VBoom: Creating A Virtual Machine Real Estate Boom

Kyung-Hwa Kim
Columbia University
New York, NY, USA
khkim@cs.columbia.edu

Hai Huang, Salman Abdul Baset and Chunqiang Tang
IBM Watson Research Center
Yorktown, NY, USA
{*haih, sabaset, ctang*}*@us.ibm.com*

***Abstract*—Cloud providers sell identically configured virtual machines (VMs) for the same price. Customers purchasing these VMs expect that they perform similarly and are allocated the same amount of *virtual* resources.**

In practice, however, the real performance of identically provisioned VMs depends on the underlying hardware, i.e., how the hardware is configured, and how much shared resources are consumed by co-located VMs. As workloads often have different resource requirements (e.g., CPU or disk I/O bound), a physical machine can be a better host to one VM than another, and swapping the locations of the two VMs can improve the performance (or any other metrics) of both VMs. However, cloud providers are unlikely to offer a VM swapping service since it is a tacit admission of providing different quality of services while charging the same rate.

We propose, VBoom, a cloud provider-agnostic system in which VMs can dynamically relocate themselves (via location swapping) in a cloud environment if the new location better serves the needs of the VM. We discuss technical and business benefits and challenges in building such a system. When location matters, we believe that through VBoom, real estate of virtual machines can be established in a completely market-driven fashion.

I. Introduction

In the Infrastructure-as-a-Service (IaaS) cloud service model, users can obtain on-demand resources by rapid provisioning of virtual machines. VMs of one user are typically co-located with VMs of other users on a physical host, especially in a public cloud. Each identically configured VM (e.g., medium EC2 instance [5]) is expected to achieve similar performance for CPU, memory, disk and network I/O irrespective of its location within the cloud, i.e., the physical machine where the VM is provisioned. However, a cursory look at the SLAs of cloud providers such as Amazon [5], Rackspace [6] and Azure [7] indicates that they do not provide any performance guarantees. Further, there are several experimental studies which indicate that the performance of identically provisioned VMs varies in EC2. Barker *et al.* [8], [12] showed variations in CPU, disk, and network performance of a single instance in EC2 over a period of several days which they attributed to failure of performance isolation. Dittrich *et al.* [12] observed performance differences across availability zones and sites[1] in EC2 and point the underlying hardware differences as the root cause. In addition, they also observed performance differences across similar instances within the same availability zone.

In this paper, our goal is to leverage the performance difference among identically provisioned VMs, that can be hosted either by the same cloud provider or by different cloud providers, to their advantage in a provider agnostic manner. The key observation is that since workloads are different, their resource needs may also be different and vary with time. For example, some workloads may be CPU-intensive while others may be more network or disk intensive. The difference in resource requirements of VMs due to their myriad workloads imply that, at any moment in time, a 'bad location' (i.e., a physical host) for one VM might be a good location for another VM. Consequently, the performance (or other metrics) of both VMs can be improved by swapping the physical hosts of two VMs.

We propose a system called VBoom, which allows VMs to take advantage of performance differences within the same or different clouds in a *provider-agnostic* manner. Our system comprises distributed *Agents* and a centralized *Broker*. The agents run on VMs interested in participating in location optimization. The centralized broker is used by agents to discover other participating VMs, verify information reported by agents, facilitate negotiations between agents, and if agreed by both parties, securely swap VMs.

The rest of the paper is organized as follows. Sections 2 discusses the pros and cons of VM relocation methods, including provider-assisted migration and swapping. Section 3 describes the design and implementation of VBoom, and Section 4 shows our experimental results. Section 5 discusses the economics and ownership issues in VBoom. Related work is described in Section 6.

II. Moving Virtual Machines Across Physical Hosts

There are multiple ways of moving virtual machines across physical hosts. We discuss these options and their pros and cons in the following sections. Even though we advocate for a provider-agnostic approach, VM relocation is relatively 'easier' with provider assistance.

[1]A site is a physical location containing availability zones which are supposed to be isolated from each other.

A. Provider Assisted Migration

The most obvious approach of moving across physical hosts is to perform live or offline migration of VMs with the assistance of cloud providers, using technologies such as VMotion [4]. However, this mechanism requires exposing the underlying physical infrastructure and a (partial) hypervisor-level control to the customer, which is not desirable from the cloud provider's perspective. Moreover, it may not be possible to move VMs across cloud providers as not all providers may expose this functionality. Nevertheless, this approach is the least disruptive and the cleanest way to move virtual machines.

B. Few From Many Provisioning

One can resort to a brute force approach by provisioning a large number of virtual machines, measuring their performance, picking the best performing ones, and decommissioning the rest. This approach has several problems: i) the result is largely dependent on where new virtual machines are provisioned (lack of randomness in the placement algorithm can yield poor result), ii) the cost of provisioning a large number of virtual machines can be prohibitive, iii) in a highly utilized environment, location of the newly provisioned virtual machines can be severely constrained, and iv) the performance of VMs may vary as the workload running in the co-located VMs changes.

The most obvious problem with this approach is its high cost. Even though most clouds use the pay-as-you-go model, users are commonly charged at an hourly granularity. For example, if a user creates 100 VMs for only a minute to run some benchmarks before decommissioning them, he will still be charged 100 hours of VM usage. Nevertheless, this approach is provider-agnostic and can be a feasible solution in determining the initial workload placement. However, if the workload of the co-located VMs change, then the performance of VMs may be affected. A strategy in which few VMs are periodically selected from a large number of provisioned VMs will likely be cost prohibitive.

C. Swapping

The third approach is to swap the locations of two consenting virtual machines after incentives have been negotiated and accepted by both. Unlike the 'few from many' approach, finding a better location for a virtual machine no longer depends on the randomness of the initial virtual machine placement, but rather on finding a cooperating virtual machine that occupies a better location and can be persuaded to trade its location.

Similar to migration, swapping two virtual machines can be easily done if hypervisor-level control is acquired. Luckily, "live swapping" is not difficult to emulate without it, which is the focus of this paper.

Swapping two virtual machines' locations (VM_1 and VM_2) is essentially the same as swapping their persistent storage, i.e., when VM_1 boots from a disk partition that contains VM_2's data and vice versa, VM_1 and VM_2 for all practical purposes have exchanged locations[2]. Swapping two disk images can be done through a temporary storage, e.g., ($tmp = Disk_1$, $Disk_1 = Disk_2$, $Disk_2 = tmp$). Swapping two virtual machines via a temporary storage is illustrated in Figure 1, and the steps in this process is listed below.

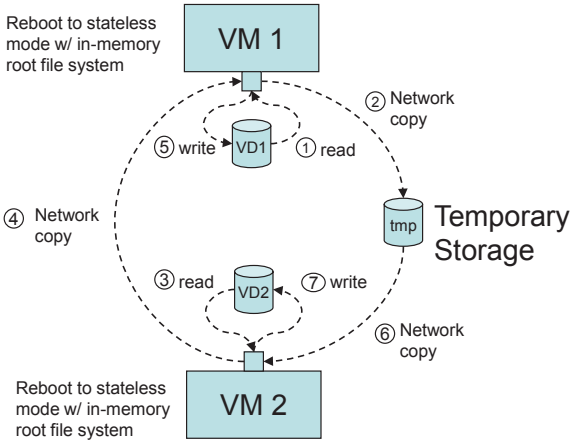


Figure 1. Swapping two virtual machines.

```
0. Reboot VM1 and VM2 to a stateless mode
1. Read data from virtual disk 1 (VD1)
2. Write VD1 content to the temporary storage
3. Read data from virtual disk 2 (VD2)
4. Send VD2 content from VM2 to VM1
5. Write VD2 content to VM1's persistent storage
6. Read VD1 content from the temporary storage
7. Write VD1 content to VM2's persistent storage
```

To maintain disk image fidelity, file system level copy is not sufficient. One would need to perform block level copy to preserve disk meta-data and data layout. Additionally, before a disk partition can be copied, it needs to be in a quiescent state, i.e., not mounted. We achieve this by rebooting its virtual machine in a stateless mode from a RAM disk (initrd). An agent within the RAM disk can then copy the content from the unmounted disk partition to the designated temporary storage via either IP network or SAN. When the two virtual machines restart, their original workload will now run in new locations. However, the VMs running in the new location must use the IP addresses of VMs that were running previously; otherwise, network connectivity will no longer work. If those VMs were located at the same subnet, IP addresses would be preserved without further steps. However, if the VMs were located in different subnets, additional steps to change IP addresses would be required. As there are works in network virtualization area

[2]We assume only similarly configured and sized virtual machines can be swapped, i.e., a small virtual machine instance cannot be swapped with a large instance and can only swap with other small instances.

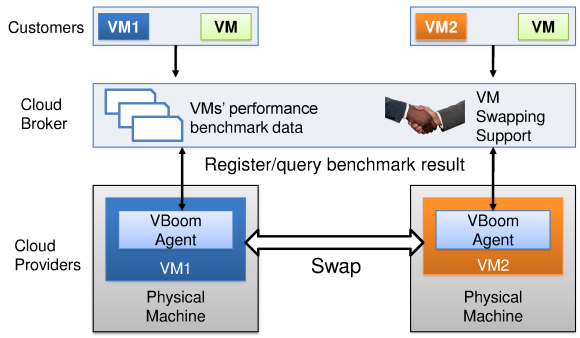


Figure 2. VBoom architecture

to address exactly this problem, we focus more on VM swapping mechanisms in this paper.

When VMs are swapped, their ownership must also be exchanged so that users cannot inadvertently access each other's VMs. Moreover, as this provider-agnostic swapping mechanism requires a reboot, it will not be suitable for all applications, especially those that cannot tolerate even a brief service interruption, e.g., those in the financial sector. On the other hand, workloads that tend to be non-interactive, run in batch mode, and/or stateless (e.g., MapReduce tasks) are good candidates to receive the full benefit of performing continuous location optimization.

We discuss VBoom architecture and these issues in the next two sections.

III. VBoom Architecture

In this section, we describe the architecture of our system that is composed of distributed agents and a centralized broker. Then, we describe the design of trustable micro-benchmarks.

A. Broker and Agents

To enable VMs to swap their locations without provider assistance, we suggest a framework that assists VMs to exchange their performance information and swap their contents. The framework includes a *broker* service, which is a central server trusted by VMs, that maintains the list of all agents participating in the VBoom network.

In order to participate in the VBoom network, an agent running on each VM first *registers* itself with the broker, and stores its network address, instance specification, i.e., provisioned CPU, memory, disk, network, and usage of above resources (e.g., CPU utilization, disk throughput). The broker stores this information in an internal database and provides it when agents *query* for matching VMs. The broker service can be compared to a real estate broker in the real world, who maintains a list of real estate and recommends the appropriate properties to the customers. As an example, consider a VM that is experiencing good CPU performance but mediocre disk and network performance. The VM queries the broker for VMs with good disk and network performance and advertises itself as having decent CPU performance. The broker then finds a list of suitable candidate VMs and returns them to the querying VM. The broker can either return the complete set of VMs matching the query if it is not too large or a subset based on a suitable criteria (e.g., fairness, load distribution).

However, a broker is simply not sufficient to provide the most accurate data of VM performance for two reasons. First, the information can be stale because the performance of each VM changes over time. Second, workloads running on a VM may have specific performance requirements that can only be determined by running an application specific micro-benchmark. Therefore, the agents that are recommended by the broker need to run an application-specific micro-benchmark to precisely gauge their performance.

B. Trustable Micro-benchmarks

The user of the probing VM defines a set of micro-benchmarks to gauge performance of other VMs in order to determine their suitability for swap. The micro-benchmarks can measure several properties such as CPU, memory, network latency (RTT between two VMs), or network bandwidth. To evaluate the CPU performance, an agent measures the elapsed time to execute a pre-defined function that contains a CPU-intensive calculation. Similarly, an agent can run micro-benchmarks to gauge the disk and network throughput.

The probing agent requests the agents running on VMs being probed to run these micro-benchmarks and passes the relevant executable of the benchmark. To protect against malicious code, the micro-benchmarks can be run in a sandbox. Alternatively, the applications can submit their micro-benchmarks to the broker which the VMs trust. The VMs only run the benchmarks that have been cryptographically signed by the broker. After running the micro-benchmarks, the probed agents return the result to the probing agent.

Once the probing agent finds a list of suitable VMs, it sends them a swap request. The agents running on VMs being probed must also ascertain the performance of probing agent by requesting it to run micro-benchmark specific to their applications. The agreeing agents can then handshake and initiate the swap process discussed in Section II.

A problem is that the probed agents can also manipulate the results of micro-benchmarks before they send it to the probing agents. It is impossible to absolutely guarantee the validity of the results unless they are run within the hypervisor, which violates the provider-agnostic property of VBoom. Thus, we suggest an alternative mechanism to verify the results. Instead of rebooting a VM from any RAM disk as discussed in Section II-C, a VM can be

rebooted from a remote disk that is under control of the broker. In this way, the broker can prohibit users from accessing their VMs temporarily. Then, the broker executes the benchmarks and verifies the results reported by the agents earlier. Since users cannot manipulate the remote disk under the broker's control, the benchmark results are trustable.

IV. Evaluation

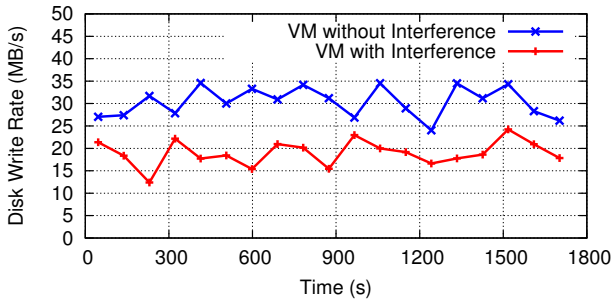
We implemented the automated provider-agnostic swapping method described in Section II-C and the VBoom components (broker and agents) discussed in Section III, and tested them on various hypervisors, including Xen, KVM, and VMware. The results verified that the VBoom's provider-agnostic approach was successful in swapping between VMs hosted by the same type of hypervisor, and furthermore, that swapping VMs across different types of hypervisors is also possible if the hypervisors support full-virtualization (e.g., KVM and VMware).

In this section, we present experimental results observed from a private cloud operated by Columbia University. This private cloud consists of 80 physical machines and hosts a number of VMware instances on demand. Our experimental setup used seven VMs located on different physical machines and co-located with other VMs owned by different users. To test a real-world scenario, we ran MapReduce tasks using the Hadoop framework, and verified that the VBoom approach significantly reduced the total elapsed time by swapping underperforming VMs.

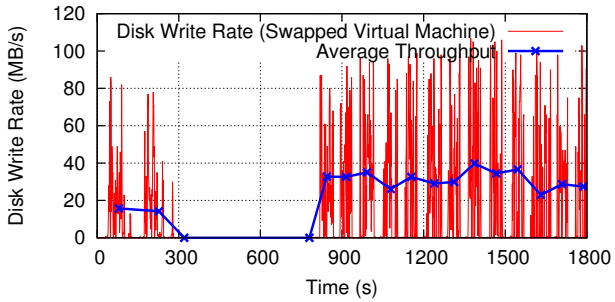
A. MapReduce Task

We ran five Hadoop nodes on different VMs - one master node and four slave nodes. Two of the salve nodes were run on normally performing machines, while the other two were run on machines with disk I/O interference. The master node ran *TestDFSIO*, a benchmark tool provided by the Hadoop framework. The role of the master node was to distribute I/O-intensive tasks to each slave node and collect statistics as they completed their tasks. We ran a total of 22 MapReduce tasks sequentially, with each task writing 800 MB to the local disk.

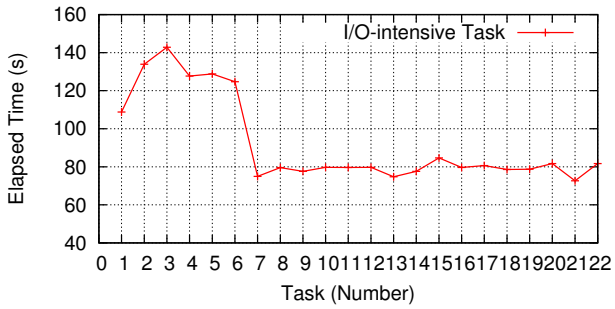
First, without applying the VBoom swapping method, we measured average disk write throughput on each VM (See Figure 3(a)). The average throughput of the VMs on normal machines was 31 MB/s, while the average for the VM on the machine with disk interference was 18 MB/s. Next, we ran VBoom agents on all VMs and executed the same MapReduce tasks again. Figure 3(b) shows disk write throughput observed for the VM with disk interference. Near the beginning, when the throughput was found to be less than expected (under 20 MB/s), the VBoom agent activated its swapping process, automatically interacting with the broker to discover the best location to swap.



(a) Throughput on a normal VM



(b) Throughput on a swapped VM



(c) Elapsed time for each task

Figure 3. MapReduce task throughput

Between 300 to 800 s, the throughput went to zero while the VM was in the process of swapping. Once the swap was completed, the average throughput increased to over 30 MB/s and remained at that level for the rest of the experiment. Figure 3(c) shows the time needed to complete each task. The swapping process begins between tasks 2 and 3 and ends before task 7. Once the swapping was finished, each task took only about 80 s to complete, with all four slaves working at normal disk throughput. In all, our 22 MapReduce tasks took 2790 s without VBoom and 2028 s with VBoom, which is a reduction of 27%. Note that, in general, this reduction ratio depends on the size of the tasks and the number of VMs showing poor performance. If the tasks are larger and more VMs are suffering from performance degradation, the VBoom swapping method will yield grater time reductions.

V. Discussion

In this section, we discuss issues related to the economics, swapping cost, and VM ownership.

A. Swapping Cost and VM performance

In our experiments, it took about 8 min to completely swap two VMs. There are several ways to reduce the length of this swap interval. For one, the content of the disk can be compressed before it is sent over the network. For another, the swapping VMs can be made to exchange only those blocks that differ. The latter method is especially effective since the VMs will often be running on the same or similar OS in a cloud environment, and will therefore have many identical blocks that do not need to be swapped. Implementing and evaluating such optimizations is left to future work.

Another issue worth addressing is the variation of VM performance over time. When two VMs agree to swap their locations, there is no guarantee that the performance of those locations will remain stable across the swapping interval. Though performance degradation is usually caused by underlying hardware problems, and is therefore unlikely to change after only a few minutes, it is also true that performance variations can be caused by non-hardware factors, such as other VMs running on the same physical machine. To address these variations, we would need a long-term pattern analysis of performance changes for a given machine, and a smart optimization algorithm that can use this analysis to reduce the risk of bad swaps.

B. Economics

We believe VBoom will introduce a new economic model for VMs based on their locations. As we have shown previously, CPU speed, I/O performance, fault behavior, and other characteristics of VMs can vary significantly depending on where they are provisioned, i.e., VM real estate. Due to these differences, some customers may choose to pay a little bit more to secure premium VM locations while others may relocate their VMs to less expensive locations. Relocation services can be provided by cloud providers, however, it makes more sense for a neutral third party to provide such services especially for relocation across different cloud providers.

1) A Market Place for VM Real Estate: In the previous sections, we discussed VM swapping that can bring mutual performance benefit. However, the number of situations where mutual benefit can be achieved might be somewhat limited. This situation changes when other types of incentives are introduced, particularly monetary incentives, as their inclusion widens the definition of mutual benefit beyond just basic VM performance. The willingness to relocate VMs for an economic benefit will encourage many customers to participate, and the resulting premium pool of VM locations would entice more customers to participate to satisfy any requirements that are beyond what cloud providers can provide.

This creates a healthy market for both sellers and buyers of a VM real estate to satisfy different needs. However, to facilitate these transactions and ensure their security and authenticity, a brokerage service is needed. A broker is a trusted service, and to ensure authenticity of a transaction, e.g., to verify the real estate that one claims is really valid, various agents are dispatched onto VMs that are to be swapped. Agents are simple programs or scripts that are secure and low-overhead. The price of a VM location can vary from time to time and is completely driven by supply and demand, but it is also a function of the standard rate that cloud providers set. An example scenario of VM bidding is as follows. A customer who is willing to swap his VM locations at a gain first registers with the broker and sets a desired price. Another customer who wants to find better locations for his VMs can query the broker with a set of desired features. If there is a match and its price is below the buyer's budget, a swapping request is sent to the broker to initiate the swapping operation and related financial transactions. Also, since it is possible that there are multiple buyers, a simple bidding system can facilitate fair competition.

In this model, some customers pay more to secure premium locations. However, one may ask why customers who are willing to spend more for a VM location do not simply purchase more resources from the cloud provider. There are two reasons. First, cloud providers usually offer only a few coarse-granular instance types. For example, an alternative option of a *small* instance in Amazon EC2 is a *large* instance, which is four times more expensive than a *small* instance ($0.085 per hour for a *small* instance and $0.34 per hour for a *large* instance [2]). Second, even if a user creates a new *large* instance, it will be instantiated at a random location. Thus, it does not necessarily guarantee the desired performance.

C. VM Ownership Transfer

In a cloud, a VM is owned by the user who initially creates the VM. The process described in Section II-C swaps the contents of two VMs, but does not change the ownership of the VMs perceived by the cloud management system. This causes several issues. First, the usage of a VM would be incorrectly billed to its original user rather than its current user. Second, even if the original user cannot log into the VM, she still has control over the VM by using the cloud's management tool, e.g., to take a snapshot of the VM's disk image and gain access to its data, which becomes a security breach.

It is obvious that, after two VMs are swapped, the ownership of the VM must be transferred to its new user so that the old user no longer has control over the VM.

This can be achieved in multiple ways. This capability may be offered by a third party that provides value-add services on top of one or more public clouds. One example is that a third party (such as Rackspace [3]) launches VMs in Amazon EC2 on behalf of its customers. In this case, the third party is the real owner of those VMs from EC2's perspective. Then the third party can offer its own layer of VM ownership management and allow VM ownership transfer between its customers. In this manner, we do not need any support from cloud providers, so that we can preserve VBoom's provider-agnostic feature. However, in the future, we can expect that cloud service providers also provide a public service to transfer the ownership of a VM from one user to another user. It is reasonable to request cloud providers to offer this kind of service since this capability is generally useful even outside the context of VM swapping (see a related topic in the Amazon Web Service discussion forum [1]).

VI. Related Work

To the best of our knowledge, there is no prior work on consumer-driven VM swapping in a public cloud, although there exists a large body of work on related topics such as resource allocation and VM placement. Tang *et al.* [13] developed a CPU and memory aware algorithm for dynamic placement of Web applications in enterprise data centers. Meng *et al.* [11] proposed a method for traffic-aware virtual machine placement. Similarly, Hyser *et al.* [10] employed user policies to guide VM-to-PM (physical machine) mapping. Bobroff *et al.* [9] focused on reducing the number of physical machines needed to run a set of VMs while minimizing SLA violations. While all these studies take a provider-centric view and use a central server for collecting performance metrics and executing the control algorithm, VBoom takes a consumer-driven approach to rearrange VMs in a distributed manner.

VII. Conclusion

We have proposed and implemented a system, VBoom, which measures and leverages the performance variations among VMs to their advantage in a decentralized fashion without any support from cloud providers. Our proposed system comprises a central broker and distributed agents running on VMs. An agent asks peer agents to run user-specified micro-benchmarks to identify better locations for its VM, and negotiation between agents happens to determine whether or not incentives to both parties can be met. If so, virtual machine locations are swapped without any help from cloud providers. We discussed several design challenges and security issues in VBoom. We believe that a VBoom like approach can engender a market place for VMs, where users can buy and sell their VMs based on the VMs' observed characteristics. Our prototype demonstrated the feasibility of provider-agnostic swapping and showed that it could significantly reduce the total runtime of sample MapReduce tasks running in a private cloud.

References

[1] Amazon AWS Forum Posting. https://forums.aws.amazon.com/message.jspa?messageID=162083. [Online; accessed July 2011].

[2] Amazon ec2 pricing. http://aws.amazon.com/ec2/pricing/. [Online; accessed July 2011].

[3] Rackspace. http://www.rackspace.com. [Online; accessed July 2011].

[4] VMware vMotion. http://www.vmware.com/products/vmotion/overview.html. [Online; accessed July 2011].

[5] Amazon EC2 SLA. https://aws.amazon.com/ec2-sla/, 2011. [Online; accessed January 2012].

[6] Rackspace Cloud Servers SLA. http://www.rackspace.com/cloud/legal/sla/, 2011. [Online; accessed January 2012].

[7] Windows Azure Compute SLA. https://www.microsoft.com/download/en/details.aspx?displaylang=en&id=24434, 2011. [Online; accessed January 2012].

[8] Barker, S. K., and Shenoy, P. Empirical evaluation of latency-sensitive application performance in the cloud. In *Proceedings of ACM MMSys* (2010).

[9] Bobroff, N., Kochut, A., and Beaty, K. Dynamic placement of virtual machines for managing sla violations. In *Proceedings of IM* (2007).

[10] Hyser, C., McKee, B., Gardner, R., and Watso, B. J. Autonomic Virtual Machine Placement in the Data Center. In *HPL-2007-189* (2008), HPL-2007-189.

[11] Meng, X., Pappas, V., and Zhang, L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of INFOCOM, 2010* (March 2010).

[12] Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. Runtime measurements in the cloud: observing, analyzing, and reducing variance. In *Proceedings of VLDB Endow.* (September 2010).

[13] Tang, C., Steinder, M., Spreitzer, M., and Pacifici, G. A scalable application placement controller for enterprise data centers. In *Proceedings of the WWW* (New York, NY, USA, 2007).