Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration

Laura Grit, David Irwin, Aydan Yumerefendi, Jeff Chase
Department of Computer Science
Duke University
{grit,irwin,aydan,chase}@cs.duke.edu

Abstract

Virtualization technology offers powerful resource management mechanisms, including performance-isolating resource schedulers, live migration, and suspend/resume. But how should networked virtual computing systems use these mechanisms? A grand challenge is to devise practical policies to drive these mechanisms in a self-managing or "autonomic" system, without relying on human operators.

This paper explores architectural and algorithmic issues for resource management policy and orchestration in Shirako, a system for on-demand leasing of shared networked resources in federated clusters. Shirako enables a flexible factoring of resource management functions across the participants in a federated system, to accommodate a range of models of distributed virtual computing. We present extensions to Shirako to provision fine-grained virtual machine "slivers" and drive virtual machine migration. We illustrate the interactions of provisioning and placement/migration policies, and their impact.

1 Introduction

After a decade of advances in virtual machine (VM) technology, robust and efficient VM systems are widely available and are fast becoming ubiquitous. Among other benefits, these systems offer powerful mechanisms for managing shared server networks and clusters. The leading VM systems support live migration, checkpoint/restart, and fine-grained allocation of server resources as a measured and metered quantity (e.g., Xen [1, 4], VMware [19]).

These capabilities create a rich policy space for system management infrastructures. How should an intelligent infrastructure "turn the knobs" to map workload and resource requests onto a server network? Coordinated usage of these basic mechanisms can make it possible to harness the potential of virtual machines for effective and efficient resource management. The broader challenge involves all of the steps to operate a computing service utility: configuring and instantiating OS images and services, binding them to server resources, and managing and controlling their interactions at both the system and application level. This general process of mapping application services onto a shared server network is widely known as *orchestration*.

In short, a key challenge today is to develop practical *policies* for on-demand, adaptive, and reliable allocation of networked computing resources from a common pool. This paper [1] explores some architectural and algorithmic challenges for resource management policy, which is difficult for at least three reasons:

- *It is heuristic.* Resource management involves projections under uncertainty and optimization problems that are NP-hard in their general form, forcing us to adopt heuristics tailored for specific needs and settings. There is no "one size fits all" solution.
- *It is dynamic.* Resource allocation policies must adapt to changing workload and demands in real time.
- *It is organic and emergent.* Policy choices must balance the needs and interests of multiple independent stakeholders, e.g., resource providers and resource consumers or hosted guests. In *federated* systems—in which independent providers contribute resources to a shared pool—brokering intermediaries may also play a role to supplant the need for pairwise peering arrangements and/or to implement community sharing policies. In general, the resource assignment emerges from choices taken by each of these actors, and the complex interactions of those choices.

The basic structure of resource management policy is common to a range of visions for networked virtual computing, encompassing managed data centers, network testbeds,

[1] This research is supported by the National Science Foundation through ANI-0330658, CNS-0509408, EIA-99-72879, and CNS-0451860, and by IBM, HP Labs, and Network Appliance. Laura Grit is a National Physical Science Consortium Fellow.

grid computing systems, and market-based utilities. With the right factoring of policy and mechanism, these systems can build on the same underlying resource management substrate. This paper explores architectural considerations for such a substrate, and summarizes challenges posed by the rich policy space realizable within it. In particular, we discuss the architecural choices for autonomic orchestration in the Shirako system [10, 16], a Java toolkit for *resource leasing* services based on the SHARP framework [7]. We illustrate some policy issues relating to the use of Shirako for fine-grained adaptive hosting of virtual machines.

2 Separating Policy from Mechanism

Our prototype infrastructure for networked virtual computing consists of a set of resource drivers and policy modules for the Shirako toolkit. Although Shirako's leasing model can generalize to many kinds of shared resources (e.g., network resources), we focus on orchestrating hosting of Xen virtual machines, as a basis for secure, adaptive, on-demand resource sharing in federated clusters.

In this setting, the system-level policies control VM *provisioning* (allocation of quantities of server resources over time) and *placement* of active VM images within the server network.

Shirako emphasizes the importance of *contracts* between resource providers and resource consumers (e.g., hosted application services or users) with respect to the commitment of resources. Shirako contracts take the form of *leases*, which define the resources allocated to a user, and the duration of the user's ownership of the resources. Some infrastructures may be useful with weak contracts (e.g., a well-provisioned network testbed), while others will require stronger contracts to meet service quality targets or to ensure reproducible results. While the underlying lease and contract mechanisms are common, the space of contract attributes and values is resource-specific, and the specific attribute sets selected for each contract is a policy choice. Stronger contracts require stronger policies.

For example, consider an on-demand system that instantiates best-effort virtual machines, with no assurance that they will make forward progress. The provisioning problem is trivial in this system: no admission control or resource arbitration is needed. The placement choice might be left to users, e.g., to seek out hosts that appear lightly loaded at any given time. Alternatively, the providers or brokers might coordinate placement to balance the load.

Now consider a system whose goal is to assure a predictable level of service quality for its guests. Best-effort contracts might be adequate if the system is sufficiently over-provisioned. The alternative is to make promises to each consumer about the resources it will receive—how much and/or when. Shirako represents such promises in the lease term and in the lease attributes.

The leading VM systems, including Xen, support the resource control mechanisms necessary to back these stronger contracts. Performance-isolating schedulers permit fine-grained allocation of host server resources as a measured and metered quantity: each guest VM is bound to a *sliver* of host resources sized along multiple dimensions (e.g., CPU capacity, memory, and network bandwidth). Slivers are sized to meet performance targets in the guest while minimizing the "crosstalk" from competing VMs on the same host. Support for VM migration [4] provides the flexibility to satisfy new and unforeseen resource demands while retaining the continuity, liveness, and sliver sizing of existing VMs. These technologies are reasonably mature, although refinement continues.

We anticipate support for resource-controlled virtualization of other resources as well. For example, while today's Internet lacks comprehensive quality-of-service mechanisms, support for bandwidth-provisioned paths is increasingly available on advanced networks below the IP layer. Resource control for shared network storage continues to be an active area of research.

A key premise of our work is that resource control and strong resource contracts are essential to deliver on the promise of distributed virtual computing. In particular, contracts enable meaningful federation, in which common infrastructure is managed across administrative domains. For federation to be sustainable the participants must have the power to understand and control what they give up and what they receive in return. Recent advances in virtualization technology motivate a stronger focus on the policy implications and the significant research challenges that they raise. Resource control features are essential mechanisms, but someone or something must control how they are used, e.g., how to size the slivers, where to place the VMs, and when and where to migrate. It is straightforward to expose these choices to human operators through a control panel, but the grand challenge is a self-managing compute utility in which the control functions are automated and respond to changes in real time.

2.1 Shirako: A Quick Tour

The Shirako toolkit offers a common, extensible resource leasing core. Leases are dynamic and renewable. The leasing abstraction applies to any kind of computing resource that is partitionable as a measured quantity. Resources are typed, and each type is associated with a controller module and a driver that controls instances of the resource. For virtual machine computing, Shirako incorporates Cluster-on-Demand (COD [3]) as a back-end manager for cluster sites, fitted with resource drivers to instantiate and control Xen VMs.

Shirako can be used to implement simple data center automation or complex federated systems. A fundamental architectural choice is the factoring of resource management functions across the different stakeholders in the infrastructure. Shirako is a toolkit for implementing *actors*, which are servers interacting with a messaging protocol such as SOAP or XMLRPC. Actors can serve any of three different roles corresponding to consumers, providers, and brokers. A resource provider represents a cluster site, data center, or other resource domain. Resource consumers request resources according to the projected needs of a hosted service, environment, user, or group. Resource providers cooperate with brokers to arbitrate requests, e.g., to allocate resources to their highest and best use. The three actor roles are referred to as service managers, site authorities, and brokers.

Shirako actors may be instantiated on the fly, and they may associate and interact in various ways. For example, a service manager may lease resources from multiple brokers and sites; a site may peer with multiple brokers; a broker may coordinate resource usage across multiple sites. Each actor embodies its own local policies, and these policies interact to automate provisioning and placement of Xen VMs in a server network.

While this paper focuses on resource management, "orchestration" for virtual computing also includes the configuration and control of hosted application services. Service managers include application-specific plugin code to interact with a guest service or application, monitor its functioning, and respond to events by requesting changes to sliver sizing or to the number of virtual machines or their location. The plugin receives upcalls to mark the passage of time and notify it of various events, e.g., when VMs *join* or *leave*, and when leases are due for renewal. The service manager has programmatic access to operator interfaces in the guest environment and application.

We recently extended Shirako/COD to support fine-grained sliver sizing along multiple dimensions (e.g., the amount of CPU cycles and network bandwidth bound to a sliver), sliver resizing on lease renewal, and policy-driven migration to adapt to changing demands. This required minor changes to the Shirako policy interface, as well as new features to protect the accountability of contracts, which are beyond the scope of this paper. To support virtual machine migration and resizing we extended the event handler set for site authority resource drivers with a new event: *modify*. The *modify* event handler for Xen virtual machines invokes Xen primitives to resize or migrate a target VM, guided by a property list passed from a policy module.

2.2 Toward a Common Control Fabric

Shirako can act as a foundational resource control plane for a wide range of applications and even different models of distributed virtual computing. Most of the differences among competing approaches to distributed virtual computing boil down to matters of policy, or questions of who the players are and how much power they have, or differing application assumptions that ultimately have little impact on the underlying resource management requirements. These superficial differences leave open the opportunity for a common management "fabric".

We use the Shirako toolkit as a substrate for three approaches to networked virtual computing:

- **Grid hosting**. Grid computing systems coordinate job management and data sharing within federated "virtual organizations". We developed a grid hosting system for Globus grids on top of a Shirako resource control plane [16]. Each grid runs a Globus-aware service manager (GROC) that interacts with a Globus monitoring service and procures resources in response to changing demand within its grid. This approach enables controlled co-location of multiple Globus grids and other unrelated services on generic clusters, and devolves grid-related management burdens from the site administrator to the grid operator.
- **Virtual batch computing**. Virtual machines introduce new opportunities for flexible job management in batch computing systems. If VM instantiation is cheap, it becomes possible to run each job within a private virtual machine workspace [6]. It is possible to customize the workspace for the needs of specific jobs, and essential checkpointing and migration features are supported in a general way at the VM level. For example, NIMO [17] can use per-job containers and sliver sizing as a basis for active learning of end-to-end application performance models.
- **Resource markets**. As networked utilities increase in scale, market-based control becomes attractive as a basis for resource allocation that is fair, flexible, adaptive, and decentralized. Strong resource contracts enable negotiated, accountable exchanges in a marketplace. We have used Shirako to experiment with market-based virtual machine hosting based on a self-recharging virtual currency [9].

3 Overview of Policy Challenges

Adaptivity to accommodate different system goals can be complex: services may request resources at any time for any length of time, VM slivers may vary in size along multiple dimensions, and VMs may be placed on host machines in many different ways. We address these challenges by factoring out decision logic into custom policies that fit the particular needs of a system.

Shirako defines interfaces for pluggable policy modules for each actor. A key principle is that provider sites control

the placement of VMs on their own clusters, but delegate limited provisioning power to brokers. This decoupling offers a general and flexible basis for managing a shared pool with many resource contributors: each broker and provider site is free to select the policies governing the resources under its control. For example, brokers can coordinate site selection and sliver sizing for virtual machines, while sites can control placement of the hosted VMs on their local servers based on local considerations, e.g., thermal load balancing within a cluster. These policies may function within a continuous optimizing feedback loop with specific objectives, e.g., fidelity to SLAs, utilization, global utility or revenue, energy efficiency, or dispersion of flash crowds.

3.1 Provisioning

Resource management poses formidable algorithmic challenges. Consider the problem of provisioning—*how much* resource is allocated to each request, and *when*. If different choices have different values (utility), then provisioning is an instance of an NP-hard knapsack problem. With multiple sites or host machines, it becomes a multiple knapsack (MK) problem. If resource requests are flexible in the number of resources they will accept, then it becomes a multiple choice knapsack (MC). VM slivering offers a finer degree of control over the resources, but then the knapsack problem becomes multidimensional (MD).

Knapsack problems are NP-hard even in their simplest form, and the MK-MC-MD variants complicate the heuristics. Kelly [12, 13] has a detailed treatment of the problem variants and approaches to solving them. Heuristics and approximations exist, but the choices may be sensitive to workload. Even so, it is an open question what the objective functions might be for resource provisioning in practice, and—if utility-driven allocation is indeed necessary—what quality of approximation is adequate given the degree of resource constraint and the presence of best-effort service classes.

Additional features such as advance reservations and market auctions further complicate provisioning policy. In Shirako-based systems, the provisioning policy module in the broker may also be responsible for site selection if multiple sites peer with the broker. Allocation of network links and paths will present new algorithmic challenges for network embedding manifested within a broker, or perhaps a set of cooperating brokers for different autonomous systems (peering networks).

3.2 Placement

Site-level placement policies determine the mapping of provisioned VMs onto a site's server network. As with provisioning, placement is subject to continuous, on-line control as conditions change. A cluster placement policy might consider several factors:

- **Failures and maintenance**. A cluster site may need to remove selected hosts from service due to failure or for scheduled maintenance.
- **Overbooking.** Proportional share schedulers on the hosts may permit overbooking [18] of resources to benefit from statistical multiplexing.
- **Thermal management.** A cluster site may respond to cooling problems by placing heat-generating workloads away from hot spots. Placement may also substantially influence data center cooling costs at typical utilizations [15].
- **Energy management.** At low utilizations, concentrating workload on a subset of servers can reduce energy consumption proportionally if the site authority powers down unused hosts [2].

3.3 Provisioning vs. Placement

A complete system for allocating cluster resources, the provisioning and placement policies must work together effectively. In practice, the decoupling of provisioning and placement creates a tension between these policies. This decoupling simplifies provisioning and enables brokered federation. But if the site and broker policies work at cross-purposes, then it may increase the need for migration, as illustrated later in this paper.

Some systems have chosen to integrate provisioning and placement tightly; for example, [11] takes an integrated approach to assigning distributed web application components in a server cluster to meet demands and balance load under dynamic conditions. A proactive model-based approach to integrated multi-tier provisioning and placement was proposed in [5].

Integrated policies could be implemented for one or more data centers in Shirako by constructing a single actor that fills the roles of both a broker and site authority for all of those sites. In essence, this structure requires that the data centers operate under central control. The lack of site autonomy may result in a less scalable system, and may be unsuitable for a federation.

4 Migration

When VMs are provisioned and resized dynamically, a cluster site may be forced to adjust its placement choices by migrating VMs. This can occur, for example, if VM slivers grow in response to load changes in a hosted service, or if requests for new VMs are granted. We have extended Shirako to enable site policies to control migration and checkpoint/restart for Xen VMs.

Migrations offer cluster sites additional flexibility to implement more complex and adaptive placement policies, as suggested in the previous section. This flexibility comes at a cost: sites must be conservative in their use of migration, since it is expensive and disruptive.

The *migration planning problem* is as follows: given a set of physical hosts, a set of slivers distributed among the physical hosts, and a target reassignment of slivers to physical hosts, find a sequence of migrations that converges to the target assignment with minimal cost, without violating the capacity constraints of individual hosts. Costs include data storage and transfer, service interruptions, and number of migrations.

It is possible to show, using a reduction from a similar problem [8], that finding a plan that minimizes the number of migrations is NP-hard. Approximation algorithms exist to solve instances of the problem at the expense of additional storage space and physical hosts.

4.1 Resource Fidelity and Migration

Migration is disruptive to the guest to varying degrees. We can quantify the disruption with the metric of *fidelity* Fidelity is a measure of the resource allotment an application receives compared to what it is promised or what it demands (whichever is less).

Current virtualization solutions typically offer two types of migration: *stop-and-copy* and *live*. The former approach suspends a VM and saves its state to stable storage. At a later time, the VM can be reinstantiated on a different host using the saved state. Stop-and-copy migration offers more flexible migration planning, because it has lower overhead and VM images may be saved on storage. However, it temporarily suspends all applications running inside the VM. Live migration addresses this limitation by significantly reducing application downtime to within a few milliseconds [4]. However, live migration may take longer to complete. Overall, migration is a powerful tool but it affects running applications and may impact on their performance.

Figure 1 illustrates the impact of both kinds of migration on fidelity. In this experiment, we run `gamut` [14] on a guest VM. `gamut` is a synthetic application that interleaves execution of a compute-intensive loop with periods of idleness to match a target CPU utilization. We fix the target CPU rate and measure the number of iterations the application performs as its VM is being repeatedly migrated between two physical hosts.

All migration operations in Figure 1 complete within a few seconds. It also shows how migration affects application fidelity. When a migration occurs, application performance degrades. Stop-and-copy migration decreases fidelity more noticeably since the application must be suspended and its VM copied to the destination. Surprisingly,

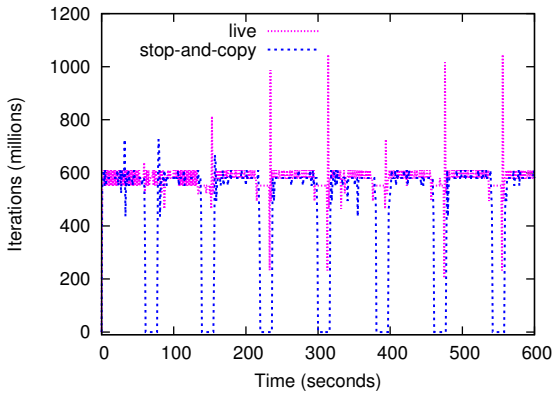

Figure 1: Progress of a synthetic `gamut` benchmark running in a Xen VM that migrates on every lease extension under Shirako/COD. Stop-and-copy migration interrupts the application for up to 20 seconds, while live migration is less disruptive.

live migration occasionally increases fidelity since the Xen hypervisor requires some time to set the resource limits on a new virtual machine. This may affect the performance of not only the migrating VM, but of all VMs hosted on the same physical host. In our policies, we use stop-and-copy migration only when live migration is not possible, e.g, the migration plan has circular dependencies.

5 Brokering and Migration in Shirako

Federated utilities, such as grids, may become very large, with many cluster sites and multiple guests competing for resources. In these cases it is important to balance the autonomy individual cluster sites have to implement their own local policies with the system's ability to allocate resources globally to their highest and best use. To accommodate this balance, we factor out policy from individual cluster sites into common intermediaries, or *brokers*.

Introducing brokers as separate entities carries further the logical separation of provisioning and placement (Section 3). Provider sites control the placement of VMs on their own clusters, but they explicitly delegate provisioning power for some or all of their resource holdings to one or more brokers. Brokers may run on behalf of third parties and may arbitrate lease requests for multiple providers. Controlling both provisioning and placement at the cluster site offers the potential to integrate provisioning and placement policy, but factoring out provisioning into a broker makes it possible to coordinate resource assignment across multiple autonomous sites. A broker can satisfy multiple requests for resources at different sites atomically. Brokers offer another level of indirection, which may be used to arbitrate contending requests from many guests, e.g., to maximize a global provisioning objective. Further, this decoupling of function enables a competitive market for third-

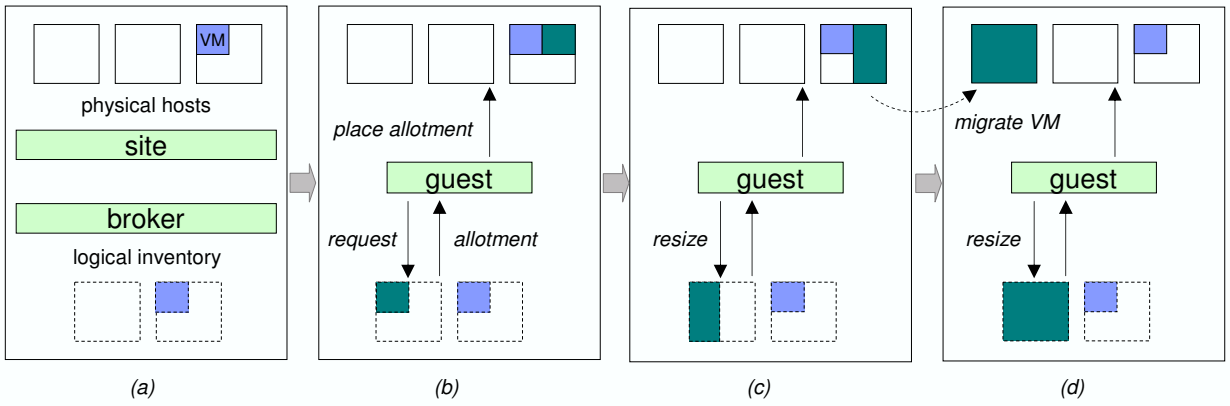

Figure 2: Sites delegate rights to provision host servers to brokers (a). Brokers determine how to allocate resource slivers from their logical inventory to serve requests. Sites use a local policy to map allocated slivers to host servers, and bind them to virtual machines (b). Guests may request to resize their slivers, e.g, in response to changing load (c). Sliver resizing or fragmentation may force the site to migrate a VM to another host (d).

party brokers to allocate shared infrastructure.

As suggested in Section 3.3 there is a tension between provisioning (in the broker) and autonomous placement policies in the site. Since the cluster site uses its own local placement policy, it may place VMs in a way that conflicts with the broker's provisioning choices, especially when VM slivers grow. If the broker is correct and honest, then some feasible placement will always exist to satisfy its provisioning choices. The broker maintains an assignment—based on a partial mapping of logical resource units onto the physical nodes at the site—as a side effect of its provisioning policy. If the site deviates from this default placement, the broker's provisioning choices may force it to migrate to obtain a feasible assignment, even if a different placement policy (or provisioning choice) could have avoided the migration.

Figure 2 demonstrates a simple example of how more migration can occur when a broker and a cluster site use policies with competing objectives. In Panel *(a)*, the broker has two hosts delegated to it from the site which it places into its *logical inventory*. As the broker provisions requests, using resources from its inventory, it places them following the *room-for-growth* policy. The *room-for-growth* policy is a greedy algorithm that fills requests in a worst-fit fixed order based on resource availability and the number of requests at the broker for every allocation period. The worst-fit heuristic gives the broker maximum flexibility to satisfy any subsequent growth in guest slivers requested when the guests renew their sliver leases.

In contrast, the cluster site places machines following a greedy best-fit policy (Panel *(b)*). This policy packs VMs onto machines as tightly as possible. We refer to it as *energy-aware* because it allows the cluster site to power down idle hosts to save energy. The conflict between these policies occurs when a guest requests its VM to be resized multiple times: due to its policy, the broker has space on its inventory host, but the site must eventually migrate the VM to a different host to accommodate the growth (Panels *(c)* and *(d)*).

If the placement policy cannot locate free space for a new sliver, it migrates VMs to create space; VMs are migrated to hosts according to the provisioning policy's placement—this ensures that space will be made for the new sliver. We are exploring a home-based migration scheme that restores a feasible placement in a bounded number of steps, at the price of reserving memory at each VM's home node. To preserve accountability, the broker annotates its directives with a verifiable proof that its sequence of VM sizing choices always has a feasible assignment at each site. The site can derive a feasible placement in linear time from the state it maintains as it verifies these proofs. This default placement defines each VM's logical "home".

Our current approach to migration in Shirako simplifies migration planning. If migration is necessary, Shirako performs live migration if there is enough space on the new host. However, if live migration is not possible Shirako suspends the VM and saves its image to disk, making use of extra storage space to reduce the complexity of the problem. Suspending the image to disk prevents potential deadlock and breaks any migration loops. The placement policy is free to place VMs on hosts independent of the provisioning policy as long as it can locate free space. We are investigating the tradeoff between the placement flexibility offered by migration and the performance overhead incurred.

5.1 Migration Experiment

To demonstrate how separating provisioning and placement may cause additional migrations at a cluster site, we perform an experiment to show the number of migrations

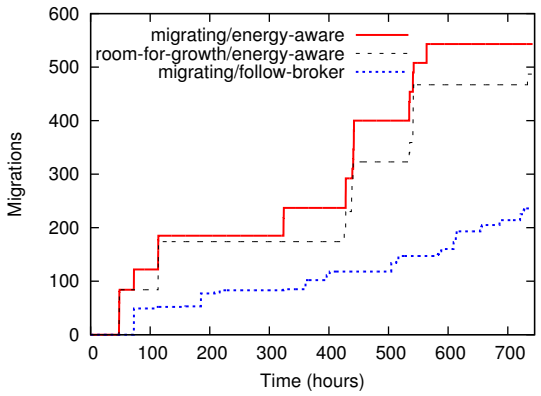

Figure 3: Number of migrations incurred for three broker/cluster site policy combinations. Different combinations of provisioning and placement policies incur different numbers of forced migrations. The largest number of migrations results when the broker provisioning policy incorporates migration, and the site selects a conflicting placement using an energy-aware placement policy.

at a cluster site when using two different provisioning and placement policies. For this experiment we run an emulation of two competing applications in a hosting site deployment of 100 physical machines. Two guests request resources; one based on a one month trace from a production application service in a major e-commerce website, and the other based on a job trace from a production compute cluster at Duke University. Guests request VMs at the granularity of half the size of a physical machine (e.g., one-half share CPU, memory, bandwidth, disk). For the purpose of this example, the cluster site delegates full control to the broker over subdividing its resources into VMs. The broker provisions VMs for guest applications and, in doing so, creates a logical placement of resources within its logical inventory.

For our experiment we use two cluster site placement policies: (1) *follow-broker* that follows the broker's provisioning decisions when performing its placements, and (2) *energy-aware* that tightly packs machines as described above. We also use two broker provisioning policies: (1) *room-for-growth* that loosely allocates VMs to machines as described above, and (2) a *migrating* variant that allows broker migration. Broker migration occurs when a broker moves an allocated VM in its logical inventory to a different logical machine; this may occur if the proposed size for the VM being resized cannot fit on its initial machine. This can allow the broker to potentially allocate a request that would otherwise fail.

Figure 3 shows the cumulative number of migrations the cluster site performs for each of three scenarios: *room-for-growth/energy-aware*, *migrating/energy-aware*, and *migrating/follow-broker*. The fourth scenario, *room-for-growth/follow-broker*, is not pictured as it leads to no migrations. By using a broker and cluster site with competing policy goals, we approximate worst case migration behavior at the cluster site. As intuition suggests, the number of migrations at the cluster site increases when the broker uses migrations to satisfy requests (*migrating*). From the three alternatives presented in Figure 3, when the cluster site uses its *follow-broker* policy, there are the least number of migrations. For *room-for-growth/energy-aware*, we observe less than 500 migrations over the month long trace.

6 Conclusion

Enhanced performance of virtualization technology exposes the need to address the resource management challenges in this space. The ability to control the resources assigned to a single VM creates a need for an intelligent infrastructure that can adapt to changing demands and conditions by automatically "turning the knobs" of performance, isolation, migration, and sliver resizing. This paper exposes some of the challenges that resource management must address.

The Shirako architecture factors provisioning and placement where provider sites retain control over VM placement, but delegate limited provisioning power to brokers. This separation is key to preserving a site's autonomy in a federated infrastructure: a cluster site is unlikely to contribute substantial resources to an infrastructure that does not allow it to control placement objectives (e.g., energy or thermal-aware) that may contribute significantly to operating costs. We show how migration is a useful mechanism to resolve conflicts between the separate provisioning and placement policies.

We also outline extensions to the Shirako resource leasing toolkit to allow sliver allocation, sliver resizing, and to enable site policies to control migration and checkpoint/restart for Xen VMs. We are using this system to explore the policy space for adaptive virtual machine hosting on federated clusters.

References

[1] P. Barham, B. Dragovic, K. Faser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[2] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP)*, pages 103–116, October 2001.

[3] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic Virtual Clusters in a Grid Site Manager. In *Proceedings of the Twelfth International Symposium on High Performance Distributed Computing (HPDC)*, June 2003.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In

Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI), May 2005.

[5] R. P. Doyle, O. Asad, W. Jin, J. S. Chase, and A. Vahdat. Model-based resource provisioning in a Web service utility. In *Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2003.

[6] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang. Virtual Clusters for Grid Communities. In *Cluster Computing and Grid (CCGRID)*, May 2006.

[7] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, October 2003.

[8] J. Hall, J. Hartline, A. R. Karlin, J. Saia, and J. Wilkes. On Algorithms for Efficient Data Migration. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2001.

[9] D. Irwin, J. Chase, L. Grit, and A. Yumerefendi. Self-Recharging Virtual Currency. In *Proceedings of the Third Workshop on Economics of Peer-to-Peer Systems (P2P-ECON)*, August 2005.

[10] D. Irwin, J. S. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum. Sharing Networked Resources with Brokered Leases. In *Proceedings of the USENIX Technical Conference*, June 2006.

[11] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic Placement for Clustered Web Applications. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, May 2006.

[12] T. Kelly. Utility-Directed Allocation. In *First Workshop on Algorithms and Architectures for Self-Managing Systems*, June 2003.

[13] T. Kelly. Generalized Knapsack Solvers for Multi-Unit Combinatorial Auctions. In *Workshop on Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems*, July 2004.

[14] J. Moore, J. Chase, K. Farkas, and P. Ranganathan. Data Center Workload Monitoring, Analysis, and Emulation. In *Proceedings of Computer Architecture Evaluation using Commercial Workloads*, February 2005.

[15] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers. In *Proceedings of the 2005 USENIX Annual Technical Conference*, pages 61–74, April 2005.

[16] L. Ramakrishnan, L. Grit, A. Iamnitchi, D. Irwin, A. Yumerefendi, and J. Chase. Toward a Doctrine of Containment: Grid Hosting with Adaptive Resource Control. In *Supercomputing (SC06)*, November 2006.

[17] P. Shivam, S. Babu, and J. S. Chase. Learning performance models in network utilities. In *IEEE International Conference on Autonomic Computing (ICAC)*, June 2006.

[18] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, December 2002.

[19] C. A. Waldspurger. Memory Resource Management in VMware ESX Server. In *Symposium on Operating Systems Design and Implementation*, December 2002.