Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 753–760
DOI: 10.15439/2014F331
ACSIS, Vol. 2

Performance Analysis of SaaS Ticket Management Systems

Pano Gushev
Innovation Dooel
1000 Skopje, Macedonia
pano.gushev@innovation.com.mk

Sasko Ristov, Marjan Gusev
Ss. Cyril and Methodius University, FCSE
1000 Skopje, Macedonia
{sashko.ristov, marjan.gushev}@finki.ukim.mk

***Abstract*—Cloud architecture has the ability of sharing hardware resources and services among multiple tenants. In this paper we measure the performance for the multi-VM (multiple virtual machines) cloud architecture and compare it with the single-VM architecture. Renting resources on a cloud usually comes with a variety of options, such as use of more and smaller virtual machines or use of less and bigger virtual machines. The objective of this research is to find out which scenario gives better performance for the same price of rented resources. This will be done by comparing the following attributes: Average response time, Pages per second, Average page time, Requests per second, CPU time. We setup a hypothesis that the multi-VM approach would be better, and the best architecture is the one offering the highest number of small virtual machines, predicting that the computational demands will spread to different virtual machines in a balanced manner. The results confirm the hypothesis and lead to a recommendation for an optimal architecture of a cloud based solution for a common transactional web solution.**

***Index Terms*—Cloud; Web Services; Windows Azure.**

I. Introduction

CLOUD computing has a growing trend in the past few years and companies increasingly understand its benefits [1]. Gartner [2] determined that the size of the cloud market is $150 Billion by 2013 and predicts that virtualised server workloads will reach a high of 60% in 2014. This trend pushes companies to migrate their services and applications in the cloud.

Benchmarking the cloud and multi-tier applications that are hosted within will help the developers to determine the most appropriate architecture, services, and settings [3]. In this paper we present results of the performance analysis of different approaches in building possible cloud architectures and solutions of the Ticket Management System (TMS).

The objective of this research is to find out which scenario performs the best for the same price of rented resources. We assume that a single processor with four cores has approximately the same price with 4 CPUs with one core, 8 GB RAM has roughly the same price with two RAMs of 4 GB etc.

Traditional distributed approach claims that the best performance is achieved when the workload is distributed to a huge number of processing units executing balanced tasks. However, it is very hard to organise the tasks with perfect load balancing in a distributed system. The cloud, on the other hand, initiates new challenges, such as predicting server CPU utilisation and resource under-provisioning. In addition, scalability and elasticity are main concerns when the customers desire to deploy their solutions on the cloud. Client expectations are very important, due to the recent huge offer of cloud providers. Summarising the above, a typical customer challenge is selection which architecture performs the best, renting more and less powerful Virtual Machines (VMs) or a smaller number of more processing powerful VMs.

We have set a hypothesis that the best performance will be achieved if the processing is distributed to more less powerful VMs. The testing environment is deploying the ticket management SaaS solution on various Windows Azure configurations with approximately the same cost.

The rest of the paper is organised as follows. In Section II, we discuss the related work and in Section III describe the testing environment, ticket management web service and test data. Section IV presents the results and Section V compares the results and evaluates the performance. The final Section VI is devoted to conclusion and future work.

II. Related work

This section presents the recent research in the area of cloud performance benchmarking, resource allocation and multi-tier transactional cloud web application.

A. Cloud performance discrepancy

Renting the same amount of IaaS (Infrastructure as a Service) resources on the Azure cloud [4] costs the same, regardless whether they are all provisioned in a single huge virtual machine or into several smaller virtual machines. This pricing scheme also follows the other most common public clouds [5], [6].

An orchestration of hardware resources on the cloud impacts the performance of the hosted cloud application or web service. Allocating the resources among many concurrent instances of virtual machines is better than using VM with multiprocessors using parallelization [7].

Gusev et al. [8] determined a region where response time is up to 10 times better if web services are spread among several small VMs, rather than in a single VM. This phenomenon motivated us to analyse how orchestration impacts the performance of the real cloud N-tier application in Azure.

Cloud multitenant virtual environment also provides discrepant performance during the time and therefore, a performance isolation is required [9]. Koh et al. reported that a VM

provides different performance on the same physical machine during the time, if it is instantiated among the other active VMs [10]. Jayasinghe et al. [11] reported that a configuration that provides the best performance in one cloud can provide the worst performance in another. Iakymchuk proposed a method to improve the performance with underutilization of physical resources by adding more nodes [12].

B. Windows Azure performance

Many authors have conducted research on Azure's performance. It is most suitable for application with small amount of communication [13] and it does not work well for tightly-coupled applications compared to Amazon EC2 or cluster [14]. Hill et al. [15] provided a performance analysis of VMs, storage services and SQL services in Azure. Azure also can decrease the costs and time for deployment, as well as support efficient TCP data transfers [16].

C. Multi-tier performance benchmarking

Today's web applications are usually 3 or 4-tier applications, commonly identified as multi-tier (n-tier) applications. Each tier is usually hosted on a separate machine or VM. These architectures are usually interesting due to several bottlenecks that can appear, which will decrease rapidly their performance.

Several authors propose benchmark tools and methodologies to simulate the multi-tier cloud web applications to analyse their performance and determine their deficiencies.

Turner et al. [17] propose a C-MART benchmark, which emulates a modern web application hosted in a cloud environment. Rubis [18] is another benchmark application that simulates an auction site and evaluates design patterns and the performance scalability of application servers. However, it cannot be used as a modern application benchmark because there are many flaws for Web 2.0 applications [19]. Also it requires a lot of effort to install, which outweighs its usefulness [20]. TPC-W is also a transactional web benchmark, which emulates an online bookstore [21]. But, it also has deficiencies [3]. In this paper, we use a real cloud SaaS application hosted in Azure to determine the scalability and user load impact on different customers' and cloud service provider's parameters.

Apart from benchmarking tools, several methodologies for benchmarking the N-tier applications are proposed and evaluated with them. Wang et al. [22] proposed a method for bottleneck detection, which correlates throughput and load. Chen et al. [23] proposed a predictive performance model that analyses cloud based N-tier web applications and determines appropriate resource allocation to each tier of an application.

III. Methodology

This section describes the architecture of the system and the environment used for testing, including the testing tools, hardware architecture, test cases and implementation procedure.

A. TMS Architecture

As a testbed-application we use our TMS SaaS solution [24] as a scalable and elastic multi-VM cloud solution. This benchmark has all features to evaluate performances of a typical transactional based SaaS solution, such as, simple operations for each transaction, and variable and huge number of users that generate different workload.

The benchmark TMS solution consists of three code modules and one optional module [24]. Core modules are: System management module (SMM), Company management module (CMM) and Core functions module (CFM) and optional is Additional functional module (AFM).

The core of this TMS cloud solution is the SMM module, shared for all companies. All resource provisioning in the cloud are managed by this module. It is always active to provide company subscription management, authentication, authorisation etc.

CMM is a company specific module, personalised and customised by TMS users. Each company offers this module as a service to its end customers. This module is responsible for all general configurations within a company.

Another company specific module is the CFM module, responsible for all essential processes to realise ticket management (bug reporting) within a company. It enables a connection between companies (providers) and their customers. This module provides functionalities such as: ticket creation, responds by a resolver, customer approvals of the responses, creation and execution of test-cases etc.

The only optional module is the AFM module, dedicated for add-on functions, such as enabling a possibility to create and execute test cases, especially useful for bug reporting systems.

All modules in this TMS solution are divided in two groups: static and dynamic modules. Static modules are always active modules, unlike dynamic modules that are activated only when some company needs them. From the modules we mentioned above SMM is a static module and the other three are dynamic modules.

This system integrates a very important Broker module in order to communicate with different company specific modules and acts as a role of system service orchestrator. Fig. 1 [24] presents three agents that are a part of the Broker module.

Let us explain shortly how this system organisation works. As we mentioned before SMM is the main module of the system and it is always active to provide features for "pay-by-use" cloud concept, including authentication, authorisation, accounting with management of the company, its' contracts and subscriptions etc. This module is realised by two agents:

- *Admin agent*, which provides the accounting features for the company.
- *Infrastructure agent*, which instantiates and closes various instances, providing the optimal resource utilisation and reducing the overall cost for renting the hardware resources.

The dynamic modules of this system have the opportunity to be hosted on the same machine or on different virtual machines, which is essential for our testing of the elasticity, because we want to analyse and compare the performances obtained in a single-VM and multi-VM environments.

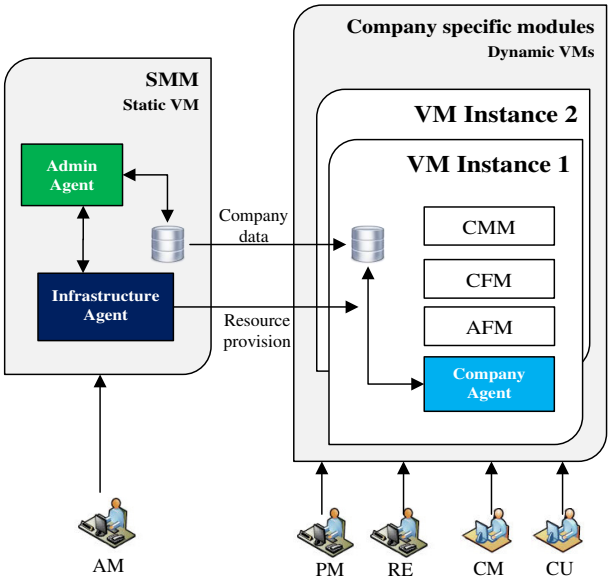

Fig. 1: Architecture of benchmark TMS SaaS solution

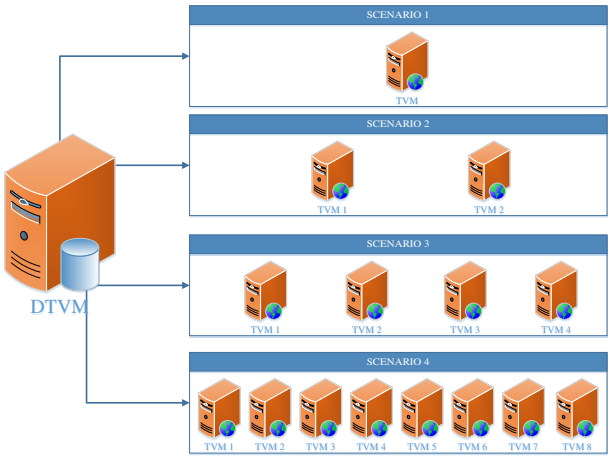

Fig. 2: Architecture of testing environment and scenarios.

TABLE I: Test case specification

Scenario	VM Type	Cores	RAM	Resources
All	DTVM	8	14 GB	8 Cores 14 GB
Scenario 1	1 TVM	8	14 GB	8 Cores 14 GB
Scenario 2	2 TVMs	4	7 GB	8 Cores 14 GB
Scenario 3	4 TVMs	2	3.5 GB	8 Cores 14 GB
Scenario 4	8 TVMs	1	1.75 GB	8 Cores 14 GB

Each dynamic VM hosts two core software modules: the CMM and CFM modules, independently for each company. Additionally dynamic part of the AFM module is hosted on each dynamic VM.

The Company agent is hosted on the same VM, to enable a communication with SMM and data synchronisation between the SMM and VM.

B. Testing environment

The testing environment consists of VMs and Windows Azure Cloud [4]. Windows Azure provides on-demand infrastructure that scales and adapts to the user ever changing business needs. With Windows Azure, the user can spin up new Windows Server and Linux VMs relatively fast and customise the environment. To create fast robust deployment, Windows Azure allows using custom images or building on the pre-configured images.

To realise the experiments we have created architecture for four different scenarios that use one *Database and Testing VM* (DTVM) and a set of *transactional VMs* (TVMs). All of them are deployed on Windows Azure as presented on Fig. 2.

The testing machine is used to run the tests for all experiments. The DTVM consists of AMD Opteron 4171 HE 2.10 GHz CPU with 8 cores and 14GB RAM. It hosts the SQL database for all scenarios. Windows Server 2012 Datacenter is running on the DTVM. The installed testing tools are selected from the Visual Studio 2012. The experiment results are measured with the *Web Performance and Load Test Project*, as a powerful tool for this kind of research.

The main function of the DTVM is to run all the tests for predefined scenarios. The first scenario is predefined as a single-VM, while the rest of them concern the multi-VM approach. The TVMs are configured to have equal cost in each scenario using resources on a machine with AMD Opteron 4171 HE 2.10 GHz CPU with 8 cores and 14GB RAM. Therefore, the first scenario is defined by one TVM that uses 8 CPU cores and has 14 GB of RAM. Two TVMs are used in the second scenario, each of them with 4 CPU cores and 7 GB of RAM. The third scenario uses 4 TVMs with 2 CPU cores and 3.5 GB of RAM, while the fourth scenario 8 TVMs with 1 CPU core and 1.75 GB of RAM.

All scenarios use VMs with same type of CPU and RAM. Windows Server 2012 Datacenter is preinstalled on TVMs including IIS and Application roles. Detailed features of the DTVM and TVMs in all scenarios are shown in Table I.

Experiments are tested by simulating two types of network: Cable DSL 1.5Mbps and Cable DSL 768Mbps. All experiments are realised with the use of a web browsers. To enable platform independence we have tested the experiments by equally using all of the following web browsers: Internet Explorer versions 9 and 10, Firefox, Chrome and Safari.

C. Web Service Description

TMS [25] is a cloud solution realised as a transactional web based software, where a user can access a dynamically created web page, browse the site, list page details and update a selected information, as most of typical web solutions.

The objective of this research is to find the best architecture when analysing the performance for the same price of rented resources. Our hypothesis is based on an assumption that multi-VM approach would be better and the best performance will be obtained renting the largest number of small VMs

for different companies. Note that the database is deployed on a separate VM in all experiments and the testing mostly relates to selecting a proper configuration for a web server that realises functions of web browsing and database update.

We define two test cases to analyse the performance. The first test case is based only on a web service providing a functionality of browsing through the application and the second uses update functionality with data store in database.

In the first test case, the user logs into the system, opens a page with presented tickets and selects a ticket to display relevant information. The user can browse the site and list the content of a given page. In the second test case, despite the previous steps realised for browsing and listing the required information, the user responds on the information by entering an additional information and changing the ticket status. It finishes with database update.

These web services are hosted by all VMs defined by each scenario. Note that the DTVM does not deploy these web services, but hosts the testing tools and the database required by the web services. The 4 scenarios realise both the single-VM and multi-VM approaches using the same amount of available resources. Hosting the web service and database on separate VMs may reduce the overall performance, because of communication needed between the VMs. But in case of large user loads all requests for a web service and interaction with the database will be distributed between VMs. Database operations are more expensive than the communication in Windows Azure, where all VMs are connected in virtual network with stable network connection between them.

D. Testing procedure

Visual Studio 2012 Web Performance and Load Test Project is a tool that offers many testing options. We decided to use the test "Tests based of the number of virtual users" for the purpose of this research providing relevant information about throughput and achieved speed.

We choose Visual studio because this tool provides a lot of possibilities for parametrisation. With Visual Studio 2012 we can choose different types of tests, depending of the number of the users or number of tests. Also we can choose the type of networks, type of browsers that we use for testing. Tests can be executed on different ways depending of number of tests or time span. A very important issue is that the results from tests are specified with sufficient details.

Visual studio 2013 provides a new option to run tests in Windows Azure without new configuration. Administrators do not have to deploy any VM or to configure different services. Through Load Test Web Service, Visual Studio 2013 loads the tests on the cloud. Behind this service is a pool of test agents that is used to run the tests. All results from a test, along with other performance counters are available.

Four experiments are defined, each for a test case defined by a corresponding scenario:

- *Experiment 1* - a single-VM with a DTVM and 1 TVM
- *Experiment 2* - a multi-VM with a DTVM and 2 TVMs
- *Experiment 3* - a multi-VM with a DTVM and 4 TVMs
- *Experiment 4* - a multi-VM with a DTVM and 8 TVMs

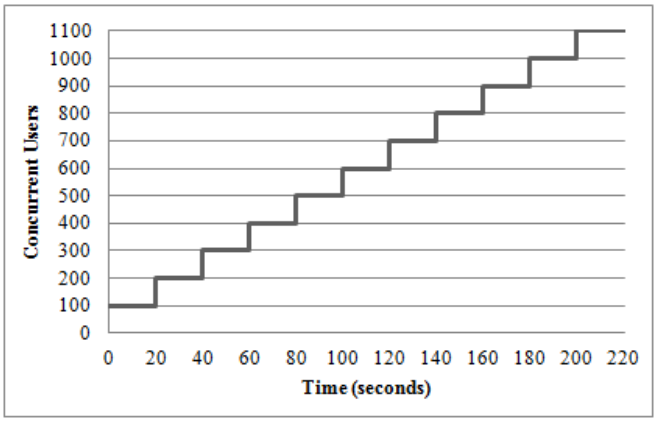

Fig. 3: Test plan for each experiment

All experiments use two VMs with 8 cores and 14GB RAM each. The first experiment simulates a single-VM environment, while the remaining a multi-VM environment with different configuration of TVMs enabled by renting the same cloud resources. Experiments perform different tests based on different loads defined by a various number of virtual users of the system per test.

The selected test runs every test for 3 minutes and 40 seconds. Certain number of users is defined per each testing interval to simulate increased load, as shown in Fig. 3. Our workload generation for multi-tier web application in the cloud is similar to other similar approaches [26]. The initial load configuration starts from 100 concurrent users and this is increased by a step of 100 users every 20 seconds, ending with a maximum of 1100 users.

All test case scenarios are executed for each experiment. To ensure good results, the tests started only after a 2 minutes warm up period of Visual Studio.

E. Test data

The selected testing tool measures and calculates a lot of parameters. For the purpose of our research, the following five different parameters are analyzed:

- T_r - Average response time,
- P - Pages per second,
- T_p - Average page time,
- R - Requests per second, and
- CPU_U - CPU utilisation.

The values of each of these criteria are based on the selected load defined by the number of users.

The performance factors for measured times are calculated as reciprocal value and indicated as throughput and speed, by the following measures:

- T - Overall throughput calculated as $T = 1/T_r$,
- S - CPU speed calculated as $S = 1/CPU_U$.

Each experiment (for a different scenario) results with different performance factors. Let $i, j \in \{1, 2, 3, 4\}$ identify

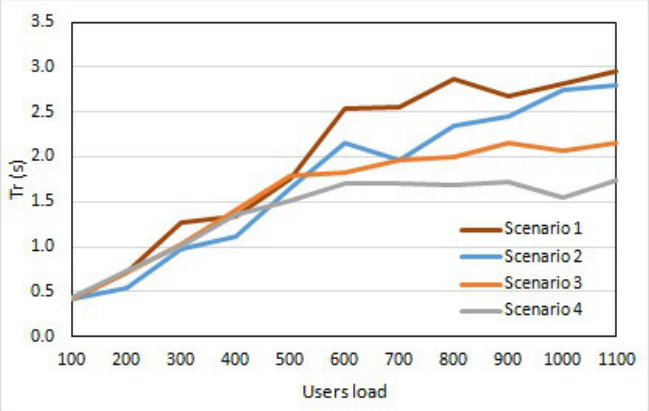

Fig. 4: Average response time as a function of user load

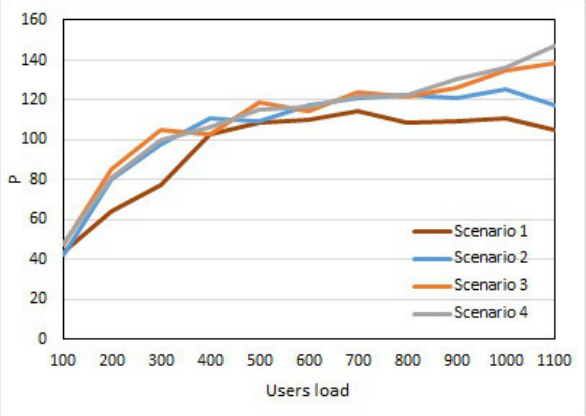

Fig. 5: Pages per second as a function of user load

two experiments. Their relative index as presented in (1) gives the relative comparison value for performance factor F, which stands instead of T, P, R, or S, corresponding to overall throughput, page throughput, request throughput and achieved processor speed. The higher the relative performance factor is, the better performance.

$$F_{ij} = \frac{F_i - F_j}{F_j} \tag{1}$$

In addition, when compared to Experiment 1 we obtain relative performance increase of multi-VM vs single-VM approach for the same number of users.

IV. Experiments and Results

This section presents the results achieved for each performance criteria: average response time, pages per second, average page time, handled requests per second and CPU utilisation. Parameters are analysed as a function of user load.

The brown line in each graph presents the results for scenario 1, the blue line scenario 2, the orange line scenario 3 and the grey line scenario 4.

A. Average Response Time T_r

Response Time refers to the average time that is necessary to receive the entire response to a request, starting from the moment when a request is sent to the web server. The difference between the Response Time and Transaction Time is in think time that occurs during a transaction. Transaction Time includes think time, but Response Time does not [27].

Fig. 4 presents the average response time results for all experiments. The average response time proportionally depends on the load increase defined by the number of concurrent users. Two different regions are observed with different behaviors. In the left region ($N \leq 500$) all four scenarios provide similar average response times. However, Scenario 1 is the worst, while Scenario 2 is slightly better than others for all test cases in the region. Scenario 1 also provides the worst performance in the right region ($N > 500$), but now Scenario 4 has the smallest average response time. All curves in the left region have a trend for linear increment until the value $1.5s$, which saturates in the right region in the range of $[1.5s, 3s]$.

B. Average pages per second P

Average pages per second refers to the number of pages that are sent per second during the load test run [27]. Fig. 5 displays the results for the performance factor P defined by achieved average pages per second.

Single-VM (Scenario 1) also shows the lowest performance for this parameter for each user load. Multi-VM scenarios provide similar performance for smaller load ($N \leq 500$), but the results are similar as the previous parameter for greater load. That is, using greater number of smaller TVMs is better than having smaller number of greater TVMs.

All curves in the left region also have a trend for linear increment starting from 40 to 110, which saturates in the right region in the range of $[100, 150]$.

C. Average page time T_p

Average page time refers to the average response time for all requests for a single web page [27]. The results of the average page time are shown in Fig. 6. Also in this case, the results show that the multi-VM approaches perform better than the single-VM scenario. Scenario 4 performs the best for a large number of concurrent users higher than 700, while Scenario 2 for smaller loads.

The curves follow the similar trend as the previous two parameters, in the regions divided with $N = 500$ concurrent requests. The average page time increases starting from 0.5 to 2.5s and saturates in the range [2.0s, 3.5s] for all scenarios.

D. Handled requests per second R

Requests per second consist of details for individual requests that are sent during a load test. This includes all HTTP and dependent requests such as images. *Request per second* refers to the rate per second of a request during the load test run [27].

Results for the overall handled requests per second are presented in Fig. 7. The obtained behavior and performance

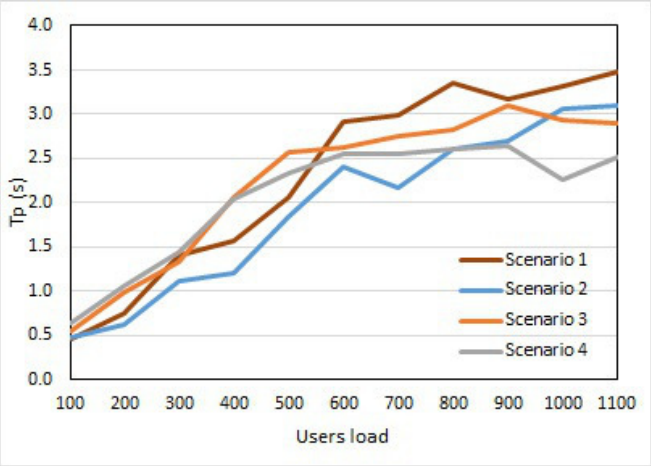

Fig. 6: Average page time as a function of user load

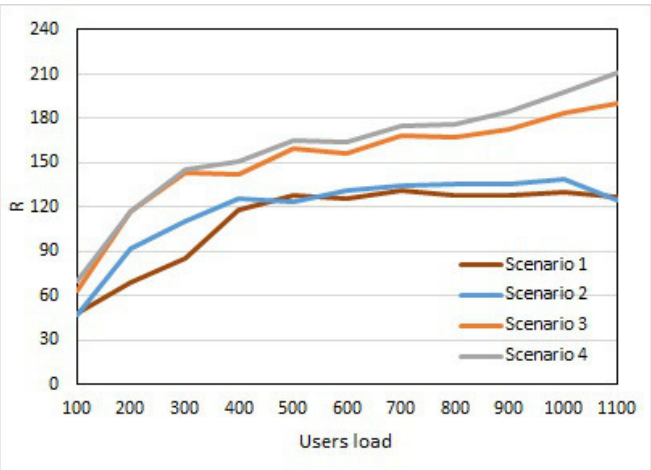

Fig. 7: Requests per second as a function of user load

trend is similar to the cases with previous analysed parameters. That is, the scenarios with greater number of smaller TVMs (scenarios 4 and 3) are better than having smaller number of greater TVMs (scenarios 2 and 1).

We observe small discrepancy compared to other parameters. That is, the curves start to saturate for different user load. Scenarios 1 and 2 increase the values for the handled requests starting from 50 to 120, while scenarios 3 and 4 provide increase from 70 to 140.

Scenarios 3 and 4 slightly saturates for $N > 300$ (they continue to increase, but with smaller intensity) in the range $[140, 210]$, while scenarios 1 and 2 saturate for $N > 400$ and their curves are almost constant in the right region around the range $[120, 140]$.

E. CPU Utilization R

Processor time denotes the percentage of time that a VM instance charges against the processor user time for individual request issued during a load test. The theoretical maximum for this parameter is $number\_of\_processors \cdot 100$.

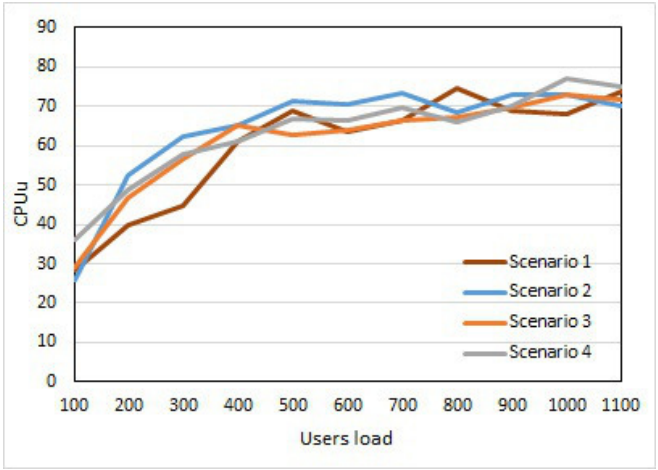

Fig. 8: CPU utilization as a function of user load

CPU utilisation is very important parameter for cloud service provider because it is directly connected with cost for power supply and cooling due to increased heating. The results for this parameter are shown in Fig. 8. Although all scenarios are very similar, lower CPU utilisation is noticed for the single-VM environment, while Scenario 2 mostly utilizes the CPU.

Similar two regions are observed, which are divided with $N = 400$. The curves increase in the left region until 65%, and then the CPU utilization saturates in the range of $[65\%, 70\%]$.

V. Discussion

This section compares the results achieved from the four cloud approaches and discussed relevant explanations for analyzed behavior and performance trends.

A. Parameter correlations

This section presents the correlations between the response time and page throughput and between the response time and CPU utilization.

1) Response time vs page throughput: The parameters T_r, P, T_p, and T_R are correlated among each other, as shown in figures 4, 5, 6, and 7. The results show that user loads for $N = 400$ or $N = 500$ are borders of two different regions. The parameters behave the same in a single region, i.e., increasing trend in the left region (smaller load) and saturating trend in the right region (heavy load).

2) Response time vs CPU utilization: Figures 9 and 10 present the overall throughput and CPU speed. We can conclude that both figures are very similar, which confirms that both parameters are very correlated as a function of user load.

B. Scenario comparisons

In most cases, especially for increased loads, we concluded that Scenario 4 performs the best compared to the others.

The load defined by the number of concurrent users has noticeable impact on the results. All performance factors lead

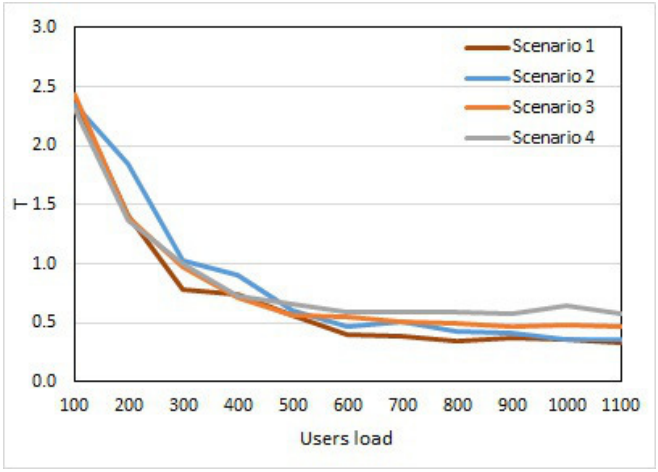

Fig. 9: Overall throughput T as a function of user load

Fig. 11: Relative throughput T_{12}, T_{13} and T_{14}.

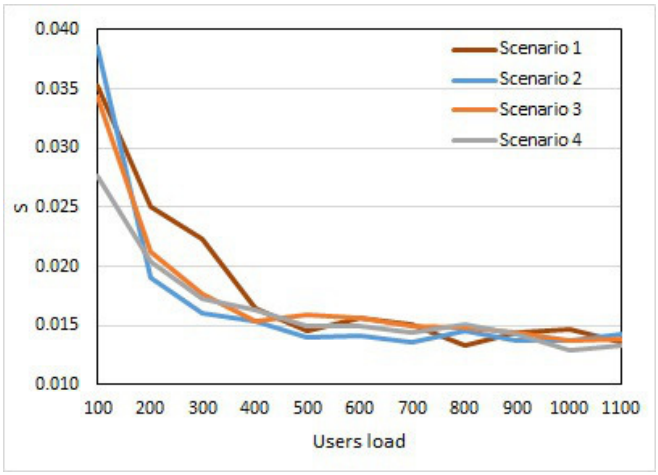

Fig. 10: Achieved CPU speed S as a function of user load

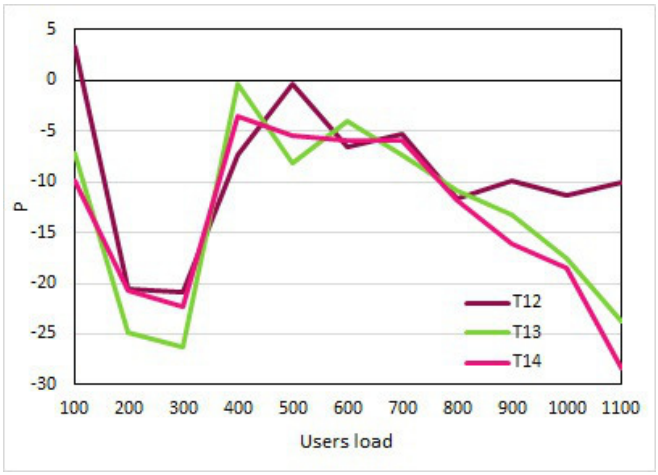

Fig. 12: Relative pages per second R_{12}, R_{13} and R_{14}.

to a conclusion that our hypothesis is proved for increased loads higher than 400 users.

Figures 11, 12 and 13 present the relative values of parameters for the three multi-VM scenarios compared to the single-VM scenario. They confirm the previous conclusions that the scenarios with greater number of smaller VMs is better than smaller number of bigger VMs as TVM.

VI. Conclusion

Performance is critical when choosing appropriate configuration for rented cloud resources. Cloud providers offer a variety of options, so the users get confused when selecting the optimal configuration. This research brings conclusion about a typical behavior in transactional web solutions for expected small and medium number of concurrent users. The ticket management solution is a transactional-based dynamic web site where a lot of information fluctuates among users, and information update is a frequent task. It uses a lot of users which access a dynamic web site.

Fig. 13: Relative page time between multi- and single-VM

The presented architecture for the cloud solution is based on separating the database layer form the business logic and presentation layers on different VMs. This solution uses a database allocated to a single and powerful VM, and distributing the transactional business logic and presentation layer to a set of other VMs. An intended user may choose whether to host a new powerful VM or to distribute the load to more less powerful VMs using the same cloud resources.

We have performed load testing with increasing number of users, testing the performance of 4 different configurations that use the same rented resources. We were motivated by the challenge to find the most optimal configuration that performs the best. This research proves our hypothesis that using a higher number of small VMs performs better than other configurations, confirming the traditional distributed concept to distribute tasks on more balanced processing units.

This result suggests an architecture for cloud based solutions that performs the best for transactional web sites, which do not perform complex computations and are mainly based on page browsing and database update operations. The recommendation is to separate the database layer from other layers, allocating the database layer to a powerful VM and distributing the business logic and presentation layers to more less powerful VMs.

REFERENCES

[1] A. Murua, I. Gonzalez, and E. Gomez-Martinez, "Cloud-based assistive technology services," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, Sept 2011, pp. 985–989.

[2] L. Toomey. 5 cloud computing statistics you may find surprising. Cloudspectator. [Online]. Available: http://cloudcomputingtopics.com/2011/11/5-cloud-computing-statistics-you-may-find-surprising/

[3] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing, "How is the weather tomorrow?: Towards a benchmark for the cloud," in *Proceedings of the Second International Workshop on Testing Database Systems*, ser. DBTest '09. ACM, 2009. doi: 10.1145/1594156.1594168 pp. 9:1–9:6. [Online]. Available: http://doi.acm.org/10.1145/1594156.1594168

[4] Microsoft. Windows azure. Microsoft. [Online]. Available: http://www.windowsazure.com

[5] Google, "Compute Engine," 2013. [Online]. Available: http://cloud.google.com/pricing/

[6] Amazon, "EC2," 2013. [Online]. Available: http://aws.amazon.com/ec2/

[7] M. Gusev and S. Ristov, "The optimal resource allocation among virtual machines in cloud computing," in *Proceedings of The 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012)*, 2012, pp. 36–42.

[8] M. Gusev, S. Ristov, G. Velkoski, and M. Simjanoska, "Optimal resource allocation to host web services in cloud," in *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, ser. CLOUD '13, USA, June 2013. doi: 10.1109/CLOUD.2013.103 pp. 948–949. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2013.103

[9] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, and H. Zhong, "Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications," in *Cloud Computing (CLOUD), 2012 IEEE 5th Int. Conf. on*, 2012. doi: 10.1109/CLOUD.2012.81 pp. 439–446. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6253536

[10] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An Analysis of Performance Interference Effects in Virtual Environments," in *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*, april 2007, pp. 200 –209.

[11] D. Jayasinghe, S. Malkowski, Q. Wang, J. Li, P. Xiong, and C. Pu, "Variations in performance and scalability when migrating n-tier applications to different clouds," in *Cloud Computing (CLOUD), 2011 IEEE Int. Conf. on*, 2011. doi: 10.1109/CLOUD.2011.43 pp. 73–80. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6008695&isnumber=6008659

[12] R. Iakymchuk, J. Napper, and P. Bientinesi, "Improving high-performance computations on clouds through resource underutilization," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. ACM, 2011. doi: 10.1145/1982185.1982217 pp. 119–126. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982217

[13] E. Roloff, F. Birck, M. Diener, A. Carissimi, and P. Navaux, "Evaluating high performance computing on the windows azure platform," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012. doi: 10.1109/CLOUD.2012.47. ISSN 2159-6182 pp. 803–810. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6253582&isnumber=6253471

[14] V. Subramanian, H. Ma, L. Wang, E.-J. Lee, and P. Chen, "Rapid 3D Seismic Source Inversion Using Windows Azure and Amazon EC2," in *Proceedings of IEEE*, ser. SERVICES '11, 2011, pp. 602–606.

[15] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early observations on the performance of windows azure," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. ACM, 2010. doi: 10.1145/1851476.1851532. ISBN 978-1-60558-942-8 pp. 367–376. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851532

[16] R. Tudoran, A. Costan, G. Antoniu, and L. Bougé, "A performance evaluation of Azure and Nimbus clouds for scientific applications," in *Proc. of the 2nd Int. Workshop on Cloud Computing Platforms*, ser. CloudCP '12. ACM, 2012. doi: 10.1145/2168697.2168701 pp. 4:1–4:6. [Online]. Available: http://doi.acm.org/10.1145/2168697.2168701

[17] A. Turner, A. Fox, J. Payne, and H. Kim, "C-mart: Benchmarking the cloud," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1256–1266, June 2013. doi: 10.1109/TPDS.2012.335. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6381404&isnumber=6507529

[18] "Rubis," 2014. [Online]. Available: http://rubis.ow2.org/

[19] E. Cecchet, V. Udayabhanu, T. Wood, and P. Shenoy, "Benchlab: An open testbed for realistic benchmarking of web applications," in *Proc. of the 2Nd USENIX Conf. on Web Application Development*, ser. WebApps'11, 2011, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002168.2002172

[20] B. Pugh and J. Spacco, "Rubis revisited: Why j2ee benchmarking is hard," in *Companion to the 19th Annual ACM SIGPLAN Conf. on Object-oriented Programming Systems, Languages, and Applications (OOPSLA '04)*. ACM, 2004. doi: 10.1145/1028664.1028751 pp. 204–205. [Online]. Available: http://doi.acm.org/10.1145/1028664.1028751

[21] "Tpc benchmarkw(web commerce) specification," 2002. [Online]. Available: http://www.tpc.org/tpcw/spec/tpcw_v1.8.pdf

[22] Q. Wang, Y. Kanemasa, J. Li, D. Jayasinghe, T. Shimizu, M. Matsubara, M. Kawaba, and C. Pu, "Detecting transient bottlenecks in n-tier applications through fine-grained analysis," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, July 2013. doi: 10.1109/ICDCS.2013.17 pp. 31–40. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6681573&isnumber=6681559

[23] X. Chen, H. Chen, Q. Zheng, W. Wang, and G. Liu, "Characterizing web application performance for maximizing service providers' profits in clouds," in *Cloud and Service Computing (CSC), 2011 International Conference on*, Dec 2011, pp. 191–198.

[24] M. Gusev, S. Ristov, and P. Gushev, "Developing a ticket management saas solution," in *MIPRO, 2014 Proceedings of the 37th International Convention, IEEE Conference Publications*, Croatia, 2014, pp. 328–333.

[25] P. Gushev, A. Guseva, S. Ristov, and M. Gusev, "Cloud solutions for bug reporting," in *XLVIII Int. Scientific Conf. on Information, Comm. and Energy Systems and Technologies*, 2013, pp. 227–230.

[26] W. Iqbal, M. Dailey, and D. Carrera, "Sla-driven dynamic resource management for multi-tier web applications in a cloud," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010. doi: 10.1109/CCGRID.2010.59 pp. 832–837. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5493374&isnumber=5493340

[27] Microsoft. Analyzing load test results and errors in the tables view of the load test analyzer. Microsoft. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms404656.aspx#analyzingloadtestresultserrorstablesviewtherequeststable