EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications

Jiang Dejun[1,2], Guillaume Pierre[1], and Chi-Hung Chi[2]

[1] VU University Amsterdam
[2] Tsinghua University Beijing

Abstract. Cloud computing is receiving increasingly attention as it provides infinite resource capacity and "pay-as-you-go" resource usage pattern to hosted applications. To maintain its SLA targets, resource provisioning of service-oriented applications in the cloud requires reliable performance from the cloud resources. In this paper, we study performance behavior of small instances in Amazon EC2. We demonstrate that the performance of virtual instances is relatively stable over time with fluctuations of mean response time within at most 8% of the long-term average. Moreover, we also show that different supposedly identical instances often have very different performance, up to a ratio 4 from each other. We consider this as an important issue that must be addressed, but also as an opportunity as it allows one to assign each instance with a task that matches its own performance profile.

1 Introduction

Cloud computing is emerging today as a new paradigm for on-demand resource provisioning for service-oriented applications. Clouds are attractive to service-oriented application hosting as such applications often observe large fluctuations in their workload. Clouds allow one to add resources very quickly to a hosted application when the performance is about to violate certain criteria such as Service Level Agreements (SLA) or system load. Similarly, clouds offer the opportunity to release resources when the load decreases. Service Level Agreements typically define requirements regarding the performance, security, availability and the like from a user perspective. We focus here on the performance aspects.

Cloud platforms typically do not give access to actual physical machines but rely heavily on virtualization techniques for reasons of cost effectiveness and technical flexibility. Virtual machine monitors such as Xen allow fine-grained performance isolation between multiple virtual instances sharing the same physical resource [1]. The usual wisdom is that CPU performance can be isolated very effectively, while I/O performance is harder to isolate [2,3].

This paper addresses the following question: *how effective can the virtual machine based cloud be for SLA-aware resource provisioning of service-oriented applications?* Resource provisioning mechanisms traditionally rely on two fundamental performance properties of the available resource units [4]:

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 197–207, 2010.

- Performance stability: the performance of the provisioned resource units should remain constant over time. In the virtual machine based clouds, the performance of the same virtual machine should be stable without being affected by the activity of other virtual machines on the same hardware.
- Performance homogeneity: the performance of different resource units should be predictable through the profiling of current deployed resource units. It requires that the performance behavior of resource units are homogeneous. In real-world applications, cloud providers commonly provide users with a set of different virtual machine types, each of which has different resource capacities in terms of CPU capacity, RAM size, disk I/O bandwidth and the like. The performance of different types of virtual machines are obviously heterogeneous. However, the performance of multiple virtual machines of the same type should be similar. Otherwise, it becomes very hard for one to quantify the number of virtual machines to be provisioned such that the performance of hosted applications meet its SLA targets.

Although these two properties are typically true for cluster-based systems where identical physical resources are exclusively dedicated to a single application, the introduction of virtualization in the cloud requires a re-examination of these properties. This paper studies the above two performance properties of virtual instances provided by Amazon EC2 [5]. We then evaluate the possibility of providing SLA-aware resource provisioning to service-oriented applications in the cloud.

As the workloads of commonly-used applications (such as three-tier web applications) can be either CPU-intensive at the application server tier or I/O-intensive at the backend database server tier, we benchmark virtual instances on EC2 using three synthetic web applications that exhibit different types of workload pattern. Based on the measured performance data, we find performance of virtual instances to be relatively stable. The mean response time of a single small instance fluctuates by at most 8% around its long-term average. On the other hand, multiple instances of the same type show very heterogeneous performance profiles, up to a ratio 4 in response time from each other. We also observed that the CPU and I/O performances of different "identical" virtual instances are not correlated. We consider this as a very important issue that must be addressed to allow effective resource provisioning in the cloud. At the same time, we see this as an opportunity to exploit these differences and assign each virtual instance with a task that matches its own performance profile.

The remain of this paper is organized as follows: Section 2 introduces research efforts related to our work. Section 3 presents our methodology to benchmark virtual instances on Amazon EC2. Section 4 discusses performance analysis results. Finally, Section 5 concludes our work and proposes future research directions for resource provisioning in virtualized cloud environments.

2 Related Work

A number of recent research works made efforts towards resource provisioning of virtual machines to multi-tier web applications. For instance, [6] assumes the

existence of a performance model which determines the number of virtual machines at each tier of the application. It then formalizes the mapping problem of assigning a set of virtual machines to a pool of physical hosts with the goal of maximizing resource usage. Similarly, [7] assumes that the most-demanding resource at each tier of the application has a well-defined capacity in terms of CPU, I/O and network. The paper then presents a general performance model for multi-tier web applications. The model determines the capacity of virtual machines provisioned to each tier with the goal of optimizing certain utility functions, such as revenues of provisioned resource. Although these works propose resource provisioning solutions that rely on virtual machines, they do not investigate the actual performance behavior of virtual machines. On the other hand, this paper focuses specifically on the performance of virtual machines in a real-world environment and aims at proposing practicable solutions for virtual machine provisioning.

To our best knowledge, few works focus on performance analysis of the virtual machines provided by clouds. For example, [8] analyzes the performance of Amazon EC2 using micro-benchmarks, kernels, and e-Science workloads. This analysis targets the evaluation of usefulness of EC2 as a scientific computing platform. In this paper we also take Amazon EC2 as our experimental cloud platform, however we focus on the challenges and opportunities of providing SLA-aware resource provisioning to service-oriented applications in the cloud.

Finally a number of works analyze the performance impact caused by the inherent virtualization mechanisms used in commercial clouds. [9] analyzes the impact of different choices of CPU schedulers and the parameters on application performance in Xen-based virtualized environments. [3] presents a system-wide statistical profiling toolkit called Xenoprof, which is implemented for the Xen virtual machine environment. Xenoprof is then used to quantify Xen's performance overheads for network I/O processing. Similarly, [10] characterizes network I/O performance in a Xen virtualized environment. [2] uses Xen Virtual Machine Monitors to measure CPU overheads when processing a set of disk I/O and network I/O intensive workloads. [11] compares the scalability of four virtual machine technologies in terms of CPU, memory, disk and network. These works help us to understand the performance behaviors of virtual machines. However, our work differs from these as we profile virtual machines at a macroscopic level with the premise of taking virtual machines as black boxes, and we finally aim to provision resources based on them.

3 Methodology

In this section we introduce our experimental environment on Amazon EC2 and present the detail of our experimental setup.

Amazon EC2 provides 5 types of virtual instances, each of which has different capacities in terms of CPU capacity, RAM size and I/O bandwidth. Table 1 shows the announced capacity details of virtual instances on EC2. To provide fault-tolerance, EC2 provides its virtual instances across multiple data centers

Table 1. Capacity detail of virtual instances on EC2

Instance type	Compute units	RAM	I/O performance
Small	1	1.7GB	Moderate
Medium - high CPU	5	1.7GB	Moderate
Large	4	7.5GB	High
Extra large	8	15GB	High
Extra large - high CPU	20	7GB	High

Table 2. Software environment in all experiments

Application server	Database server	JDK	Operating system	Kernel
Tomcat 6.0.20	MySQL 5.1.23	JDK 6 update 14	Ubuntu 8.10	Linux 2.6.21

organized in so-called availability zones[1]. Two virtual instances running in different availability zones are guaranteed to be executed in different data centers. Of the six availability zones, four are located in the U.S. and the other two are in Europe.

In this paper, we examine the performance of small instances on EC2 as they are the most widely used. To demonstrate that the same performance features appear on different types of virtual instances as well, we also partially benchmark medium instances with high CPU[2].

In practice, service-oriented applications are commonly deployed in different data centers for fault tolerance and to deliver good quality of service to users in different locations. To match this common case we examine the performance of small instances in all six availability zones. This also allows us to make sure that the experimental instances do not interfere with each other.

In order to provision virtual machines for service-oriented applications, it is important for one to predict its future performance if given one more or one less resource. This performance predictability in turn requires that performance of the same virtual machine remains constant over time. In addition, it requires that the performance of newly allocated virtual instances is similar to that of currently-deployed instances. We therefore carry out three groups of experiments to benchmark small instances on EC2.

Performance stability: The first group of experiments studies the performance stability of small instances under the constant workload intensity. As workloads of service-oriented applications can be CPU-intensive and database I/O intensive, we develop the following three synthetic web applications to simulate different types of workload patterns:

[1] There are four zones located in the United States (US-EAST-1A, US-EAST-1B, US-EAST-1C and US-EAST-1D) and two zones in Europe (EU-WEST-1A and EU-WEST-1B).

[2] We leave the experiments on other types of instances for future work.

- T1: a CPU-intensive web application. This application consists of a servlet processing XML transformation based on client inputs. It issues no disk I/O (except for reading configuration file when starting up) and very little network I/O (each request returns one html page of size around 1,600 bytes). The request inter-arrival times are derived from a Poisson distribution. The average workload intensity is 4 requests per second.
- T2: a database read-intensive web application. This application consists of a servlet and a database hosted on two separate virtual instances. The database has 2 tables: "CUSTOMER" and "ITEM." The "CUSTOMER" table holds 14,400,000 records while the "ITEM" table holds 50,000,000 records. The size of data set is 6.5 GB, which is nearly 4 times the RAM size of small instances. The servlet merely issues SQL queries to the backend database. It first gets customer order history based on customer identification, and then fetches items related to those ones in customer's historical orders. Here as well, the request inter-arrival times are derived from a Poisson distribution. The average workload intensity is 2 requests per second.
- T3: a database write-intensive web application. This application consists of a servlet and a database hosted on two separate virtual instances. The servlet issues UDI (Update, Delete and Insert) queries to execute write operations on 2 tables: "CUSTOMER" and "ITEM." The servlet first inserts 1,440,000 records into "CUSTOMER" table and then inserts 1,000,000 records into "ITEM" table. After populating the two tables, the servlet sends queries to sequentially update each record. Finally, the servlet sends queries to delete the two tables.

In this group of experiments, we randomly select one small instance in each availability zone and run T1, T2 and T3 on each instance separately. Each run of the tested application lasts for 24 hours in order to examine the potential interference of other virtual instances on the tested application performance. We compare the statistical values of mean response time of each hour within the whole experiment period to evaluate the performance stability of small instances.

Performance homogeneity: The second group of experiments evaluates the performance homogeneity of different small instances. We randomly select one small instance in each availability zone and run T1 and T2 on each instance separately. Each run of the tested application lasts 6 hours such that database caches can fully warm up and the observed performance becomes stable. We repeat this process 5 times at a few hours interval such that we acquire different instances on the same availability zone[3]. We compare the mean response times of tested application among all tested instances in order to evaluate the performance homogeneity of different instances.

CPU and I/O performance correlation: The third group of experiments studies the correlation between the CPU and I/O performance of small instances.

[3] If one requested a virtual instance very quickly after another one is released, Amazon EC2 might recycle the virtual instances and return the previous one.

Table 3. Response time of T1 on small instances

	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D	EU-WEST 1A	EU-WEST 1B
Mean value (mean)	684.8ms	575.5ms	178ms	185.2ms	522.9ms	509.9ms
Standard deviation (std)	46.7ms	31.3ms	4.99ms	3.5ms	20.6ms	18.9ms
std/mean	6.8%	5.4%	2.8%	1.9%	3.9%	3.7%

Table 4. Response time of T2 on small instances

	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D	EU-WEST 1A	EU-WEST 1B
Mean value (mean)	75.9ms	76.3ms	79.9ms	71.8ms	83.8ms	71.6ms
Standard deviation (std)	1.28ms	2.05ms	6.36ms	1.14ms	2.1ms	2.34ms
std/mean	1.7%	2.7%	8.0%	1.6%	2.5%	3.3%

The experiment process is similar to the second group. Instead of running T1 and T2 separately, we run them sequentially on the same instance, each one for 6 hours. We finally correlate the CPU performance and I/O performance in all tested instances to observe their relationships.

In all above experiments the clients run on a separate virtual instance in the same availability zone as the virtual instances hosting application servers and database servers. Table 2 shows the software environment used in all experiments.

4 Evaluation

This section first presents the results of CPU and disk I/O performance stability on small instances in each availability zone. We then show the performance behavior of different small and medium instances. Finally, we discuss the implications of the (lack of) correlation between CPU performance and disk I/O performance of small instances for resource provisioning. We measure the response time at the server side in all experiments in order to avoid the latency error caused by the network between servers.

4.1 Performance Stability Evaluation

To study the performance stability of small instances, we measure the response time of each request and calculate the mean response time at a one hour granularity. We then compute the standard deviation of the 24 mean response times (each for 1 hour in the whole 24-hours experiment period). Table 3 shows the mean value and the standard deviation of the 24 mean response times for T1 in all zones.

From the perspective of long-running time periods the CPU performance is quite stable. As shown in Table 3, the standard deviation of mean response times of each hour is between 2% and 7% of the mean value, which may be acceptable in real-world hosting environments.

Table 5. Response time for INSERT operation of T3 on small instances

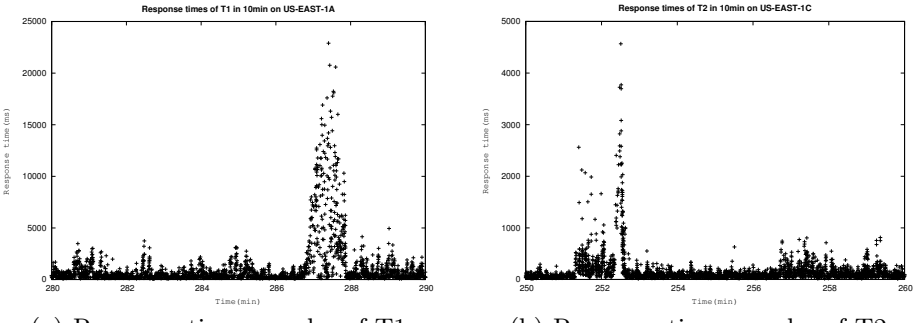
	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D	EU-WEST 1A	EU-WEST 1B
Mean value (mean)	0.33ms	0.33ms	0.33ms	0.53ms	0.47ms	0.46ms
Standard deviation (std)	0.0012ms	0.0006ms	0.0022ms	0.0021ms	0.0019ms	0.0044ms
std/mean	0.4%	0.2%	0.7%	0.4%	0.4%	0.9%

Table 6. Response time for UPDATE operation of T3 on small instances

	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D	EU-WEST 1A	EU-WEST 1B
Mean value (mean)	3.84ms	3.5ms	2.67ms	3.09ms	3.72ms	3.91ms
Standard deviation (std)	0.026ms	0.035ms	0.061ms	0.03ms	0.005ms	0.026ms
std/mean	0.7%	1%	2.3%	1.0%	0.1%	0.6%

Table 7. Response time for DELETE operation of T3 on small instances

	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D	EU-WEST 1A	EU-WEST 1B
Mean value (mean)	30.1ms	21.7ms	30.3ms	17.8ms	15.9ms	21.7ms
Standard deviation (std)	21.4ms	0.52ms	20.2ms	4.6ms	0.33ms	0.71ms
std/mean	71.1%	2.4%	66.7%	25.8%	2.1%	3.3%



(a) Response time samples of T1 (b) Response time samples of T2

Fig. 1. Response time samples of T1 and T2 over a period of 10min

However, we also observed that the CPU performance could be temporarily affected by the underlying resource sharing mechanism. Figure 1(a) shows the response time of T1 over a period of 10 minutes. We observe short periods during which the response time significantly increase. The duration of such peaks is relatively short (on average 1 to 2 minutes). We attribute these peaks to external factors such as the creation of a new virtual instance in the same physical machine. (the duration of such peaks is similar to the observed delay for creating a new virtual instance). Apart from these short peaks, the CPU-capacity sharing

has little impact on the performance stability of CPU-intensive web applications when considering performance behavior in long-running periods.

We also observed that the performance of multiple small instances vary wildly from each other, from 185ms to 684ms of average response time. We further compare the performance of different small instances in section 4.2.

Similarly, we measure the response time of application T2 and compute the mean response times of each hour. Table 4 shows the mean value and the standard deviation of mean response times of each hour within the 24-hours experiment period. The database read performance of small instances is also very stable from the perspective of long-running time periods. For all tested instances, the mean response time of each hour deviates between 1.6% and 8% from the mean value. Similarly to CPU performance, the database read performance is affected by short interferences with the underlying I/O virtualization mechanism. Figure 1(b) shows the peak of response time of T2 over 10 minutes. The disturbances are also short, in the order of 2 minutes.

The database read performance of different small instances (such as in different zones) are also different from each other. We further examine database read performance homogeneity in section 4.2.

We finally evaluate the performance stability of database write-intensive workloads. Tables 5 to 7 show response time statistics for database UDI operations. As shown in Tables 5 and 6, the performance of database INSERT and UPDATE operations is very stable. We observed that the standard deviation of those mean response times is small, between 0.2% and 2.3%.

However, Table 7 shows that the performance of database DELETE operation varies a lot. The cause of this performance behavior of database DELETE operations remains to be found. Similarly to CPU and database read operations, the performance of database UDI operations on different small instances are different from each other.

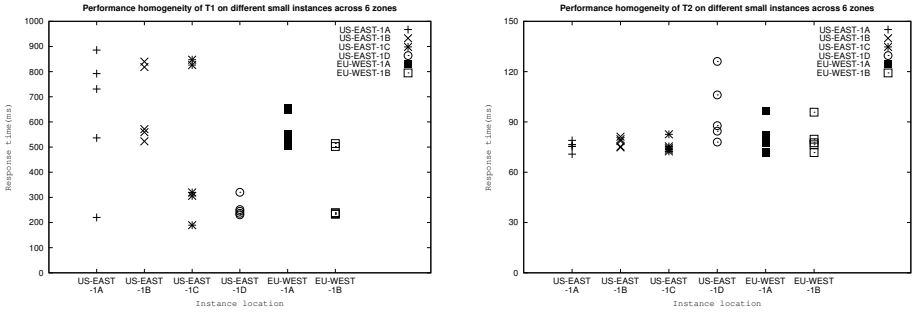
To demonstrate that other types of virtual instances on EC2 exhibit similar performance features, we partially benchmark medium instance with high CPU on EC2. We run T1 to profile CPU performance of medium instances after adjusting the request rate to match the capacity of medium instances. Table 8 shows the statistical values of mean response times of each hour on medium instances across the four US availability zones. Similar to the performance behavior of small instances, CPU performance is also relatively stable. The standard deviation of mean response time of each hour is between 2% and 10%. The CPU performance of different medium instances also varies a lot. One would however need more samples to fully explore the performance behavior of medium and large instances.

4.2 Performance Homogeneity Evaluation

So far we observed that the CPU performance and disk I/O performance of the same small instance are relatively stable from the perspective of long-running time periods (except for database DELETE operation). However, typical resource provisioning algorithms also expect that different small instances have

Table 8. Mean response time of T1 on medium instances (high CPU)

	US-EAST 1A	US-EAST 1B	US-EAST 1C	US-EAST 1D
Mean value (mean)	307.7ms	791.5ms	197.3ms	199.8ms
Standard deviation (std)	27.4ms	26.1ms	3.6ms	5.8ms
std/mean	9.6%	3.3%	1.8%	2.9%



(a) CPU performance homogeneity (b) Disk I/O performance homogeneity

Fig. 2. Performance homogeneity of different small instances across all zones

homogeneous performance behavior such that the performance of future small instances is predictable based on current performance profiles. Thus, we evaluate the performance homogeneity of different small instances through the second group of experiment.

Figure 2(a) shows the mean response times of T1 for 30 different small instances across all zones (5 instances for each zone). Different instances clearly exhibit very different CPU performance when serving the exact same workload. Response times of different virtual instances vary up to a ratio 4. The same pattern appears both inside each zone and between different zones. Figure 2(b) shows similar heterogeneous performance behavior of different small instances for disk I/O operations, even though the variations are less important than for CPU. Thus, when provisioning small instances to host service-oriented applications, it will be very hard for one to predict the performance of the newly-allocated virtual instances based on the observed performance profiles of currently deployed ones. This property challenges traditional resource provisioning approaches which assume that the underlying infrastructure provides homogeneous resources.

4.3 Implications for Resource Provisioning

As we observed that different small instances behave differently when serving CPU-intensive and disk I/O intensive workloads, we further explore this phenomenon and run the third group of experiment to check if the CPU and disk I/O performances are correlated on supposedly identical small instances.

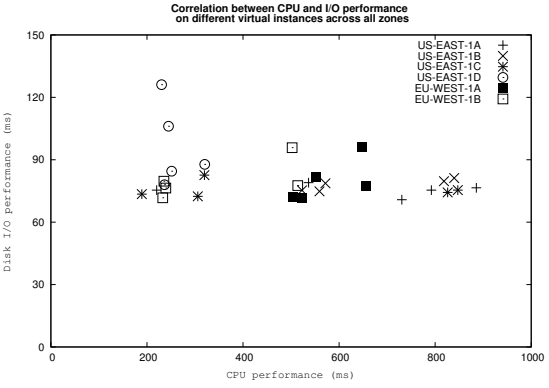


Fig. 3. Correlation between CPU and I/O performance of small instances

Figure 3 shows 30 samples of the correlation of CPU performance and disk I/O performance on identical small instances across all availability zones. Each point depicts the CPU and I/O performances of a single virtual instance. We do not observe any obvious correlation between the respective CPU and I/O performances of single instances. On the other hand, small instances can clearly be classified into three or four clusters with similar performance. Often (but not always) instances from the same availability zone are clustered together.

These results suggest that different small instances on Amazon EC2 may be suitable to process different types of workload. Within commonly-used multi-tier service-oriented applications, different tiers have different workload patterns. For example, an application server tier is commonly CPU-intensive while a database server tier is rather I/O intensive. Thus, one may consider employing well-suited small instances to provision resources to hosted applications such that each instance runs a task that matches its own performance profile. A virtual instance with fast CPU could be given an application server to run, while an instance with fast I/O would run a database server and a virtual instance with slow CPU and I/O may carry a modest task such as load balancing. We name this performance feature of small instances on EC2 as workload affinity. Although adapting resource provisioning algorithm will be a challenge, we believe that exploiting such workload affinity properties could result in improvement of the overall resource usage, even compared with a fully homogeneous case.

5 Conclusion

Cloud platforms such as Amazon EC2 are attracting attention in the service-oriented application hosting community as they provide near-infinite capacity and a "pay-as-you-go" business model. Good usage of cloud facilities may help application providers to reduce their IT investments as well as operational costs.

However, fluctuating workloads and the necessity to maintain SLAs require clouds to provide SLA-aware resource provisioning to hosted applications.

To dynamically provision virtual machines to hosted applications with guaranteed performance, it is necessary to understand the performance behavior of virtual instances provided by clouds. In this paper, we took the popular commercial cloud Amazon EC2 as an example, and evaluated the performance stability and homogeneity of small instances on EC2. We demonstrated that the CPU and disk I/O performance of small instances are relatively stable from the perspective of a long-running periods. However, the performance behavior of multiple "identical" small instances is very heterogeneous. We claim that this property challenges the effectiveness of current resource provisioning approaches if employed in the virtual machine based cloud as these approaches assume homogeneous sets of underlying resources. We consider this as an important issue that must be addressed by the resource provisioning community. At the same time, we believe that this variety of workload affinity provides opportunities for future algorithms to make more effective use of the resources offered by the cloud.

References

1. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proc. SOSP (2003)
2. Cherkasova, L., Gardner, R.: Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In: Proc. USENIX Annual Technical Conf. (2005)
3. Menon, A., Santos, J.R., Turner, Y., Janakiraman, G.J., Zwaenepoel, W.: Diagnosing performance overheads in the Xen virtual machine environment. In: Proc. Intl. Conf. on Virtual execution environments (2005)
4. Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P.: Dynamic provisioning of multi-tier internet applications. In: Proc. Intl. Conf. on Autonomic Computing (2005)
5. Amazon.com: Amazon Elastic Compute Cloud, `http://aws.amazon.com/ec2/`
6. Campegiani, P., Presti, F.L.: A general model for virtual machines resources allocation in multi-tier distributed systems. In: Proc. Intl. Conf. on Autonomic and Autonomous Systems, pp. 162–167 (2009)
7. Wang, X., Du, Z., Chen, Y., Li, S.: Virtualization-based autonomic resource management for multi-tier web applications in shared data center. Jounal of Systems and Software 81(9), 1591–1608 (2008)
8. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: An early performance analysis of cloud computing services for scientific computing. Technical Report PDS-2008-006, Delft University of Technology (December 2008)
9. Cherkasova, L., Gupta, D., Vahdat, A.: When virtual is harder than real: Resource allocation challenges in virtual machine based it environments. Technical Report HPL-2007-25, HP Laboratories Palo Alto (February 2007)
10. Apparao, P., Makineni, S., Newell, D.: Characterization of network processing overheads in Xen. In: Proc. Intl. Workshop on Virtualization Technology in Distributed Computing (2006)
11. Quetier, B., Neri, V., Cappello, F.: Scalability comparison of four host virtualization tools. Journal of Grid Computing 5(1), 83–98 (2007)