Virtual Machine Boot Time Model

Thuy Linh Nguyen, Adrien Lebre

▶ **To cite this version:**

Thuy Linh Nguyen, Adrien Lebre. Virtual Machine Boot Time Model. PDP 2017 - 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing, Mar 2017, St Peterbourg, Russia. pp.430 - 437, 10.1109/PDP.2017.58 . hal-01586932

HAL Id: hal-01586932
https://inria.hal.science/hal-01586932

Submitted on 13 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Virtual Machine Boot Time Model

Thuy Linh Nguyen
ASCOLA Research Group, Mines Nantes
INRIA/LINA, Nantes, France
Email: thuy-linh.nguyen@inria.fr

Adrien Lebre
ASCOLA Research Group, Mines Nantes
INRIA/LINA, Nantes, France
Email: adrien.lebre@inria.fr

***Abstract*—Cloud computing by far brings a lot of undeniable advantages. Accordingly, many research works aim to evaluate the characteristics of cloud systems on many aspects such as performance, workload, cost, provisioning policies, and resources management. In order to setup a cloud system for running rigorous experiments, ones have to overcome a huge amount of challenges and obstacles to build, deploy, and manage systems and applications. Cloud simulation tools help researchers to focus only on the parts they are interesting about without facing the aforementioned challenges. However cloud simulators still do not provide accurate models for Virtual Machine (VM) operations. This leads to incorrect results in evaluating real cloud systems.**

Following previous works on live-migration, we present in this paper an experiment study we conducted in order to propose a first-class VM boot time model. Most cloud simulators often ignore the VM boot time or give a naive model to represent it. After studying the relationship between the VM boot time and different system parameters such as CPU utilization, memory usage, I/O and network bandwidth, we introduce a first boot time model that could be integrated in current cloud simulators. Through experiments, we also confirmed that our model correctly reproduced the boot time of a VM under different resources contention.

I. Introduction

In order to study large scale cloud systems, a cloud simulation framework that is capable of simulating as close as possible real cloud behaviors is a huge advantage. To deliver such a framework, VM operations such as booting, migrating, suspending, resuming and turning off should be correctly modeled. While models for some actions such as turning off can be a simple constant, operations such as booting or migrating require advanced models. Following previous works that focused on the migration operation and showed that minimalist models are not appropriate [1], [2], [3], we deal, in this paper, with the boot action. The time it takes to boot a VM is an important element for many operations of a cloud system. For instance, if a sporadic VM is mandatory to perform a task, the time to boot it, is an important information that may become critical if this time is significant. More generally, considering the boot time is essential for VM allocation strategies.

However, one just has to give a look to the literature to understand that an accurate model is required. In Costache et al. [4], the authors used a simple VM boot time model based on a small constant. More recently, a survey on cloud simulation tools [5] showed that current models of VM boot time are simply ignored because they assumed that the VM boot time duration is negligible. Because the VM booting process consumes resources, it is obvious that this assumption is erroneous: any other co-located workloads will impact the booting process of a new VM.

In this work, we discuss several experiments that aim at identifying the factors that govern the boot time of a VM. Relying on the gathered information, we propose a first-class model that is able to calculate the boot time by taking into account resources contentions caused by other computations. In addition to CPU utilization, memory usage, I/O and network bandwidth, we show that the storage device as well as the type of VM image can affect the boot time and should be considered in the model. We underline that the model we propose, is only for the VM boot operation and do not consider the whole chain to obtain a VM on a IaaS system. Considering prior actions such as the provisioning request or the VM image deployment is let as future works. We claim that delivering a first model for the VM boot operation is an important contribution as several people erroneously consider that the time of VM booting process is non-significant.

The rest of this article is organized as follows. Section II explains the basic of the VM launching process and booting process. Section III presents a number of experiments that allowed us to understand the relationship between resource consumptions and the duration to boot a VM. In Section IV, we introduce our VM boot time model. Validation of the model is discussed in Section V. Section VI presents some works related to the VM boot time. Finally, Section VII concludes and gives some perspective of this work.

II. VM Booting Process

In a cloud system, when a user requests to start a VM, three stages are performed:

- Stage 1: The scheduler identifies a suitable physical machine to allocate the VM based on the resource requirements.
- Stage 2: The VM image (VMI) is transferred from the repository to the compute node. Then, the disk for the VM is created from this image.
- Stage 3: The VM is booted on the node.

Depending on properties of the client's request (*i.e.*, VM type, resources), the available physical resources and the scheduling algorithm purpose (*i.e.*, to shorten boot times, to save energy, or to guarantee QoS, etc.), the duration of operation in Stage 1 can vary. In Stage 2, the size of image, the I/O throughput and the network bandwidth seem to be the most dominant factors. Researchers usually consider that the

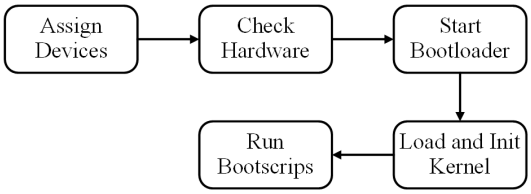


Fig. 1: VM booting process

VM launching process time takes place mostly in this stage and they try to speed it up [6], [7]. In Stage 3, when the boot request is sent from the hypervisor, the normal boot process occurs inside the VM. Because the environment for the VM is ready, researchers assumed that boot process consumes little resources and so, the time to boot a VM can be ignored.

Figure 1 illustrates the different actions that occur in Stage 3. First, resources are allocated to the VM by the hypervisor. Second, the BIOS of the VM (or more recent technologies such as UEFI) locates and loads the boot loader into the memory. Next, the boot loader loads the corresponding kernel binary, initializes various devices and mounts the root file system. After that the kernel runs some scripts to initialize the OS and starts the configured services such as SSH.

These actions that occur in Stage 3 are the ones we would like to model.

III. Experiments

Workloads/VMs that are already executed on a node can significantly increase the time to boot an additional VM and this should be considered. In order to have an idea of the influence level of the factors on the VM boot time, we performed experiments using representative workloads. This helps us identify what are the dominant factors that govern the boot time of a single VM in the presence of concurrent workloads on different storage devices.

A. Experimental Protocol

All experiments have been performed on top of the Grid'5000 testbed [8] and have been automatized thanks to scripts using the Execo framework [9] and libvirt [10].Each physical node had 2 CPUs (8 physical cores each); 64 GB of memory; 10 Gigabit Ethernet. We had 10.000 rpm Seagate Savvio HDDs and Toshiba PX02SS SSDs as storage devices on the compute node. The hypervisor was Qemu/KVM (Qemu-2.1.2 and Linux-3.2). The I/O scheduler of VMs and the compute node was CFQ. We set up all VMs with 1 vCPU and 1 GB of memory.

We used *LINPACK* benchmark [11] to produce CPU workloads; *CacheBench*[1] to produce memory workloads [12]; *Iperf* [2] to stress network bandwidth and *Stress*[3] to produce competition on the I/O path. For the combination of CPU, memory,

[1]http://icl.cs.utk.edu/llcbench/cachebench.html

[2]https://iperf.fr/

[3]http://people.seas.harvard.edu/ apw/stress/

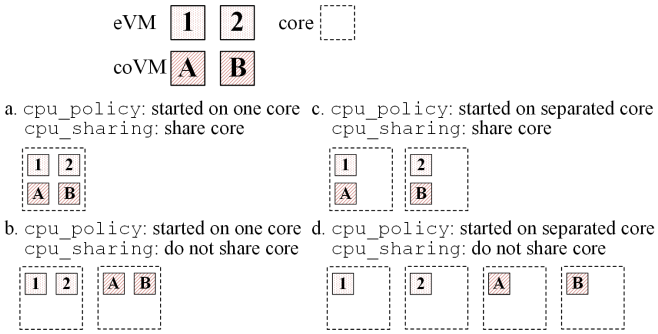


Fig. 2: VM allocation on physical cores with 2 cpu parameters: cpu_policy and $cpu_sharing$

and I/O workloads, we used *Pgbench*[4] program. Running these synthetic workloads on VMs allowed us to stress different resources of a system. In a cloud system, VMs are not ready to be used unless clients can log into the VMs, therefore, we calculated the boot time as the time to have SSH service started. This can be retrieved by checking the system log, and it is measured with millisecond precision. We agree that more complex services than SSH are generally launched in cloud VMs (hadoop, apache, mysql ...). However, understanding whether this last step of the boot process is important is let as future work. Our goal is to deliver a first boot model that can be generic and accurate enough to simulate VM allocation strategies.

In addition to the aforementioned information, we used these following parameters for our experiments:

- eVM: stands for experimenting VM, that is the VM used to measure the boot time;
- $coVM$: stands for co-located VM, that is the VM allocated on the same host and that runs competitive workloads (i.e., workloads that stress CPU, memory, or I/O resources);
- $boot_policy$: defines whether all eVMs are booted at the same time or iteratively.
- $image_policy$: is the way of creating the disk from the image for a VM. There are two possibilities:
  - (a) - ***shared image***: the disk image is created from an original image - called backing file. Initial read accesses are obtained from the backing file whereas write accesses are stored on a specific disk image;
  - (b) - ***no shared image***: the disk image is fully cloned from the original image. All read/writes accesses will be performed on this standalone image.
- cpu_policy: whether all eVMs (or all coVMs) are started on one single physical core or isolated each on a dedicated core.
- $cpu_sharing$: whether a coVM can share the same physical core with or without another eVM.
  Figure 2 illustrates the two CPU-related parameters, let consider that we have 2 eVMs and 2 coVMs. When all VMs are started on one physical core and eVMs can share a physical core with coVMs, all 4 VMs stay on the same

[4]http://www.postgresql.org/docs/devel/static/pgbench.html

physical core (Figure 2a). If $cpu_sharing$ is set to false, 2 eVMs stay in one core while 2 coVMs are assigned to another core (Figure 2b). If we set cpu_policy to start each VM on a separated core, we can have 4 VMs on 4 different physical cores with no $cpu_sharing$ (Figure 2d), or a pair of eVM and its coVM on two separated cores (Figure 2c). Using these two parameters enabled us to finely control the VM placement

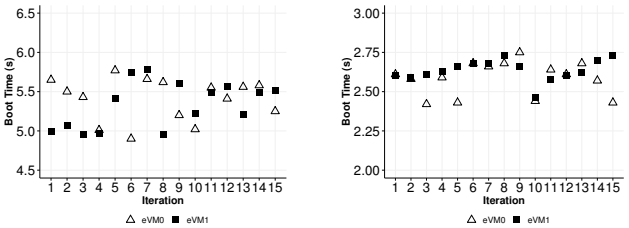
We combined the above parameters to cover a maximum of boot scenarios. Indeed, the VM boot time might not only be impacted by resource competition on the compute node but also the way VMs are booted and the way they are assigned to resources. We underline that we did not take into account the number of cores as well as the memory size assigned to a VM. Indeed, a recent study [13] showed the capacity of a VM does not impact the booting time (*i.e.*, a VM with 1 core and 2G memory takes the same time to boot than a VM with 16 cores and 32GB). Finally, we also did not consider the size of the VM image. Indeed, it is important to understand that only the kernel image should be loaded into the memory. In other words, while other application files can significantly increase the size of the VM image, they will not impact the booting time.

B. Multiple Boot

In this part, there is no coVM involved in the experiments. We discuss the required time to boot multiple eVMs iteratively and all at once.

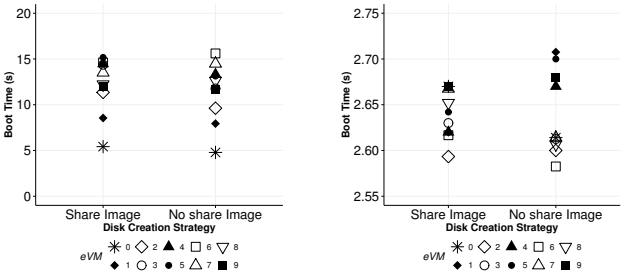
1) Iterative Boot: When a VM is booted, its OS kernel is loaded from the disk into the memory. By considering the *shared image* disk creation strategy, we can expect that the image should be loaded and stayed in the memory after the completion of the first VM's boot. This should benefit to the other VMs sharing the same backing file image. In order to understand whether VM boots can take the advantage of read caching behaviors, we conducted the following experiment: 2 eVMs (each eVM stays on 1 physical core) are booted iteratively; these two eVMs share one common image. First, we boot eVM_0 completely, then we boot eVM_1. When eVM_1 is booting, we observe that there are still read accesses to disk. This means the image is not cached in the memory. We repeat this experiment 15 times. Results are presented in Figure 3. On average, there is no difference between the boot time of the two eVMs, the delta is just 0.5 seconds for HDD (Figure 3a) and 0.25 seconds for SSD (Figure 3b) in the worst case. If there is a cached copy in the memory, the boot time of eVM_1 should always be faster than that of eVM_0. Based on this result and although it is surprising, we had to conclude that there is no read access caching policy for VM by default.

2) Simultaneous Boot: The goal of this second experiment is to evaluate how is the VM boot time affected when several VMs boot at the same time on one node. For such a purpose, the number of eVMs has been increased from 1 to 16. Each eVM has been assigned to a single core to avoid CPU contention and we performed this experiment for both types of disk creation strategies. The expected impact on boot time



(a) On HDD (b) On SSD

Fig. 3: Boot time of two consecutively booted eVMs on HDD and SSD



(a) On HDD (b) On SSD

Fig. 4: Bootting time of 10 eVMs simultaneously on HDD and SSD

in this experiment should be mostly the I/O throughput from loading the kernel image and writing the system log from VMs. Figure 4 shows the results. When there is only one eVM, all the I/O bandwidth is allocated to this eVM, which explains why the boot time is small. But when we have more eVMs that boot simultaneously, the boot time of each eVM increases. Even though we start all eVMs at once in a parallel fashion, there is still a very small gap between the start signal of each eVM. This causes the very first eVM to boot faster in general. The second eVM suffer from the I/O violation of 2 eVMs, the third eVM suffer from 3 eVMs and so on. As a result, the boot time of the last eVMs will converge because they suffer from the same amount of violation in I/O throughput: For HDD, approximately 5 seconds were necessary to boot one eVM but it took as much as 17 seconds to boot 10 eVMs at the same time on a fresh server (*i.e.*, a server that does not host any VM). Because the performance of SSDs are not impacted by seek operations and because the performance can be maintained in this scenario, the difference is negligible: 2.6 seconds were required to boot one eVM compared with 2.8 seconds to boot 10 eVMs on SSDs.

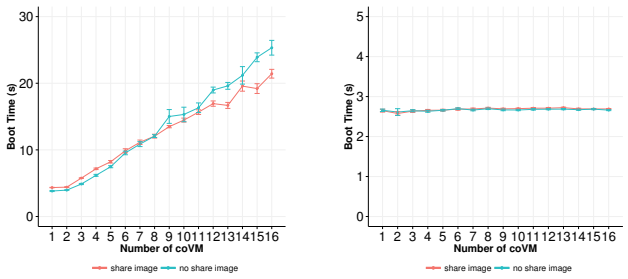
Figure 5 presents the completion time for booting several VMs simultaneously. This results is also valuable as it shows the linear trend for HDDs while the time looks to be constant for SSDs. The difference between the two strategies for HDDs is explained in Section III-C4

C. Single Boot on HDD

In these experiments, we measured the boot time of one eVM while increasing the number of coVMs gradually. On the host, first, we started several coVMs and ran workloads on them for a while so that the workloads can reach their peak

resource usage, then we started one eVMs and measured the boot time. For the experiments where CPU contention should be avoided (*i.e.*, when we wanted to stress either only the memory or the I/O bus or network bandwidth), every VM (eVM and coVM) has been assigned to one dedicated core. Since the maximum number of cores we had on our compute nodes was 16 cores, we set up n coVMs so that $n \in [0, 15]$. The VM disk image has been created by two strategies due to the parameter $image_policy$. With only one eVM, the parameters $boot_policy$ was not relevant. All the experiments have been repeated at least 10 times.

1) Impact of the CPU: LINPACK estimates a system's floating point computing power by measuring how fast a computer solves a dense n by n system of linear equations $Ax = b$. All coVMs have been allocated to the same physical core of the eVM (cpu_policy = True and $cpu_sharing$ = True). This placement generated high CPU competition. After all coVMs have been running at full capacity, we started the eVM and measured the boot time. Results in Figure 6a shows that when the number of coVMs increases, the boot time of the eVM increases as well. This is a clear linear correlation between boot time and number of coVMs. Since the workload on each coVM has been always at 100% of CPU usage, the more coVMs running on one physical core, the more the eVM had to wait until it can access the CPU core to achieve the booting process. This results in a longer boot time. However, we note the difference of boot time between two disk creation strategies. We measured the VM boot time in the environment where there is no resource contention. The average boot time of the VM that had the *shared image* disk is 4.95 seconds while the *no shared image* disk is 4.45 seconds. The gap between the two disk creation strategies is around 0.5 seconds which is maintained when CPU contention happens. The reading mechanism of backing file is the reason. In the *shared image* strategy, the hypervisor has to check where it should read the data from: the VM disk image or the backing file. If the needed data is not on disk, it reads from the backing file. With the strategy *no shared image*, the hypervisor reads the data directly from the VM disk. As a result, reading data with the *no shared image* strategy is generally faster than the *shared image* one. We also highlight that assigning the eVM to one specific core and put other coVMs in other cores did not generate CPU competition. In other words, the boot time is only impacted when there is competition on the core where the eVM has been pinned.



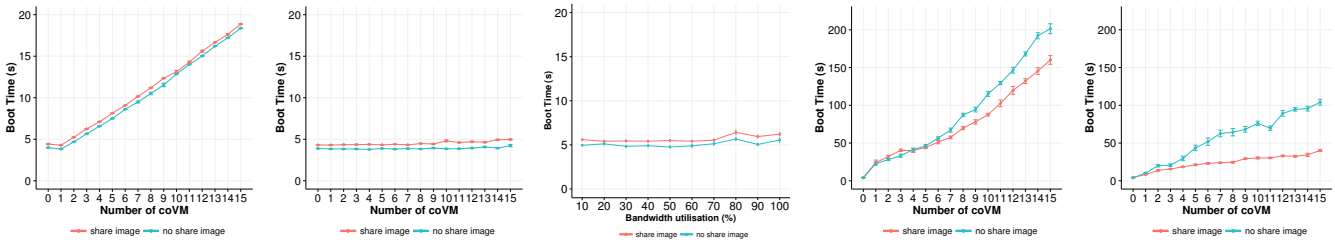
(a) On HDD (b) On SSD

Fig. 5: Bootting time of multiple eVMs simultaneously on HDD and SSD

2) Impact of the memory: CacheBench is a benchmark to evaluate the raw bandwidth in megabytes per second of the memory of computer systems. It includes read, write and modify operations on the memory to fully simulate the effect of memory usage. For this experiments, we avoided I/O conflict and also CPU competition because every coVM and eVM ran on different cores. As shown in Figure 6b, the impact of the memory is not significant in comparison to the CPU one. On average, the boot time with the *shared image* disk creation strategy only raises around 0.8 seconds (from 4.3 seconds to 5.1 seconds) and around 0.5 seconds with the *no shared image* strategy. We still see the gap between boot time of the two disk creation strategies. The explanation is the same as the one for the CPU experiment.

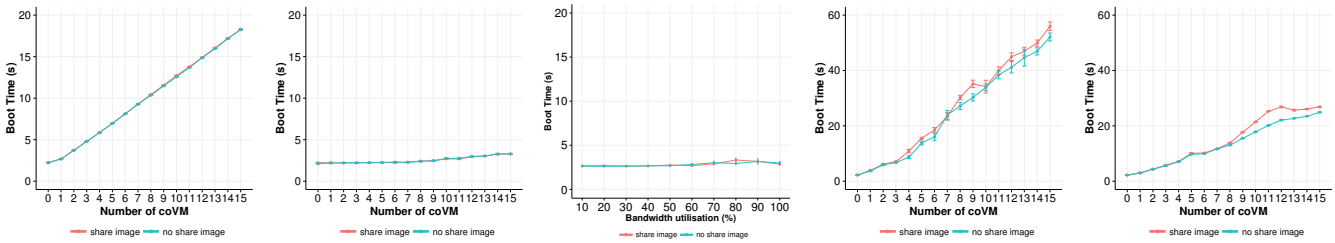
3) Impact of the network stress: Iperf is a tool for measuring of the maximum achievable bandwidth on IP networks. Iperf creates TCP and UDP data streams to measure the throughput of a network that is carrying them. We generated workload on the network of the physical machine by running Iperf to create and send data streams to another physical machine in the same cluster. The experiment protocol is the same as the previous experiment but for the network dimension. That is, we increased the network bandwidth utilization by 10% each time. Figure 6c depicts that the VM boot time under different network bandwidth utilisation is stable. In other words, network competition at the host level does not impact the boot operation.

4) Impact of I/O: The *Stress* command allows us to simulate an I/O stress by spawning a number of workers to continuously write to files and unlink them. We placed every eVM and coVM on separated physical cores to avoid CPU competiton. Figure 6d depicts the boot time for the different scenarios. When we have more coVMs that stress the I/O path, the I/O bandwidth left to be allocated to the eVM becomes smaller. Therefore, this eVM has very little I/O bandwidth to load its kernel image for the booting process so that it takes much longer time to finish booting up. While the boot time grows linearly with the increased number of coVMs for both disk strategies, the boot time of eVM with *no shared image* increases faster when the I/O contention becomes important (from 4 seconds to over 201 seconds, compared to 4.5 seconds to 160 seconds). This can be explained as follows. With *no shared image* disk, a disk is populated with its full size so in case we have many coVMs, the space that is used on the host HDD is larger than the one used for the *shared image* approach. Given the mechanism of the HDD with the seek operation, the probability to have longer seeks is proportional to the space used on the HDD (the HDD's arm needs to seek back and forth often and through a larger distance in order to serve all requests coming from all the hosted VMs). Although in the no I/O contention environment, an *shared image* eVM takes 0.5 seconds longer than a *no shared image* eVM due to the overhead to check where the data is located. This overhead becomes negligible in comparison to the penalty due to the



(a) Impact of CPU usage (b) Impact of memory usage (c) Impact of network stress (d) Impact of I/O throughput (e) Impact of I/O and CPU

Fig. 6: The correlation between VM boot time on HDDs (in seconds) and resource contentions for the two disk creation strategies.



(a) Impact of CPU usage (b) Impact of memory usage (c) Impact of network stress (d) Impact of I/O throughput (e) Impact of I/O and CPU

Fig. 7: The correlation between VM boot time (in seconds) and resources contentions with two disk creation strategies, on SSD.

numerous seeks. Consequently, the boot time for a *shared image* VM is smaller than for a *no shared image* VM under high I/O contention ($n_coVM > 5$) as depicted in Figure 6d. We verified such an explanation by conducting additional experiments with *blktrace* tool [14] to extract the I/O access data on the compute node while coVMs run I/O stress.

5) Impact of mixed workload: Pgbench is an official benchmark that performs operations on a postgresql-9.4.5 database server. Pgbench runs SQL commands on multiple concurrent database sessions to measure the performance of a database server. As Postgres database has to handle the load generated by Pgbench, it consumes not only I/O bandwidth but also CPU capacity and memory to run queries. In our experiment, Pgbench has been eployed on each coVM. Because Postgres database has a caching mechanism that prevents reading from the disk to query data all the time, this test did not saturate the I/O path as in the previous experiment, leading to smaller boot times. However, when there are more workloads from coVMs, the VM boot time can increase up to 36 seconds in case *shared image* and 110 seconds with *no shared image*, so boot time of eVM with *shared image* disk is 60% faster than that of *no shared image*. The explanation here is also related to the cost of HDD seeks to serve I/O requests. Moreover and because *Pgbench* generates both read and write requests to the physical disk, the number of seeks is increased overall producing a larger difference in boot time for the two disk creation strategies in this case when compared with the I/O contention scenario (in Figure 6d).

D. Single Boot on SSD

As we performed the same five previous experiments on SSD-backed machines, we found that the similar trend occurs in Figure 7 but the boot time in I/O contention experiments on SSD is much faster than on HDD disk. We do not see the difference between two disk creation strategies for CPU and memory scenarios (Figure 7a and Figure 7b). SSD disks do not need to seek for data when reading or writing because there is no moving part compared to traditional electromechanical magnetic disks. As a consequence, writing and reading from SSD is significantly faster. The better performance is also the reason why under high I/O bandwidth of SSD, the 0.5 seconds delay between the two disk strategies is minimized. Regarding Figures 7d and 7e, the trends between the two strategies are similar. Small deviations related to share image are due to the reading mechanism as explained in Section III-C1 (even with SSDs, the hypervisor should check whether it should read data on the backing file or from the qcow). Since the I/O bus is more stressed, such an overhead becomes more significant under high I/O contention.

E. Discussion

By comparing all metrics as depicted in Figure 8, we observe that I/O throughput and CPU capacity are the dominant factors that clearly govern VM boot times for both HDD and SSD scenarios. However, it is noteworthy that the I/O factor is much more impacting than the CPU: when the workload increases, the boot time of VM in I/O contention ranges from 4 to 200 seconds with HDDs and 2.8 to 60 seconds with SSDs, while in CPU contention, the boot time increases approximately from 4 to 20 seconds for HDDs and from 2.8 to 17 for SSDs. In other words, where CPU contention increases the boot time by a factor between 5 and 6, IO stress leads to a boot time that is from 20 to 50 times higher. Even though the impact of CPU on VM boot time is rather small compared to I/O factor, we cannot simply ignore CPU factor, since it is still significant enough to alter the boot duration.

Moreover, the two disk creation strategies lead to different VM boot time fluctuations. For each factor, the general trend of VM boot time remains the same for both strategies, however the magnitude of VM boot time changes. With the experiments on I/O and mixed workload, we can confirm that the *no shared image* strategy is more impacted by I/O contentions on HDD nodes.

IV. VM Boot Time Model

By using coVMs to generate stress on the shared resources of the physical machine, we identified the effect of different resources on the VM boot time. As concluded in Section III-E, the VM boot time is mainly influenced by CPU and I/O factors. This corresponds to the actions that have been highlighted in Section II (Stage 3) : during the boot process, CPU has to check devices, set up necessary initializations; while part of the VM image should be read from the disk to the memory. In other words, a VM boot time model should integrate the I/O and CPU dimension. Keeping this in mind, we propose to model the boot time around two components:

$$t = time_{IO} + time_{CPU} \tag{1}$$

Using the experiment results in Figure 8, we made a transposition from the number of coVMs into the utilization percentage of I/O throughput and CPU capacity. Figure 9 presents this conversion. The correlation between the boot time and the utilization percentage of both CPU capacity and I/O throughput is the upward curve. In fact, when the utilization reaches 100%, theoretically the booting process can last indefinitely. Therefore, we propose a boot time model that follows he form $\frac{\alpha}{1-x}$ where x is the resource utilization percentage. Moreover, we add the exponential growth function to this equation for the I/O dimension in order to reflect the fact that the boot time for I/O throughput utilization increases exponentially. Furthermore, when there is no resource contention, which means the utilization percentage is zero, the boot time can be modeled through two values, denoted α and β. The equation for each dimension is:

$$time_{IO} = \frac{e^x \times \alpha}{1-x} \tag{2}$$

$$time_{CPU} = \frac{\beta}{1-y} \tag{3}$$

with:

- x is the utilization percentage of I/O throughput
- y is CPU utilization percentage
- α is the time that a VM needs to perform I/O operations in booting process when there is no resources contention
- β is the time that CPU takes to run operations in booting process when there is no resources contention

V. Model Evaluation

To validate the correctness and the accuracy of our VM boot time model, we compared the boot time from Grid5000 experiments with the boot time calculated from the model. To perform such a comparison, we first, identified the α and β parameters of our model.

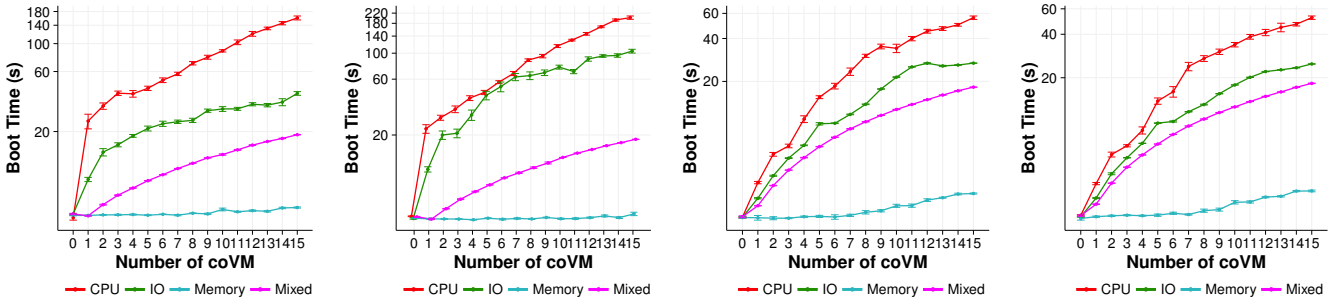
$$t = \frac{e^x \times \alpha}{1-x} + \frac{\beta}{1-y} \tag{4}$$

Those parameters vary according to different hardware configurations and guest operating systems. Therefore, to evaluate the model against the experiment results in Section III, we ran a calibration experiment, *i.e.*, an experiment where one VM is booted with no resource contention at all. In this case, the boot time model is simply $t = \alpha + \beta$. We measured t and because the time consumed by the CPU, *i.e.*, β, can be retrieved from the system logs (`/proc/stat`), we can easily determine α. It is noteworthy that with large and complicated systems (*i.e.*, composed of different physical hosts and many OSes), this initial benchmark should be performed one each kind of machines.

Figures 10 and 11 show the comparison of boot time, with HDD and SSD machines under resources contentions, between the Grid5000 experiments, our model and a naive one (*i.e.*, with a constant boot time of 30 seconds as used in [4]). We evaluated the model on three resource contention cases: CPU, mixed workload (using *Pgbench*) and I/O contention. It is clear that the naive model cannot represent the variation of VM boot time under different conditions. The naive model overestimates the boot time in CPU contention while it underestimates by a large margin in high I/O contention.

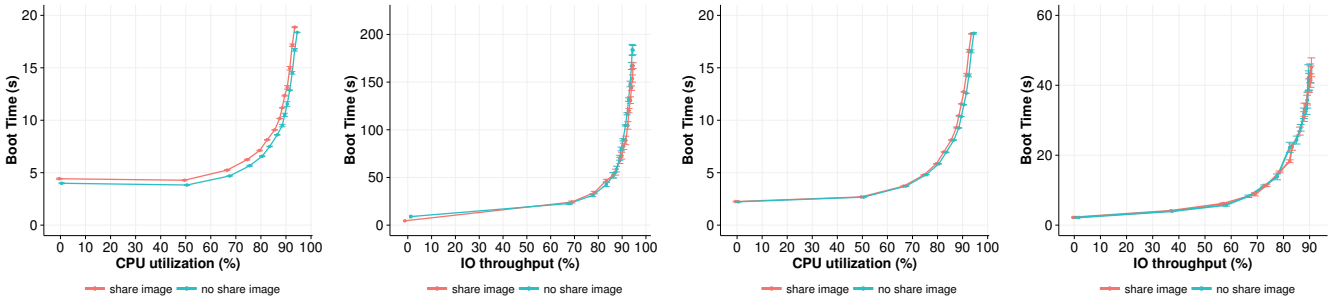
At the opposite, Figure 10 shows that our model can keep up with the upward curve of VM boot time under different workloads stress on HDD. The deviation in CPU contention case is within 2 seconds in Figure 10a. For the mixed workload (Figure 10b), the average deviation is 15 seconds, but for high resource contention (*i.e.*, 90% CPU and I/O throughput utilization), the deviation increases up to 37 seconds (keeping in mind that the boot time can reach up to 130 seconds). Under I/O stress, the boot time can rise as high as 210 seconds. The difference between our model and the Grid5000 experiments is 10 seconds on average for *shared* disk (Figure 10c) and 25 seconds for *no shared* disk (Figure 10d). However, when the I/O resource utilization goes over 90%, the deviation can be up to 23 seconds and 89 seconds for each disk type (i.e. SSD,HDD), respectively. In the case of HDD, under I/O contention, the model is less accurate due to the large difference in boot time of the two disk creation strategies (Figure 6d). With the *no share* disk, there is a higher probability that the HDD needs to do more seek operations, which leads to the high deviation of the model. However, our VM boot time model does not take into account such a phenomenon yet.

The graph in Figure 11a shows that our model also successfully represents the upward curve of VM boot time for SSDs. The deviation is within 1.5 seconds for CPU stress and 4 seconds for mixed worloads. The comparison between the Grid5000 experiment and the model under I/O contention using SSDs is illustrated in Figure 11c. Under high I/O contention, the boot time can rise up to 70 seconds. The



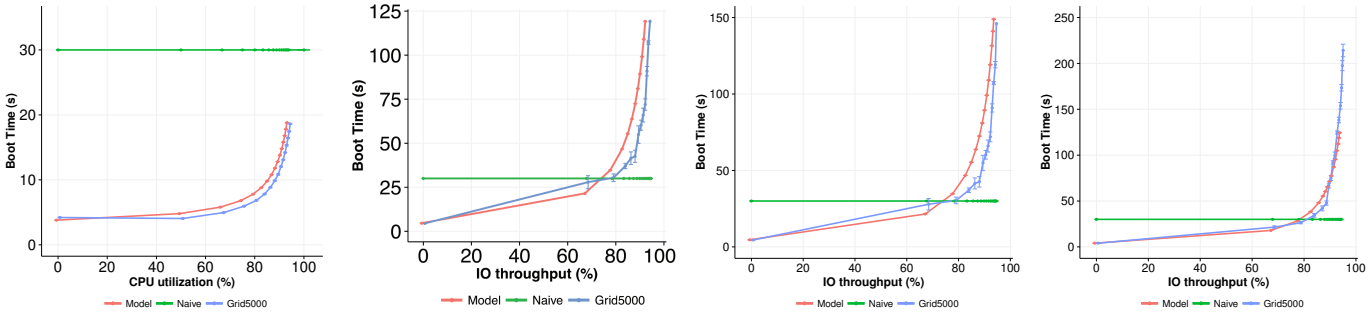
(a) *Shared* image - on HDD (b) *No Shared* image - on HDD (c) *Shared* image - on SSD (d) *No Shared* image - on SSD

Fig. 8: The comparison of VM boot time with 4 factors: I/O throughput, CPU capacity, memory utilization and the mixed factors, on HDD vs SSD
The y-axis shows VM boot time in seconds on a log scale with base 10



(a) CPU utilization on HDD (b) I/O utilization on HDD (c) CPU utilization on SSD (d) I/O utilization on SSD

Fig. 9: The correlation between VM boot time and resources utilization with two disk creation strategies, on HDD and SSD



(a) CPU (b) Mixed workload (c) IO, *Shared* image (d) IO, *No Shared* image

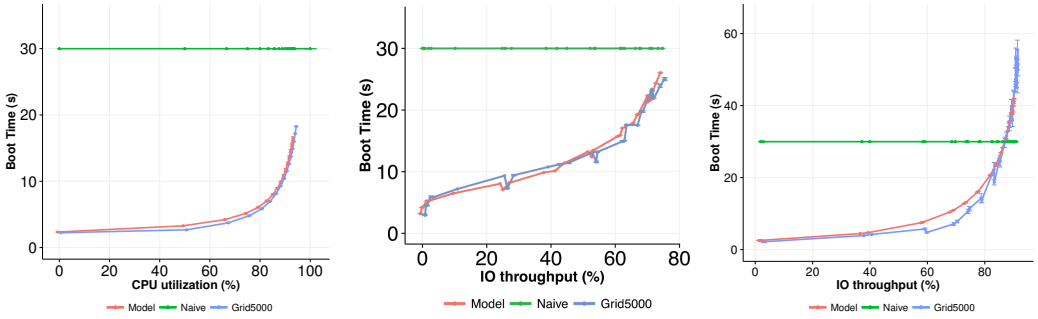
Fig. 10: Boot time with different resource contentions on HDD

model differs from the Grid5000 experiment by 25 seconds maximum when the I/O utilization reaches 90% and 5 seconds on average.

VI. Related Work

There are several works that analyzed/improved the provisioning chain of a VM (*i.e.*, the three steps introduced in Section II). For instance, the works [15] and [6] focused on stage 2, the transferring process. In [15], they studied the startup time of VMs across real-world cloud providers such as Amazon EC2, Window Azure and Rackspace. They analyzed the relationship between the VM startup time and VMI, instance type, time of day and the data center location. In [6], they also analyzed the startup time based on the VMI disk size and the content on Amazon EC2. They let users select the desired set of services, based on which the strictly necessary set of software packages is determined and installed into the VMI. Their solution reduces the disk size in four times and speeds up the VM startup time up to three times. Amazon EC2 uses EBS-backed VMIs (*i.e.*,. remote attached disks). This means that the VMI is mounted on the node where the VM will be started [7]. Based on their evaluations, it seems that it reduces the time to start the VM.

Other works have focused on the boot process in Stage 3. Razavi et al. [16] proposed the *Prebaked* capability to improve the startup time by addressing the VM boot process itself. VMs are pre-booted and snapshotted so that they can be resumed in any location of the infrastructure. The authors in [4] implemented a model in the CloudSim simulator for a few VM operations including the VM boot. However, the model is



(a) CPU (b) Mixed workload (c) IO

Fig. 11: Boot time with different resource contentions on SSD

simple as it is just a constant for the VM boot time. In [13], Wu et al. performed an in-depth analysis of the overhead that occurs when VM are launched (*i.e.*, transfer of the VMI and boot process of the VM). They proposed a global resource utilisation model for these two steps. They concluded that the boot overhead is relatively steady compared to the time to transfer the VMI. However, our experiments showed that the boot time can fluctuate between a few seconds up to several minutes according to the conditions of the system.

VII. Conclusion

In this work, we discussed several experiments we performed in order to identify which factors affect the VM boot time and their influence levels. Based on the results, we can confirm that a simple model based on constants are definitely not accurate since it cannot handle the variation of the boot time in presence of co-located workloads. This is critical as the boot time may increase up to 50 times under high I/O contentions. Concretely, the time to boot a VM grows from a few seconds to several minutes according to the contention that occurs on the I/O and CPU resources. To take into account these variations, we introduce a VM boot time model that gives a more accurate estimation of the time a VM needs to boot in a shared resource environment. When resource utilization is lower than 90%, our model succeeds to follow the times we measured with deviations within 2 seconds in case CPU contention, and 10 seconds in average for I/O contention (*i.e.*, 82% and 90%, respectively)

Regarding the future work, we plan to extend the model to capture the HDD seek time when there are I/O competitive workloads. Moreover, we will conduct the same set of experiments on network storage. Network storage is intensively used in Desktop as a service scenarios. Furthermore, we also plan to evaluate the model in other public and private cloud systems. Finally, we are interested in integrating our model in the cloud simulator such as SimGrid [17].

References

[1] T. Hirofuchi, A. Lèbre, and L. Pouilloux, "Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 96–103.

[2] V. Kherbache, F. Hermenier *et al.*, "Scheduling live-migrations for fast, adaptable and energy-efficient relocation operations," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2015, pp. 205–216.

[3] V. De Maio, R. Prodan, S. Benedict, and G. Kecskemeti, "Modelling energy consumption of network transfers and virtual machine migration," *Future Generation Computer Systems*, vol. 56, pp. 388–406, 2016.

[4] S. Costache, N. Parlavantzas, C. Morin, and S. Kortas, "On the use of a proportional-share market for application slo support in clouds," in *Euro-Par 2013 Parallel Processing*. Springer, 2013, pp. 341–352.

[5] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 866–872.

[6] K. Razavi, L. M. Razorea, and T. Kielmann, "Reducing vm startup time and storage costs by vm image content consolidation," in *Euro-Par 2013: Parallel Processing Workshops*. Springer, 2013, pp. 75–84.

[7] "Creating an Amazon EBS-Backed Linux AMI," http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html.

[8] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[9] M. Imbert, L. Pouilloux, J. Rouzaud-Cornabas, A. Lèbre, and T. Hirofuchi, "Using the execo toolbox to perform automatic and reproducible cloud experiments," in *1st International Workshop on UsiNg and building ClOud Testbeds (UNICO, collocated with IEEE CloudCom 2013*, 2013.

[10] Red Hat, "libvirt: The virtualization api," *http://libvirt.org*, 2012.

[11] "LINPACK_BENCH - the LINPACK benchmark," http://people.sc.fsu.edu/~jburkardt/c_src/linpack_bench/linpack_bench.html.

[12] P. J. Mucci, K. London, and J. Thurman, "The cachebench report," *University of Tennessee, Knoxville, TN*, vol. 19, 1998.

[13] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, K. Chadwick, and S.-Y. Noh, "A reference model for virtual machine launching overhead," 2014.

[14] A. D. Brunelle, "Block i/o layer tracing: blktrace," *HP, Gelato-Cupertino, CA, USA*, 2006.

[15] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 423–430.

[16] K. Razavi, G. Van Der Kolk, and T. Kielmann, "Prebaked μvms: Scalable, instant vm startup for iaas clouds," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 245–255.

[17] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*. IEEE, 2008, pp. 126–131.

[18] T. Knauth and C. Fetzer, "Fast virtual machine resume for agile cloud services," in *Cloud and Green Computing (CGC), 2013 Third International Conference on*. IEEE, 2013, pp. 127–134.