Overhead Matters: A Model for Virtual Resource Management

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

Abstract

Virtual machines provide a promising vehicle for controlled sharing of physical resources, allowing us to instantiate a precisely defined virtual resource, configured with desired software configuration and hardware properties, on a set of physical resources. We describe a model of virtual machine provisioning in a Grid environment that allows us to define such virtual resources and efficiently instantiate them on a physical Grid infrastructure. We argue that to properly account for, and manage, the overhead resulting from instantiating and managing virtual resources, overhead must be scheduled at the same level as virtual resources, instead of being deducted from a user's resource allocation. We present preliminary results that demonstrate the benefits of such an approach.

1. Introduction

In most grid deployments today, clients only have limited control over the resource platform on which computations are performed. Two types of control are often lacking: control over the availability and quantity of the resource, on the one hand, and control over its software configuration, on the other.

Control over availability and quantity is particularly important for deadline-sensitive applications, in which a resource needs to be made available in response to a specific event, such as data becoming available from a sensor, a class starting at a specific time (in educational settings), and input from a human client. Such control can be provided via reservation mechanisms that allow clients to request that resources be available either immediately ("immediate reservation") or at a specified time in the future ("advance reservation").

Control over software configuration can reduce the barriers to the use of remote resources and thus increase demand for remote computing resources. Various approaches to automated, user-driven software configuration have been explored, but virtual machines are particularly useful in this regard.

With these requirements in mind, Keahey et al. defined *virtual workspaces* (VWs) [1], a construct that allows clients to negotiate the creation of a virtual computing resource with specified software environment and resource allocation. The workspace interface allows a remote client to negotiate and manage a virtual resource allocation securely using Web Services-based protocols for state access and management [2]. Virtual machines (VMs), such as Xen [3] and VMware [4], with their isolation and virtualization properties, provide a particularly promising platform for workspaces.

In this paper, we present and evaluate strategies designed to enable the accurate and efficient creation of VM-based virtual workspaces with specified availability and configuration. By "accurate," we mean that a request to create a virtual workspace at a particular time T (either immediately, or in the future) is satisfied at that time T, not later. By "efficient," we mean that the overheads incurred by the server(s) that process requests for virtual workspace creation are low. As we shall see, the often large size of virtual machine images can cause problems for both accuracy and efficiency. We show that by annotating virtual machine images with descriptive metadata, we can allow a scheduler to improve both accuracy and efficiency by prefetching images, caching images, and reusing images.

The rest of this paper is structured as follows. We begin, in Section 2, by describing the resource management scenarios that motivate our work. Section 3 explains our virtual resource model, and also introduces the concept of VM image templates. Section 4 describes how we implemented our virtual resource model, Section 5 presents our experimental results, Section 6 discusses related work, and Section 7 presents our conclusions.

2. Resource Management Scenarios

We argue for a model in which a virtual workspace associated with a well-defined resource allocation, in particular its availability, can be procured by negotiating an agreement with the resource provider, via for example WS-Agreement [5]. In other work [7], we focus on protocols and enforcement methods defining the shape of a resource platform (in terms of memory, CPU%, etc.). Here, we concern ourselves with availability.

Within that model we define availability as a period defined by agreed-upon events happening anytime while the agreement is valid. Such events may constitute for example the arrival of data, a user submitting a request, a scheduling event, or the expiration of a timer. The resource availability will be more or less strict depending on the definition and priority assigned to these events. Depending on the strictness of the definition, we encounter the following frequently occurring definitions of availability:

- *Batch platforms*: virtual resources suitable for running batch jobs have loosely defined availability requirements with open-ended start requirements and a variety of stop conditions. One particular case, which we explore in this paper, is platforms where the resources must be provisioned as soon as possible ("*ASAP*")
- *Advance Reservation (AR) platforms*, where both the start and stop of resource availability are clearly defined in advance by a time event, as described above.

We focus in this paper on AR platforms, and discuss the management of ASAP deployments only briefly.

3. Modeling Virtual Resources

We describe how we model virtual resources and overhead, and introduce the concepts of workspace metadata and image templates.

3.1. Virtual Resources and Overhead

Our scheduling model assumes a set of physical resources providing a set of *resource slots* (e.g., all the physical memory is one resource slot, each CPU is another resource slot, etc.). A quantum of a slot (e.g., 512 MB of memory, out of the 4 GB available) may be bound to a virtual workspace to provide it with some hardware resources needed to support the workspace's activities for a well-defined period of time. We term such a binding of a portion of a slot to a virtual workspace a *virtual resource*.

However, deploying and managing virtual resources involves two types of overhead*: preparation overhead* and *runtime overhead.* The former refers to the cost of preparing the environment where the virtual workspace will run (most notably, deploying the VM images required by that workspace), while the latter refers to the memory, CPU and I/O overhead incurred by the VM hypervisor itself. Furthermore, these overheads are not necessarily constant, and may depend on the size of the requested virtual resources, the hypervisor used, and the quality of base resources.

To adequately manage this overhead, we propose a model in which portions of resource slots can be bound either to virtual resources *or* to overhead associated with the creation or maintenance of those virtual resources. This approach is distinct from the idea of having overhead *deducted* from a user's resource allocation, which burdens the users with having to factor in overhead when deciding how many virtual resources to request. Since this model treats preparation and runtime overhead in the same way as virtual resources, the resource provider can thus budget a slot for overhead in the same way that virtual resource slots are budgeted. As discussed in previous work [1], resources can be subdivided in this way across many different layers.

The management of runtime overhead for the Xen hypervisor was explored by Keahey et al. [7]. We concern ourselves here primarily with preparation overhead. In particular, workspace deployment can involve the (potentially expensive) transfer of a VM image to a node, a task that requires I/O and network usage that has to be accounted for by our scheduler.

3.2.Workspace Metadata and Image Templates

A virtual workspace is composed of three elements [1]: a VM *image* (or images), the *workspace metadata*, and the *deployment request*.

The workspace metadata contains information about the workspace that may be preserved between deployments (such as the IP address of each node in the workspace), and is therefore deployment-independent. The deployment request describes the resource allocation that should be assigned to the workspace (such as memory and CPU%), which will generally vary between deployments.

This approach allows workspaces to be described in terms of *VM image templates*, generic reusable VM images with the system software and tools for some specific purpose (e.g. a worker node for an Open Science Grid cluster), but lacking all the configuration specific to a particular type of deployment. This configuration, contained in the metadata file, is *bound* to the VM image at runtime to produce an *image instance*, which is allocated the resources specified in the deployment request.

The reusability of image templates enables certain optimizations because an image template, deployed to a physical worker node, can be used multiple times by making local copies and binding those copies to different metadata files and deployment requests. Without image templates, each distinct image instance would have to be transferred to the physical nodes; we, in contrast, can potentially reuse already deployed image templates. Workspace metadata itself is an appealing feature because it contains information that a VW scheduler can use to adequately manage overhead. More specifically, our scheduler uses the (1) image descriptor (currently the location of the image file within an image repository node), (2) image size, and (3) number of nodes in the VW to estimate what the preparation overhead will be.

In the following sections we show how workspace metadata and image templates enable us to optimize VW scheduling in ways that are not possible when deploying VM images with hardcoded configurations.

4. Scheduling Workspaces

We have developed the Workspace Service, a Web Service interface that allows clients to request the creation, monitoring, and control of virtual workspaces. This service, implemented on top of the Globus Toolkit version 4 [7], enables authorized users to submit virtual workspace creation requests to a resource provider via a Web Services interface. In this section we describe how we extend the Workspace Service to support the AR and ASAP scenarios described in Section 2.

4.1 Service Interface

We extend the Workspace Service interface to allow the user to specify the time constraints of the virtual resources required by a VW. Specifically, we extend the ResourceAllocation element of a VW request [7] to include the start time and end time of the virtual resource(s) to be allocated to the workspace. These times can be expressed as exact timestamps, as a time interval within which an asynchronous event can be received (triggering the start or end of the workspace), or omitted (indicating an immediate reservation, in which case the workspace must be scheduled as soon as possible).

4.2 Implementation

We modify the Workspace Service implementation to use Sun Grid Engine (SGE) [8] as a local resource manager backend. Our extensions provide SGE with application-specific information that will allow it to arrive at better scheduling decisions for the VMs that we are concerned to schedule. Specifically, we (a) use *deadline-driven file staging* to schedule a separate resource slot to accommodate the overhead of transferring VM images, instead of having that overhead absorbed by the virtual resource allocated to the client, and (b) add *image caches* to worker nodes in an effort to reduce the number of image transfers performed.

4.2.1. Deadline-driven file staging. Jobs submitted to batch schedulers generally assume that required files are available in the worker nodes (e.g., through an NFS drive) or that the input files will be staged to the worker nodes when the job starts. This assumption presents problems for deploying time-sensitive VWs, VM images can be large and costly to transfer, and transfer times can consume a significant portion of the time allocated to the user. Thus, even if a resource is made available at a requested time T, it may not be ready for use until a significantly later time T+D.

These problems can be solved in some cases by providing the scheduler with application-specific information (selected metadata information, as described in Section 3.2) about what data needs to be transferred for each deployment. We modified our Workspace Service to provide this information, and extended SGE to support the following file staging strategies:

- *Just In Time (JIT)*. The scheduler estimates the time required to transfer the image and starts the transfer before the start time, allocating just enough time to transfer the image.
- *Aggressive*: This strategy attempts to transfer images immediately after the request has been accepted, regardless of the deadline for the image transfer.
- *Hybrid*: This approach is a combination of *JIT* and *Aggressive*. Although the image transfer can potentially start when the VW is submitted, the transfer is scheduled based on a priority assignment that is a function of proximity to deadline and estimated transfer time relative to other images. In this way, transfers with loose deadlines can give way to transfers with tighter deadlines. This approach involves actually *scheduling* a resource slot for the preparation overhead. In contrast, *JIT* and *Aggressive* are naïve file staging strategies.

4.2.2. Image caches. The strategies just described benefit clients, who are interested in *accuracy*, that is, getting the virtual resources at the start time they are promised. The provider, on the other hand, is interested in *efficiency*, that is, optimizing resource usage.

To address resource provider optimization, we include an *image template cache* in every worker node. This cache keeps a copy of a subset of all available template images according to a caching strategy. (For now, we support LFU and LRU, with a configurable cache size.) When an image has to be deployed to a worker node, the scheduler favors nodes that already have a cached copy of the required template image.

Because caches enable some images to be instantly available in the nodes, they benefit both AR deployments, since it will be possible to accept starting times that would ordinarily be unfeasible if the image first had to be transferred to the nodes, and ASAP deployments, by allowing workspaces to start running sooner. As described in Section 3.2., this optimization is possible because an image template can be reused several times on the same node by binding it to different configurations (different metadata files). Additionally, our image caches are implemented to avoid redundant transfers when deploying a new image template to a node, as for example when the same image template is required to produce two image instances on a node, in which case the template is transferred only once.

5. Experiments

We present a series of experiments that illustrate the effect of using the scheduling model and techniques discussed in the previous section. These experiments focus on evaluating our techniques for managing the overhead of transferring VM images to the nodes where they are deployed.

Our experiments were run on a testbed composed of 10 dual-CPU Pentium III 500 MHz systems, each with 512 MB of RAM and 9G of local disk. One node was used as a cluster head node, eight nodes for VM deployment, and the remaining node as an image repository node, from which the VM images would be transferred to the worker nodes. Nodes were connected using 100 Mb/s switched Ethernet.

Virtual machine images were deployed using the SGE scheduler with the extensions described above, based on traces that we developed for both the advance reservation (AR) and batch (ASAP) cases. For the ASAP cases, we used real workload traces, while for the AR cases, lacking real AR submission workloads, we produced artificial traces using a trace generator.

For all our experiments, we assumed that all virtual workspace requests involved the same amount of CPU% and memory for each virtual node. We allowed at most 2 VMs to be deployed to a single physical node. Since we focus on preparation overhead, the VW remains idle during its runtime, and we assume that the VM generates no network traffic that would share bandwidth with preparation overhead. This assumption is reasonable in the case of highly parallel applications.

5.1. Scheduling Jobs versus Scheduling VMs

Our first set of experiments investigates to what extent using information on the relatively manageable overhead of VM scheduling can improve the accuracy of providing a virtual resource to a deadline-sensitive

Table 1: Traces used in experiments

	Trace I	**Trace II**	**Trace III**	**Trace IV**
Trace duration (s)	7200		4700	
# VW submissions	36	35	62	
Nodes per VW	2-4 (Uniformly distributed)		2-16 (Derived from original trace)	
Total images to deploy	110 (66.0GB)	106 (63.6GB)	114 (86.4GB)	
VW Duration	1800s		Avg=53.0s StDev=4.24s	
Starting times	Uniformly Distributed	Clustered in 100s windows, every 900s	ASAP	
Images used	6 600MB images, uniformly distributed		See Table 2	

client. We assume that the client requests the resource for a fixed time interval and we calculate accuracy (or "client satisfaction") as the ratio of the time the client actually got to the requested time. Lacking any AR traces or AR trace generator, we developed a simple trace generator capable of generating a large number of requests according to a set of parameters. We then ran an offline admission control algorithm (derived from Earliest-Deadline First [22]) on those requests to ensure that there exists a feasible schedule for the submissions in the trace. Each submission represents the deployment of a virtual cluster configured with the software required to applications commonly run on the Open Science Grid (OSG), and includes (1) the descriptor of the image template to use, (2) the number of nodes, (3) the starting time of the workspace, and (4) its duration. The Xen VM image with OSG worker node support that we used in this experiment is 600 MB in size [6].

Table 1 describes the two traces (I and II) used in this experiment. These two traces differ in how the starting times of the VWs are distributed throughout the duration of the experiment. In Trace I, the starting times are distributed uniformly throughout the trace, while in Trace II the starting times appear only during 100s windows (each occurring every 900s), simulating VWs that are submitted in a bursty fashion.

Figure 1 shows the results of running Trace I with the three scheduling strategies described in Section 4.3 (*JIT, Aggressive, and Hybrid*) and pits them against the baseline unmodified SGE (*Job-Style)* that does not seek to prestage VM images but instead stages required files during the availability of the physical resources. We see that *Hybrid* and *JIT* achieve 100% client satisfaction in most cases, followed by *Aggressive* with most submissions in the 96%-100% range. Since this trace represents a best-case scenario, where the start times are uniformly distributed throughout the experiment, even the naïve *JIT* and *Aggressive* strategies achieve good performance.

Figure 2 shows results with Trace II. Since *JIT* allows just enough time to transfer the image, regardless of what other VWs are scheduled, this strategy can result in multiple image transfers being scheduled during the same period of time (right before the "window"). As those transfers share available bandwidth, they take longer than estimated. *Hybrid* addressed this problem by prioritizing image transfers and making use of network idle time, resulting in the best performance, with few non-100% satisfaction instances (and always at the end of a "window"). *Aggressive*, on average, also has good performance, but the images with tighter deadlines suffer as the result of having to share bandwidth with other transfers.

As described in Section 4.2.1, *Hybrid* is the only approach that actually schedules a resource slot for the preparation overhead with the goal of maximizing client satisfaction, while the other strategies naïvely start the image transfers at fixed times. By scheduling overhead in the same way as virtual resources, instead of assuming that overhead should be absorbed into the client's requested virtual resource, *Hybrid* achieves the best client satisfaction in the two submission patterns present in traces I and II.

5.2. Optimizing Image Transfer

Our second set of experiments investigate two things: (1) how much in terms of bandwidth usage the resource provider can save through judicious use of image caching based on workspace metadata and (2) to what extent image caching can improve deployment time (and thus also client satisfaction) in situations where VM availability is requested to start ASAP.

Since this experiment focuses on ASAP submissions, the same type of submissions commonly found in batch systems, we were able to use a real workload. In particular, we used the San Diego Supercomputer Center (SDSC) DataStar log, available at the Parallel Workloads Archive [21]. We chose this workload because it explicitly distinguished submissions for the DataStar's express queue (eight 8-CPU nodes, accepting jobs lasting at most 2 hr), which allows us to test a scenario in which minimizing deployment overhead is specially important: short-lasting jobs. Since the SDSC DataStar log spans several months, we selected an 80 minute stretch of submissions (submissions #21543 to #21665 on queue #1) which we could run on our testbed. This extract

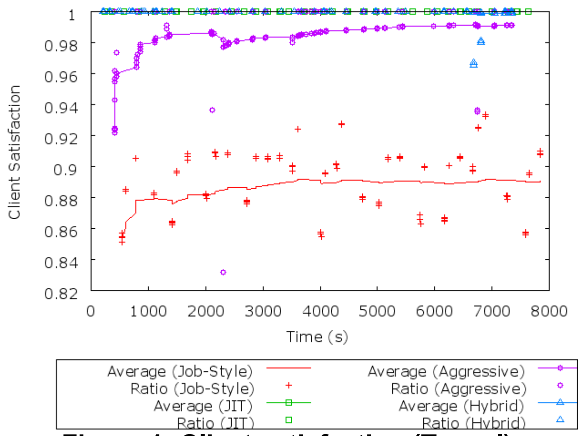

Figure 1: Client satisfaction (Trace I)

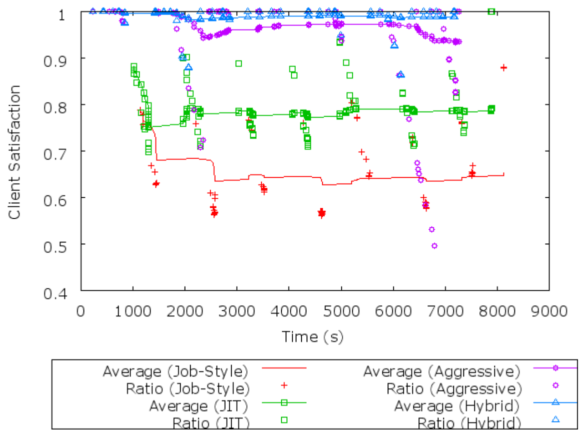

Figure 2: Client satisfaction (Trace II)

was selected because it represented a flurry of short-lasting jobs, which would allow us to test how our system copes with the bandwidth requirements of deploying a large amount of VM images.

When adapting the trace to our own experiments, each submission was converted to a virtual cluster submission in which the number of nodes was the number of requested processors in the original trace, scaled down by four (the express queue has 64 processors; our testbed has 16), with submission times and VW duration left unaltered. Each submission was furnished with one of six 600 MB images. Two traces where produced, with the only difference being the distribution of images assigned to each submission. The first trace (Trace III) has images uniformly distributed amongst the submissions, while in the second trace (Trace IV) two images account for more than 80% of the submissions. The characteristics of theses traces are summarized in Table 1, while the distribution of images is shown in Table 2. The rate at which VWs are submitted is shown in Figure 3.

To minimize deployment time and increase throughput, we used a 1.8 GB LFU cache in each node (enough to cache three images). Figure 4 shows the

Table 2: Distribution of images in traces III, IV

	Trace III		Trace IV	
	Submissions	Images	Submissions	Images
img.1	23%	21%	55%	60%
img.2	18%	15%	27%	25%
img.3	15%	12%	6%	6%
img.4	18%	15%	5%	4%
img.5	24%	14%	3%	3%
img.6	3%	12%	3%	3%

cumulative number of MB transferred, overlaid with the VW submissions. We can observe how, after the 2000s mark, the rate at which images are deployed starts to increase, thanks to the reduced transfer time resulting from the use of a cache. The difference in effectively deployed MB is greatest at the 3550s mark, where the cached approach results in a 25.2 GB "advantage" over the non-cached approach. Figure 5, shows the same data from running trace IV, where the distribution of images favors a much larger number of cache hits. Throughput is slightly better, with a difference of 27.6 GB at that same 3550s mark.

Deployment time (not shown in graphs) is also improved. The average deployment time for a single image, when not using a cache, is 440s for both traces. This time is reduced to 305s and 247s, in Trace III and IV respectively, when using an image cache.

These two experiments highlight how using the VW metadata to cache image templates and avoid redundant transfers benefits both the provider, by offering a better utilization of resources leading to higher throughput, and the consumer, by reducing the deployment time of ASAP workspaces.

5.3. Effect of cache size

When submitting workspaces set to begin ASAP, preparation overhead may prevent short-duration workspaces from being cost-effective. Our final experiment explores the impact of cache size on the costs associated with short-duration workspaces. Although these results are not exhaustive, they nonetheless allow us to extract some useful information regarding the use of image caches.

For this experiment we used two artificial stress traces in which a series of VWs, of size one to six nodes each, were submitted at random intervals (between 5s and 30s) for 20 minutes. Given the limited disk space in our testbed machines, and to allow for a configuration where all possible images could be cached in a node, we used eight 256 MB images. In the first trace, 257 images had to be deployed, with images

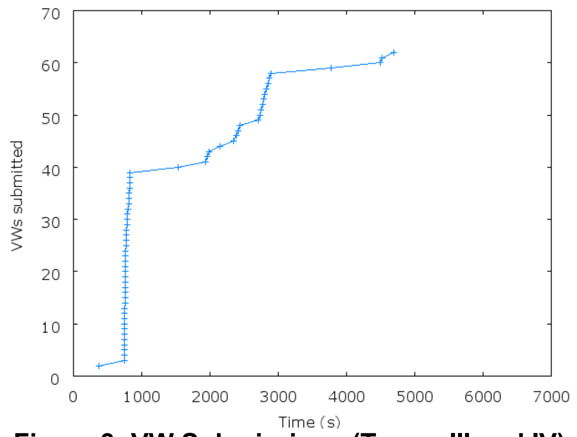

Figure 3: VW Submissions (Traces III and IV)

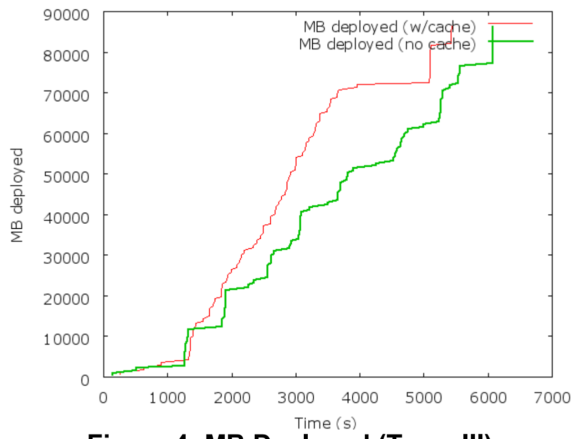

Figure 4: MB Deployed (Trace III)

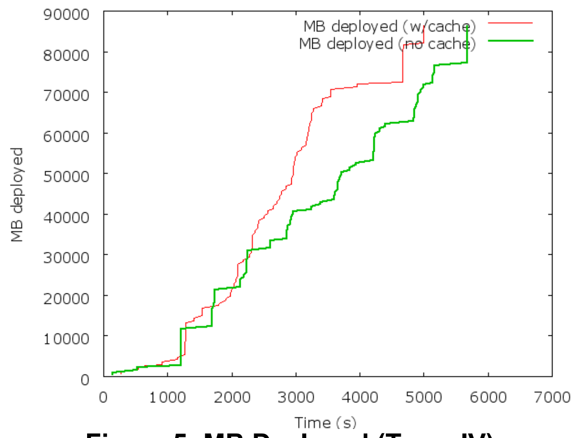

Figure 5: MB Deployed (Trace IV)

being selected according to a uniform distribution. In the second trace, 235 images had to be deployed, and two of the images accounted for 77% of all submissions.

Table 3 shows the result of running these two traces both without a cache and with a cache capable of holding 25%, 50%, 75%, or 100% of the images. Note

Table 3: Effect of Cache Size

	Stress Trace I					Stress Trace II				
	No cache	**25%**	**50%**	**75%**	**100%**	**No cache**	**25%**	**50%**	**75%**	**100%**
Cache hits	-	48	123	150	165	-	144	169	167	159
Cache misses	-	126	80	60	54	-	59	36	39	42
Avg. image deploy time	141s	163s	103s	96s	94s	124s	89s	80s	81s	86s

that, when a redundant transfer is avoided (as described in Section 4.2.2) it is not counted as a hit or a miss.

We see that the first trace produces similar results when using a 50%, 75%, or 100% cache, but the 25% configuration is *worse* than not using a cache at all, due to the large number of cache misses (a cache miss results in both a download of the image *and* performing a local copy of the image). In the second trace, on the other hand, all cache sizes produce similar results, since the 25% cache is already capable of holding the two most used images.

To make short-duration cost-effective, caches can help to reduce the deployment time of images, but this time will still be bound by the time of making a local copy in the case of a cache hit. In this experiment, the average time to do a local copy is 60s. By precaching images typically used for short-term deployments, locking them in the image caches, and optimizing the local copy time, the cost-effectiveness of short-duration VWs can be improved.

6. Related Work

Many projects tackle the problem of dynamically overlaying virtual resources on top of physical resources by using virtualization technologies, and do so with different resource models. These models generally consider overhead as part of the virtual resource allocated to the user, or do not manage or attempt to reduce it. A common assumption in related projects is that all necessary images are already deployed on the worker nodes. Our requirements for dynamic deployment of AR and ASAP workspaces make it impossible to make this assumption.

The Shirako system [12] developed within the Cluster-On-Demand project [10, 11] uses VMs to partition a physical cluster into several virtual clusters. Their interfaces focus on granting *leases* on resources to users, which can be redeemed at some point in the future. However since their model focuses on batch cases the adopted overhead management model is to absorb it into resources used for VM deployment and management. As we have shown, this model is not sufficient for AR-style cases.

The VIOLIN and VioCluster projects [13, 14, 15] allow users to overlay a virtual cluster over more than one physical cluster, leveraging VM live migration to perform load balancing between the different clusters. The VioCluster model assumes that VM images are already deployed on potential hosts, and only a "binary diff" file (implemented as a small Copy-On-Write file), expressing the particular configuration of each instance, is transferred at deploy-time. This is less flexible than using image metadata, as COWs can be invalidated by changes in the VM images. Furthermore, our work focuses on use cases where multiple image templates might be used in a physical cluster, which makes it impractical to prestage all the templates on all the nodes.

The Maestro-VC system [18] also explores the benefits of providing a scheduler with application-specific information that can optimize its decisions and, in fact, also leverages caches to reduce image transfers. However, Maestro-VC focuses on clusters with long lifetimes, and their model does not schedule image transfer overhead in a deadline-sensitive manner, and just assumes that any image staging overhead will be acceptable given the duration of the virtual cluster. Our work includes short-lived workspaces as a case that must perform efficiently under our model.

The Virtuoso Project [19] and, in particular, its VSched component [17], is capable of co-scheduling both interactive and batch workloads on individual machines in a deadline-sensitive manner, but does not factor in the overhead of deploying the VMs to the nodes where they are needed.

The In-VIGO project [16] proposes adding three layers of virtualization over grid resources to enable the creation of virtual grids. Our work, which relates to their first layer (creating virtual resources over physical resources), is concerned with finer-grained allocations and enforcements than in the In-VIGO project. Some exploration of cache-based deployment has also been done with VMPlant [20], focusing on batch cases.

7. Conclusions

We described a virtual resource model, and a set of scheduling strategies for that model, based on scenarios that, in our experience, frequently arise in the Grid. These scenarios combine batch job platforms as well as platforms whose deployment is deadline-sensitive, such as interactive platforms. VM deployment and management overhead can be both large and highly variable, factors that conflict with the deadline-sensitive availability needs of interactive and time-critical platforms. Thus, our proposed model separates resource use devoted to the overhead of VM deployment from resources available to the VM itself, enabling us to schedule overhead resource slots equally with VM slots.

Our results show that by modifying a scheduler to schedule workspace preparation overhead, rather than leaving workspace preparation to the user, we can achieve significantly better adherence to requested availability times. (One may argue that the user could simply request start times sooner than required, to allow time for image staging, but this in effect corresponds to the JIT scenario, which we show is not always effective.) Providing the scheduler with information about the VM images needed by a virtual workspace can have two benefits. First, the required transfer operations can be scheduled in advance. Second, we can use information about a VM image as defined in the workspace metadata to optimize the resource usage devoted to VM deployment by caching the bulk of the data associated with frequently used images. Both strategies have benefits for the resource provider and for the client in the case of heavy load and/or immediate reservations. Further, by reducing preparation overhead, these strategies also make the deployment of short-lived VMs more cost-effective.

Interesting challenges arise when one considers that the network bandwidth has to be shared not only with other VM image transfers but also with virtual resources allocated to applications. Our further work on this subject will involve developing models that will accommodate such sharing, as well as managing other resource overhead arising in the hypervisor context.

Acknowledgements

This work was supported by NSF CSR award #527448 and in part, by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, SciDAC Program, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38

References

[1] Keahey, K., I. Foster, T. Freeman, and X. Zhang, Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. accepted for publication in the Scientific Progamming Journal, 2005.
[2] Foster, I.C., K., D.F. Ferguson, J., S. Graham, and S.S. Maguire, D. Tuecke, S., Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. Proceedings of the IEEE, 2005. 93(3): p. 604-612.
[3] Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. in ACM Symposium on Operating Systems Principles (SOSP).
[4] VMware: http://www.vmware.com/.
[5] Andrieux, A., K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, Web Services Agreement Specification (WS-Agreement) Draft 20. 2004: https://forge.gridforum.org/projects/graap-wg/.
[6] Foster, I., T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. Virtual Clusters for Grid Communities. CCGRID 2006, Singapore, May 2006.
[7] Keahey, K., I. Foster, R.D. Freeman, A. Rana, B. Sotomayor, and F. Wuerthwein, Division of Labor: Tools for Growth and Scalability of the Grids. ICSOC, 2006.
[8] Foster, I., Globus Toolkit version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, 2005.
[9] Sun Grid Engine: http://gridengine.sunsource.net/
[10] David Irwin, Jeff Chase, Laura Grit, Aydan Yumerefendi, David Becker, and Ken Yocum. Sharing Networked Resources with Brokered Leases. in USENIX Technical Conference, 2006.
[11] COD: Cluster-On-Demand project: http://www.cs.duke.edu/nicl/cod/
[12] Shirako (part of Cereus project): http://www.cs.duke.edu/nicl/cereus/shirako.html
[13] Paul Ruth, Junghwan Rhee, Dongyan Xu, Rick Kennell, Sebastien Goasguen, Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure. in ICAC'06
[14] Paul Ruth, Phil McGachey, Dongyan Xu. VioCluster: Virtualization for Dynamic Computational Domains. in Proceedings of the IEEE International Conference on Cluster Computing (Cluster'05), 2005.
[15] Paul Ruth, Xuxian Jiang, Dongyan Xu, Sebastien Goasguen. "Virtual Distributed Environments in a Shared Infrastructure", IEEE Computer, Special Issue on Virtualization Technologies, May 2005.
[16] Adabala, Sumalatha; Chadha, Vineet; Chawla, Puneet; Figueiredo, Renato; Fortes, Jose; Krsul, Ivan; Matsunaga, Andrea; Tsugawa, Mauricio; Zhang, Jian; Zhao, Ming; Zhu, Liping; Zhu, Xiaomin From Virtualized Resources to Virtual Computing Grids: The In-VIGO System. In Future Generation Computer Systems, vol 21, no. 6, April, 2005. DOI:10.1016/j.future.2003.12.021.
[17] B. Lin, and P. Dinda, VSched: Mixing Batch and Interactive Virtual Machines Using Periodic Real-time Scheduling, Proceedings of ACM/IEEE SC 2005 (Supercomputing), November, 2005.
[18] Nadir Kiyanclar, Gregory A. Koenig, William Yurcik. "Maestro-VC: A Paravirtualized Execution Environment for Secure On-Demand Cluster Computing". CC-Grid 06.
[19] Virtuoso: Resource Management and Prediction for Distributed Computing Using Virtual Machine. http://virtuoso.cs.northwestern.edu/
[20] Krsul, I., A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. in SC04. 2004. Pittsburgh, PA.
[21] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/
[22] Kirk Pruhs, Jiri Sgall, Eric Torng. "Online Scheduling", in "Handbook of Scheduling". Chapman & Hall, 2004.