Enabling Cost-Effective Resource Leases with Virtual Machines

Borja Sotomayor[1], Kate Keahey[1,2], Ian Foster[1,2], Tim Freeman[1,2]
[1]*University of Chicago, Chicago, IL*
[2]*Argonne National Laboratory, Argonne, IL*
borja@cs.uchicago.edu {keahey, foster, tfreeman}@mcs.anl.gov

1. Introduction

Leasing resources for short periods of time can be of great value to many applications. Applications consisting of workflows of small tasks (such as Montage [5], GADU [6] or fMRI [7]), can be more efficiently scheduled by a workflow engine (e.g., Pegasus [8] or Swift [9]) when using leased resources than when each request must pass via a traditional scheduler. Interactive applications (where the user must use the application at a specific time), real-time applications, or applications requiring resource coscheduling (e.g., to provision resources for a parallel job running across several sites) may further require an advance reservation capability [18]. While the first group of requirements can be addressed to some extent by using multi-level scheduling (e.g., as implemented by Condor [10], MyCluster [12], and Falkon [13]), or task clustering [11], achieving stricter lease semantics (such as advance reservations) is typically difficult. This is because advance reservations often lead to utilization problems in the scheduler caused by the need to "drain" jobs off of a group of resources before the job/reservation starts.

We argue that the use of virtualization can help overcome those problems. The ability of virtual machines (VMs) to seamlessly suspend and resume computations can enable a scheduler to use batch computations, which have loosely defined availability requirements, to backfill around leases with strict availability requirements, such as advance reservations. Providing cost-effective leasing mechanisms, which would also use VMs to allow deployment of arbitrary software environments, would increase the usefulness of short-term leases to clients and make providing them more attractive to the resource provider.

We also argue that, by making leasing more cost-effective, we can support a multi-level scheduling model, by decoupling resource provisioning from execution management [1]. The ability to provision short-term leases creates an opportunity for scientific applications that require multi-level scheduling to support application-specific scheduling (e.g., a Swift [9] workflow where groups of tasks are managed by the Falkon [13] execution framework, not by a site-specific scheduler).

Thus, we propose an architecture that uses VMs to make on-demand short-term leasing of resources cost-effective. This architecture allows resource providers to satisfy short-term leasing requests while continuing to support existing workloads (i.e., batch processing). We show via experiment that using virtualization for this purpose can achieve improved performance, both from the provider's perspective (throughput) and the user's perspective (running time), even in the presence of overhead incurred by using VMs. Our approach also allows a provider to both offer leases and run jobs associated with a particular execution environment (implemented by a VM) and rapidly switch between such software environments, providing added incentives for resource leasing.

This work is done in the context of our research into virtual workspaces [1, 2]. We represent and manage short-term leases as VM-based virtual workspaces. Our results build upon previous work that explored the combination of workspaces with traditional batch computation [3, 4].

2. Approach

Our architecture enables a multi-level scheduling approach by separating resource provisioning from job management and providing interfaces for each. The former is handled by a *lease manager* component developed by us, while the latter can be handled by the resource provider's existing scheduler. We can extend existing schedulers [16, 17] to support virtualization in such a way that resource providers can provide short-term leases to their users in a cost-effective way, while continuing to use their existing job management software stack for batch computations.

In our system, resource provisioning is handled by the lease manager, which more properly becomes the site's LRM. Users can request short-term leases directly to the lease manager, and use the allocated resources for any purpose, including task-driven computations where users could deploy their own task managers, and not necessarily rely on the site's existing one. Users can continue to submit batch requests through the existing LRM, which can internally leverage the lease manager to provision resources.

Our leasing semantics can be more expressive than those provided by existing schedulers, allowing availability periods to be defined by a variety of agreed-upon events, such as specific timer events (a lease that must start at a specific time) or resource events (best-effort provisioning as resource become available). We will also explore resource renegotiation semantics for resource leases.

3. Implementation

We are implementing a proof-of-concept that has already resulted in encouraging results [4]. Our ongoing implementation work shows that, by using the suspend/resume capability of virtual machines, batch computations and timer-driven leasing requests (such as advance reservations) can be interleaved in such a way that resources do not have to be backfilled and "drained" before the start of a lease (see Figure 1), resulting in improved resource utilization. Suspending batch computations is not a novel concept, as many existing resource managers, such as Condor and SGE, allow checkpointing. However, this feature is primarily used for fault tolerance purposes and generally requires modifying a job's executable to explicitly support checkpointing, or depends on the presence of a checkpointing-capable operating system.

We aim to process batch workloads and short-term leases efficiently even in the presence of the runtime and deployment overhead associated with VMs. The latter results from the need to manage and transfer potentially large VM images, necessary to dynamically deploy different software environments, and is handled by integrating *application-specific* knowledge into our scheduling policies. In particular, our scheduling algorithms must be aware that VMs are being scheduled, which requires overhead (such as image transfers) to be managed adequately if we want to guarantee accuracy in meeting the availability requirements of leases.

4. Experiments

To validate the proposed architecture, we ran experiments that simulate both artificial and real workloads combining batch workloads with short-term

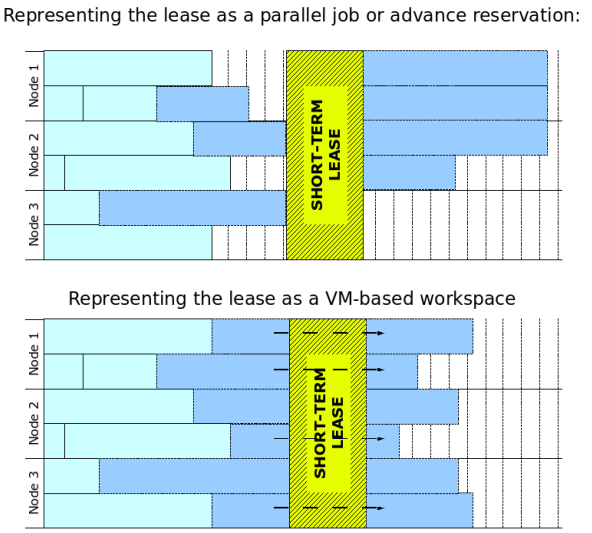

Figure 1: Difference between job and VM representation of a lease

leases. The real workloads are based on execution traces collected from the Jazz cluster [15] at Argonne National Laboratory. The experiments compare the performance obtained when using advance reservations (as currently supported by LRMs like PBS Pro and SGE) to represent short-term leases and the performance when those short-term leases are represented using VMs in our approach. We measure performance from the point of view of the resource provider (throughput, or the rate at which batch jobs and short-term leases are completed) and the point of view of the user (the completion time of jobs and short-term leases). These experiments are a continuation of previous work [3, 4].

Our preliminary results (not included due to lack of space; some of these results are presented elsewhere [4]) show that using the suspend/resume capability of VMs can result, under most conditions, in better performance even when accounting for the slowdown resulting from running inside a VM. These results also show that, even when working with a significant number of different VM images (which may result in considerable amounts of deployment overhead as large VM images need to be transferred to the nodes where they are needed), we can still achieve performance that is equal or better to running on physical nodes by adequately managing the deployment overhead.

5. References

[1] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid", *Scientific Programming Journal*, vol 13, No. 4, 2005, Special Issue: Dynamic Grids and Worldwide Computing, pp. 265-276.

[2] T. Freeman, K. Keahey, I. Foster, A. Rana, B. Sotomayor, F. Wuerthwein, "Division of Labor: Tools for Growth and Scalability of Grids", *ICSOC 2006*.

[3] B. Sotomayor, K. Keahey, I. Foster, "Overhead Matters: A Model for Virtual Resource Management", *VTDC 2006* (workshop), in Supercomputing 06, November 2006.

[4] B. Sotomayor, "A Resource Management Model for VM-Based Virtual Workspaces", Masters paper, University of Chicago, February 2007.

[5] G.B. Berriman, et al., "Montage: a Grid Enabled Engine for Delivering Custom Science-Grade Image Mosaics on Demand", *Proceedings of the SPIE Conference on Astronomical Telescopes and Instrumentation*, 2004.

[6] GADU. http://compbio.mcs.anl.gov/gaduvo/

[7] The Functional Magnetic Resonance Imaging Data Center. http://www.fmridc.org/

[8] Pegasus. http://pegasus.isi.edu/

[9] Swift. http://www.ci.uchicago.edu/swift/

[10] J. Frey, T. Tannenbaum, I. Foster, M. Frey, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Cluster Computing*, vol. 5, pp 237-246, 2002.

[11] G. Singh, C. Kesselman, E. Deelman, "Performance Impact of Resource Provisioning on Workflows", Technical Report 05-850, Department of Computer Science, University of South California, 2005.

[12] E. Walker, J.P. Gardner, V. Litvin, E.L. Turner, "Creating Personal Adaptive Clusters for Managing Scientific Tasks in a Distributed Computing Environment", *CLADE 2006* (workshop), in HPDC-15, June 2006.

[13] Raicu, I., Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, "Falkon: a Fast and Light-weight tasK executiON framework". Submitted to *SuperComputing 2007*.

[14] S. Srinivasan, R. Kettimuthu, V. Subrarnani, and P. Sadayappan, "Characterization of backfilling strategies for parallel job scheduling", *ICPPW 2002*.

[15] Jazz Cluster. http://www.lcrc.anl.gov/jazz/

[16] Sun Grid Engine. http://gridengine.sunsource.net/

[17] PBS. http://www.openpbs.org/

[18] W. Smith, I. Foster, V. Taylor, "Scheduling with Advanced Reservations", in *Proceedings of the 14th International Parallel and Distributed Processing Symposium*, May 2000.