Performance-aware server consolidation with adjustable interference levels

Luis Carlos Jersak
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Av Ipiranga 6681 - Porto Alegre, Brazil
luis.jersak@acad.pucrs.br

Tiago Ferreto
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Av Ipiranga 6681 - Porto Alegre, Brazil
tiago.ferreto@pucrs.br

ABSTRACT

Virtualization technologies and server consolidation are the main drivers of high resource utilization and energy efficiency in modern Data Centers. However, some combinations of virtual machines into the same server may lead to severe performance degradation. This performance degradation is known as virtual machine interference. In a typical Data Center, different measures of virtual machine interference can be employed, depending on applications importance. Supporting a higher virtual machine interference may result in a higher consolidation, while strict low interference requirements may demand more resources. This paper presents an algorithm for server consolidation that uses an adjustable virtual machine interference threshold to map virtual machines into physical servers, allowing users to get a better trade off between amount of resources and performance according to their needs. Simulation results show that the solution succeeds in maintaining the interference levels below a defined threshold while also providing efficient server consolidation.

CCS Concepts

•Computer systems organization → Cloud computing;

Keywords

virtualization; server consolidation; interference

1. INTRODUCTION

Virtualization is a core technology in modern Data Centers. It provides higher abstraction and flexibility to deal with IT resources, and enables a substantial reduction in power consumption and footprint. In the last years, enhancements in hardware and software enabled the utilization of virtualization technologies in production systems with minimal performance impact. Furthermore, virtualization is also recognized as one of the main pillars of cloud computing.

Nevertheless, multiplexing computing resources of a physical machine between several virtual machines (VMs) is not trivial and may lead to performance issues between virtual machines. There are several evidences that show that virtual machine interference (i.e. the performance penalty caused by the dispute of several VMs or applications over the same computational resource) may occur, depending on the workload running on each VM. In some cases, the impact is so high that may even lead a virtual machine to stop responding [8, 12, 19]. Despite the existence of certain hypervisors specially suited for specific workloads in order to minimize the interference, all hypervisors present some interference on VMs, depending on the characteristics of the VMs' workloads.

In order to minimize the interference, VMs may be grouped together based on the impact they cause on each other. Current server consolidation algorithms focus on minimizing the number of physical machines used [7]. Some algorithms also include rules to minimize the interference [20, 3, 10]. However, the effectiveness of server consolidation, i.e., minimization of the number of physical machines, and interference, are opposite goals. Depending on the level of interference desired, the consolidation may be more or less aggressive.

Costs should also be taken into account when focusing on reducing virtual machine interference, since extra physical servers will be needed to accommodate the virtual machines. Considering that different scenarios have different performance demands (e.g., a test environment versus a production environment), the trade off between performance and cost is an important aspect.

This paper proposes a server consolidation algorithm that minimizes the number of required physical servers, given an accepted, user defined, virtual machine interference threshold. It uses a virtualization interference model and a classification of each virtual machine regarding its resource utilization profile. The algorithm is evaluated using simulation and tests show that it succeeds in consolidating servers while keeping the virtual machine interference under the defined thresholds.

SAC 2016,April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851625

The paper is organized as follows. Section 2 presents related work with proposals to minimize the virtualization interference through server consolidation. Section 3 presents the server consolidation algorithm using adjustable interference thresholds, as well as a virtual machine interference model used by the algorithm. Section 4 presents the evaluation of the algorithm through simulation. Finally, Section 5 concludes the paper.

2. RELATED WORK

In this section we present related work with proposals to minimize the interference in server consolidation. These works have different main objectives like virtual machine interference reduction [20, 3, 10], load balancing among servers [11, 1] or energy saving [16, 9].

On Zhu et al. [20], the authors developed a consolidation algorithm split in 2 different parts: an interference model and the on-line consolidation algorithm. The interference model is further divided in two other stages: off-line model training and the resource interference matrix. The initial placement of the VMs is performed based on an off-line profile, and then further placements are done by the consolidation algorithm based on the characteristics of the VM to be consolidated and the influence matrix obtained through off-line model training. Also, the algorithm periodically monitors the VMs and updates its resource utilization profile. This work's approach focus at guaranteeing that an application (VM) will finish its work before a defined deadline, while trying to do the best possible consolidation without affecting the deadlines established.

Similar to the previous work, TRACON [3] is a framework for task and resource allocation composed of three major components: an interference prediction model, an interference-aware scheduler, and a task and resource monitor that collects data about the applications and feeds it to the interference prediction model. The scheduling algorithm may use three different strategies: the "Minimum interference online scheduler" make quick scheduling decisions, but these may not be optimal. The "Minimum interference batch scheduler" puts tasks in a queue before scheduling them, making better scheduling decisions but delaying the scheduling. The last strategy is a mix of the former two.

In Nathuji et al. [10], the authors propose the concept of "Q-states", where a higher q-state means a higher level of QoS for the VMs. To achieve this, first the VMs are profiled by a "staging server" which identifies the amount of resources needed by the VM to maintain a certain level of QoS, then each VM is scheduled by a bin packing-like scheduling technique by the "cloud scheduler". This scheduling has one variation: the cloud scheduler keeps an amount of resources reserved to be allocated to a VM impacted by resource interference. If a VM is never impacted, a user may choose to pay a greater fee to have part of the reserve allocated to a VM to achieve higher Q-States. The downside of this method is clear: if none of the VMs allocated in a server suffer from resource interference and the users don't "buy" the reserve, the resource reserve is wasted.

The work presented in Ni et al. [11] proposes a policy for mapping virtual machines that focus on reducing the interference among servers. The algorithm checks the current load in each available server and then generates a "roulette of probabilities" that contains the probability of each server receiving the current VM. As the mapping generated is based on the roulette, even with the probability being lower, there is still a chance that the VM will be allocated in a server that is overloaded.

Bobroff et al. [1] proposes an iterative algorithm that periodically verifies the load in each server and then remaps the virtual machines using migration. The initial mapping is done using the first-fit heuristic and then the algorithm enters a cycle composed by three stages: load measurement, load prediction and remapping to balance the load among servers. There are two main disadvantages in this method: firstly, the initial mapping does not consider virtual machine interference which can result in an increased number of migrations already in the first cycle. Also, it is known that migration of virtual machines can impose overhead in the servers and the network of a datacenter [17, 6].

The work presented in Srikantaiah et al. [16] focus on reducing the server's energy consumption while maintaining its performance above an established threshold. The authors analyze the relation between energy consumption and server usage and, based on this analysis, propose an algorithm that maps the VMs focusing on reducing the energy consumption, while maintaining a certain degree of server performance. The solution uses heuristics because, as mentioned by the authors, instead of finding an optimal solution producing a better mapping overall, there are cases where the time needed to generate the mapping may be too high.

Similar to the previous work, in Moreno et al. [9] the authors propose an algorithm that focuses on optimizing the energy consumption by reducing the virtual machine interference caused by resource disputes. Tests show that in cases where high levels of resource interference exist, the energy consumption increases while the performance of the VMs mapped in that physical server decreases. The algorithm initially classifies the workload to be mapped, pre-selects a set of servers that are able to receive the request and then, using a dynamic monitor that keeps track of the detailed state of each pre-selected server, maps the VM to the server with the lowest interference.

Despite these works focus on the virtual machine interference problem, most of them try to minimize it, but do not consider the increase in costs that may be generated by such approaches. As mentioned before, the possibility to adjust the trade off between costs and performance is a desirable characteristic for a consolidation algorithm, which is the main contribution of our proposal.

3. SERVER CONSOLIDATION STRATEGY

The interference between VMs in a virtualization platform depends strongly on the type of workload being executed on each VM. Depending on the computing resource being intensively used by the workload, different levels of interference can occur.

We define virtual machine interference as the penalty in

performance suffered by a VM, caused by the sharing of resources. For instance, considering CPU interference, the metric evaluated is the execution time in which, for instance, a 20% interference indicates that the execution time was increased by 20%. In the case of RAM and Disk I/O, the metric evaluated is throughput in which, for instance, a 20% interference indicates that the throughput is reduced by 20%. In the case of RAM and Disk I/O, 100% or more interference indicates that the VM has stopped responding.

3.1 Virtual machine interference model

In order to devise a simple interference model to be used as a proof of concept with the consolidation algorithm, we executed tests to identify how much interference each type of resource sharing produces. We used specific benchmarks that stress each computational resource separately. The intention of this setup is to simulate virtual machines that make intensive use of one or more computing resources. Initially, a single VM running alone in a server was deployed and its performance was measured. The results of this test were taken as the base case for comparison. After that, the amount of VMs mapped to the same server was increased in order to generate interference and the results compared to the base case. Each virtual machine is configured with one virtual core pinned to an exclusive physical core, 4GB of RAM and 10GB of disk space. Up to 16 virtual machines were mapped to the same server. Table 1 shows the characteristics of the server used in the tests.

Table 1: Server configuration used in the experiments

Server configuration	
CPU	2 Xeon Octa-Core E5-2650, 2GHz, 20MB cache L3
RAM	64GB DDR3 1333 Mhz
Disk	SAS 300GB - 6Gbps
Hypervisor	KVM
OS	Linux Ubuntu 12.04

For each resource, one or more benchmarks were chosen to be used as the VMs workload. The benchmarks are presented in Table 2. The goal is to identify the interference in a worst case scenario, i.e., when the VM is heavily using a specific resource. For the CPU resource, a combination of two benchmarks were used. Figure 1 presents the interference obtained for each resource (dotted line) and a logarithmic trend line using least squares fit regression used to predict the interference caused by sharing resources.

The virtual machine interference model is used to predict the interference generated by a combination of virtual machines hosted in the same physical machine. The model was devised from the equations that represent the logarithmic trends of the interferences obtained for each resource. Each resource has a different interference equation, since the interference levels for each resource are different. The equations obtained in the experiment are presented in Table 3 where X represents the amount of VMs mapped to a physical server and Y represents the level of interference produced by such combination. The equations are used by the consolidation algorithm explained in Section 3.2. It should be noted that different equations may be obtained for other hypervisors and hardware configurations without requiring changes in the server consolidation algorithm.

3.2 Server Consolidation algorithm

The server consolidation algorithm has two goals: maintain the virtual machine interference below a threshold defined by the user (in this case, a cloud infrastructure manager), and minimize the amount of physical servers needed. The interference control is achieved by the interference model presented previously, but the algorithm should be easily adaptable to use other types of more complex modeling as well. For the consolidation part of the algorithm, we use heuristics as a proof of concept, since these methods, despite not finding optimal solutions, usually provide good solutions at lower computational costs [4]. Similarly to the interference model, the algorithm can be adapted to use more advanced consolidation techniques.

We used three well known heuristics to test the consolidation algorithm. The First-fit Decreasing (FFD) heuristic maps the virtual machine to the first server found with enough capacity to accommodate the VM (Algorithm 1). The Best-fit Decreasing (BFD) heuristic evaluates all servers and maps the virtual machine to the server that will leave the least free space after the mapping (Algorithm 2). The Worst-fit Decreasing (WFD) heuristic does the opposite and maps the virtual machine to the server that will leave the most free space after the mapping (Algorithm 3).

Since we use a "decreasing" version of the heuristics, all three algorithms initially sort the virtual machines by size, from the largest to the smallest (procedure *SortDescending()*). After that, each algorithm does the VM mapping based on parameters that represent the size of the VM being mapped (*vm.size*), the available capacity of the physical servers (*srv.AvailableCapacity*), the virtual machine interference that will be produced by the combination of VMs (procedure *VMInterf()*) and the interference threshold defined by the user.

In the case of the BFD and WFD algorithms, there is also a parameter that represents the residual capacity that would be left on the physical server in case a VM was mapped on that server (*residualCapacity*), which is used to evaluate the physical server that will have the least capacity left (BFD algorithm) or the most capacity left (WFD algorithm).

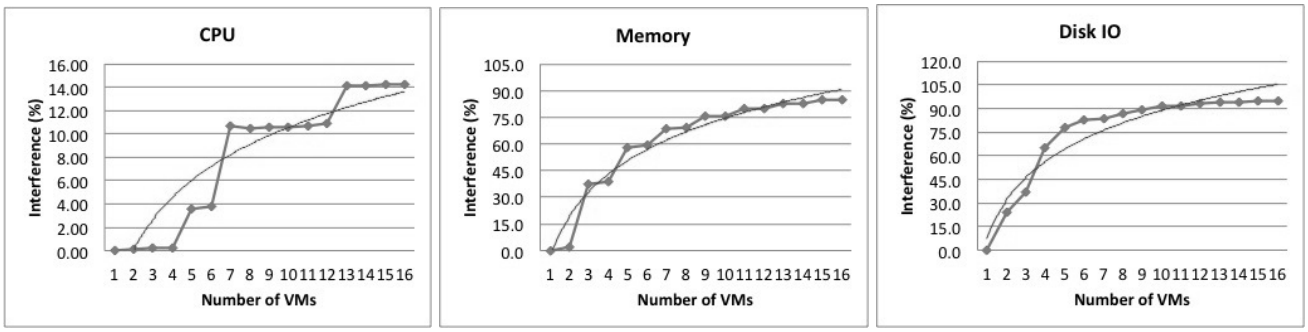
All heuristics were modified in order to integrate the interference model presented in the previous section. Therefore, the algorithm is able to predict the interference that will be created by the combination of virtual machines and makes the mapping according to the threshold defined by the user.

4. EVALUATION

In order to evaluate the proposed server consolidation algorithm, we simulated the mapping of 1000 virtual machines in a typical Data Center infrastructure. The algorithm minimizes the amount of servers required without exceeding a pre-established interference threshold. These results are compared to the mapping generated by a simple server consolidation algorithm using the best-fit decreasing heuristic,

Table 2: Benchmarks used in the experiments

Resource	Benchmark	Description
CPU	CRAFTY [5]	Plays chess matches and focus on logic and integer operations.
	C-RAY [2]	Performs operations with floating-point numbers.
RAM	RAMSPEED [13]	Performs 4 types of memory operations (copy, scale, add, triad) to measure the memory bus throughput.
Disk I/O	TIOBENCH [18]	Multithreaded benchmark that performs 4 types of operations (sequential read, random read, sequential write and random write) to evaluate the data throughput.



(a) CPU (b) RAM (c) Disk I/O

Figure 1: Interference analysis

Table 3: Interference model

Resource	Equation
CPU	$y = 6,5346ln(x) - 4.4983$
RAM	$y = 34,398ln(x) - 4,7183$
Disk	$y = 35,347ln(x) + 7,2785$

Data: virtual machines, servers, threshold

```
1 SortDescending(virtual machines);
2 foreach vm in virtual machines do
3 |   foreach srv in servers do
4 |   |   if srv.availableCapacity >= vm.size AND
  |   |   VMInterf(srv.vms, vm) <= threshold
5 |   |   |   srv.add(vm);
6 |   |   |   break;
7 |   end
8 end
```

Algorithm 1: Server consolidation algorithm using FFD heuristic

which does not consider virtual machine interference. Each virtual machine has a defined size, which establishes the percentage of occupation in the server, and a profile of resource utilization, which defines which resources (CPU, RAM and Disk I/O) the VM uses intensively. Table 4 shows the sizes of the VMs used in the tests and the distribution of each VM size in the set. The distribution of VM sizes was based on a survey [14] that shows the popularity of different instance types on Amazon EC2 [15].

An important parameter used in our experiments is the interference threshold. Lower thresholds result in an increased number of required servers, since fewer VMs will be mapped to each physical server in order to maintain the interference low. The thresholds evaluated vary from 0% up to 90%, in

Data: virtual machines, servers, threshold

```
1 residualCapacity = MaxCapacity;
2 SortDescending(virtual machines);
3 foreach vm in virtual machines do
4 |   foreach srv in servers do
5 |   |   if srv.availableCapacity >= vm.size AND
  |   |   srv.availableCapacity-vm.size < residualCapacity
  |   |   AND VMInterf(srv.vms, vm) <= threshold
6 |   |   |   chosenSrv = srv;
7 |   end
8 |   chosenSrv.add(vm);
9 end
```

Algorithm 2: Server consolidation algorithm using BFD heuristic

10% increments. The assignment of each VM profile, i.e. resources that present intensive utilization, follows an uniform distribution. VM profiles are based on the combination of all metrics (CPU, RAM and Disk I/O) regarding high or low utilization, resulting in 8 different profiles, ranging from no resource with intensive utilization till all resources being used intensively.

Figure 2 shows the amount of physical servers required to allocate the set of 1000 virtual machines for each threshold. It is possible to observe that, comparing to the algorithm that does not take interference into account, the solution proposed uses approximately 360% more servers to guarantee that no performance interference will exist. However, it is possible to guarantee that the interference will stay below 50% with an increase of roughly 25% in the number of servers. In addition, with almost no increase in the amount of servers, it is possible to guarantee that the interference will stay below 80%.

Figure 3 shows the maximum interference for each resource

Data: virtual machines, servers, threshold

```
1 residualCapacity = 0;
2 SortDescending(virtual machines);
3 foreach vm in virtual machines do
4 |   foreach srv in servers do
5 |   |   if srv.availableCapacity >= vm.size AND
  |   |   srv.availableCapacity-vm.size > residualCapacity
  |   |   AND VMInterf(srv.vms, vm) <= threshold
6 |   |   |   chosenSrv = srv;
7 |   end
8 |   chosenSrv.add(vm);
9 end
```

Algorithm 3: Server consolidation algorithm using WFD heuristic

Table 4: Virtual machines sizes, server occupation and distribution

VM Size	Server occupation	Distribution
1	5%	20%
2	10%	27%
3	15%	14%
4	20%	23%
5	25%	10%
6	30%	6%

using different thresholds and heuristics, and also for the simple server consolidation algorithm which does not control the maximum interference (no control). Based on the virtual machine interference model presented in Section 3, minimum interference for RAM and Disk I/O are higher than 10% and 30%, respectively. Therefore, below these thresholds there is no interference for these resources, i.e., the algorithm places each VM with profiles including intensive RAM or Disk I/O utilization in a distinct server, in order to attain the desired threshold.

CPU is the resource with lowest interference. It never exceeds 15%, even with high thresholds. Therefore, VMs with profiles of intensive CPU utilization can be consolidated into a single server, even when low interference thresholds are required. However, the benefits of server consolidation are only clear when a higher interference is used (40%), which is when VMs with high Disk I/O utilization can be consolidated into a single server. Disk I/O is the metric which presents most interference. The RAM metric does not have a deep impact in the consolidation, since it starts being considered for interference threshold of 20%, but without a significant reduction in the number of required servers.

It is important to observe that the mapping produced by the algorithm that does not control interference presents a Disk I/O interference of over 95%, while the proposed solution guarantees that the interference will stay below 80% without increasing the number of servers needed. Comparing the heuristics, WFD presented slightly better results, producing lower maximum interference under all thresholds. This behavior is related to the load balancing characteristic of WFD, which aims at distributing the load on each server, instead of concentrating the VMs on a single server.

5. CONCLUSION

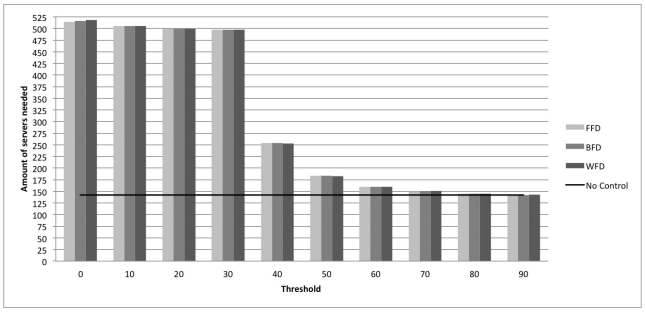


Figure 2: Number of servers needed

Server consolidation is considered one of the main features of virtualization in modern Data Centers. Grouping together several VMs in a reduced number of servers results in substantial gains, specially in lower energy consumption and smaller footprint. However, gathering VMs in servers according to its capacity may not provide satisfactory results if the resources utilization profile of each VM is not taken into account. The behavior of each VM workload may interfere with other VMs in the same server, resulting in performance problems and affecting end users. Several solutions have been proposed to address this problem, but most of them focus on minimizing the interference caused by resource disputes among virtual machines without considering that different scenarios may have different performance demands and costs limitations.

This paper presented a server consolidation algorithm which uses a virtualization interference model to guarantee that virtual machine interference will be maintained under a user defined threshold, providing better control of the trade off between performance and cost. The experiments performed indicate that the proposal attends the constraint of maximum interference level, while accordingly reducing the amount of physical servers needed.

6. REFERENCES

[1] Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. In: Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on. pp. 119–128 (2007)

[2] C-RAY: C-ray (2013), http://openbenchmarking.org/test/pts/c-ray

[3] Chiang, R.C., Huang, H.H.: Tracon: Interference-aware scheduling for data-intensive applications in virtualized environments. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. p. 47. ACM (2011)

[4] Coffman Jr, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: A survey. In: Approximation algorithms for NP-hard problems. pp. 46–93. PWS Publishing Co. (1996)

[5] Crafty: Crafty (2013), http://www.spec.org/cpu2000/CINT2000/186.crafty/docs/186.crafty.html

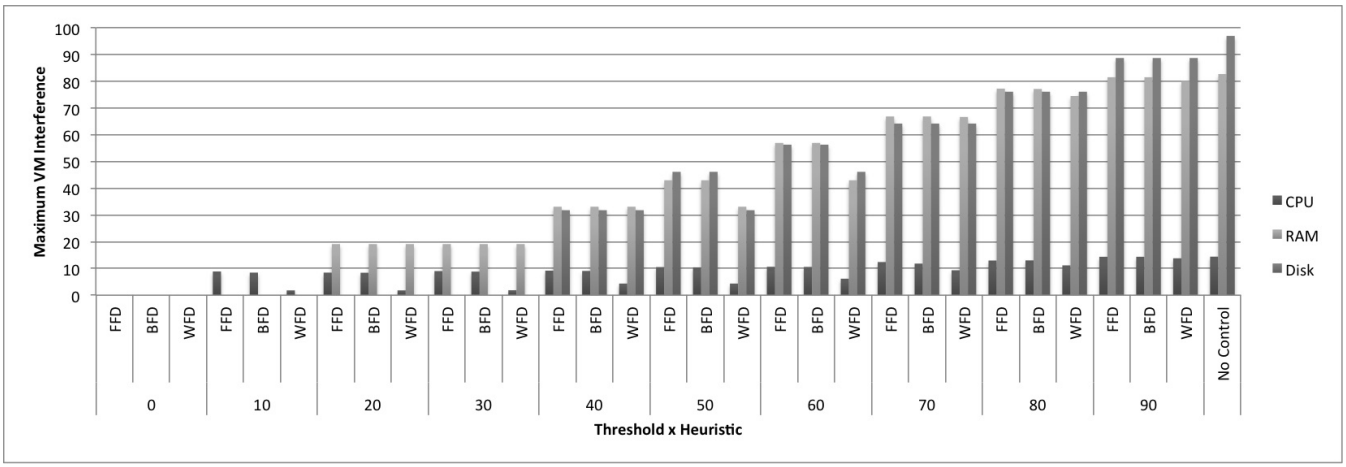


Figure 3: VM Interference for different thesholds and heuristics

[6] Ferreto, T.C., Netto, M.A., Calheiros, R.N., De Rose, C.A.: Server consolidation with migration control for virtualized data centers. Future Generation Computer Systems 27(8), 1027–1034 (2011)

[7] Marzolla, M., Babaoglu, O., Panzieri, F.: Server consolidation in clouds through gossiping. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a. pp. 1–6 (June 2011)

[8] Matthews, J.N., Hu, W., Hapuarachchi, M., Deshane, T., Dimatos, D., Hamilton, G., McCabe, M., Owens, J.: Quantifying the performance isolation properties of virtualization systems. In: Proceedings of the 2007 workshop on Experimental computer science. ExpCS '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1281700.1281706

[9] Moreno, I.S., Yang, R., Xu, J., Wo, T.: Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement. In: Autonomous Decentralized Systems (ISADS), 2013 IEEE Eleventh International Symposium on. pp. 1–8 (2013)

[10] Nathuji, R., Kansal, A., Ghaffarkhah, A.: Q-clouds: managing performance interference effects for qos-aware clouds. In: Proceedings of the 5th European conference on Computer systems. pp. 237–250. ACM (2010)

[11] Ni, J., Huang, Y., Luan, Z., Zhang, J., Qian, D.: Virtual machine mapping policy based on load balancing in private cloud environment. In: Cloud and Service Computing (CSC), 2011 International Conference on. pp. 292–295 (2011)

[12] Padala, P., Zhu, X., Wang, Z., Singhal, S., Shin, K.G.: Performance evaluation of virtualization technologies for server consolidation. Tech. rep. (2007)

[13] RamSpeed: Ramspeed (2013), http://openbenchmarking.org/test/pts/ramspeed

[14] Reports, D.C.: Aws ec2 instance type survey - what is popular? (2013), http://www.newvem.com/aws-ec2-instance-type-survey-what-is-popular/

[15] Services, A.W.: Amazon elastic compute cloud (2013), http://aws.amazon.com/pt/ec2/

[16] Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 conference on Power aware computing and systems. vol. 10. USENIX Association (2008)

[17] Stage, A., Setzer, T.: Network-aware migration control and scheduling of differentiated virtual machine workloads. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. pp. 9–14. IEEE Computer Society (2009)

[18] TIOBench: Tiobench (2013), http://openbenchmarking.org/test/pts/tiobench

[19] Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T., De Rose, C.A.: Performance evaluation of container-based virtualization for high performance computing environments. In: Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on. pp. 233–240. IEEE (2013)

[20] Zhu, Q., Tung, T.: A performance interference model for managing consolidated workloads in qos-aware clouds. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. pp. 170–179. IEEE (2012)