Capacity Leasing in Cloud Systems using the OpenNebula Engine

Borja Sotomayor[1,2], Rubén Santiago Montero[1], Ignacio Martín Llorente[1], and Ian Foster[2,3]

[1]Facultad de Informática, Universidad Complutense de Madrid (Madrid, Spain),
{rubensm,llorente}@dacya.ucm.es
[2]Department of Computer Science, University of Chicago (Chicago, IL),
{borja,foster}@cs.uchicago.edu
[3]Mathematics and Computer Science Div., Argonne National Laboratory (Argonne, IL),
foster@mcs.anl.gov

1 Introduction

Clouds can be used to provide on-demand capacity as a utility. Although the realization of this idea can differ among various cloud providers (from Google App Engine[1] to Amazon EC2[2]), the most flexible approach is the provisioning of virtualized resources as a service. These virtualization-based clouds, like Amazon EC2 or the Science Clouds[3] (which uses the Globus Virtual Workspace Service [4]), provide a way to build a large computing infrastructure by accessing remote computational, storage and network resources. Since a cloud typically comprises a large amount of virtual and physical servers, in the order of hundreds or thousands, efficiently managing this virtual infrastructure becomes a major concern. Several solutions, such as VMWare VirtualCenter, Platform Orchestrator, or Enomalism, have emerged to manage virtual infrastructures, providing a centralized control platform for the automatic deployment and monitoring of virtual machines (VMs) in resource pools.

However, these solutions provide simple VM placement and load balancing policies. In particular, existing clouds use an immediate provisioning model, where virtualized resources are allocated at the time they are requested, without the possibility of requesting resources at a specific future time and, at most, being placed in a simple first-come-first-serve queue when no resources are available. However, service provisioning clouds, like the one being built by the RESERVOIR project[4], have requirements that cannot be supported within this model, such as resource requests that are subject to non-trivial policies, capacity reservations at specific times to meet peak capacity requirements, variable resource usage throughout a VM's lifetime, and dynamic renegotiation of resources allocated to VMs. Additionally, smaller clouds with limited resources, where not all requests may be satisfiable immediately for lack of resources, could benefit from more complex VM placement strategies supporting queues, priorities, and advance reservations.

In this work we explore extending the capacity provisioning model used in current clouds by using *resource leases* [3, 10, 9] as a fundamental provisioning abstraction. To do this, we have integrated the OpenNebula[5] virtual infrastructure engine with the Haizea[6] lease manager to produce a resource management system that can be used to support a variety of leases in clouds. We focus in this work on advance reservation leases, which can be used to satisfy capacity peaks known in advance, or for a variety of well-documented use cases where advance reservations are used (such as coscheduling of multiple resources [12, 5, 1, 2], urgent

[1] http://code.google.com/appengine/

[2] http://www.amazon.com/ec2/

[3] http://workspace.globus.org/clouds/

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

[5] http://www.opennebula.org/

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

computing applications [6], and reservations for multilevel scheduling [7, 11]). Previous work by Sotomayor et al. [9] presented results obtained from scheduling workloads that include advance reservation leases, but did so only in simulation. In this work, we present preliminary experimental results that show the effect of preempting an existing lease on a virtualized physical testbed to satisfy the requirements of an advance reservation lease.

2 OpenNebula

The open source OpenNebula virtual infrastructure engine 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 infrastructure configurations.

OpenNebula is composed of three main components. The *OpenNebula Core* is a centralized component that manages the life-cycle of a VM by performing basic VM operations (e.g. deployment, monitoring, migration or termination). The core also provides a basic management and monitoring interface for the physical hosts. The *Capacity Manager* is a pluggable module that governs the functionality provided by the OpenNebula core. The capacity manager adjusts the placement of VMs based on a set of pre-defined policies. The default capacity scheduler implements a simple matchmaking policy and supports user-driven consolidation constraints. In order to provide an abstraction of the underlying virtualization layer, OpenNebula uses pluggable *Virtualizer Access Drivers* that expose the basic functionality of the hypervisor (e.g. deploy, monitor or shutdown a VM). Thus, OpenNebula is not tied to any specific environment, providing a uniform management layer regardless of the virtualization technology used.

In this way, OpenNebula shapes a physical infrastructure to support the execution of a given service workload. Moreover, OpenNebula is able to dynamically scale-out this infrastructure by interfacing with an external cloud; an Amazon EC2 virtualizer driver is currently included with OpenNebula, and drivers could be developed to interface with other clouds (such as the Science Clouds, through the Globus Workspace Service interface). This seamless integration of an external cloud with in-house resources allows for effective access of outsourced computational capacity.

3 Haizea

Haizea is an open source lease management architecture developed by Sotomayor et al. [9] that OpenNebula can use as a scheduling backend. Haizea uses *leases* as a fundamental resource provisioning abstraction, and implements those leases as virtual machines, taking into account the overhead of using virtual machines (e.g., deploying a disk image for a VM) when scheduling leases. Using OpenNebula with Haizea allows resource providers to lease their resources, using potentially complex lease terms, instead of only allowing users to request VMs that must start immediately.

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" [9]. In Haizea, the lease terms include the hardware resources, software environments, and the availability period during which the hardware and software resources must be available. Currently, Haizea supports *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 (this corresponds to the type of provisioning currently supported by OpenNebula's default scheduler). 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.

To use Haizea as an OpenNebula scheduling backend, a resource provider simply has to run Haizea instead of OpenNebula's default Capacity Manager. To request a lease, instead of a VM that must start

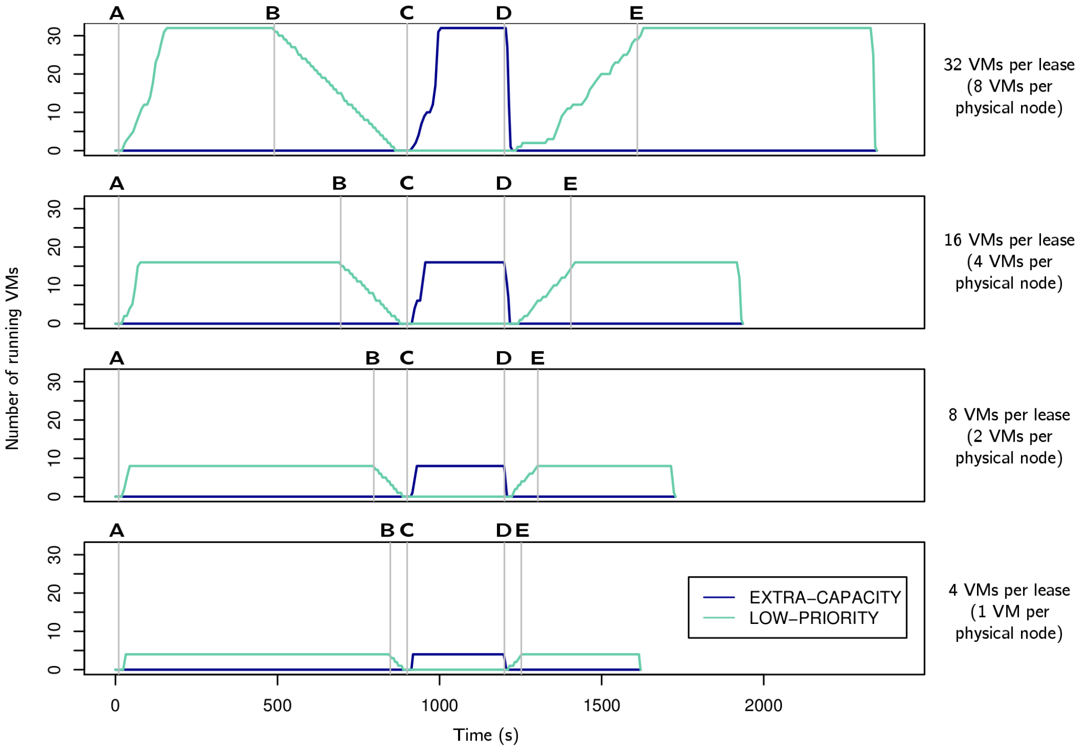

Figure 1: Number of VMs available for each of the two leases through time. LOW–PRIORITY is scheduled to start at **A** and is suspended between **B** and **C** (the time scheduled for the suspend operation varies with the number of VMs to suspend), in time for EXTRA–CAPACITY to start at **C**. At **D**, lease EXTRA–CAPACITY ends and the resumption of LOW–PRIORITY takes place from **D** to **E** (again, the time varies with the number of VMs). After **E**, LOW–PRIORITY continues until it completes.

immediately, users have to specify an `HAIZEA` option in the OpenNebula request. The `HAIZEA` option allows users to request the three types of leases described above. Haizea currently works with OpenNebula by polling it for new requests, and sending OpenNebula enactment commands (start VM, stop VM, etc.) and polling for any changes in state (e.g., VMs that end prematurely). This will be replaced in future versions by event-based communication that will allow both pieces of software to keep consistent state.

4 Experiments

We have tested OpenNebula's support for leases by running simple workloads that combine different types of leases. We used OpenNebula 1.0 and Haizea Technology Preview 1.1 (the latest versions at the time) on a testbed of five SunFire x4150 servers, each with two Intel Xeon QuadCore L5335 2GHz processors (i.e., 8 cores per server) and 8GB of RAM. 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. The remaining four nodes all run Xen 3.2 (i.e., 32 cores are available to run VMs). The disk images needed by the VMs and the memory image files (created when a VM is suspended) are stored on the NFS filesystem.

Our experiment involves submitting two leases to OpenNebula/Haizea: LOW–PRIORITY and EXTRA–CAPACITY. The former is a lease for low–priority work, while the latter is a lease to handle a spike in capacity requirements that is known in advance. LOW–PRIORITY needs to be available for 20 minutes, requires N nodes, each with P CPUs and M MB of memory, and is requested as a preemptible best-effort lease at the start of the experiment. In our experiments, we set M to 512MB, and tested values of N and P equal to $(4, 8)$, $(8, 4)$, $(16, 2)$, and $(32, 1)$ (i.e., the lease always requires all 32 cores, and we vary whether 1, 2, 4, or 8 VMs can run in each physical node). Additionally, the lease's VMs use copies of a 2GB Debian Etch disk image, located on the NFS filesystem. EXTRA–CAPACITY has the same resource requirements, but the resources are only needed from 00:15:00 to 00:20:00 (where 00:00:00 is the start of the experiment). Thus, an advance reservation lease is requested at the start of the experiment for that timeframe. Since EXTRA–CAPACITY also requires all 32 cores, Haizea will have to preempt LOW–PRIORITY by suspending all its VMs for the duration of EXTRA–CAPACITY. Knowing that the read/write throughput of our NFS filesystem is 40MB/s, Haizea estimates the time to suspend as $\frac{M \cdot N}{40}$ (i.e., the time to save to disk the memory of all the virtual machines in the lease).

We are interested in observing the effect of preempting a lease, using VM suspend/resume, in favor of another lease. We ping all the VMs in both leases, every five seconds, and consider a VM to be available if it responds to pings. Figure 1 shows the availability of the leases' VMs throughout the experiment in the four configurations we tested. We observed that, while the VMs boot and shutdown promptly, despite storing the disk images in NFS, suspending and resuming from NFS (which, in this experiment involves writing/reading up to $32 \cdot 512 = 16,384$MB to NFS) becomes an important bottleneck, specially since the suspend time will increase with the number of VMs to be suspended. The version of OpenNebula we used requires a global filesystem, necessitating NFS to store the memory files, although this assumption will be removed in future versions (allowing memory files to be saved locally and, if necessary, migrated elsewhere).

In the results shown here, the VMs are suspended and resumed sequentially, so only one memory file is read/written from/to NFS at any given time. In other experiments (not shown here for lack of space), we observed that suspending/resuming all the VMs simultaneously led to unpredictable suspend/resume times, and even Xen failures, specially when 8 VMs had to be suspended simultaneously on the same physical node. We also observed that allowing the suspension of the LOW–PRIORITY VMs to overlap with the start of the EXTRA–CAPACITY VMs (even partially) would delay their booting process.

5 Conclusions and future work

We have shown that OpenNebula and Haizea can be used together to provide a VM management solution supporting a variety of lease types, and have presented experimental results showing how an advance reservation lease can be created by leveraging the suspend/resume capability of VMs to preempt lower-priority leases. Our results stress the importance of accurately and separately scheduling not just VM deployment overhead (which had been previously explored by Sotomayor et al. [8, 9]) but also *runtime* overhead, such as suspend/resume actions. By accurately estimating the time to perform these actions, Haizea can schedule a suspension to complete before the start of a preempting lease, avoiding performance hits resulting from overlaps in VM actions. Greater accuracy could be achieved by also modelling and scheduling the time to boot the VMs (the VMs currently boot at the start of the lease). Nonetheless, the results presented here are preliminary, and future work will explore workloads with more leases and combinations of different types of leases. We will also explore policies to resolve conflicts in resource requirements between leases, and what other types of leases, besides the ones described here, are required in cloud systems.

On the technological side, the integration between OpenNebula and Haizea is still in a relatively early stage. While Haizea's resource model supports scheduling leases with multiple nodes in parallel, managing the transfer of VM images, and assumes that, when suspending a VM, its memory can be saved to local disk (instead of a global file system), OpenNebula currently does not support these features natively. The next release of OpenNebula 1.2 will support the concept of "VM groups", which will allow Haizea to manage the VMs in a lease as a group, instead of each VM individually. Future versions of OpenNebula will also allow memory state to be saved to local disk, and will also allow OpenNebula to hook into *transfer managers* to handle the deployment of VM disk images. Ongoing work also focuses on improving Haizea's scheduling algorithms and data structures to allow them to react to hardware failures or unscheduled events (e.g., a VM suspension that takes longer than estimated). Most of the future work on OpenNebula and Haizea will be driven by the requirements of the RESERVOIR project.

6 Acknowledgements

We gratefully acknowledge the hard work of the OpenNebula developers: Javier Fontán, Luis González, and Tino Vázquez. This research was 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.

References

[1] Karl Czajkowski, Ian Foster, and Carl Kesselman. Resource co-allocation in computational grids. In *HPDC '99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, page 37, Washington, DC, USA, 1999. IEEE Computer Society.

[2] 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.

[3] David Irwin, Jeffrey Chase, Laura Grit, Aydan Yumerefendi, David Becker, and Kenneth G. Yocum. Sharing networked resources with brokered leases. In *USENIX Technical Conference*, June 2006.

[4] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life on the grid. *Scientific Programming*, 13(4):265–276, 2005.

[5] Martin W. Margo, Kenneth Yoshimoto, Patricia Kovatch, and Phil Andrews. Impact of reservations on production job scheduling. In *13th Workshop on Job Scheduling Strategies for Parallel Processing*, 2007.

[6] P.Beckman, S.Nadella, N.Trebon, and I.Beschastnikh. SPRUCE: A system for supporting urgent high-performance computing. *IFIP International Federation for Information Processing, Grid-Based Problem Solving Environments*, 239:295–311, 2007.

[7] G. Singh, C. Kesselman, and E. Deelman. Performance impact of resource provisioning on workflows. Technical Report 05-850, Department of Computer Science, University of South California, 2005.

[8] Borja Sotomayor, Kate Keahey, and Ian Foster. Overhead matters: A model for virtual resource management. In *VTDC '06: Proceedings of the 1st International Workshop on Virtualization Technology in Distributed Computing*, page 5, Washington, DC, USA, 2006. IEEE Computer Society.

[9] Borja Sotomayor, Kate Keahey, and Ian 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, New York, NY, USA, 2008. ACM.

[10] Borja Sotomayor, Kate Keahey, Ian Foster, and Tim 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.

[11] Henan Zhao and Rizos Sakellariou. Advance reservation policies for workflows. In *12th Workshop on Job Scheduling Strategies for Parallel Processing*, 2006.

[12] Final report. teragrid co-scheduling/metascheduling requirements analysis team. `http://www.teragridforum.org/mediawiki/images/b/b4/MetaschedRatReport.pdf`.