Performance Evaluation in a Cloud with the Provisioning of Different Resources Configurations

Bruno G. Batista, Júlio C. Estrella, Marcos J. Santana, Regina H. C. Santana
Institute of Mathematics and Computer Science
University of São Paulo - USP
São Carlos, Brazil
batista, jcezar, mjs, rcs@icmc.usp.br

Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester, United Kingdom
srm13@le.ac.uk

***Abstract*—Cloud computing is a computing style where resource providers can offer on-demand services in a transparent way and clients usually pay as they go. It introduces a new level of flexibility and scalability for IT users addressing challenges such as the rapid change in IT and the need to reduce cost and time of infrastructure management. However, to be able to offer QoS guarantees without limiting the number of accepted requests, providers must be able to dynamically adjust the available resources to serve requests. This dynamic resource management is not a trivial task, bringing its own challenges related to workload and performance modelling, and deployment and monitoring of applications on virtualised IT resources. An efficient mapping between resources and applications ensures workload balancing and good resource utilization and allows to meet the QoS levels required by clients. This paper presents a performance evaluation that considers different resource configurations in a cloud environment to define which dimension of resource scaling has real impact on client applications.**

***Keywords*-Cloud Computing; Resources Provisioning; Quality of Service; Performance Evaluation.**

I. Introduction

In recent years, one of the most discussed topics in Information Technology has been cloud computing. According to NIST (National Institute of Standards and Technology), "Cloud computing is a model that allows ubiquity, convenience and on-demand access to a set of configurable shared resources and can be quickly delivered with minimal management effort on the clients part" [10].

Cloud computing allows to dynamically adapt the capacity of a company to meet clients' service requests without investing in infrastructure such as hardware and software licenses as well as maintenance staff training. However, clients expect that the computational system that composes a cloud operates properly, with no interruption in its service or loss of data/messages. Therefore, service providers should ensure Quality of Service aspects (QoS) to gain the confidence and satisfaction of their clients.

The term quality of service refers to the effect caused by the service performance characteristics, determining the degree of client satisfaction, i.e. the set of system characteristics necessary to achieve certain functionality [4]. It can also be described as a set of parameters that characterize the quality of a particular data flow, such as bandwidth or priority assigned to a specific client. These parameters are attributes of a system that can be measured quantitatively by metrics and used in the definition of QoS levels [3]. However, ensuring QoS in a cloud environment is not a trivial task, because there are different types of clients with varied service requirements [15] in addition to the usual issues arising with distributed systems.

Considering the great scalability in a cloud environment and the fact that service demand can change instantly, automatic resource allocation to meet this demand becomes an interesting topic both in academia and in industry. Correct provisioning allows for better use of available computational resources and, consequently, of all infrastructure that composes the cloud, because the mapping between the workload and the resource becomes more efficient. Furthermore, it helps in fulfilling the QoS levels demanded by the clients and provides a greater dynamism to the system.

Providing more computational resources to a client, for example, is highly workable and easily achieved in a cloud environment. However, it has great impact on the final cost that is paid by the client and requires efficient mechanisms by providers.

It is desirable that resource allocation in a cloud environment can be performed automatically and dynamically, based on client high-level needs and on a fair price. For this to occur, firstly it is important to analyse and understand the computational resources and the quantity of these resources that should be allocated to a client in a cloud. To address this, this paper presents results considering different configurations of virtual machines (VMs) running different workloads. With the results, the influence of different configurations of number of VMs and VCPUs, disk size, network type and memory (RAM) quantity on the system performance can be quantified. Furthermore, it is possible to analyze if an environment composed by i VM with j virtual cores (VCPUs) is more efficient than an environment composed by j VMs and i VCPU, for instance.

The remainder of the paper is organised as follows:

section II presents the background for physical and virtual resources provisioning; section III shows the experimental design and the performance evaluation; section IV describes some related works; and section V summarizes the main results, contributions and future work.

II. Resource Provisioning

Cloud computing is a model for on-demand service provisioning based on already established techniques such as virtualization, distributed, utility and autonomic computing. It aims to reduce the overhead of ownership of information technology to the final client and allows great flexibility and scalability as well as cost reduction in the acquisition, management and maintenance of all infrastructure owned by a company [12] [13]. However, in order for clients and providers to enjoy these benefits, an efficient resource provisioning mechanism is required.

In a cloud environment, assuming efficient resource provisioning, applications can operate more efficiently, with reduced financial and environmental costs, reduced underutilization of resources, and better performance at times of peak load. However, the process of resource provisioning in clouds is a complex undertaking. It requires the application provider to compute the best software and hardware configuration to ensure that QoS targets of application services are achieved, while maximizing the overall system efficiency and utilization. Moreover, there are significant problems that exist with regard to efficient provisioning and delivery of applications using cloud-based IT resources, such as workload modeling, virtualization, performance modeling, and deployment and monitoring of applications on virtualized IT resources [1].

The way in which the hypervisor associates the physical and virtual resources is another important factor that should be considered, because it is the base for efficient resource provisioning. However, this association varies according with the hypervisor used.

The Xen hypervisor, for instance, uses three scheduling algorithms: BVT (Borrowed Virtual Time), SEDF (Simple Earliest Deadline First) and Credit Scheduler [2]. The Credit Scheduler is Xen's default scheduling algorithm. BVT and SEDF algorithms are still available, but it is indicated that they will be removed from Xen in the next versions.

In the Credit Scheduler algorithm, each physical core (CPU) is responsible for a queue composed of virtual cores (VCPUs). In this way, there is an association between physical and virtual cores, where the number of VCPUs exceeds the available CPUs.

For the presented work we used the Xen hypervisor because it is one of the most important available hypervisors in the literature. Experiments were performed using the Credit Scheduler algorithm. The experiment design and results analysis are detailed in the next sections.

Table I: Environment specification

Machine	Physical	Virtual
Processor	Core 2 Quad 2.4GHz	Core 2 Quad 2.4GHz
Cores	4	Varies
Memory RAM	8GB	Varies
Disk	160GB	Varies
Network	-	Varies
Operational System	Ubuntu Server 11.10	Ubuntu Server 10.04
Hypervisor	Xen 4.1	-

Table II: Factors and levels

Factors	**Levels**
Disk size	8GB and 16GB
Network type	Megabit and Gigabit
Memory RAM quantity	512MB and 1024MB
VMs number	1, 2 and 4
VCPUs number	1, 2, 4 and 8

III. Experiments Design

The experiments shown in this section aim to analyze which computational resources should be provided to improve the performance in a cloud and furthermore the proportion in which they should be increased. For this the *Apache* benchmark[1] was used. This benchmark uses the quantity of served requests per second as response variable. All experiments were run 10 times[2] and the results show the average of served requests with 95% of confidence.

In the experiments we used a physical machine based on an *Intel Core 2 Quad* processor to host the virtual machines that execute the workload. The environment configurations are shown in Table I. In the results presented in this section, five factors with different quantities of levels were considered (Table II). These factors and levels were used to compose the different scenarios. Each level was increased by 100% compared with the inferior level.

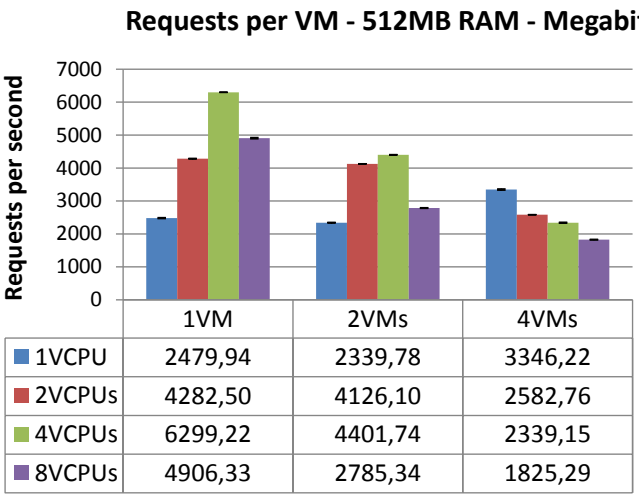
A. Results Analysis

Figure 1 is composed of four graphs that show the average quantity of served requests per second per one VM during the experiments execution time. The graphs show different combinations of levels that compose the factors defined in the experiment design. However, even with different configurations, the experiments' behaviours were practically the same.

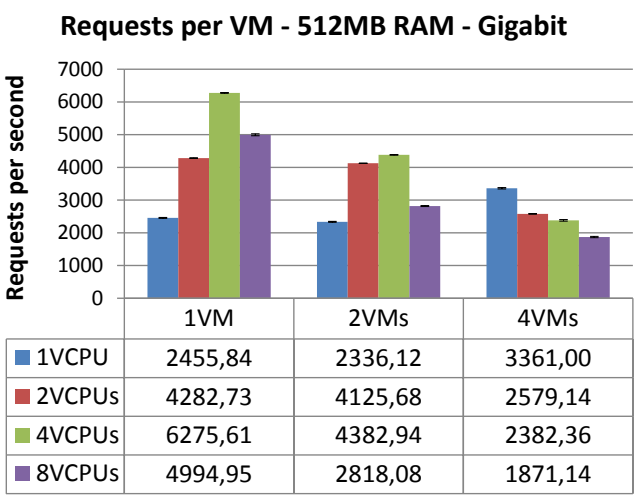
According with the graphs, as new VMs were added to the system, the competition for computational resources becomes greater, consequently reducing the average quantity

[1] http://openbenchmarking.org/test/pts/apache

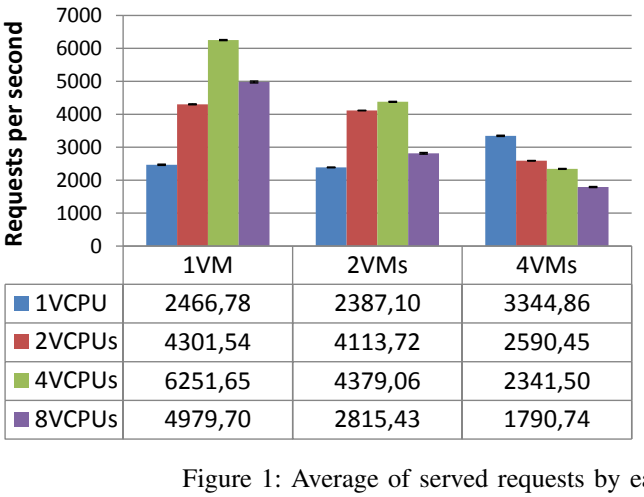
[2] Through 10 repetitions it was noted that the results did not show large variations. Therefore, it was concluded that 10 was a reasonable amount for the number of repetitions.



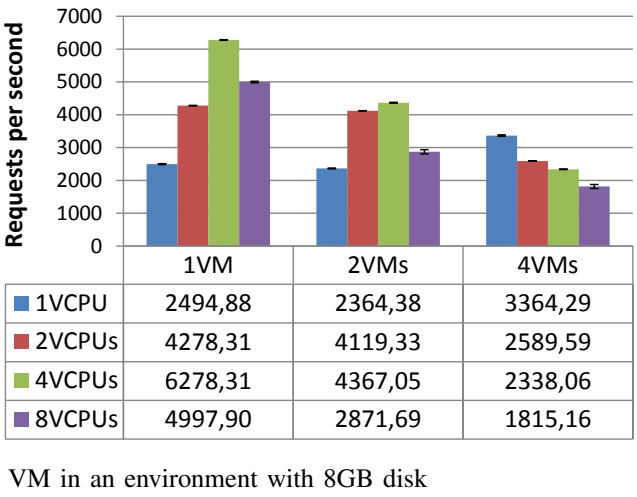
	1VM	2VMs	4VMs
1VCPU	2479,94	2339,78	3346,22
2VCPUs	4282,50	4126,10	2582,76
4VCPUs	6299,22	4401,74	2339,15
8VCPUs	4906,33	2785,34	1825,29



	1VM	2VMs	4VMs
1VCPU	2455,84	2336,12	3361,00
2VCPUs	4282,73	4125,68	2579,14
4VCPUs	6275,61	4382,94	2382,36
8VCPUs	4994,95	2818,08	1871,14



	1VM	2VMs	4VMs
1VCPU	2466,78	2387,10	3344,86
2VCPUs	4301,54	4113,72	2590,45
4VCPUs	6251,65	4379,06	2341,50
8VCPUs	4979,70	2815,43	1790,74



	1VM	2VMs	4VMs
1VCPU	2494,88	2364,38	3364,29
2VCPUs	4278,31	4119,33	2589,59
4VCPUs	6278,31	4367,05	2338,06
8VCPUs	4997,90	2871,69	1815,16

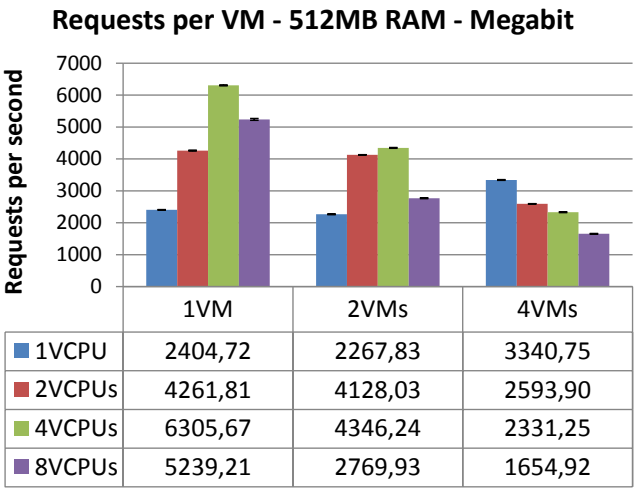
Figure 1: Average of served requests by each VM in an environment with 8GB disk

of served requests per VM. This behaviour was noticed in the comparison between the experiments with 4 VCPUs and 1, 2 and 4 VMs (green columns/ third column of each group). For these experiments, an increase of 100% in the VMs number generated a reduction of, approximately, 30% (1 to 2 VMs) and 46% (2 to 4 VMs) in the quantity of served requests of all graphs. No CPU had been idle since the start of the experiments execution.

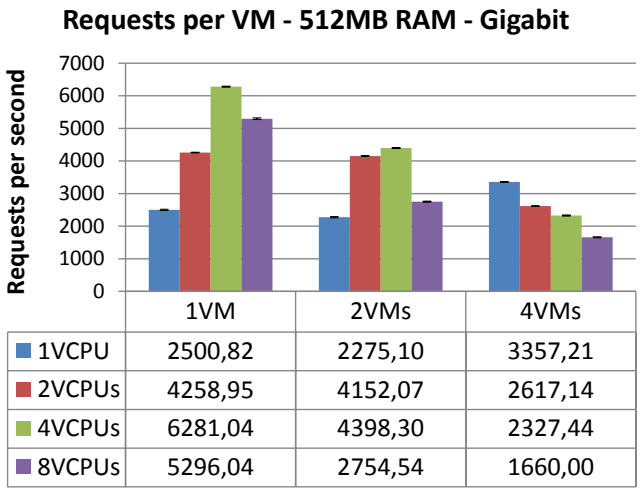
On the other hand, the experiments with 1 VM and 1 VCPU had results similar to the experiments with 2 VMs and 1 VCPU. This happened because some resources were idle during the experiments execution. This idleness occurred because the number of virtual cores in the VMs was less than the number of available physical cores. The same behaviour occurred in the experiments with 1 and 2 VMs and both with 2 VCPUs. However, in the experiment with 1 VM and 2 VCPUs, there was a total of 2 VCPUs to be run in 4 physical cores. In the other case, 2 VMs with 2 VCPUs, there was a total of 4 VCPUs to be executed in 4 physical cores. In this case, each CPU received one VCPU to run and all VCPUs were executing in parallel. Therefore, the results were similar, as we consider the average quantity of served requests per second per each VM. Figure 3 illustrates the described behaviour.

Continuing in Figure 1, in the experiments with 4 VMs, the higher the number of VCPUs, the lower the quantity of served requests per second. For this experiments set, the number of physical cores [3] was a limiting factor, because, with the number of VCPUs increasing, the competition for these resources increases and, for this reason, there is a reduction in the quantity of served requests. However, in the experiments with 1 and 2 VMs, the competition for physical resources is lower, resulting in a higher quantity of served requests. For 1 VM, the increase of 1 to 2 VCPUs and, later, from 2 to 4 VCPUs in the number of VCPUs

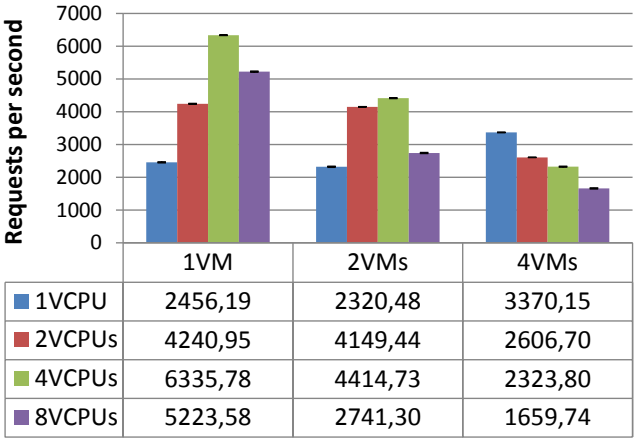
[3]Intel Core 2 Quad processor was used in the experiments.



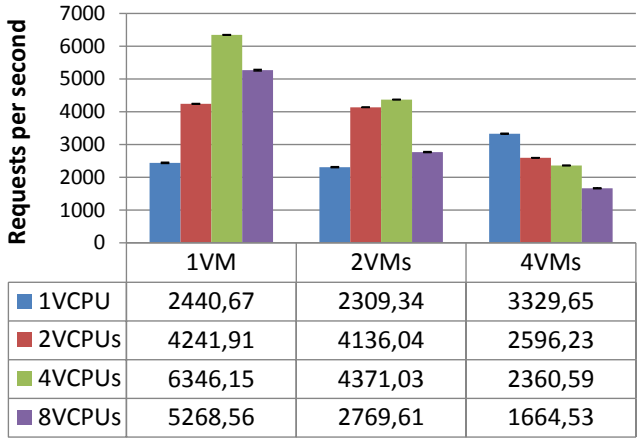
	1VM	2VMs	4VMs
1VCPU	2404,72	2267,83	3340,75
2VCPUs	4261,81	4128,03	2593,90
4VCPUs	6305,67	4346,24	2331,25
8VCPUs	5239,21	2769,93	1654,92



	1VM	2VMs	4VMs
1VCPU	2500,82	2275,10	3357,21
2VCPUs	4258,95	4152,07	2617,14
4VCPUs	6281,04	4398,30	2327,44
8VCPUs	5296,04	2754,54	1660,00



	1VM	2VMs	4VMs
1VCPU	2456,19	2320,48	3370,15
2VCPUs	4240,95	4149,44	2606,70
4VCPUs	6335,78	4414,73	2323,80
8VCPUs	5223,58	2741,30	1659,74



	1VM	2VMs	4VMs
1VCPU	2440,67	2309,34	3329,65
2VCPUs	4241,91	4136,04	2596,23
4VCPUs	6346,15	4371,03	2360,59
8VCPUs	5268,56	2769,61	1664,53

Figure 2: Average of served requests by each VM in an environment with 16GB disk

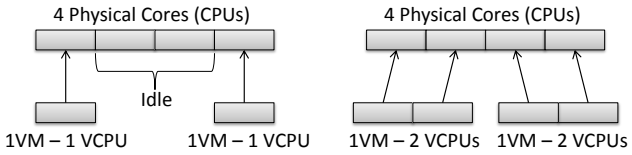


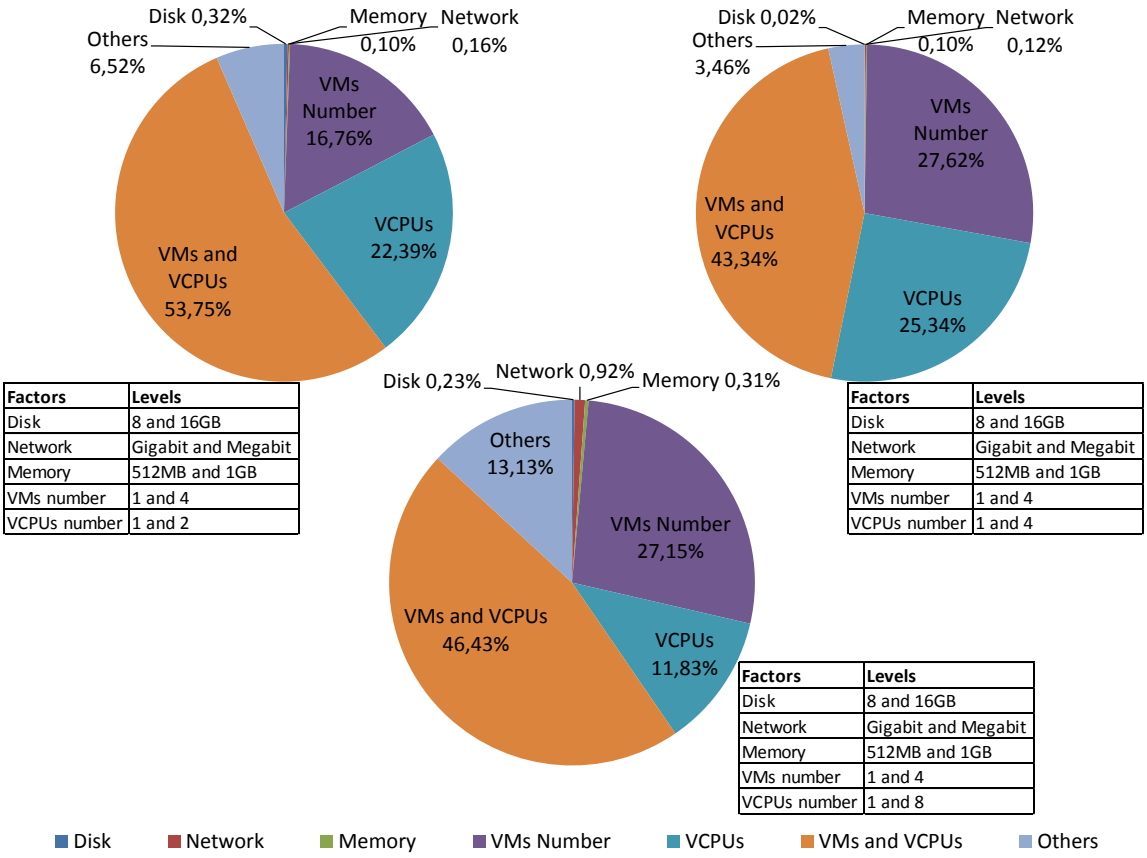
Figure 3: Processor use

increases the response variable by, approximately, 73% and 46%, respectively. For 2 VMs, the same increase in the number of VCPUs results in an increase of, approximately, 75% and 6%, respectively.

The behaviour described in Figure 1 is applied to Figure 2 where the disk size was changed from 8 to 16 GB. In this way, considering both Figure 1 and 2, the change in the memory RAM quantity (from 512MB to 1GB), network (from Megabit to Gigabit) and disk size (from 8 to 16GB) does not generate big changes in the results, i.e., the system performance behaviour remains the same. Therefore, considering the experiment design and the workload used, there were no statistical differences in the results with the combination and change of these factors. This assertion is proven in the next section by the factors influence analysis.

B. *Factors Influence*

In this paper the full factorial model was used to measure the influence of each factor and the respective levels on response variable. A full factorial design for N factors and k levels requires N^k experiments [7]. In this way, analyses considering the factors Disk size, Network type, Memory (RAM) quantity, VMs number and VCPUs number were performed. Because the VCPUs number factor has 4 levels, an analysis combining the levels in 2 – 2 to determine the influence of each factor in the response variable was performed. The VMs number factor (with 3 levels), the extreme levels were considered, i.e., 1 and 4 VMs.



Factors	Levels
Disk	8 and 16GB
Network	Gigabit and Megabit
Memory	512MB and 1GB
VMs number	1 and 4
VCPUs number	1 and 2

Factors	Levels
Disk	8 and 16GB
Network	Gigabit and Megabit
Memory	512MB and 1GB
VMs number	1 and 4
VCPUs number	1 and 4

Factors	Levels
Disk	8 and 16GB
Network	Gigabit and Megabit
Memory	512MB and 1GB
VMs number	1 and 4
VCPUs number	1 and 8

Figure 4: Factors influence

In all graphs in Figure 4[4], the number of VMs and VCPUs factors combination was the greatest influence in the results with 53.75%, 43.34% and 46.43%. As previously mentioned, as new VMs and VCPUs are added to the system, competition for physical resources becomes greater, impacting on the response variable.

Note that only in the first graph, the number of VCPUs was the second factor with most influence (22.39%), followed by number of VMs (16.76%). Through the combinations shown in this graph, moments where the VCPUs number was lower, equal and greater than the number of VMs are registered. For this reason, the respective influence percentages were obtained.

In the second graph, although the number of VMs influence (27.32%) was larger than the number of VCPUs influence (25.34%), the combination between these factors had a reduction in the influence percentage of, approximately, 19% compared to the first graph. This percentage was added to the VMs and VCPUs number influence, providing an increase of, approximately, 65% and 13%, respectively. In the third graph, the third most influential factor, i.e., number of VCPUs (11.83%), had a reduction of, approximately, 53% in influence percentage compared with the second graph.

In this way, though the combination shown in the second graph, the idleness of processing resources during the experiments was lower than the first graph, while the competition by processing resources also was lower than the third graph. Therefore, given the experiments design and the workload used in the system, the combination of the factors and their levels shown in the second graph provided the most efficient use of available resources.

Finally, the influences exerted by the factors Disk size, Network type and Memory RAM quantity were minimal as mentioned in the previous section, less than 1% each one. Therefore, based in the shown analyses, new experiments were performed considering a strictly CPU-bound load. The next section shows the results analysis.

C. CPU-Bound Analysis

The last sections showed that the VMs and VCPUs number were the factors with more influence on the response variable considering a System-bound workload. Based on these results, additional experiments were performed using a strictly CPU-Bound workload. The goal is to verify if the same behaviour is maintained with different kinds of

[4]All the values were obtained using [7].

workload. For this the *Smallpt* benchmark[5] was used. This benchmark renders an image using a Monte Carlo algorithm and shows the execution time (seconds) as response variable. The hardware configuration was as before. The environment specification is shown in Table III.

Table III: Environment specification in CPU-Bound analysis

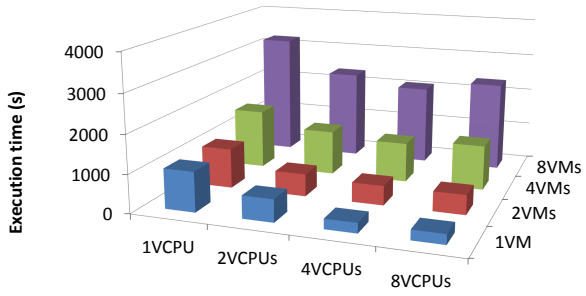
Factors	Levels
Disk size	8GB
Network type	Gigabit
Memory RAM quantity	512MB
VMs number	1, 2, 4 and 8
VCPUs number	1, 2, 4 and 8

In Figure 5, as new VMs were added in the system, the competition for computational resources became greater, increasing, consequently, the workload execution time. This behaviour is noted in the experiments with 4 VCPUs, where the increase of 100% in the VMs number (from 1 to 2, 2 to 4 and 4 to 8) provided an increase of, approximately, 95%, 105% and 100% in the response variable, respectively.

An interesting analysis consists in the comparison among the columns that compose the diagonals. Although all experiments that compose the main diagonal in Figure 5 had a total of 8 VCPUs, the experiments with a smaller VM number had better results. In other words, the experiment with 1 VM and 8 VCPUs obtained a better execution time than the experiment with 2 VMs and 4 VCPUs. This difference was of, approximately, 49%. Considering the comparison between 1 VM with 8 VCPUs and 4 VMs with 2 VCPUs, the first case obtained a lesser result than the second case, approximately, 78%. For the diagonal extremities, i.e., experiments with 1 VM with 8 VCPUs and 8 VMs with 1 VCPU, the difference in the execution time was, approximately, 92% based on the change from 8 to 1 VM. In this way, considering an environment with the same total number of VCPUs, the combination of 1 VM with 8 VCPUs had better performance than the combination with 8 VMs and 1 VCPU. The same behaviour occurred with environments with 2 VMs with 4 VCPUs and 4 VMs with 2 VCPUs, where the change from 4 to 2 VMs provided a reduction in the response variable in, approximately, 57%.

Furthermore, it is noteworthy that the experiment with 1 VM with 1 VCPU had similar result to the experiment with 2 VMs with 1 VCPU. This is justified by the occurrence of idle physical resources in these scenarios during the experiments execution. This behaviour was discussed in the experiments analysis described in Figures 1 and 2. The same behaviour occurred in the experiment with 2 VMs with 1 VCPU that had similar result to the experiment with 2 VMs with 2 VCPUs. This behaviour can be exemplified by Figure 3.

[5]http://openbenchmarking.org/test/pts/smallpt



	1VCPU	2VCPUs	4VCPUs	8VCPUs
1VM	1040,20	586,00	260,10	260,20
2VMs	1039,15	585,50	508,20	512,40
4VMs	1556,40	1170,03	1040,18	1171,45

Figure 5: Execution time in seconds

Finally, the VCPUs number increase provided a reduction in the system execution time. However, this affirmation was valid until the VCPUs number reached the physical cores number. From this point, the increase in the VCPUs number provided a greater competition for physical resources, harming the system performance. Analysing the experiments with 8 VMs (purple columns), the increase of 100% in the VCPUs number, from 1 to 2 and 2 to 4, provided a reduction in the execution time of, approximately, 27% and 11%, respectively. From this point, the VCPUs number increase, from 4 to 8, increased the execution time in, approximately, 13%.

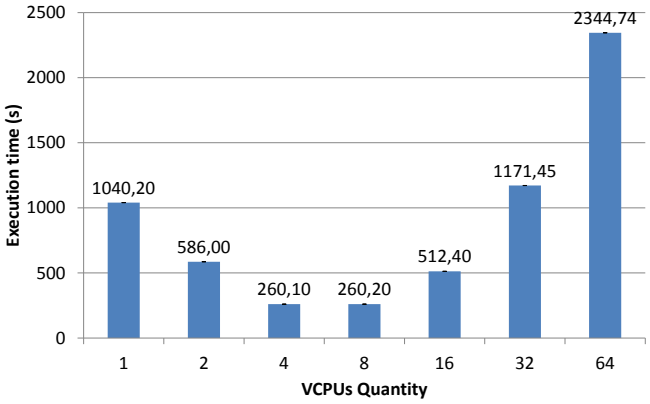


Figure 6: Execution time vs. VCPUs quantity

Figure 6 shows a comparison considering the VCPUs number increase. Through this analysis, is possible to prove the described behaviour, where the increase in VCPUs number, from 8 to 16, 16 to 32 and 32 to 64, provided an increase of, approximately, 97%, 129% and 100%, respectively, in the response variable.

IV. Related Work

We consider two different types of related work: papers available in the academic literature and business scenarios

applied by companies like Amazon.

There are several papers in the academic literature with the aim of analysing and proposing efficient mechanisms for resource self-management in a cloud environment. In [5], for instance, the authors present a study about resource allocation among multiple HPC (High Performance Computing) systems like cluster, grid and cloud. Sharkh et al. [14] discuss various internal and external factors that should be considered in the resource allocation process. Manvi and Shyam [9] conduct a more embracing study that considers resource management techniques as provisioning, allocation, mapping and adaptation in an Infrastructure as a Service (IaaS). In [16], various resource allocation strategies and their challenges are discussed.

The model proposed in [8] consists of an adaptive model for allocation and provisioning of resources based on analysis of historical factors. A statistical method is applied to choose the most influential characteristics to improve the application performance and determine where a VM is allocated and which instance type is more suitable for the specific application.

In [17], the authors propose resource allocation algorithms for SaaS providers allowing to reduce infrastructure cost. They modelled a simulated environment where VMs are assigned following a pre-defined mapping strategy. However, the environment does not change the VMs capacity dynamically. It tries to maximize the profit minimizing the cost by reusing the created VMs and applying the multi-tenancy approach, which can induce an SLA (Service Level Agreement) violation.

In [1] the authors propose an adaptive approach with focus on automation of tasks management and resources scalability to ensure the contracted QoS. For this, an analytical performance model and workload information is used which enables a better resources allocation avoiding undesired performance, measured in response time and rejected requests. The analytical performance model defines which resource configurations are better suited for a specific workload and when resources should be increased or reduced [11].

In the analysed papers the authors present autonomous mechanisms of resource provisioning where VM configurations are fixed. VMs are split in different classes according with their capabilities and the number of VMs is increased or reduced during the resource provisioning. In this way, there is no dynamicity in the resource configurations that compose the VMs during the workload execution. Considering that SLAs can be renegotiated, an environment with fixed configurations cannot be the most efficient.

Considering the business scenarios, there are cases where a client pays a specific price for a service that is not fair. He controls the virtual resources and not the physical resources. For this reason, he can pay a price for a dedicated resource quantity, for instance, and the quantity allocated by the provider can be less than that requested or it can be shared with other clients. Also, the provider can offer an increase in the resource quantity at a specific price and this increase can be illusory, because sometimes the resources quantity increase does not provide a better system performance.

Various services as Amazon EC2[6] and Windows Azure[7] work with a billing system where virtual machines are classified according with their configurations of memory, virtual cores, network type and disk size, for instance, and the price is defined according with the class. In these cases, there is no plausible explanation for the resources increase from a class to another. Furthermore, they leave the clients responsible for precisely estimating the amount of needed resources [6].

Moreover, providers such as Compiere ERP[8] provide an individual VM for each client to maintain service level requirements in terms of response time and capacity. However, this can cause overload and/or idleness of hardware resources which results in high infrastructure cost, since clients may not have the contracted QoS level.

The study presented in this paper showed an analysis that considers the impact on the system performance with the increase in the VMs capabilities and the influence of each resource on the response variable. This analysis is important, because is possible to quantify the resources that compose a VM instead for it to be configured with random quantities. In this way, clients can contract the ideal quantity of resources to run their applications and, consequently, save money. On the other hand, providers can achieve a better use of the available resources and more reliably ensure the QoS contracted by clients.

V. Conclusion and Future Work

In this paper we presented some experiments with the goal of identifying the computational resources and quantity of resources that should be allocated to a specific client in a cloud in order to gain most appropriate resource utilisation and performance gains. The correct automatic resource allocation allows a greater load balancing and a better use of the available resources, besides to help in the enforcement of the QoS levels requested by clients. Therefore, is important know which computational resources should be allocated and the quantity because this provisioning directly influences the final cost of service. In this way, experiments were performed considering various virtual machines configurations with different workloads.

In the results, for the workload used, changes in the memory quantity, disk size and network type did not provide significant impact on the response variable. In this way, some providers can be offering more computational resources, combined with an increase in the cost, promising a performance increase that is not real. On the other hand, it

[6]http://aws.amazon.com/ec2/

[7]http://www.windowsazure.com/en-us/pricing/details/virtual-machines/

[8]http://www.compiere.com/

was more efficient to increase the number of VCPUs instead of the number of VMs. However, as shown in Figure 6, this increase rate should consider the number of available physical cores. In an environment where the number of VCPUs exceeded the physical CPU cores, the environment performance was harmed, since the competition for physical resources was greater.

Furthermore, Figure 5 showed that, an environment with 1 VM with 4 VCPUs was more efficient than one with 4 VMs and 1 VCPU. It happened because the memory access and messaging between cores are faster in a multicore environment than an environment with multiples VMs, while the delay in the VCPUs context switching is smaller.

The next steps consist in the proposal of an autonomic mechanism to control the elasticity in a cloud environment. This mechanism should consider different workloads composed by different security levels. Techniques and methodologies available in the literature that aim to ensure the integrity, availability and confidentiality of data will be used to compose the different security levels. In this way, it will be possible to verify the impact of the security policies on the system performance and, from the presented analysis, automatically define the additional resource quantity that is necessary to cope with the overload caused by the security policies. Therefore, the possibility of maintaining the system performance with the security attributes from the allocation of more computational resources to the VMs will be analysed Finally, a new business model that reflects the impact of these additional computational resources will be proposed.

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support provided by CNPq, CAPES and FAPESP for the projects under development at the Distributed System and Concurrent Program, group of the Computer Systems Department at ICMC-USP. This study was conducted while Bruno G. Batista was on a research exchange at the University of Leicester, generously supported by the Brazilian Science without Borders program.

REFERENCES

[1] Rodrigo N Calheiros, Rajiv Ranjan, and Rajkumar Buyya. Virtual machine provisioning based on analytical performance and qos in cloud computing environments. pages 295–304, 2011.

[2] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. Comparison of the three cpu schedulers in xen. *SIGMETRICS Performance Evaluation Review*, 35(2):42–51, 2007.

[3] Julio Cezar Estrella, Regina HC Santana, and Marcos J Santana. Wsarch: An architecture for web services provisioning with qos support performance challenges. 2011.

[4] Paul Ferguson and Geoff Huston. *Quality of service: delivering QoS on the Internet and in corporate networks*. Wiley New York, NY, 1998.

[5] Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Wang Yongji, Nasir Ghani, et al. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736, 2013.

[6] Tram Truong Huu and Johan Montagnat. Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 612–617. IEEE, 2010.

[7] Raj Jain. *The art of computer systems performance analysis*, volume 182. John Wiley & Sons Chichester, 1991.

[8] Seoyoung Kim, Jik-Soo Kim, Soonwook Hwang, and Yoonhee Kim. An allocation and provisioning model of science cloud for high throughput computing applications. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 27. ACM, 2013.

[9] Sunilkumar S Manvi and Gopal Krishna Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 2013.

[10] Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800(145):7, 2011.

[11] Daniel A Menascâe, Virgilio AF Almeida, Lawrence Wilson Dowdy, and Larry Dowdy. *Performance by design: computer capacity planning by example*. Prentice Hall Professional, 2004.

[12] George Reese. *Cloud application architectures: building applications and infrastructure in the cloud*. O'Reilly, 2009.

[13] John W Rittinghouse and James F Ransome. *Cloud computing: implementation, management, and security*. CRC press, 2009.

[14] Mohamed Abu Sharkh, Manar Jammal, Abdallah Shami, and Abdelkader Ouda. Resource allocation in a network-based cloud computing environment: design challenges. *Communications Magazine, IEEE*, 51(11):46–52, 2013.

[15] Srinivas Vegesna. *IP quality of service*. Cisco Press, 2001.

[16] V Vinothina, Dr R Sridaran, and Dr PadmavathiGanapathi. A survey on resource allocation strategies in cloud computing. *International Journal of Advanced Computer Science and Applications*, 3(6), 2012.

[17] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 195–204. IEEE, 2011.