencing cloud modeling tool is CloudSim [6] developed by CLOUDS lab from the University of Melbourne. The CloudSim is a java based cloud simulation tool that supports modeling and simulation of large scale cloud computing environments. It provides a cloud modeling that models cloud infrastructure physical machines', and virtual machines' characteristics and behaviors, a cloud market modeling that models the cost of resources, a network modeling that models the network behavior of inter-networking of clouds, a cloud federation modeling that models the communication between clouds, a power consumption modeling that models the power consumptions in the datacenter, and a resource allocation modeling that models virtual machine allocation policies. The CloudSIm provides a relative comprehensive modeling tool that covers almost all the basic elements under a cloud environment.

Recently, Huber *et al.*'s work evaluated the virtualization performance and proposed a virtualization overhead model [9]. In their work, they mainly focus on two virtualization platforms, XenServer and VMware ESX. They test the performance downgrades that is brought by the virtualization. They test the CPU, memory, disk IO, and network performance degradations on both XenServer and VMware ESX platforms. Based on the experiments, they categorized the virtualization performance influencing factors into four major categories: virtualization type, hypervisor's architecture, resource management configuration and workload profile. However, Huber's model does not consider virtual machine launching overhead, it only provides the computation overhead that is brought by the virtualization.

Researchers adapted the above cloud models and cloud simulations tools and proposed significant contributions to resource allocation on clouds, such as resources provisioning algorithms from QoS perspective [7], from service providers' profit perspective [14] and from energy consumption perspective [5]. Recently, Mengxia Zhu *et al.* proposed a cost effective scheduling for scientific workflow under cloud environments [11]. Their scheduling algorithm aims to shorten the application's response time and reduce the energy consumption simultaneously by considering the virtual machine launching overhead.

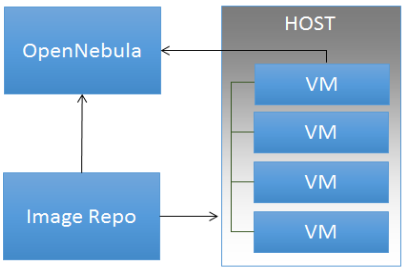

Fig. 1: System Architecture

However, they did not consider the variation of the virtual machine launching overhead. Not only Zhu's work, including CloudSim, few other researchers have taken virtual machine launching overhead variation as an key variable for designing resource allocation algorithms. However, the virtual launching overhead may have a large variation that may cause significant impact on the resource allocation process. In the FermiCloud bursting project, the design of the resource allocation algorithm aims to automatically allocate resources for the scientific workflows that need extra computational resources. If the virtual machine launching overhead variation is not well modeled and calculated, the system utilization and efficiency may be pulled down dramatically. Furthermore, it may cause resource and energy waste. Hence, we need an accurate mathematical model for the virtual machine launching overhead. The reference model we propose in the paper is drawn from a large amount of experimental observations. The formal analysis of the experiments is discussed in the next section.

III. Actual VM Launching Overhead on FermiCloud

In this section, we study the patterns of the virtual machine launching overhead based on the virtual machine operations in the FermiCloud production cloud environment.

A. FermiCloud System Configuration

The FermiCloud uses OpenNebula [3], [12] as its cloud platform. As illustrated in Fig. 1, the system has an OpenNebula front-end server that manages the entire cloud infrastructure, an image repository that stores all VM images, and a set of host machines on which VMs are deployed.

The OpenNebula front end server has 16-core Intel(R) Xeon(R) CPU E5640 @ 2.67GHz, 48GB memory. Fifteen homogeneous hosts are used for the experiments. All the fifteen hosts are configured with 8-core Intel(R) Xeon(R) CPU X5355 @ 2.66GHz and 16GB memory. All these machines are connected through high speed Ethernet.

Under OpenNebula [4], the VM launching process consists of four major states. Fig. 2 illustrates the state change during a VM launching process in OpenNebula [4]. In particular, when a user creates a new VM, the VM enters the *pending* state. In the pending state, the cloud scheduler decides where to deploy the VM. Once the VM has been deployed on a specific host, it enters into the *prolog* state in which all VM related files (images in our case) are transferred from the image repository to the host machine. After all the files are copied to the host, the VM enters the *boot* state, during which it is booted from the host. Finally, after the VM is successfully booted, it enters

into the *running* state. Once a VM is in its running state, it is ready to execute tasks.

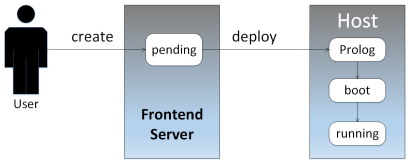

Fig. 2: VM Launching State Diagram[4]

B. Base VM Launching Overhead

We first obtain the baseline utilization overhead of launching a new VM. In order to get the baseline utilization overhead of launching a new VM, we let all the host machines in the private cloud be empty, i.e. have no application being deployed, before launching a VM. Each time, a single VM is launched. All the launched VMs are configured with one virtual core and 2 GB memory. We retrieve the exact virtual machine launch time from each virtual machine's system log. The experiment is repeated ten times.

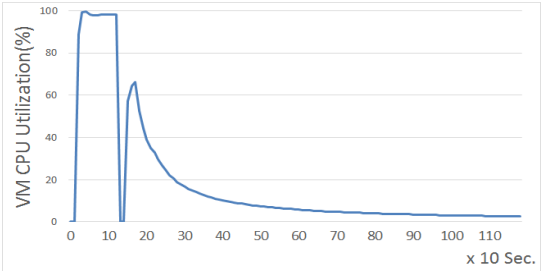

Fig. 3: VM Launching CPU Utilization Overhead

Fig. 3 shows the average system CPU utilization variation in the process of launching a VM. The x-axis represents the time instance of sampling points. The sampling interval is every 10 seconds and it is used for all the other experiments. The y-axis indicates the host machine's CPU utilization consumed by the process of a single virtual machine. For convenience, throughout the paper, we refer the CPU utilization consumed by a VM on a host machine as the VM's CPU utilization.

Since the host machine consists of multiple CPU cores, the VM's CPU utilization represents a single CPU utilization consumption by the VM's process. If the VM's CPU utilization exceeds 100%, it means the VM occupies more than one CPU cores.

As shown in Fig. 3, there are two different CPU utilization variation trends. The first part, from time 0 to time 14, is due to the *prolog* procedure, which fully consume a CPU until the image is copied to the host. The second part, from time 15 to time 120, is due to the *booting* procedure. Once the booting procedure starts, it immediately reaches a high CPU utilization and the CPU utilization slowly decreases after the services are started. When VM's CPU utilization remains close to constant, the VM is considered to be in *running* state. We denote the VM's CPU utilization variation trends in Fig. 3 as the baseline VM launching CPU overhead and use it as the base for comparisons in following experiments.

C. CPU Utilization Impact

The above experimental data indicates that VM launching overhead causes system CPU utilization to change on an empty machine. In reality, most of the VMs are not launched on empty hosts. Thus, it is interesting to see how the system utilization influences the VM launching process. The following sets of data are obtained to investigate the influence of system utilization on the VM launch overhead.

1) Different system utilization: In this experiment, VMs are launched under different system utilizations. Fig. 4 depicts the system CPU utilization change when VMs are launched under different system CPU utilizations. It indicates that every time a new VM is launched, the system utilization is suddenly increased to a high level and remains at the level for a while before it goes down.

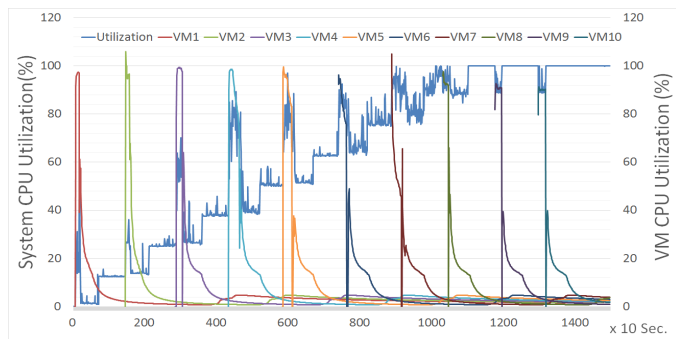

Fig. 4: VM Launching Overhead under Different System Utilization

Fig. 5 shows the booting utilization variations for different VMs. As indicated in the figure, the variations converges to the same value. In fact, the variation of the booting process is at most 10% while the peak utilization of booting a VM has a variation of 40%.

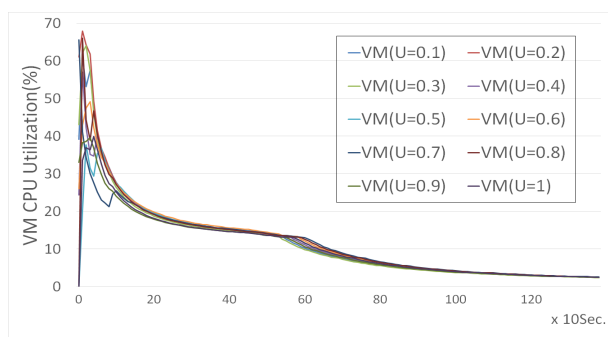

Fig. 5: VM Booting Overhead

To clearly distinguish the VM launching overhead change from the system utilization change, we extract individual VM launching overheads from Fig. 4 and depict the results in Fig. 6. Fig. 6 clearly demonstrates that the time of the VM *prolog* process changes quite significantly when launching VMs under different system utilizations.

Table I summarizes the variations. "Util" column indicates the system's CPU utilization when a new virtual machine is created; the column of "Prolog" represents the time increases for the prolog process when it is compared with the baseline virtual machine prolog time; the column of "Boot" represents

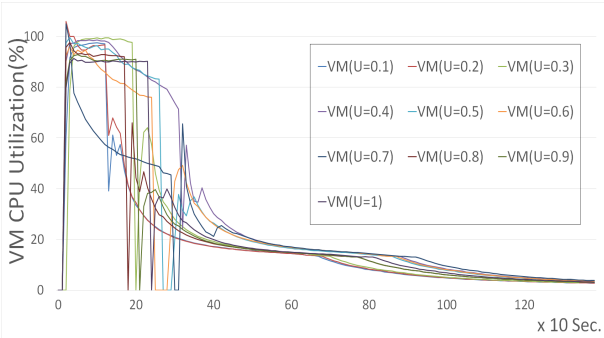

Fig. 6: VM Launching Overhead Comparison

Util (CPU)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Prolog	0	0	0.38	0.61	0.69	0.84	1.23	1.23	1.46	1.46
Boot	0	0	0	0.03	0.05	0	0.12	0.03	0.01	0.03
Peak Util.	0	0.10	0.02	-0.05	-0.38	-0.19	0.07	0.07	-0.35	-0.34

TABLE I: VM Launching Overhead Comparison

the time increases for the booting process compared with the baseline virtual machine booting time; and the "Peak Util." column represents the VM's peak CPU utilization increases for the booting process compared with the baseline virtual machine's booting CPU peak utilization.

As table I indicates, when the system's CPU utilization reaches 100%, launching a new VM takes 1.5 times compared with launching a VM on an empty host. However, if the VM booting process is isolated out, surprisingly, all the booting processes take similar amount of time no matter how high the system utilization is as shown in Fig. 5.

2) Different VM configuration: For previous experiments, all VMs have the same configuration. To know how the configurations of the VMs may impact the VM launching overhead, we repeat the above experiments but with different VM configurations (2 virtual cores and 4 GB memory). Table II lists the results of the experiments. The incremental times are compared with the baseline VMs from table I.

Intuitively, the VM prolog time will not change much as the same VM image is used for the VMs and the only changes are the number of CPU cores and the size of memory. The results confirm that the prolog times remain the same trends as the baseline VMs. Furthermore, without a surprise, the VM booting processes also take the same amount of time as the single core VMs do. Hence, we can conclude that the VM configurations do not have significant impact on the VM launching overheads.

D. Disk IO Utilization Impact

The baseline experiments show that the VM launching overhead consists of two parts, one is the image transferring and copying process and the other is the VM booting process. From the above CPU utilization experiments, we have learned

Util (CPU)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Prolog	0	0	0.15	0.23	0.30	0.38	0.84	1.69	1.69	1.76
Boot	-0.03	-0.01	0	0	0.05	0	0.07	0.05	0.07	0.05
Peak Util.	0.24	0.26	0.02	0.06	-0.04	0.07	-0.04	-0.14	-0.31	-0.31

TABLE II: VM Launching Overhead Comparison under Different VM Configurations

that the VM booting overhead does not change much when the system utilization changes. However, it is possible that during the image copying process, the overhead may be influenced by the disk IO operations and network traffic. We discuss each below.

1) Launching overhead under different IO utilization: Since disk IO operations also consume CPU resources. In order to focus on the impact of disk IO utilization variations, we keep the system's CPU utilization as low as possible. As illustrated in Fig. 7, even when the IO utilization reaches 100%, the CPU utilization still remains at a relatively lower level (less than 20%). Similar to the VM's CPU utilization, we refer VM's IO utilization as the host machine's disk IO utilization consumed by the single VM and system's IO utilization as the host machine's total utilization.

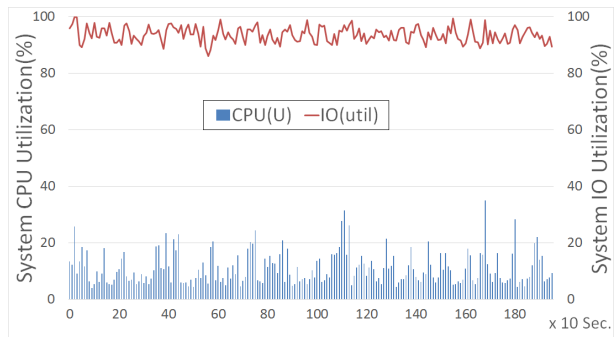

Fig. 7: IO Utilization v.s. CPU Utilization

We use the same VM configuration as used for the baseline experiments. VMs are launched under different system disk IO utilization and the results are shown in Fig. 8. As Fig. 8 indicates the VM launching overhead has large variations when VM starts under different disk IO utilizations. If we isolate the VM booting overhead, as shown in Fig. 9, we can clearly notice that the VM booting overhead under different IO utilizations has significant changes when the disk IO utilization changes. The VM's booting time is almost doubled when launched under fully IO utilized situation when it is compared to the one launched under an idle host.

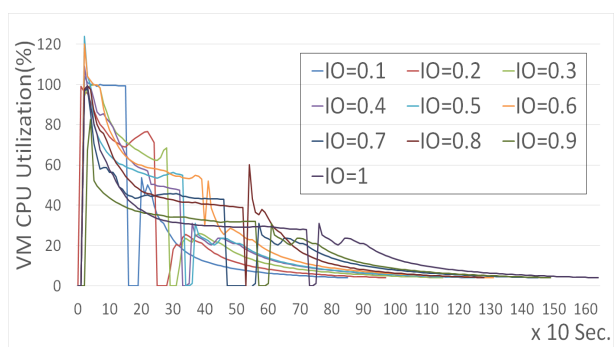

Fig. 8: VM Launch Overhead Comparison Under Different IO Utilization

Table III gives the detailed VM launching overhead increments under different disk IO utilizations in comparison with the baseline overhead. In particular, when compared with the baseline launching overhead, the prolog process takes a much longer time to copy images to the host machine when the host machine's disk IO utilization is high. When the host machine's

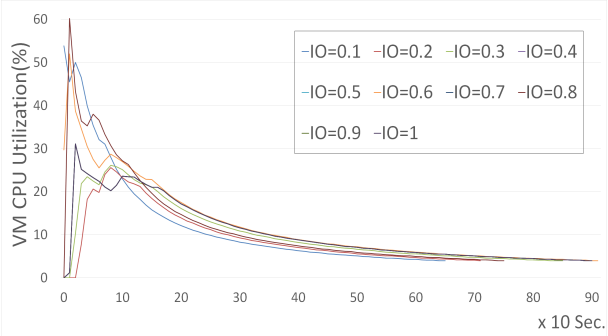

Fig. 9: VM Booting Overhead Comparison Under Different IO Utilization

Util(io)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Prolog	0.61	1.07	1.31	1.61	1.69	2.07	2.54	3.23	3.53	4.69
Boot	0.03	0.06	0.12	0.20	0.32	0.40	0.41	0.41	0.78	1.09
Peak Util.	-0.10	-0.57	-0.56	-0.48	-0.40	-0.13	-0.34	0.03	-0.31	-0.38

TABLE III: VM Launching Overhead Comparison under Different IO Utilization

disk IO utilization reaches 100%, it take almost 5 times to copy an image compared with copying an image to idle host machines. Notice that the VM booting processes also take a longer time (almost twice as much) when the host machine has frequent disk IO operations. An interesting finding is that the peak utilization caused by the booting process is much lower when it is compared with the baseline overhead when the host disk IO utilization is high.

2) Simultaneous Launching Overhead: Above experiments give the insight of how host machine's disk IO operations can impact the VM launching overhead. This set of experiments is to investigate if there are mutual influences on launching overhead among the different VMs when multiple VMs are simultaneously launched to the same host machine. Under the same system disk IO utilization, we launch two and later multiple VMs simultaneously and deploy them on the same host machine under different IO utilization. Fig. 10 depicts the results. The data clearly indicates that the *prolog* time for the VM is prolonged significantly (700%) when more than one VMs are launched at the same time.

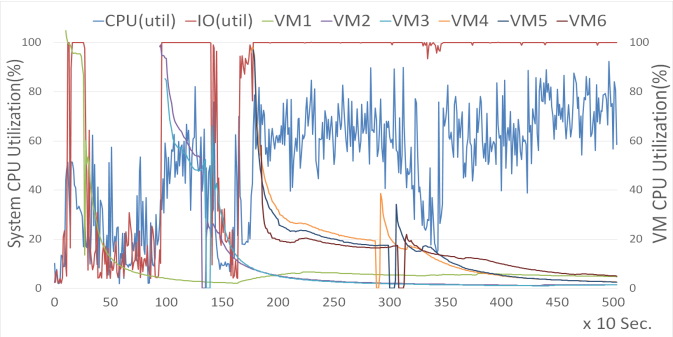

Fig. 10: System Utilization Variation On Simultaneous Launching

Figure 11 illustrates the individual VM launching overhead when multiple VMs are started simultaneously. It is interesting to see that when multiple VMs are launched, the system evenly distributes CPU resources to each VM for the *prolog* processes. The peak utilization of the VM booting process

No. VMs	2VMs(U=0)	3VMs(U=0)	2VMs(U=1)	3VMs(U=1)
Prolog	2.15	6.92	3.07	7.84
Boot	0.21	0.80	0.70	0.88
Peak Util.	-0.20	-0.36	-0.40	0.08

TABLE IV: Simultaneous VM Launching Overhead Comparison under Different IO Utilization

also decreases proportionally to the reduction of the number of simultaneously launched VMs.

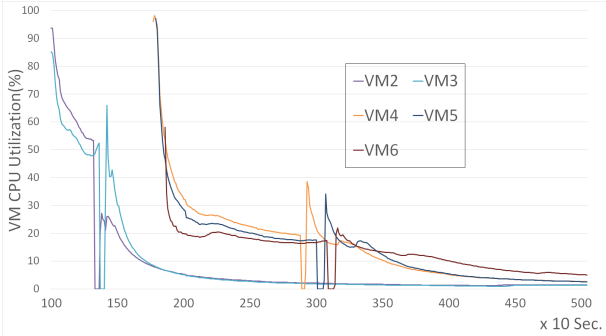

Fig. 11: Simultaneous VM Launching Overhead Comparison

As depicted in Fig. 11, the VM *prolog* process time increases as the system disk IO utilization increases. It is because the *prolog* process not only competes with the other newly launched VMs, it also competes with other VMs that are running. Table IV shows the statistic comparisons of simultaneous VM launching overheads under different system IO utilizations. The first row of the table indicates the number of VMs created simultaneously and the system IO utilization on which these VMs are deployed. As shown in the table, when the system disk IO is idle, simultaneously launching two VMs takes twice the time to copy the images to the host compared with the baseline copying process; and seven times the time to transfer an image to the host when three VMs are launched in the meantime. When the disk IO is fully utilized, the time of copying an image is three times as much for two simultaneous launches and eight times as much for three simultaneous launches compared with the baseline image transferring process. While the booting time for each VM is also increased when the disk IO is fully utilized, the booting time increase is rather much slower compared with the prolog process — it is only 1.9 times as much compared to the baseline booting time.

E. Network Traffic Impact

As discussed above, the image copying process may also be influenced by the network traffic. In this section, we discuss the impact of network traffic on the VM launching overhead. We consider two scenarios for the experiments, the impact of downstream and upstream bandwidth utilization on the VM launching overhead, respectively.

1) Network downstream bandwidth Impact: We first test the influences when the downstream bandwidth is utilized for other running VMs on the hosts. Intuitively, the downstream bandwidth will affect the image transferring time. If the available spare bandwidth for the newly launched VM is relatively small, the bandwidth then will become the bottleneck for launching VMs. However, after VM images are copied to the host, the booting process will not be affected.

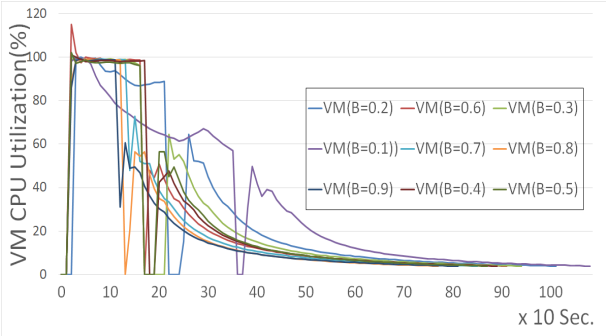

Fig. 12: VM Launching Overhead Comparison Under Different Network Downstream Bandwidth

Bandwidth (down)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Prolog	2.16	1.08	0.66	0.83	0.66	0.50	0.16	0.16	0
Boot	0	0.10	0.04	0.05	-0.02	0.02	-0.08	0	0.02
Peak Util.	0.07	0.075	-0.20	-0.05	-0.15	-0.05	0.21	-0.05	0.01

TABLE V: VM Launching Overhead Comparison under Downstream Bandwidth

The overall VM launching overhead comparison is dispatched in Fig. 12. It is not difficult to see that when the downstream bandwidth is highly utilized, the prolog process of launching a VM is increased. However, without a surprise, all the VM booting processes take almost the same amount of time as indicated in Fig. 13.

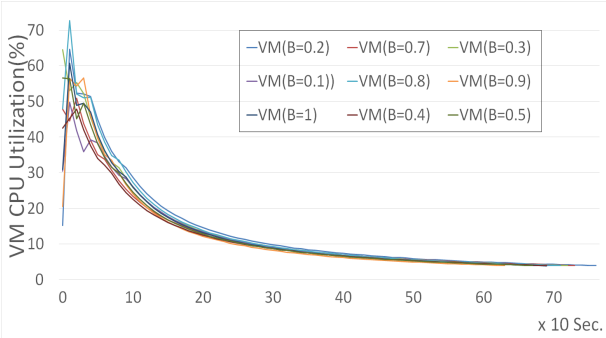

Fig. 13: VM Booting Overhead Comparison Under Different Network Downstream Bandwidth

Furthermore, as shown in table V, even when the bandwidth is 90% utilized, the prolog time is only twice of the baseline prolog time; and if the bandwidth utilization is low, the prolog time decreases quickly and remains at a steady level when the utilization reaches 0.3. The reason for such prolog time variation is that the total network downstream bandwidth is very large compared with the disk IO bandwidth. When the spare bandwidth available for transferring an image becomes larger than the available disk IO bandwidth, the disk IO bandwidth becomes the bottleneck. Hence, the minimum available network downstream bandwidth and disk IO bandwidth decides the image transferring overhead.

2) Network upstream bandwidth Impact: Evaluating the impact of upstream bandwidth utilization on the VM launching overhead takes the same steps as for the downstream bandwidth limitations. Intuitively, the upstream bandwidth utilization does not have significant impact on the VM launching process and our data confirms this.

As shown in Fig. 14 and Fig. 15, almost all the VMs' overheads match with each others'. However, there is one

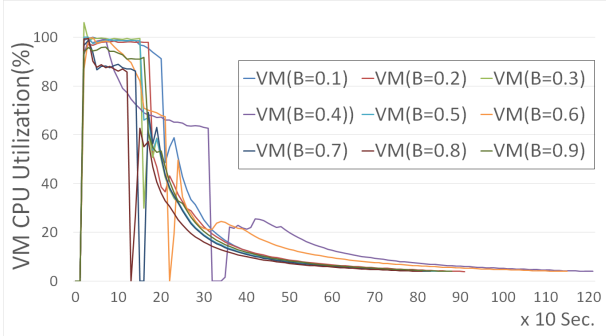

Fig. 14: VM Launching Overhead Comparison Under Different Network Upstream Bandwidth

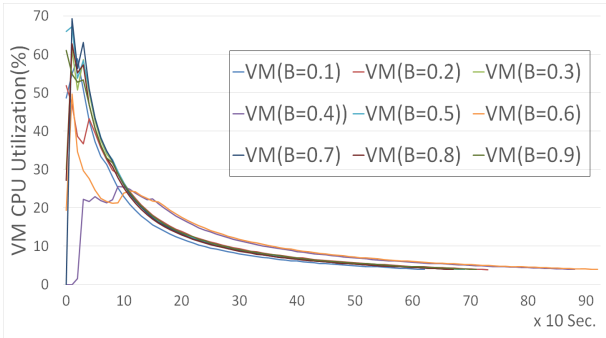

Fig. 15: VM Booting Overhead Comparison Under Different Network Upstream Bandwidth

exception. The VM launched under 40% upstream bandwidth utilization takes an extremely long time for the entire launching process. Our further investigation of the system log indicates that at the time when the VM is launched, the host machine happens to have IO operations for some system critical services.

F. Image Repository Impact

The FermiCloud architecture as shown in Fig. 1 contains an image repository which can also become a bottleneck when large number of VMs are launched simultaneously even when they are deployed on different hosts. In order to evaluate the impact of sudden large number of simultaneous launches on the VM launch overhead, we set up another experiment using the baseline VM configuration. In particular, we launch a VM to a host, and simultaneously launch more VMs to different hosts. Fig. 16 illustrates the overall VM launching overheads when different number of VMs are launched simultaneously. It is obvious that when more VMs are launched simultaneously, the image repository's disk IO/network bandwidth become a bottleneck. In particular, when seven VMs are launched simultaneously, the prolog time increases 3.5 times compared with the baseline prolog time.

Notice that the CPU utilization for the prolog process becomes lower when a VM is launched with more VMs simultaneously. This is because when multiple VMs are launched

Bandwidth(down)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Prolog	0.61	0.38	0.23	1.53	0.23	0.76	0.23	0.07	0.31
Boot	-0.10	0.05	-0.01	0.27	0	0.32	-0.02	-0.02	0.028
Peak Util.	-0.02	0.03	0.016	-0.55	0.11	-0.18	0.15	0.03	0.016

TABLE VI: VM Launching Overhead Comparison under Upstream Bandwidth

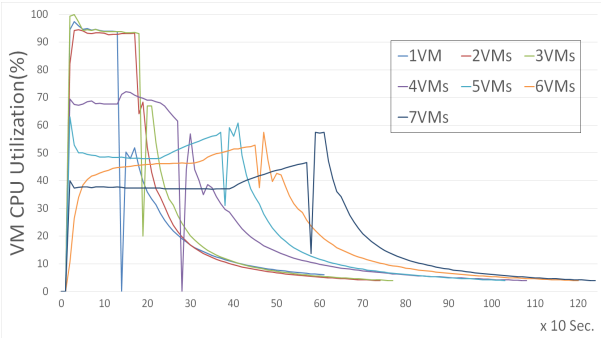

Fig. 16: VM Launching Overhead Under Different Simultaneous Launches on Different Hosts

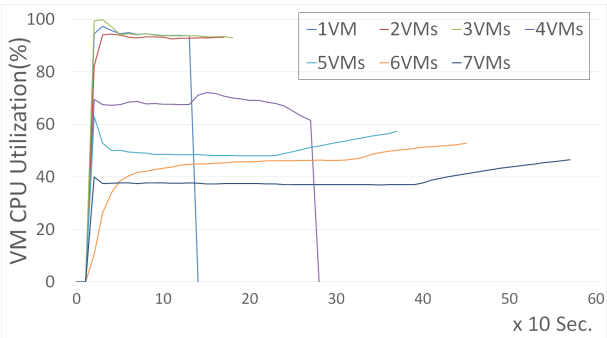

Fig. 17: VM Prolog Overhead Under Different Simultaneous Launches on Different Hosts

together, the disk IO bandwidth/network bandwidth of image repository is evenly distributed among each of them. For each machine, its prolog process does not fully occupy the disk IO utilization, hence the CPU utilization for the prolog process becomes lower.

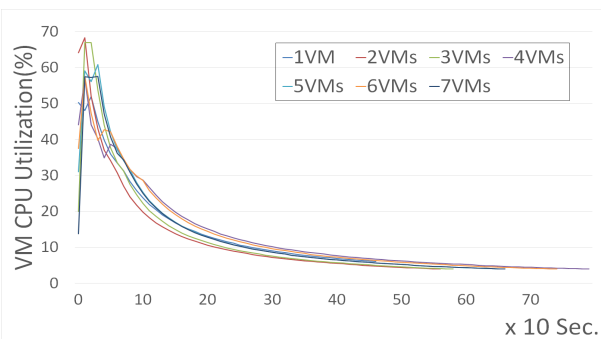

Fig. 18: VM Booting Overhead Under Different Simultaneous Launches on Different Hosts

As shown in Fig. 18 and table VII, the overhead for the VM booting processes remains at the same level as the baseline VM booting overhead.

G. Summary

From these experiments, we can conclude the following:

Simul. Launches	1VM	2VMs	3VMs	4VMs	5VMs	6VMs	7VMs
Prolog	0.15	0.38	0.46	1.23	1.92	2.53	3.46
Boot	-0.12	-0.13	-0.10	0.21	0.02	0.13	0.03
Peak Util.	-0.13	0.13	0.11	-0.05	0.01	-0.04	-0.04

TABLE VII: VM Launching Overhead Comparison Under Different Simultaneous Launches on Different Hosts

- VM launching overhead mainly contains two parts: prolog (image copying/transferring) overhead and booting overhead;
- booting overhead is relatively steady, i.e. has less variations, when it is compared to the prolog overhead;
- prolog overhead, on the other hand, has significant variations under different disk IO utilization, network bandwidth utilization on both host machines and image repository; and
- disk IO utilization has significant impact on both prolog overhead and booting overhead.

In the next section, we present a reference model for the VM launching overhead based on the data obtained.

IV. VM Launching Overhead Modeling

Before we present the reference model for the virtual machine launching overhead in private cloud, we first introduce notations to be used in defining the reference model. In particular, A virtual machine in a private cloud is defined as $v = (f, t, h)$, where f is the image size of the virtual machine, t is the virtual machine launch time and h is the host machine that the virtual machine is to be deployed on. For each host h_i, we denote $V_{h_i} = \{v_1, v_2, \ldots, v_n\}$ as the set of virtual machines the host has. In the set V_i, virtual machines are sorted according to their launch time in none decreasing order. The B_n and B_d denote host machine network bandwidth and disk IO bandwidth, respectively; $av_n(h_i, t)$, $av_d(h_i, t)$ and $av_i(t)$ denote the available network bandwidth on host h_i, disk bandwidth on host h_i, and network bandwidth on image repository at time t, respectively.

The proposed reference model contains three different overheads: timing overhead which is the time needed for launching a VM untill it is ready to execute tasks; disk IO utilization overhead and CPU utilization overhead. We first model the CPU utilization and disk IO utilization that a single VM consumes on the host machine during the launching process. Then we model the host machine's entire system CPU utilization and disk IO utilization. As discussed in section III, the complete VM launching process mainly consists of two parts: prolog and boot process. We discuss the reference models for these two steps below.

A. Prolog Overhead Model

The prolog overhead we modeled in here also contains three different overheads: timing overhead which is the time needed for transferring an image from image repository to the host machine; disk IO utilization overhead and CPU utilization overhead. Let $AV_{band} = min\{av_d(h_i, t_i), av_n(h_i, t_i), av_i(t_i)\}$. The image transfer time for virtual machine v_i is defined below:

$$Trans_i = \frac{f_i}{AV_{band} * w * U_s(h_i, t-1)} \quad (1)$$

where $U_s(h_i, t)$ is the system's CPU utilization that is defined in section IV-E, and w is a constant that represents how much impact that the system's CPU utilization has on the image transferring process.

From the experiments we know that if the disk IO is fully utilized for the image transferring process, the process also fully utilizes one physical core of the host machine. If the disk IO is not fully utilized for the image transferring process, then the CPU utilization is the available disk IO bandwidth proportional to the total IO bandwidth. We first define the base CPU utilization function for image transferring process as follows:

$$U_{tr\_base}(i,t) = \frac{1}{1+e^{-0.5(Trans_i+t_i)(t-t_i)}} - \frac{1}{1+e^{-0.5(Trans_i+t_i)(t-(Trans_i+t_i))}} \quad (2)$$

The IO utilization consumed by a VM's prolog process is modeled as the IO bandwidth occupied by image transferring process to the total bandwidth. Hence, the IO utilization of transferring an image for virtual machine v_i is modeled as:

$$IO_{tr}(i,t) = \begin{cases} \frac{AV_{band}}{B_d} & t_i \le t \le t_i + \textit{Trans}_i \\ 0 & otherwise \end{cases} \quad (3)$$

Then, the CPU utilization of transferring an image for virtual machine v_i is modeled as:

$$U_{tr}(i,t) = IO_{tr}(i,t) * U_{tr\_base}(i,t) \quad (4)$$

B. Booting Overhead Model

The virtual machine booting overhead also refers to the timing overhead and CPU utilization overhead. As once the image is copied to a host, it will not consume any disk IO utilization for the booting process. We consider that there is no disk IO overhead for the virtual machine booting process. The experiments also indicate that the system CPU utilization impact the booting overhead. Hence, we model the CPU utilization overhead for the virtual machine v_i's booting process as follow:

$$U_b(i,t) = c * \frac{1}{m} e^{-\gamma(1-IO_s(h_i,t-1))(t-Trans_i))} \quad (5)$$

where c and γ are two constants,m is the number of cores on the host machine and $IO_s(h_i,t)$ represents the system's disk IO utilization at time t. We will formally define the system disk IO utilization in section IV-E.

In OpenNebula, VMs are not immediately ready for use until all the necessary services, such as ssh, are started. As there is no accurate way to tell the actual time when a virtual machine is booted and ready to use unless entering the running virtual machine and check the log, therefore, we base our estimation fo the time points on the variation of the virtual machine's CPU utilization consumption. If the virtual machine's CPU utilization consumption remain stable, then we consider the virtual machine is booted and ready to use. We define the time point $t_b(i)$ of a virtual machine v_i is ready to use as:

$$t_b(i) = \max\{t | U'_b(i,t) \le \epsilon\} \quad (6)$$

where ϵ is the threshold to determine whether the virtual machine's CPU utilization consumption become stable. Then, we can calculate the virtual machine booting time is as $t_b(i) - \textit{Trans}_i$.

C. Virtual Machine Launching Overhead Model

We have formally modeled image transferring overhead and virtual machine booting overhead. Combining the two compoments together, we derive virtual machine launching overhead functions. In particular, combining equation 4 and equation 5, the virtual machine v_i's launching CPU utilization function is modeled as:

$$U(i,t) = \begin{cases} U_{tr}(i,t) & t \le t_{tran} \\ U_b(i,t) & t > t_{tran} \end{cases} \quad (7)$$

Since the virtual machine booting process does not consume any IO utilization, the IO utilization function for virtual machine v_i's launching process is still equation 3.

The total time needed for launching a virtual machine v_i then can be calculated as image copying time plus virtual machine booting time. It is formally defined as follow:

$$t_{overhead}(i) = t_b(i) - t_i \quad (8)$$

D. Virtual Machine Utilization Consumption Model

The complete virtual machine utilization functions consist of the virtual machine launching overhead utilization functions and the utilization functions after workloads are deployed on the virtual machine. We assume at time $t' \ge t_b(i)$, the virtual machine v_i starts executing tasks; and the CPU and disk IO utilization consumption function of v_i at t' are $U_w(t)$ and $IO_w(t)$, respectively. Then the virtual machine CPU utilization consumption model is defined below:

$$U_c(i,t) = \begin{cases} U_{tr}(i,t) & t \le t_{tran} \\ U_b(i,t) & t > t_{tran} \\ U_w(i,t) & t \ge t' \end{cases} \quad (9)$$

The virtual machine IO utilization consumption model is defined as:

$$IO_c(i,t) = \begin{cases} IO_{tr}(i,t) & t_i \le t \le t_i + \textit{Trans}_i \\ IO_w(i,t) & t \ge t' \\ 0 & otherwise \end{cases} \quad (10)$$

E. System Utilization Model

We assume that host machines only run virtual machines and all other critical system services consumes a small portion of the system CPU and IO utilization. Then we can calculate the system CPU and disk IO utilization as the summation of the virtual machines' CPU and IO utilization consumptions. The system CPU utilization of host h_i is modeled below:

$$U_s(h_i,t) = \max\{1, \sum_{j=1}^{|V_{h_i}|} \{U_c(j,t)\}\} \quad (11)$$

The system IO utilization of host h_i can be modeled as:

$$IO_s(h_i,t) = \max\{1, \sum_{j=1}^{|V_{h_i}|} \{IO_c(j,t)\}\} \quad (12)$$

V. Evaluation

We build the reference model for the virtual machine launching overhead from a large amount of real system experimental data. However, we cannot guarantee the accuracy of the model unless we compare the calculated data using the model we built with the real system data and prove the accuracy of the model. In order to measure the accuracy of the proposed reference model, we introduce an evaluation criteria called average utilization difference. We denote N as the total number of sampling points. The average difference is defined as follows:

$$dif = \frac{1}{N}\sum_{i=1}^{N}|U_r(i) - U_s(i)| \tag{13}$$

where $U_r(i)$ and $U_s(i)$ represents the real data and calculated data at ith sampling point.

Another important criteria needed to be evaluated is the launching time overhead. To check the real time point for the virtual machine that is ready to use, we use the virtual machine system log to check the starting time point of the ssh service. We also calculate the difference between the real VMs' ready time and the calculated ready time to evaluate the accuracy of the proposed model.

We first compare the baseline overhead obtained by calculating the value based on formula 7, and the real data obtained on FermiCloud.

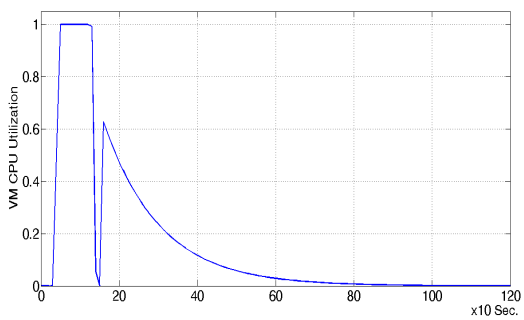

Fig. 19: Baseline VM Launching Overhead using Proposed Model

Fig. 19 draws the CPU utilization during a virtual machine's launching process using the proposed VM launching overhead model. Compare the graph with the utilization variations shown in Fig. 3 as for the baseline virtual machine launching overhead. The calculated data using our proposed model is very close to the real data.

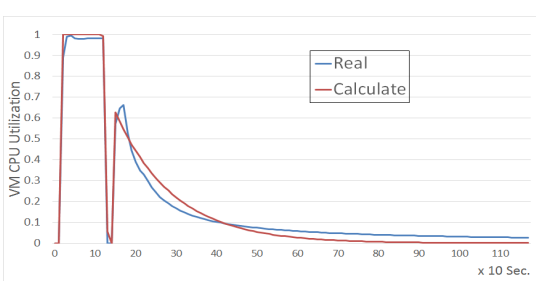

Fig. 20: Baseline VM Launching Overhead Comparison

	Utilization Difference	Time Difference
Baseline	0.0003	-0.023
2 Sim. Launches	0.0446	-0.051
3 Sim. Launches	0.0628	-0.017
Random Launches	0.0491	-0.069
Overall	0.0392	-0.040

TABLE VIII: Performance of the Proposed Model

If we put two data sets at the same page, as shown in Fig. 20, our model accurately represents the baseline CPU utilization variation. Table VIII gives a more detailed comparison between the real data and calculated data. From the table, we can observe that the difference between the real data and the calculated data using our proposed model is only about 0.03% of the system CPU utilization. The estimated launching overhead calculated by our model is only 2.3% below the actual launching overhead.

We further evaluate when more than one VMs are launched simultaneously. Fig. 21 shows the CPU utilization variation comparison between the real data and calculated data from the reference model, i.e., formula 7, when two virtual machines are launched at the same time. Fig. 21 indicates that the CPU utilization difference between real data and calculated data is 4.46% of the system CPU utilization, launching time overhead is also very close to the actual time, only 6% difference. The detailed analysis is given in table VIII.

We increase the number of simultaneous launches to three VMs. The results are depicted in Fig. 22. The CPU utilization difference between real data and calculated data is 6.28% of the CPU utilization; and the estimated launching time overhead is about 1.7% less than the actual measured value.

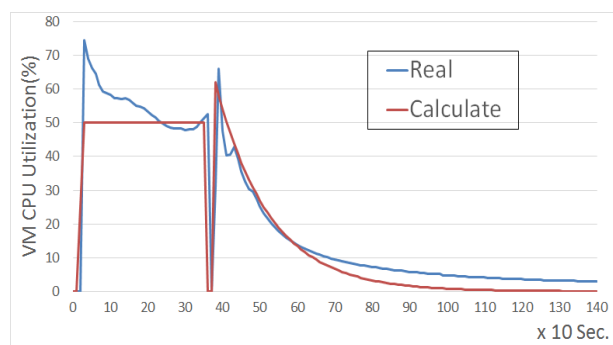

Fig. 21: VM Launching Overhead Comparison with 2 Simultaneous Launches

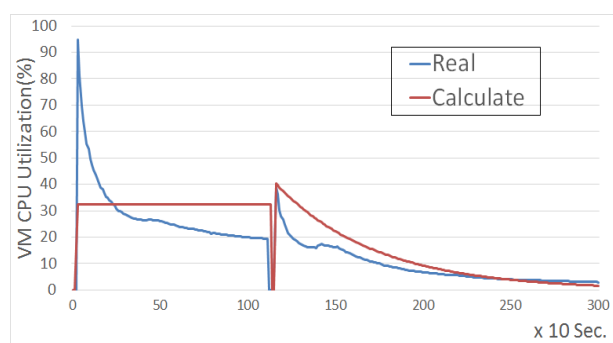

Fig. 22: VM Launching Overhead Comparison with 3 Simultaneous Launches

For the last set of evaluations, we randomly launch multiple

virtual machines under different CPU and IO utilization and at different time instances. We use the reference model to calculate the same scenario for the virtual machine launching process in a real cloud environment. The results are shown in Fig. 23.

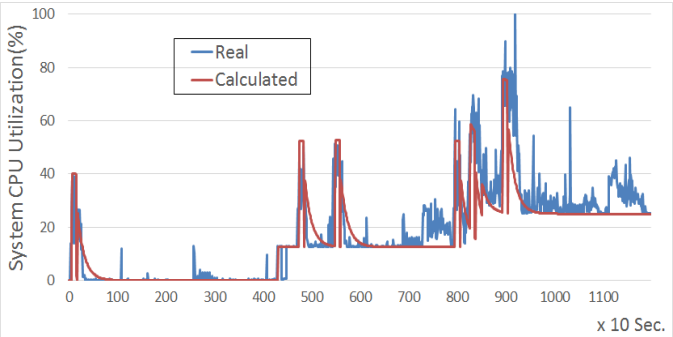

Fig. 23: System Utilization Variation Comparison

As shown in the figure, the two lines are almost merged together. In real environment, there are additional services running on the host machine other than the virtual machines. As a result, they whole system CPU utilization variation is more unpredictable as there are additional CPU utilization consumption in addition to virtual machines. However, the whole system utilization difference between the real data and calculated data is rather small, only 4.91% of the system CPU utilization. When we compare the virtual machine launching time overhead, the difference is still very small, less than 7%.

VI. Conclusion

The FermiCloud is a private cloud built by Fermilab for the scientific workflow. The Cloud Bursting project on the FermiCloud enables the FermiCloud, when more computational resources are needed, to automatically launch virtual machines to available resources such as public clouds. One of the main challenges in developing the cloud bursting module is to decide *when* and *where* to launch a VM so that all resources are most effectively utilized and the system performance is optimized. We have found that the VM launching overhead has a very large variation under different system states, i.e. CPU/IO utilizations can have significant impact on cloud bursting strategies. Hence, being able to model accurately the dependency between VM launching overhead and system resource utilization is critical in deciding *when* and *where* a VM should be launched. This paper has studied the VM launching overhead patterns based on data obtained on FermiCloud and presented a VM launching overhead reference model to guide cloud bursting process. To our best knowledge, this is the first reference model for virtual machine launching overhead that incorporates the dynamics and variation during virtual machine launching process. Our next engineering step is to integrate the reference model into the cloud bursting decision algorithms.

It is worth pointing out that during our experiments, we find that virtual machine launching overhead is mainly caused by the image transferring process. It is not hard to understand that if the image copying/transferring process can be well controlled, the virtual machine launching overhead will become relatively stable and easy to model. As overhead reference model we proposed in this paper consists of two different parts, i.e., prolog overhead and virtual machine booting overhead, the model can easily fit the situation when image transferring process is well managed. We believe that the proposed model is applicable to other private cloud in general.

References

[1] Feature - clouds make way for STAR to shine. http://www.isgtw.org/feature/isgtw-feature-clouds-make-way-star-shine.

[2] Nimbus and cloud computing meet STAR production demands. http://www.hpcwire.com/hpcwire/2009-04-02/nimbus_and_cloud_computing_meet_star_production_demands.html.

[3] Opennebula. http://opennebula.org.

[4] Opennebula managing virtual machines. http://opennebula.org/documentation:archives:rel3.0:vm_guide_2.

[5] A. Beloglazov and R. Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 577–578. IEEE, 2010.

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[7] R. N. Calheiros, R. Ranjan, and R. Buyya. Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In *Parallel Processing (ICPP), 2011 International Conference on*, pages 295–304. IEEE, 2011.

[8] G. G. S. T. G. B. H. W. K. K. C. S.-Y. N. H.-J. J. Hao Wu, Shangping Ren. Automatic cloud bursting under fermicloud. *Workshop on Cloud Services and Systems*, 2013.

[9] N. Huber, M. von Quast, M. Hauck, and S. Kounev. Evaluating and modeling virtualization performance overhead for cloud environments. In *CLOSER*, pages 563–573, 2011.

[10] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Scientific workflow applications on amazon ec2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59–66. IEEE, 2009.

[11] Y. Z. Mengxia Zhu, Qishi Wu. A cost-effective scheduling algorithm for scientific workflows in cloud. *Proceedings of 31st IEEE International Performance Computing and Communications Conference*, 2012.

[12] R. Moreno-Vozmediano, R. Montero, and I. Llorente. Iaas cloud architecture: from virtualized data centers to federated cloud infrastructures. 2012.

[13] J. Qiu, J. Ekanayake, T. Gunarathne, J. Y. Choi, S.-H. Bae, H. Li, B. Zhang, T.-L. Wu, Y. Ruan, S. Ekanayake, et al. Hybrid cloud and cluster computing paradigms for life science applications. *BMC bioinformatics*, 11(Suppl 12):S3, 2010.

[14] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 279–287. IEEE, 2011.

[15] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman. Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, pages 15–24. ACM, 2011.