Resource Leasing and the Art of Suspending Virtual Machines

Borja Sotomayor
University of Chicago
borja@cs.uchicago.edu

Rubén Santiago Montero Ignacio Martín Llorente
Universidad Complutense de Madrid
{rubensm,llorente}@dacya.ucm.es

Ian Foster
University of Chicago
Argonne National Laboratory
foster@mcs.anl.gov

Abstract

Using virtual machines as a resource provisioning mechanism offers multiple benefits, most recently exploited by "infrastructure-as-a-service" clouds, but also poses several scheduling challenges. More specifically, although we can use the suspend/resume/migrate capability of virtual machines to support advance reservation of resources efficiently, by using suspension/resumption as a preemption mechanism, this requires adequately modeling the time and resources consumed by these operations to ensure that preemptions are completed before *the start of a reservation. In this work we present a model for predicting various runtime overheads involved in using virtual machines, allowing us to efficiently support advance reservations. We extend our lease management software, Haizea, to use this new model in its scheduling decisions, and we use Haizea with the OpenNebula virtual infrastructure manager so the scheduling decisions will be enacted in a Xen cluster. We present both physical and simulated experimental results showing the degree of accuracy of our model and the long-term effects of variables in our model on several workloads.*

1 Introduction

Virtual machines (VMs) have become a key technology to realize an "infrastructure-as-a-service" (or IaaS) model where computational capacity, in the form of virtual machines deployed in a resource provider's datacenter, is provisioned on-demand as a service. Virtual machines offer multiple benefits, such as the ability to securely partition physical servers and to provide users with customized software environments, but pose the problem of how to efficiently schedule virtual machines on multi-host environments. In previous joint work with K. Keahey (Argonne National Laboratory) [20, 21] we proposed a solution to this problem that relied on *leases* as a fundamental resource provisioning abstraction; a lease is "a negotiated and renegotiable agreement between a resource provider and a resource consumer, where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumer." We defined lease terms supporting both best-effort leases, where requests are queued until resources can be allocated, and advance reservation (AR) leases, where resources must be available at a specific time. We designed and implemented Haizea,[1] a lease management system that allows users to request a lease for computational resources, which is then scheduled and implemented as a set of VMs. We also showed that, by leveraging the suspend/resume/migrate capabilities of VMs, our approach allowed ARs to be supported efficiently, without the utilization problems commonly associated with them [5, 11, 18, 19].

Supporting ARs accurately and efficiently, however, requires an accurate model of the time and resources consumed by certain VM operations. For example, starting a set of VMs may require transferring multiple VM disk images to the physical nodes where the VMs will run. If those VMs have been scheduled for a lease starting at a time t, the images must be transferred before that starting time, or the lease agreement will be broken. This requires not only estimating the transfer

[1] http://haizea.cs.uchicago.edu/

times accurately but also scheduling to meet a deadline, possibly in the presence of other image transfers that have to be completed before other deadlines. Similarly, if an AR lease requires preempting other leases, which can be accomplished by suspending the VMs in the preempted lease, those suspension operations must also be scheduled in such a way that they will finish before the start of the reservation.

Our previous work focused on modeling *preparation overhead*, or the actions that must take place before the start of a lease, such as transferring VM disk images. However, we used a simple model for the *runtime overhead* of suspending and resuming VMs that incorporated certain assumptions that we knew to be limiting in practice, such as allowing only a single VM to be deployed per physical server and requiring that memory state images (resulting from suspensions) be saved to the local filesystem of the physical node, without allowing for the option of a global filesystem. This made VM scheduling simpler but did not capture the interactions that may arise when multiple VMs are scheduled on the same physical server. Additionally, we did not validate this model through experiments on physical resources; all our results were simulation-based. Here, we develop a more general model and validate this model by conducting a range of experiments. More specifically, our contributions are the following:

1. A model for predicting the runtime overhead of suspending and resuming VM-based leases (with potentially multiple VMs that must be co-scheduled within the lease) under a variety of conditions, removing many of the assumptions made in previous work.

2. Experimental results showing the degree of accuracy of our model in predicting the runtime overhead of suspending and resuming leases on a physical testbed. To run these experiments, we implemented our model in the Haizea lease manager, which relied on our model's estimations to schedule VM suspensions and resumptions. Furthermore, we used the OpenNebula[2] [10] virtual infrastructure manager to run experiments on a physical testbed.

3. Simulation results showing the long-term effects of modifying parameters in the model, such as the amount of memory requested by VMs or the amount of network bandwidth available.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 describes the design and implementation of our model, including the formulas to estimate suspension/resumption times (Section 3.1) and a description of the OpenNebula and Haizea systems (Section 3.2), and the modifications that were necessary to support our model. Next, Section 4 discusses experimental results testing the degree of accuracy of our model and the long-term effects of modifying certain variables in our model. Section 5 presents our conclusions and discusses future work.

[2] `http://www.opennebula.org/`

2 Related Work

Several research groups have explored the use of VMs as a mechanism to allocate and manage computational resources, addressing a variety of issues such as cluster contextualization [7], rapid deployment [13,23], job scheduling on virtual clusters [3,4,8,22], VM consolidation [2,12], virtual networking and load balancing between multiple physical clusters [16,17], and automatic configuration and creation of VMs [1,9]. However, all of these focus on meeting the requirements of a single provisioning scenario (either immediate or best-effort, but not ARs), except for Walters et al. [22], who consider workloads combining both batch jobs and interactive jobs with near-immediate availability requirements. Both Walters et al. and Fallenbeck et al. [4] leverage the suspend/resume capability of VMs to improve utilization of physical resources by using preemption-based scheduling, but the overhead of suspending/resuming is not modeled or explicitly scheduled so it will complete before a deadline (e.g., the start of a reservation). Yamasaki et al. [23] developed a model for predicting the time to completely set up a new software environment on potentially heterogeneous physical nodes, allowing their scheduler to choose nodes that minimize the time to set up a new virtual cluster. Although their model is effective for rapid deployment of virtual clusters, it does not model suspension and resumption of VMs. Our lease model supports both best-effort provisioning and ARs, including preempting VMs in favor of an AR, and accordingly models the overhead of preemption (via suspension and resumption of VMs) to guarantee that resources are preempted in time before the start of a reservation.

Several solutions have emerged, both in industry and academia, to manage *virtual infrastructures*, allowing dynamic deployment of virtual machines in a pool of physical servers. Datacenter-based solutions include commercial products such as VMWare VirtualCenter[3] and Platform Orchestrator[4] and open-source

[3] `http://www.vmware.com/products/vi/vc/`

[4] `http://www.platform.com/Products/platform-vm-orchestrator`

projects such as Enomalism and Ovirt.[5] Amazon EC2, GoGrid, FlexiScale, and ElasticHosts are examples of "clouds" that provide infrastructure-as-a-service using VMs. Eucalyptus [14] and Nimbus [6] are open source projects that allow existing infrastructure to be transformed into an infrastructure-as-a-service cloud with EC2-compatible interfaces. However, all these solutions use an immediate provisioning model where VMs must be allocated right away, or not at all, with no support for ARs. As such, they do not support resource preemption and thus have no need to model suspension/resumption in VMs.

Our work is relevant to use cases where ARs, and thus the ability to support them efficiently, are desirable. Service provisioning clouds, such as the one being built by the RESERVOIR project[6] [15], have requirements that cannot be supported with only an immediate provisioning model, including the need for capacity reservations at specific times to meet service-level agreements or peak capacity requirements. Additionally, smaller clouds with limited resources, where not all requests may be satisfiable immediately because of lack of resources, stand to benefit from more sophisticated VM placement strategies supporting queues, priorities, and ARs.

3 Design and Implementation

In this section, we present our lease suspension/resumption time model. The development of this model was an iterative process, where we started with a basic model, implemented it using Haizea and OpenNebula, experimented with it, and refined the model based on our observations. Although we describe the stages the model went through, the results presented later in the paper rely on the final model. We also briefly describe the Haizea lease manager and the OpenNebula virtual infrastructure manager, and we discuss how we implemented our model using them.

3.1 Modeling and Scheduling VM Suspension/Resumption Times

When scheduling leases, we assume we manage P identical physical nodes with the ability to instantiate VMs on them. Each node has C cores, M megabytes of memory, and D megabytes of local disk storage. When a VM is suspended on a physical node, it does so at a rate of s megabytes of VM memory per second. We define r similarly for VM resumption. Suspending a VM results in a memory state image that can be saved to either a local filesystem or a global filesystem ($f \in \{\mathsf{local}, \mathsf{global}\}$). All nodes are connected by a switched network. A lease is implemented as a set of N VMs, one for each node requested in the lease. We assume that all VMs have the same resource requirements, described by a tuple (c, m, d, b), where c is number of CPUs, m is memory in megabytes, d is disk space in megabytes, and b is network bandwidth in megabytes per second.

Our first model to estimate suspension/resumption time, used in our previous work [20], assumed that $C = 1$ and $f = \mathsf{local}$. Thus, a lease can be suspended by suspending all its N VMs in parallel, since each physical node will have only one VM to suspend, and each physical node's local filesystem is independent of the others. Thus, t_s and t_r, the time to suspend and resume an entire lease, can be estimated by the following simple formulas.

$$t_s = m/s \tag{1}$$

$$t_r = m/r \tag{2}$$

If we remove the $C = 1$ assumption, a lease may have more than one VM in a physical node, and the above formulas become invalid because we can no longer assume that all the VMs can be suspended in parallel. Although each physical node can suspend VMs independent of what is happening on other nodes, the time to suspend will now depend on the number of VMs scheduled on each physical node. Additionally, we observed that suspending or resuming multiple VMs simultaneously on the same physical node results in longer and more unpredictable suspension times because of contention for I/O. Thus, we choose to schedule suspension and resumptions sequentially, so each has exclusive access to the filesystem. More specifically, if n_i is the number of VMs mapped to physical node i, then suspending all the VMs in i will require $n_i \cdot \frac{m}{s}$. If we remove the $f = \mathsf{local}$ assumption and allow memory state files to be saved to a global filesystem, we can no longer assume that physical nodes can suspend independently of others. Instead, we need to account for contention when accessing the shared filesystem.

Thus, the time to suspend an entire lease becomes the following.

$$t_s = \begin{cases} \max(n_0, n_1, \ldots, n_P) \cdot \frac{m}{s} & \text{if } f = \mathsf{local} \\ N \cdot \frac{m}{s} & \text{if } f = \mathsf{global} \end{cases} \tag{3}$$

We can define t_r similarly.

Each suspend/resume command sent to physical nodes will incur a communication overhead. In preliminary experiments we observed that this overhead

[5] http://ovirt.org/

[6] http://www.reservoir-fp7.eu/

cannot be assumed away, since OpenNebula sends commands to physical nodes sequentially over TCP. Thus, even if a command takes only 1–2 seconds to be processed, suspending 64 VMs would result in a total overhead of 64–128 seconds. Thus, if e is the enactment overhead of sending a suspend/resume command, the formula becomes the following.

$$t_s = N \cdot e + \begin{cases} \max(n_0, n_1, \ldots, n_P) \cdot \frac{m}{s} & \text{if } f = \mathsf{local} \\ N \cdot \frac{m}{s} & \text{if } f = \mathsf{global} \end{cases} \quad (4)$$

Our model also takes into account h, the time to shut down a VM. When suspending/resuming a lease A to free resources for another lease B, the end of B is, in general, followed by the resumption of A. Previously, we did not model the shutdown time, as we assumed that resumption could begin as soon as the ending lease started shutting down (and assumed the cost of shutdown was negligible). However, we observed in preliminary experiments that allowing B's shutdown to overlap with the resumption of A noticeably delayed the first resumption operations, resulting in a longer t_r than expected. Although modeling h does not affect the formulas for t_s and t_r, it is taken into account by the scheduler.

3.2 Haizea and OpenNebula

Haizea is an open source lease management architecture supporting *advance reservation leases*, where the resources must be available at a specific time; *best-effort leases*, where resources are provisioned as soon as possible and requests are placed on a queue if necessary; and *immediate leases*, where resources are provisioned when requested, or not at all. Haizea maps leases to VMs, scheduling the deployment overhead of starting those VMs separately (so it will not be deducted from the lease's availability period), and leveraging backfilling algorithms and the suspend/resume/migrate capability of VMs to schedule the leases more efficiently. More specifically, best-effort leases can be preempted by AR leases by suspending the VMs of the best-effort lease before the start of the AR, and resuming them after the end of the reservation.

Haizea originally estimated suspension/resumption times using formulas (1) and (2), which effectively limited it to scheduling one VM per physical node. Implementing the model described in the previous section required more than just updating the formula used to estimate the time. Since the model requires that each suspension/resumption have exclusive access to a filesystem (local or global), Haizea had to be modified so that these operations were spaced apart in such a way to guarantee exclusion. Additionally, the scheduler had to take into account the shutdown time h of leases to avoid the shutdown of a VM from overlapping with the resumption of another VM (e.g., when a lease was being resumed after the end of an AR lease).

Although Haizea is capable of scheduling leases, however, it has no way of enacting scheduling decisions. By itself, Haizea can be used only as a scheduling simulator. To operate on physical hardware resources, we also integrated Haizea with the OpenNebula virtual infrastructure manager. OpenNebula provides the functionality needed to deploy, monitor, and control VMs on a pool of distributed physical resources. The OpenNebula architecture has been designed to be flexible and modular to allow its integration with different hypervisors and third-party components. In particular, this modular architecture allowed us to use Haizea as a drop-in replacement for OpenNebula's default scheduler, thus allowing Haizea to send its enactment commands to OpenNebula instead of to a simulated testbed.

4 Experimental Results

To evaluate the accuracy and effects of our model we carried out two sets of experiments on physical hardware and in simulation. The first set of experiments, carried out on our Xen testbed, focuses on determining what factors can affect the accuracy of the model, and showing the effect of underestimating and overestimating the values of s and r in practice. The other set of experiments, carried out by simulating 30 days of lease requests, shows the long-term effects of different parameter values in our model. This section begins by describing our experimental setup; then each set of experiments is described.

4.1 Experiment Setup

Our testbed is made up by five SunFire x4150 servers, each with two Intel Xeon QuadCore L5335 2 GHz processors (i.e., 8 cores per server, $C = 8$) and 8 GB of RAM ($M = 8192$). All the nodes are connected with a switched Gigabit Ethernet network. One of the nodes is used as a head node that hosts a shared NFS filesystem for all the nodes, while the remaining four nodes are used to run virtual machines (i.e., $P = 4$, with $P \cdot C = 32$ cores available to run VMs). The head node also runs OpenNebula 1.0 and Haizea TP1.2 (the latest versions at the time of writing this paper), which manage all the VMs during the experiments. The testbed is configured to operate with Xen 3.2 or KVM,

although our current results are based on Xen 3.2. All virtual machines used in the experiments have 2 GB disk images on the shared NFS filesystem. When suspending or resuming a virtual machine, OpenNebula can be instructed to save the file to the NFS filesystem or to the local filesystem of the node where the virtual machine is running.

For the first two sets of experiments, we need to provide Haizea with values for s and r. We determined these values by suspending and resuming a single virtual machine (with $m = 1024$) 25 times and measuring the times v_s and v_r to suspend and resume that single VM (these times were extracted by parsing the Xen logs). When $f = \mathsf{local}$, $\overline{v_s} = 15.4$ ($\sigma = 0.58$) and $\overline{v_r} = 14.12$ ($\sigma = 0.67$). When $f = \mathsf{global}$, $\overline{v_s} = 14.00$ ($\sigma = 1.04$) and $\overline{v_r} = 11.60$ ($\sigma = 0.50$). We conservatively estimate s to be $\frac{m}{\overline{v_s}+2\cdot\sigma_{v_s}}$ and estimate r similarly. Thus, for $f = \mathsf{local}$, s is 61.86 MB/s and r is 66.27 MB/s; for $f = \mathsf{global}$, s is 63.67 MB/s and r is 81.27 MB/s.

4.2 Experiment #1: Suspending and Resuming Leases

In our first experiment, two leases are scheduled on our hardware: a best-effort lease (BE_LEASE) initially scheduled to use all available resources, which is preempted to free up resources for an AR lease (AR_LEASE), which involves suspending all the VMs in BE_LEASE, and then resuming them once AR_LEASE ends. The purpose of this experiment is to measure how accurately lease suspension and resumption times are estimated, comparing the value predicted by our model to the actual times measured on our testbed. Although these results highlight only how these leases are scheduled on our testbed, they provide insights into which factors can have an impact on the accuracy of the model.

Both leases are requested at the start of the experiment and require all the physical resources on the testbed. AR_LEASE has a duration of 5 minutes and must start 15 minutes into the experiment. BE_LEASE has a duration of 20 minutes; and since there are no other leases at the start of the experiment, BE_LEASE can start immediately (but will be preempted by AR_LEASE when it starts 15 minutes into the experiment; see Figure 1). In this experiment we explore the following three parameters:

- $c \in \{1, 2, 4, 8\}$: The number of VMs per physical node. Since each lease uses all available resources, then N, the total number of VMs, will be $c \cdot P$.
- $f \in \{\mathsf{local}, \mathsf{global}\}$: as defined in Section 3.1.

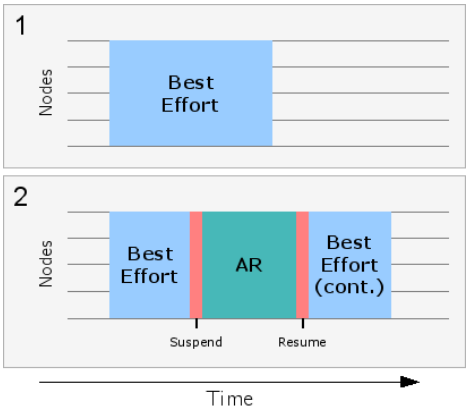


Figure 1. (1) A best-effort lease is scheduled to use all available nodes. (2) An AR lease is requested at the time when the best-effort lease is using the testbed resources, necessitating preempting the best-effort lease. The best-effort lease is suspended before the AR and then resumed once the AR ends.

- $m \in \{512, 768, 1024\}$: as defined in Section 3.1.

This experiment is performed in 22 configurations (one for each combination of parameters, except those with $c = 8$ and $m = 1024$; since the Xen Dom0 domain uses 512 MB of memory, we cannot start VMs that consume a total of 8192 MB), and each configuration is run five times. In each run, we measure the following metrics:

- $\tilde{t}_s$: The *observed* time used to suspend BE_LEASE. We obtain this time by pinging each VM in the lease every 2 seconds and recording whether it responds. We analyze the sequence of responses to determine how much time the lease took to suspend.
- a_s: The *accuracy* of suspension, or how close the observed time to suspend was to the predicted time. Thus, we define a_s as $\frac{\tilde{t}_s}{t_s}$. A value of 1.0 indicates perfect accuracy. Values less than 1.0 indicate the time was *overestimated*, meaning that the VMs in the lease finished suspending earlier than the time predicted by the model; overestimation has the effect of leaving resources idle between the end of the suspension and the start of the AR. Values larger than 1.0 indicate the time was *underestimated*, meaning the lease took longer to suspend than estimated; underestimation has the effect of delaying the start of the AR.

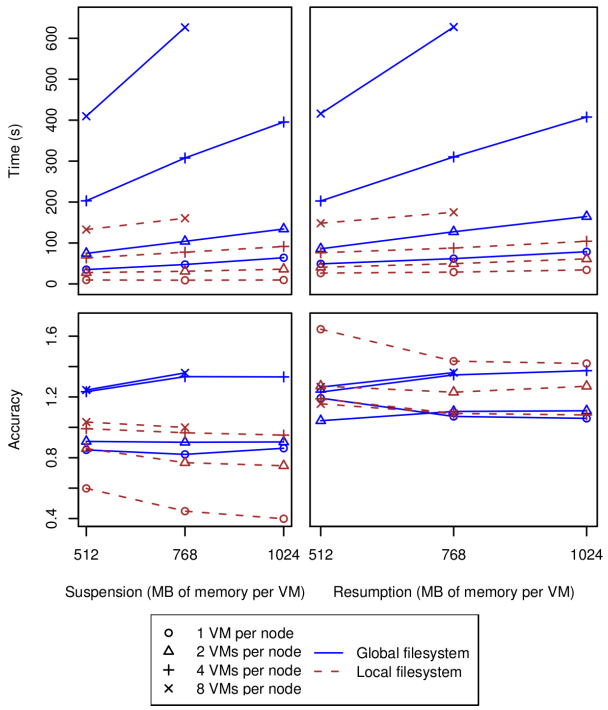


Figure 2. Experiment #1: Observed time to suspend and resume an entire lease (top) and accuracy of estimation (bottom)

- $\tilde{v}_s$: The *observed* time used to suspend a single VM in Be_Lease. We parse the Xen log files to determine this time.
- $\tilde{t}_r, a_r, \tilde{v}_r$: Defined similarly for resumption.

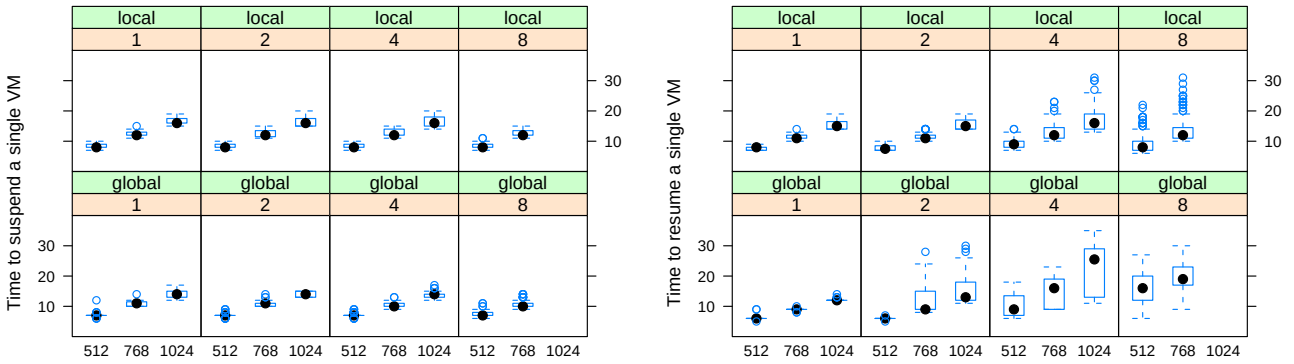
Figure 2 shows the average values for $\tilde{t}_s$, $\tilde{t}_r$, a_s, and a_r in each experiment configuration. The averages are taken from the five runs of each configuration. The standard deviation is not shown in the graphs but is at most 13.9% of the average (with most values below 10%). These graphs show that, as expected, the time to suspend and resume increases with the amount of memory requested, with the rate of increase being more pronounced when $f = \mathsf{global}$. The observed times show that the model tends to overestimate the time to suspend, with only six configurations having an accuracy larger that 1.0 (overestimation), with a maximum value of 1.39 (i.e., the lease took 39% longer to suspend than estimated). On the other hand, resumption times are all underestimated, with all accuracies above 1.0 and a maximum value of 1.64.

The explanation for these values can be found by looking at how individual VMs behave when suspending and resuming. Figures 3a and 3b show the distribution of values for $\tilde{v}_s$ and $\tilde{v}_r$, respectively. We see that values $\tilde{v}_s$ show little dispersion. However, the graph does not show 19 outliers (out of 1388 measurements) with times ranging between 45 seconds and 370 seconds. By inspecting the Xen logs, we found that these are caused by suspensions that, for no apparent reason, Xen will block on. In other words, Xen correctly receives and processes the suspension command, including pausing the VM, but does not actually save the memory state to disk until an arbitrary amount of time has passed (although it does not block all other operations; e.g., other suspensions are processed correctly). We do not know why this situation occurs, and we have found no mention of it in other work.

On the other hand, the values of $\tilde{v}_r$ tend to become more dispersed as c increases. We have found that the direct cause is resource contention between overlapping resumptions. Although the scheduler plans the resumptions in such a way that they will not overlap (either globally or locally, depending on f), sometimes a resumption might take slightly longer than estimated, affecting the time of the next planned resumption, delaying both in the process. We found that the underlying cause is the following:

1. *The shutdown of* Ar_Lease *can overlap with the first resumptions.* If Ar_Lease is still in the process of shutting down when the resumptions start, there will be contention for resources, delaying the first resumption. In our experiments, we set h to be a fixed value (15) regardless of the number of nodes in a lease, and this turned out to be insufficient when $c = 8$ (even assuming that a single VM can be shut down in 1 second, with a 1-second enactment overhead, that still adds up to 64 seconds).
2. *Delays in enactment commands.* Occasionally, enactment commands sent from OpenNebula (which uses SSH to send commands) would be delayed by the SSH server itself, sometimes up to 10 seconds.
3. *The larger c is, the greater the likelihood of a cascade effect.* For larger values of c, the more likely it is that a delayed resumption will affect other resumptions, resulting in more dispersed values overall. This situation is especially true for the shutdown overlap, where delaying the first resumption could delay up to 31 other resumptions thereafter (when $c = 8$ and $f = \mathsf{global}$).

As we can see, the above factors can have an impact on the time to suspend and resume leases and can potentially delay other leases, such as AR leases that depend on other leases being preempted before



(a) Suspension times

(b) Resumption times

Figure 3. Experiment #1: Distribution of times required to suspend and resume individual VMs, for all combinations of cores per physical nodes (1, 2, 4, 8), memory used by each VM (1024, 2048, 3076, 4096), and filesystem used (local or global).

they can start. We are currently working on minimizing the impact of these factors by making Haizea more adaptive, by reacting to suspensions and resumptions that take longer than expected, to avoid contention for resources when these operations take place. We also plan to improve the way enactment commands are sent from OpenNebula, by exploring mechanisms that are not based on SSH (reducing the overhead of each command) and that can distribute commands to n hosts in sublinear time.

4.3 Experiment #2: Long-term effects

The previous experiment explored small, self-contained cases on real hardware to validate the accuracy of our model in predicting and scheduling the overhead of preempting a lease. However, it reveals nothing about the long-term implications of using resource leases and how changes in the model's parameters can compound over time, particularly those that affect suspension/resumption times. In this second set of experiments, we run Haizea in simulation mode to process 30 days of lease requests.

The workloads we have used are a subset of those used in a previous Haizea paper [20]. These workloads were constructed by taking the first 30 days of requests from the SDSC Blue Horizon cluster job submission trace from the Parallel Workloads Archive [24] and treating each job request as a best-effort lease request. Next, we interleave with this set of best-effort requests a set of AR requests, generated according to three parameters. Here, ρ is the aggregate duration of all AR leases in a trace, computed as a percentage of the total CPU-hours in the simulation's run, which is the number of nodes multiplied by the time when the last best-effort request is submitted; δ is the average duration of each AR lease; and ν is the number of nodes requested by each lease. The interlease arrival interval is derived from the values for ρ, δ, and ν. Given a value for ρ, since the number of AR CPU-hours is fixed, smaller values of δ will result in more frequent requests (since there will be more AR lease requests).

In this paper, ν will always be a random value between 25 and 48 (the *medium* range in our previous paper), and we use only three combinations of the other two parameters:

- T10: $\rho = 10\%, \delta$ =3 hours
- T20: $\rho = 20\%, \delta$ =2 hours
- T30: $\rho = 30\%, \delta$ =1 hour

These three workloads were chosen because they test two extremes of ARs. Given that, according to the Parallel Workloads Archive the SDSC Blue Horizon trace's utilization is 76.2%, T10 introduces a relatively small amount of ARs, with more time between each reservation, whereas T30 introduces a large amount of ARs. T20 is meant as an intermediate point between the two. We limit our discussion to these three workloads because the effect of varying ρ, δ, and ν was already explored in our previous paper, and the focus of this experiment is on the effect of other parameters.

The simulated cluster in our experiment is modeled after the SDSC Blue Horizon cluster. However, whereas our previous paper assumed a single processor per physical node ($C = 1$) and that memory state files are saved to a local filesystem ($f = \mathsf{local}$), here we allow C to equal 1, 2, 4, or 8 (and vary P accordingly so the total number of cores in the cluster is 144). We also allow memory state files to be saved to a global filesystem. Furthermore, we assume the values for s and r are the same as the ones in our testbed (indicated in Section 4.1). Moreover, we assume that VM disk images do not need to be deployed before the start of a lease (e.g., because they are on a global filesystem or are predeployed to the local filesystems; this corresponds to configuration VM–PREDEPLOY in our previous paper).

In sum, the parameters in this experiment are the following:

- trace $\in \{\mathsf{No\ ARs}, \mathsf{T10}, \mathsf{T20}, \mathsf{T30}\}$
- $C \in \{1, 2, 4, 8\}$ (number of cores per physical node)
- $f \in \{\mathsf{local}, \mathsf{global}\}$
- $m \in \{1024, 2048, 3076, 4096\}$

In this experiment, we explore all 128 combinations of these four parameters. In each run, we record for each lease the following values: the arrival time, or time when the lease request is submitted (t_a), the time at which the lease starts (t_b), and the time the lease ends (t_e). We define the all-best-effort metric as the time when the last best-effort lease is completed, or $\max(t_e)$. For a given workload, this value is normalized by presenting the relative difference between this time and the time required to process the same workload under the assumption that suspension and resumption can be done instantaneously (e.g., a value of 1.1 indicates that processing the entire workload took 10% longer than if we could suspend and resume virtual machines instantaneously).

Figure 4 shows the values of all-best-effort for each combination of the experiment parameters. We see that when using traces T10 and T20, the effect of all parameters on the running time of the best-effort leases is relatively small, with all-best-effort being at most 1.018. However, with trace T30, where preemptions are more likely to happen, the effect is more noticeable, especially when using a global filesystem, with values of all-best-effort up to 1.10 (for this 30-day workload, this translates to 3.37 extra days of work relative to the baseline).

Table 1. Experiment #3: Effect of modifying the bandwidth

Workload	1000 Mbps	100 Mbps	Difference
No ARs	0.99	1.00	+0.1%
T10	0.99	1.00	+0.7%
T20	1.00	1.05	+5.1%
T30	1.04	1.16	+11.1%

Another parameter that may affect performance is the network bandwidth, which directly affects s and r when $f = \mathsf{global}$. Fixing $C = 1$ and $m = 1024$, we reran the simulations assuming a 100 Mbps Ethernet network. Since we do not have a 100 Mbps network in our testbed with which to determine the values of s and r, we assume that they will be one-tenth of what they are in our 1000 Mbps network (i.e., $s = 6.367$ MB/s, and r is 8.127 MB/s). Table 1 summarizes the results of these simulations. As we can see, the effect on all-best-effort is still small with workload T10 but more noticeable when using workloads T20 and T30.

5 Conclusions

We have described a model for estimating the time required to suspend and resume a resource lease implemented with virtual machines. Such a model is necessary to accurately and efficiently support VM-based resource lease preemption, where a given lease may need to start at a specific time, requiring other leases to be preempted to free up resources. In order to guarantee that those resources are freed up on time, the time to suspend the VMs must be accurately estimated.

Through our experiments, we have shown that while our model underestimates suspension times in only a few cases (which would delay the start of an AR), it also tends to overestimate suspension times, resulting in idle times between the end of the suspension and the start of the AR. Moreover, our model tends to underestimate resumption times. However, we found that these inaccurate estimations were due to specific factors — overlapping of shutdowns and resumptions, delays in enactment commands, and cascade effects — highlighting how an incomplete model can have a significant impact, not just on lease resumption times, but also on the rest of the schedule. In the short term, we will work on refining our model and minimizing the impact of these factors. In particular, we are focusing on making Haizea more adaptive to unexpected events, such as suspensions/resumptions that take longer than expected (e.g., because of a delay in an enactment com-

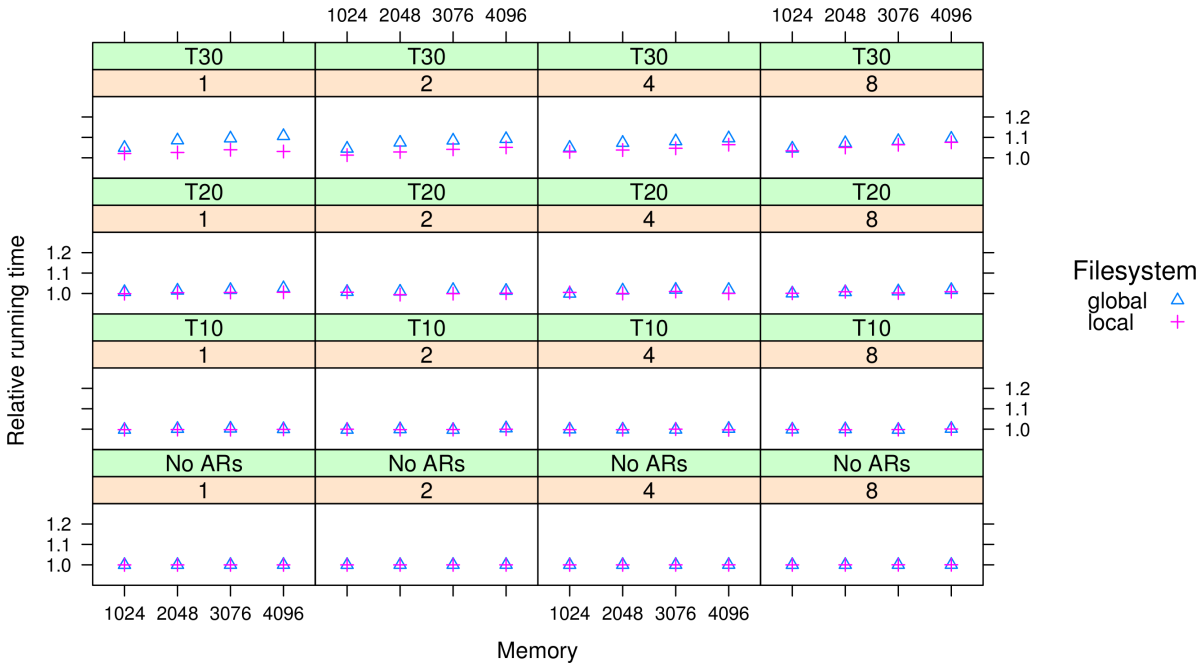


Figure 4. Experiment #3: Values of the all-best-effort **metric for all the combinations of workloads (**No ARs**,**T10**,**T20**,**T30**), cores per physical nodes (1, 2, 4, 8), memory used by each VM (1024, 2048, 3076, 4096), and filesystem used (local or global).**

mand), to avoid contention for resources when these operations take place. Additionally, in this paper we explored only one model, along with fixed values for s and r; we plan to explore different estimation models and variable values of s and r, showing their effect on the probability that ARs will be delayed (instead of presenting only the degree of accuracy of the estimations). In our simulation experiments, we have shown that varying some of the parameters in our model has relatively small long-term effects on performance, as measured by the all-best-effort metric, except when using workloads with a large amount of ARs, which results in a larger number of lease preemptions (in our case, the T30 required that 30% of resources be devoted to ARs, while the best-effort requests required 76.2% of resources). However, we still need to analyze the long-term effects on other metrics, such as waiting times and slowdowns in best-effort requests.

Besides addressing the issues identified above, we will extend the model to cover more cases. In particular, we are interested in rerunning our experiments using KVM virtual machines, instead of Xen virtual machines, to observe whether suspension and resumption times behave the same way or whether additional factors have to be taken in account.

Acknowledgments

We gratefully acknowledge the hard work of the OpenNebula developers Javier Fontán and Tino Vázquez. Development of OpenNebula is supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, and through the research grant TIN2006-02806, and by the European Union through the research grant RESERVOIR Grant Number 215605. Development of Haizea is supported by RESERVOIR, the University of Chicago, and the U.S. Department of Energy under Contract DE-AC02-06CH11357. Early work on Haizea was done in collaboration with Dr. Kate Keahey (Argonne National Laboratory) and funded by NSF grant #509408 "Virtual Playgrounds."

References

[1] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. From vir-

tualized resources to virtual computing grids: The In-VIGO system. *Future Gener. Comput. Syst.*, 21(6):896–909, June 2005.

[2] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing SLA violations. *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on Integrated Management*, pages 119–128, 2007.

[3] W. Emeneker and D. Stanzione. Efficient Virtual Machine Caching in Dynamic Virtual Clusters. In *SRMPDS Workshop, ICAPDS 2007 Conference*, December 2007.

[4] N. Fallenbeck, H.-J. Picht, M. Smith, and B. Freisleben. Xen and the art of cluster scheduling. In *VTDC '06: Proceedings of the 1st International Workshop on Virtualization Technology in Distributed Computing*. IEEE Computer Society, 2006.

[5] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, 1999.

[6] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science clouds: Early experiences in cloud computing for scientific applications. In *Cloud Computing and Applications 2008 (CCA08)*, 2008.

[7] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *eScience 2008*, 2008.

[8] N. Kiyanclar, G. A. Koenig, and W. Yurcik. Maestro-VC: A paravirtualized execution environment for secure on-demand cluster computing. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, page 28. IEEE Computer Society, 2006.

[9] I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In *SC '04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, page 7. IEEE Computer Society, 2004.

[10] I. Llorente, R. Moreno-Vozmediano, and R. Montero. Cloud computing for on-demand grid resource provisioning. To appear in *Advances in Parallel Computing*, 2009.

[11] M. W. Margo, K. Yoshimoto, P. Kovatch, and P. Andrews. Impact of reservations on production job scheduling. In *13th Workshop on Job Scheduling Strategies for Parallel Processing*, 2007.

[12] S. Mehta and A. Neogi. Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers. *Network Operations and Management Symposium, 2008. IEEE*, April 2008.

[13] H. Nishimura, N. Maruyama, and S. Matsuoka. Virtual clusters on the fly — fast, scalable, and flexible installation. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2007.

[14] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cloud Computing and Applications 2008 (CCA08)*, 2008.

[15] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galán. The reservoir model and architecture for open federated cloud computing. *IBM Systems Journal*, October 2008.

[16] P. Ruth, P. McGachey, and D. Xu. VioCluster: Virtualization for dynamic computational domains. *Proceedings of the IEEE International Conference on Cluster Computing (Cluster'05)*, 2005.

[17] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. *IEEE International Conference on Autonomic Computing*, 2006.

[18] W. Smith, I. Foster, and V. Taylor. Scheduling with advanced reservations. In *IPDPS '00: Proceedings of the 14th International Symposium on Parallel and Distributed Processing*, page 127. IEEE Computer Society, 2000.

[19] Q. Snell, M. J. Clement, D. B. Jackson, and C. Gregory. The performance impact of advance reservation meta-scheduling. In *IPDPS '00/JSSPP '00: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 137–153, London, UK, 2000. Springer-Verlag.

[20] B. Sotomayor, K. Keahey, and I. Foster. Combining batch execution and leasing using virtual machines. In *HPDC '08: Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 87–96. ACM, 2008.

[21] B. Sotomayor, K. Keahey, I. Foster, and T. Freeman. Enabling cost-effective resource leases with virtual machines. In *Hot Topics session in ACM/IEEE International Symposium on High Performance Distributed Computing 2007 (HPDC 2007)*, 2007.

[22] J. P. Walters, B. Bantwal, and V. Chaudhary. Enabling interactive jobs in virtualized data centers. In *Cloud Computing and Applications 2008 (CCA08)*, 2008.

[23] S. Yamasaki, N. Maruyama, and S. Matsuoka. Model-based resource selection for efficient virtual cluster deployment. In *VTDC '07: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, 2007.

[24] Parallel workloads archive. `http://www.cs.huji.ac.il/labs/parallel/workload/`.