Windows Azure: Resource Organization Performance Analysis

Marjan Gusev, Sasko Ristov, Bojana Koteska, Goran Velkoski

Windows Azure: Resource Organization Performance Analysis

Marjan Gusev[1], Sasko Ristov[1], Bojana Koteska[1], and Goran Velkoski[2]

[1] Ss. Cyril and Methodius, Faculty of Computer Science and Engineering,
Rugjer Boskovikj 16, 1000 Skopje, Macedonia
{marjan.gushev,sashko.ristov,bojana.koteska}@finki.ukim.mk
[2] Innovation LTD,
Vostanichka 118, 1000 Skopje, Macedonia
goran.velkoski@innovation.com.mk

Abstract. Cloud customers can scale the resources according to their needs in order to avoid application bottleneck. The scaling can be done in two ways, either by increasing the existing virtual machine instance with additional resources, or by adding an additional virtual machine instance with the same resources. Although it is expected that the costs rise proportionally to scaling, we are interested in finding out which organization offers scaling with better performance. The goal of this paper is to determine the resource organization that produces better performance for the same cost, and help the customers decide if it is better to host a web application on a more "small" instances or less "large" instances. The first hypothesis states that better performance is obtained by using more and smaller instances. The second hypothesis is that the obtained speedup while scaling the resources is smaller than the scaling factor. The results from the provided experiments have not proven any of the hypotheses, meaning that using less, but larger instances results with better performance and that the user gets more performances than expected by scaling the resources.

Keywords: Cloud Computing; Microsoft Azure; Performance; SaaS.

1 Introduction

One of the customers' motivations to migrate their applications onto a cloud is the cost. Currently cloud service providers (CSPs) use a linear "pay-by-the-drink" cost model, where the costs are proportional to the rented resources.

Although the cloud offers a possibility to reduce the costs, we are eager to find a platform that will offer the best performance for the same price. The customers can choose among various possibilities, such as using large or small instances. When the web services are hosted on the cloud, the performance is usually discrepant due to several reasons, such as the additional virtualization layer, cloud multi-tenant and shared resources environment. The overall web service performance depends on the quantity and capacity of rented hardware

resources, platform environment, the number of active virtual machines (VMs) on a physical server, the total number of active VMs in the cloud, and so on.

One of the advantages of the cloud compared to the traditional IT hosting platforms is that cloud is scalable and elastic. In this paper we analyze the performance trade-off to make conclusions about which resource organization of the cloud offers the maximum performance for a given cost.

A typical scenario that initiates a huge dilemma for the customer happens with scaling. Imagine that the customer has migrated the web service onto a cloud and measures the response time of the applications. Increased popularity of the web service might initiate a need for more transactions, and soon the rented resources will not be sufficient to keep the response times low enough to ensure a good quality of service. A typical CSP answer is that the customer should rent more resources, usually expressed with more CPU cores. For example, let the initial configuration be a small single tenant VM with one CPU core and the customer would like to rent 2 CPU cores instead of one. The dilemma is due to the fact that the customer faces several options, such as a medium VM with 2 CPU cores, or two small VMs with 1 CPU core each. The dilemma increases if the customer decides to use 4 CPU cores, or higher number of cores. In this case, the possibilities are even higher, for example, for 4 CPU cores, the options are 1x4, 2x2, and 4x1, corresponding to one large VM, or 2 medium VMs, or 4 small VMs. CSP is using a linear cost model, so all the configurations using a total of 4 CPU cores will approximately cost the same. In this paper we conduct experiments on Azure, as one of the most commonly used commercial clouds.

The research problem is to find out the Azure resource organization that performs the best and scores the highest performance trade-off. In addition, we set a hypothesis that renting more "smaller" VMs for the same cost is better than renting less "greater" instances, hoping that the tasks will be completed faster if we distribute them to more smaller instances. The second hypothesis is that the performance is smaller than the scaling factor. For example, it means that by renting double size resources, we will not obtain double performance. This is set due to virtualization and Gustafson's bounded linear speedup [10].

The paper is organized as follows. Related work in the area of cloud performance is given in Section 2. Section 3 presents the testing methodology, plan and infrastructure. The results from the experiments are described in Section 4. Section 5 is dedicated to a discussion of the outcome and performance trade-off, comparison of the results and analysis which environment provides the best performance. The conclusion and future work are specified in Section 6.

2 Related Work

The performance of various cloud applications and services is analyzed by many authors. For example, Brebner and Liu conducted empirical evaluations of different cloud infrastructures using a suite of cloud testing applications [2]. They also use those experimental evaluations to predict the resource requirements in terms of application performance, cost and limitations of a realistic application

for different deployment scenarios. Gao et al. proposed formal graphic models and metrics in order to preform SaaS evaluation and analyze system scalability in clouds [3]. Their case study is Amazon's EC2 cloud technology.

Several papers analyze the performances on Azure. Hill et al. [11] present the results of the performance experiments conducted on Azure. They present a detailed performance evaluation and give some recommendations for Azure users. Scaling as performance measure was analyzed by Mao et al. [14]. They present a cloud auto-scaling mechanism which scale computing instances automatically based on workload information and performance desire. They have also implemented their mechanism in Azure platform and made evaluations of simulations and real scientific cloud application.

The performance trade-off was also analyzed in the literature. A comparison between the performance and monetary cost-benefits of clouds for desktop grid application was reported by Kondo et al. [12]. They conducted performance measurements and monetary expenses of real desktop grids and the Amazon elastic compute cloud. Ostermann et al. evaluated the usefulness of the cloud computing services for scientific computing [16]. They found that current cloud services need performance improvement in order to be used in scientific community.

Other performance aspects were also analyzed, such as storage services, data transfer etc. For example, Agarwal and Prasad describe a benchmark suite for the storage services of Azure platform called AzureBench [1]. Tudorian et al. concluded that Azure can support the efficient TCP data transfers and it can decrease the costs and time for deployment [17]. Several pitfalls in the Azure cloud are examined during several days of performing the experiments: Instance physical failure, Storage exception, System update [13]. The authors also discovered several pitfalls resulting in waste of active VM idling.

Recently, Gusev and Ristov [8] have reported that it is better to use many smaller VM instances for a web service hosted in the cloud. They achieved similar result for parallel implementation of cache intensive matrix multiplication algorithm [7], i.e., maximum speedup is obtained if matrices are scattered and multiplied on 8 concurrent (XS) VMs using a single thread in a VM, rather than using OpenMP with 8 threads in a single XL VM. This was the initial motivation to define the hypothesis in this paper and to check if this holds for a real web service that includes transactions in a 3-tier architecture, using a database server, a web server and an application server.

In this paper, we measure the performance of the 3-tier cloud SaaS application hosted on Azure. It acts as a transaction based application, where a database transaction is started and then the results are transferred back to the customer. The application is loaded with different number of requests, while it is being hosted in different number of instances with different resources.

3 Testing Methodology

This section describes the testing environments of Azure cloud, test cases and design implementation.

3.1 Test Goal

We are trying to determine which organization of the resources in the Azure cloud will provide the best performance for the most common 3-tier application using Web, Application and Database Servers.

The idea is to conduct three experiments. The first is about analyzing large (L) and small (S) configurations for the same amount of resources. By large configuration, we define a configuration that uses a small number of large VMs with greater number of CPUs. Correspondingly, a small configuration will be the one that uses more small VMs, where each VM consist of 1 or 2 CPUs. The results would give answer to the first hypothesis.

The next two experiments should provide an answer for the second hypothesis, whether the scaling of CPUs or the number of VMs will benefit with proportional performance. These experiments will also confirm which is better, to increase the number of CPUs or number of VMs when scaling is desired.

Let the number of VMs be v and the number of CPU cores c. Then the total number of CPU cores n used in a given configuration will be $n = v \cdot c$.

To answer the research questions and confirm validity of both hypotheses we conduct three experiments. Experiment 1 provides the tests with different configurations by keeping the same total number of CPU cores (n), as defined by (1); Experiment 2 provides the tests with different configurations by keeping the same number of VMs and scaling the number of CPU cores in each VM, as defined by (2); and Experiment 3 provides the tests with different configurations by keeping the same number of CPUs in each VM and scaling the number of VMs, as defined by (3).

$$\text{change } v, c \qquad n = const \tag{1}$$

$$\text{scale } c \text{ (and thus } n), \qquad v = const \tag{2}$$

$$\text{scale } v \text{ (and thus } n), \qquad c = const \tag{3}$$

3.2 Cloud Testing Environment

The testing environment is a client-server environment hosted on Windows Azure. The server side consists of the SaaS application "PhluffyFotos" [15], as a sample cloud application developed for public use under the Microsoft Public License (Ms-PL). The application uses the following technologies: ASP.NET MVC 4, Azure SQL Databases and Azure Storage, including Tables, Blobs, and Queues. This application is used because it frequently interacts with cloud storage services [18].

The PhluffyFotos is a Picture Gallery Service which can be accessed by web or mobile device. Users can create new albums, upload photos and share their photo albums. They must register and login in order to perform these actions. Unregistered and unlogged users can see all albums and search photos by tag, but

they neither can create their own albums nor upload photos. There are database records for 100 albums, each containing one picture.

This web system is programmed specially for the Azure platform, which means that it is designed to be scaled by using different number of VMs and CPUs offered by the cloud service. The images and their meta-data are sent over to Windows Azure for processing. The information stored in Azure Storage Queues is picked up and processed by the the Cloud Service. After that, it is stored in Azure Table Storage. The content of the image is stored in binary blobs using Azure Blob Storage. Multiple user profiles can be stored in a Azure SQL Database and accessed by using the Universal Profile Providers. Everything image-centric is stored in Azure Storage, once it has been processed via the Cloud Service [4].

We created a storage, web site, cloud service and 1 GB SQL database. Another storage was created when the project was published. Also, the worker role was added in the cloud service. Each storage contains three services: Blobs, Queues and Tables.

The client uses Apache JMeter to test the performance by varying the load (the number of requests) and using different number of instances with various number of resources. To minimize the network latency, both the client and the cloud are placed in the same data center (the West Europe's center).

In order to perform the load testing we add a Thread Group element to configure the number of clients that will send HTTP requests. Each user sends one HTTP request. HTTP requests of all clients are processed in parallel. The Rump-up period value is set to 0 seconds in order to immediately start all clients' requests.

The experiments consist of accessing a web page which displays the albums created by other users. The web page retrieval includes reading information from the SQL database that is common for all application instances, and can be assumed as an IO intensive operation, like most of 3-tier web applications.

3.3 Test Cases

Each test case is denoted with v x c, expressing the corresponding number of VMs and CPU cores per VM in each configuration.

We tested all possible configurations, where the total number of VMs is less or equal to 10, and the number of CPUs per VM is at most 4, explicitly expressed with $1 \leq v \leq 10$ and $1 \leq c \leq 4$.

In addition we have also tested several configurations that are beyond this threshold, required to realize necessary comparisons for the scaling experiments. The list of all test cases covers at least the following configurations: 1x1, 1x2, 1x4, 2x1, 2x2, 2x4, 3x1, 3x2, 3x4, 4x1, 4x2, 4x4, 5x1, 5x2, 5x4, 6x1, 6x2, 6x4, 8x1, 8x2, 8x4, 10x1, 10x2, 10x4.

Each test case is executed at least 5 times with different number of HTTP requests N = 1, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950 and 1000. JMeter deals with the server load until 1000 requests are complete, since it reports an error message for increased load.

For each test case, we measure the average response time when all requests will be completed.

Our goal is to obtain reliable tests in each cloud environment by changing the number of application instances and CPU cores. The Azure load balancer decides how to serve the concurrent HTTP requests.

Each experiment realizes two different cloud configurations denoted as L and S testing various total numbers of CPU cores. For example, a part of the first experiment assumes that the application will serve the HTTP requests with such a capacity that will require a configuration with a total of 16 CPU cores. Fig. 1 (left) depicts the test case defined as large configuration, denoted by L, where 4 instances of the SaaS application are hosted on 4 VMs, each allocated with 4 CPU cores. Fig. 1 (right) presents the test case defined as small configuration, which is denoted by S. It is a cloud environment where 8 instances of the SaaS application are hosted on 8 VMs, each allocated with 2 CPU cores.

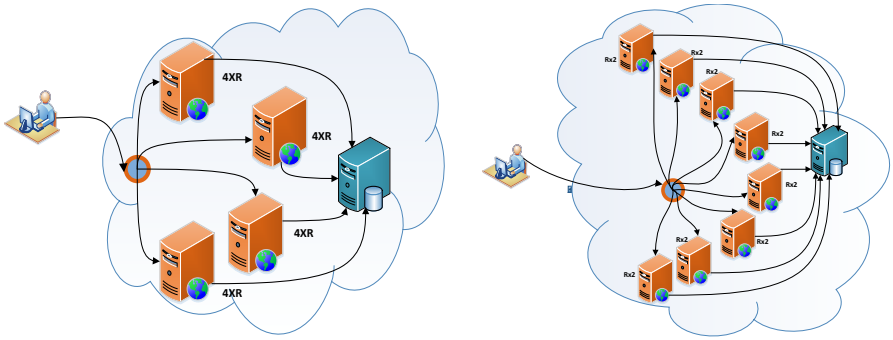

Fig. 1. 4x4 (left) and 8x2 (right) environments

Now, the dilemma a customer might have, is to decide what is better, a configuration in Fig. 1 (left) or in Fig. 1 (right). Our hypothesis assumes that the configuration with a bigger number of small VMs (Fig. 1 right) performs better for the same price.

3.4 Test Data

The performance of the SaaS application is calculated by measuring the average response time T for experiments. Denote by T_{vc} the measured average response time in each test case using v instances of the SaaS application hosted on v VMs, each with c CPU cores.

In the evaluation of results we usually compare two configurations denoted by indexes L and S. We calculate the *Relative Speedup* as ratio of average times measured for S configuration over L by relation $R = R(v_S, c_S, v_L, c_L) = T_{v_S c_S}/T_{v_L c_L}$.

Value $R > 1$ will mean that response times of the L configuration are smaller than those of the S configuration, and we can conclude that the L configuration is R times better than S.

In Experiment 1 we realize test cases where (1) is satisfied, keeping the total number of CPUs n in analyzed pair of configurations the same. Our first hypothesis in this case assumes that we will obtain $R < 1$.

Let us compare two experiments and find the scaling factor. Denote the number of required CPUs in the configuration by n_L and n_S corresponding to the configurations L and S. In order to analyze the scaling relation between these two experiments there should be a positive integer number m, such that $n_L = m \cdot n_S$ is valid. In this case, m is the scaling factor.

Scaling speedup S is the ratio of obtained relative speedup R and scaling factor m when two configurations L and S are compared, that is, $S = R/m$. It means that the scaling speedup S will give information how much the L configuration performs better than S with factor compared to the scaling factor.

Suppose that the L configuration uses m times more resources than the S configuration. One will expect that the performance will also scale with at most the same factor. A value of scaling speedup $S > 1$ will mean that the performance is at least m times better, meaning that it is worth enough to scale. Hypothesis 2 assumes that $S < 1$.

4 Analysis of Experimental Results

In this section, we present and analyze the experimental results of the three conducted SaaS performance experiments. We have measured response times for each configuration and in the next sections we will elaborate the relative speedup and scaling speedup, along with explanation of obtained results.

4.1 Experiment 1: Same total number of CPUs n

For this experiment we define L configuration to be the large configuration which uses less number of large VMs, each with large number of CPUs. The S configuration will be the small configuration using more small VMs, each with small number of CPUs. We assume that (1) is valid, keeping the same total number of CPUs. The formal definition of criterion for this experiment is given in (4).

$$n = v_L \cdot c_L = v_S \cdot c_S, \quad v_L < v_S, \quad \text{and} \quad c_L > c_S \tag{4}$$

Table 1 presents the test cases and their identification as large and small configurations. For example, the first comparison assumes using a total of 4 CPUs, and the L configuration is a single VM environment with 1 application instance hosted on a VM with 4 CPU cores, while the S configuration is a multi VM environment with 4 application instances hosted on 4 small VMs, each with 1 CPU.

We have provided analysis on the following relative speedups $R_4 = R(4, 1, 1, 4)$; $R_6 = R(6, 1, 3, 2)$; $R_8 = R(8, 1, 2, 4)$; $R_{10} = R(10, 1, 5, 2)$; $R_{12} = R(6, 2, 3, 4)$;

Table 1. L and S configurations for Experiment 1

CPUs n	4	6	8	10	12	16	20
L configuration v_L x c_L	1x4	3x2	2x4	5x2	3x4	4x4	5x4
S configuration v_S x c_S	4x1	6x1	8x1	10x1	6x2	8x2	10x2

$R_{16} = R(8, 2, 4, 4)$; and $R_{20} = R(10, 2, 5, 4)$, which correspond to pairs of L and S configurations in Table 1. The index denotes the total number of used CPU cores for the analyzed configurations.

The performance of the SaaS, measured for a total of 4 CPU cores is presented in Fig. 2 (left). The response time is presented on the Y axis, measured in seconds. The input parameter, which is the number of requests sent to the server is presented on the X axis. Blue bars present the results for large configurations, and red for small configurations. For example, in L configuration, the time to complete 1000 requests is 8.916 seconds, while in the S configuration is 10.401 seconds. We conclude that for all tests, L configuration performs better. The differences rise when the number of concurrent HTTP requests rises.

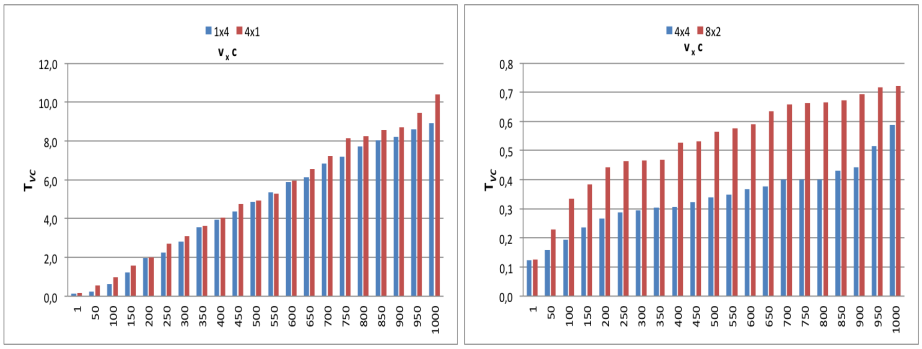

Fig. 2. Response time for test cases with a total of 4 (left) and 16 CPU cores (right)

The behavior of the response time is similar for all the cases. For example, the results for test cases, which use a total of 16 CPU cores, are presented in Fig. 2 (right). In all tests the L configurations perform better, and in this case the differences are bigger than in previous example.

Fig. 3 depicts the relative speedup for all experiments. We can clearly observe that the relative speedup is always greater then 1, for all experiments, meaning that the large L configurations perform better than the small S configurations. Therefore, our first hypothesis, does not hold, and renting less more powerful VMs is better than renting more smaller VMs in Azure.

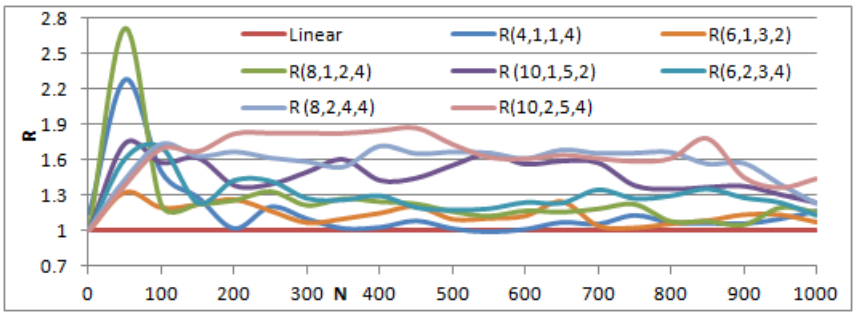

Fig. 3. Relative Speedup between large and small configurations for all experiments.

4.2 Experiment 2: Scaling the number of CPU cores *c*

In the previous section we concluded that L configurations perform better than their corresponding S configurations, expressing the cases with same number of CPU cores, as defined by (1). In this section we will analyze what happens when the number of CPUs is increased by a scaling factor $m > 1$ keeping the number of VMs the same, as expressed in (2).

Scaling c from 1 to 2. The first part of the experiment consist of analysis of the case when the number of VMs v is constant and the number of CPU cores c is doubled from 1 CPU core to 2 CPU cores, or choosing the scaling factor to be $m = 2$. We assume that each application instance runs on a separate VM and the customer decided to use more powerful VMs upgrading the number of CPUs in each VM.

Table 2 presents the L and S configurations. For example, the first test case assumes using S configuration with a total of 3 CPU cores, and the L configuration with 6 CPU cores. Both configurations are multi VM environment with 3 application instances hosted on 3 VMs, where the S configuration specifies a VM with 1 CPU and the L configuration a VM with 2 CPUs.

Table 2. L and S configurations for Experiment 2a: CPU cores doubled from 1 to 2

CPU increase $n_S \rightarrow n_L$	$3 \rightarrow 6$	$4 \rightarrow 8$	$5 \rightarrow 10$	$6 \rightarrow 12$	$8 \rightarrow 16$	$10 \rightarrow 20$
S configuration v_S x c_S	3x1	4x1	5x1	6x1	8x1	10x1
L configuration v_L x c_L	3x2	4x2	5x2	6x2	8x2	10x2

Fig. 4 (left) presents the scaling speedup for all test cases defined in Table 2 comparing configurations when the number of CPU cores is doubled from 1 to 2 CPU cores per each VM. The results show that it is worthwhile to upgrade from 1 to 2 CPUs and the performances will mostly scale more than the scaling factor. There is a slight deviation of general behavior when comparing the 3x1 and 3x2 configurations, which is mostly due to unaligned usage of processing power in the

CPU. Most of the scaling in the CPU is aligned to 2 or 4 cores and when 3 cores are used, slight performance deviations may appear due to task scheduling in the Azure's balancer and current load distribution of the distributed environment.

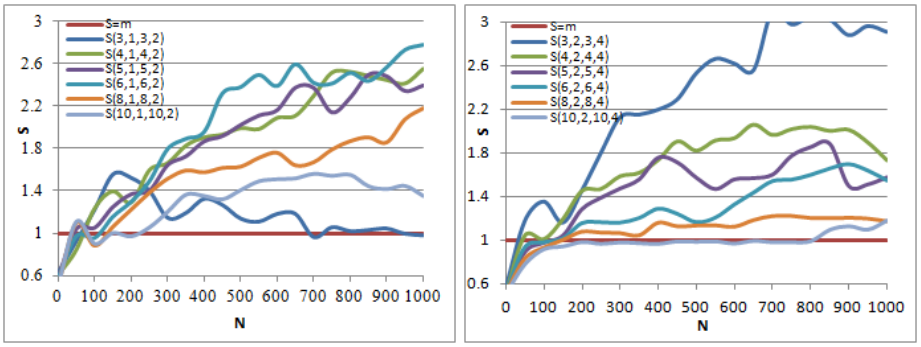

Fig. 4. Scaling speedup when c is doubled from 1 to 2 (left) and 2 to 4 (right)

The average behavior shows that an upgrade of a VM from 1 to 2 CPUs makes an impact of approximately 94,5% more than the scaling factor (in this case the scaling factor is 2, the average speedup 3.89 and the average scaling speedup 1.945). We can notice that higher speedup is reached by increasing the workload, and the obtained impact stabilizes for $N > 200$.

Scaling c from 2 to 4. The second part of this experiment analyzes the case when the number of CPU cores is doubled from 2 to 4 per VM. Test cases for L and S configurations are presented in Table 3. For example, the first test case defines the S configuration 3x2 with a total of 6 CPU cores, and the L configuration 3x4 with 12 CPU cores, doubling c from 2 to 4.

Table 3. L and S configurations for Experiment 2b: CPU cores doubled from 2 to 4

CPU increase $n_S \rightarrow n_L$	6 → 12	8 → 16	10 → 20	12 → 24	16 → 32	20 → 40
S configuration v_S x c_S	3x2	4x3	5x2	6x2	8x2	10x2
L configuration v_L x c_L	3x4	4x4	5x4	6x4	8x4	10x4

The results are presented in Fig. 4 (right). Once again we observe it is worth to upgrade the VMs from 2 to 4 CPUs. The impact is similar to the previous case with upgrade from 1 to 2 CPUs. An average of 69,2% better performance than the scaling factor is obtained.

The results show that the same trend is also observed for these cases. The workload has impact on the overall performance. Note that the impact factor is smaller for configuration using greater number of CPUs, and it can also be

observed for increased workload. This happens due to the processing power of measured configurations. For example, the 10x2 configuration is quite powerful in handling the workload, and the 10x4 configuration achieves a scaling speedup equal to the scaling factor 2 for almost complete domain of testing when $N \leq 800$. The analyzed trend starts to rise for heavier workload when $N > 800$. In all other cases the achieved speedup is much higher than the scaling factor of 2.

We have not analyzed the cases when the number of CPU cores is upgraded from 1 to 4, since the value can be easily calculated as a multiple of the two previous values. The average impact in this case is even higher than 2.5 times more than the scaling factor (in this case 4).

The conclusion from the experiments 2a and 2b is that the scaling with upgrade of the CPU cores in a VM will impact more than the scaling factor, contrary to our hypothesis 2. Next we analyze the scaling speedup by scaling v.

4.3 Experiment 3: Scaling the number of VMs v

The case when (2) is satisfied for scaling with upgrade of the number of CPUs per VM was discussed in the previous section. Here we analyze the scaling when the number of CPUs per VM c is fixed and scaling is obtained by increasing the number of VMs, as specified in (3). We conduct three parts of the experiments, each part defined with different number of CPUs.

Scaling v with 1 CPU per VM. The first part of the experiment consists of analysis of the case when doubling v, with 1 CPU per VM. We assume that each application instance runs on a separate VM and the customer decided to use more VMs instead of upgrading the number of CPUs in each VM.

Table 4 presents the L and S configurations for the experiment. For example, the first test case assumes using S configuration with a total of 3 CPU cores, and the L configuration with 6 CPU cores. Both configurations are multi VM environment where a VM is defined with 1 CPU, and the S configuration with 3 application instances hosted on 3 VMs, and the L configuration with 6 VMs.

Table 4. L and S configurations for Experiment 3a: v is doubled, $c = 1$

CPU increase ($n_S \rightarrow n_L$)	S configuration (v_S x c_S)	L configuration (v_L x c_L)
$3 \rightarrow 6$	3x1	6x1
$4 \rightarrow 8$	4x1	8x1
$5 \rightarrow 10$	5x1	10x1

Fig. 5 (left) presents the scaling factors comparing the configuration when the number of VMs is doubled and each VM has 1 CPU core. A deviation from the trend exposed in all analyzed cases is observed for the configuration 3x1. We have previously discussed that this deviation happens due to alignment of processing power in real CPUs, since most of the configurations demand 2 or 4 CPUs and it depends on current cloud workload. In all other cases the upgrade by doubling the number of VMs makes an impact with average value of 32,2%

more than the scaling factor, so it is worth to upgrade. Even in the case where a deviation is observed we measure the achieved speedup to be equal to the scaling factor as average behavior.

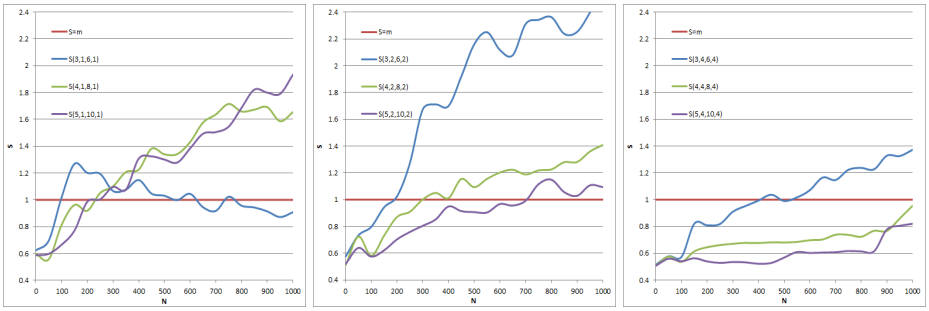

Fig. 5. Scaling speedup with doubled number of VMs v, each with 1 (left), 2 (middle), and 4 CPU cores (right)

Scaling v with 2 CPUs per VM. The second part of the experiment analyses the case with $c = 2$ CPUs per VM and the upgrade is done by doubling the number of VMs v. The test cases are presented in Table 5.

Table 5. L and S configurations for Experiment 3b: v is doubled, $c = 2$

CPU increase ($n_S \rightarrow n_L$)	S configuration (v_S x c_S)	L configuration (v_L x c_L)
$6 \rightarrow 12$	3x2	6x2
$8 \rightarrow 16$	4x2	8x2
$10 \rightarrow 20$	5x2	10x2

The results of the second part of Experiment 3 are shown in Fig. 5 (middle). Also in this case the upgrade makes a good impact better than the scaling factor, which is slightly better than in the previous case with the configuration using VMs with 1 CPU. The rising trend depends on the workload. Higher number of concurrent messages will show better performance of L configurations, rather than the S configurations. We can also observe that, for example, the configuration 5x2 can handle a sufficient number of concurrent messages, and the 10x2 configuration can achieve speedup more than the scaling factor only for heavier workload, such as for $N > 700$.

Scaling v with 4 CPUs per VM. The last part of Experiment 3 analyses powerful VMs with $c = 4$ CPUs each and the scaling is done by doubling the number of VMs v. Table 6 presents the L and S configurations.

Fig. 5 (right) presents the scaling speedup comparing the configurations of Table 6. In this case the impact is lower than in the previous cases, which is

Table 6. L and S configurations for Experiment 3c: v is doubled, $c = 4$

CPU increase ($n_S \rightarrow n_L$)	S configuration (v_S x c_S)	L configuration (v_L x c_L)
$12 \rightarrow 16$	3x4	6x4
$16 \rightarrow 32$	4x4	8x4
$20 \rightarrow 40$	5x4	10x4

due to the reasons explained previously that the configurations are capable to handle large number of concurrent messages. The speedup is obvious (more than 0.5 means it is positive but still under the value of scaling factor 2). Desired speedup greater than the scaling factor happens for heavier workload, when, for example, the configuration 10x4 can have greater impact over 5x4.

As a conclusion of analysis in Experiment 3, we realize that the second hypothesis is also disproved as in Experiment 2. We obtain better performance when scaling the number of VMs keeping fixed number of CPU cores.

5 Discussion

Experiment 1 showed that the first hypothesis is not valid and Azure performs better with large sized VMs with 4 CPUs. Both experiments 2 and 3 covered all cases to show that the second hypothesis is also disproved, meaning that by scaling the resources, a customer gets more performance than the scaling factor.

We can also conclude from experiments 2 and 3 that it is better to scale in such a way to use more powerful VMs instead of using more VMs, a fact that also is shown by the Experiment 1 for comparing different configurations with same number of CPUs. So we have concluded that the linear cost model is not unfair to the customer, once the customer needs more power, the solution by renting more resources achieves performance higher than the scaling factor.

The results of Experiment 1 oppose the findings [8] where the test cases are performed for compute intensive and memory demanding web services instead of a transactional web service. Gusev et al. [9] also achieved opposite results for transactional web services without using worker role on Azure. The authors believe that a great role in this behavior is mostly due to the Azure's balancer and organization of the system using a predefined database. The results show that the balancer for transactional web services using databases should be improved because internal VM web server's task scheduler handles the load much better.

We have observed two side effects. The first one addresses the alignment to the number of CPU cores. This is highly dependent on the CSP availability and the current cloud workload. The alignment of needs is mostly to the 2 or 4 CPU cores per active CPU, meaning that the requests for 3 CPU cores are rare. In this case depending on the availability, the CSP may schedule 3 CPU cores in one CPU or several CPUs, which can change the obtained performance. We have provided a lot of experiments and the average performance (with very small discrepancy among each repeated test case) is presented in this paper. We have

concluded that with this configuration there is a deviation in trend behavior which was observed for other test cases.

The second side effect was the capability to handle the number of requests. More powerful configurations (with higher number of CPUs) were capable to provide a good throughput and processing speed. Scaling the resources for these cases will produce speedup equal to the scaling factor. A speedup bigger than the scaling factor (superlinear) is observed for heavier demands and number of requests. A superlinear speedup is a well known phenomenon achieved in distributed environment for cache intensive algorithms [6], where more CPU cache memory is used in parallel implementations with low inter-CPU communication, despite the Azure's virtualization layer [5]. Obviously, the superlinearity in this distributed environment is totally different. We believe that increasing the incoming requests over-utilizes smaller VMs faster than a greater VM.

6 Conclusion and Future Work

In this paper we have conducted three experiments to conclude about the performance trade-off for transactional web services on Azure cloud.

We concluded that when a customer wants to scale the existing configuration it is better to choose a configuration which is using less number of larger instances, i.e. to choose VMs with 4 CPUs if possible. Also we have concluded that by scaling the resources, a customer usually gets more performance than the expected scaling factor, i.e. if there is an upgrade which doubles the number of CPU cores in the configuration, then the achieved performance is more than double. In this case, a customer should choose configurations that use powerful instances with 4 CPU cores.

These results will solve the customer's dilemma to choose the most optimal configuration that performs the best. For example, if the customer is using a 4x4 configuration with total of 16 CPUs and would like to upgrade to total of 20 CPU core, should chose 5x4 having advantage over the 10x2 or 20x1 configurations.

Although most offers follow the linear pricing model we have observed that the number of CPU cores is not the only parameter that a customer should analyze. A customer can choose among great variety of available RAM and storage, or throughput etc. Although there are a lot of available online calculators for this purpose, we plan to make deeper analysis of impact of these factors on the overall performance.

We will continue to realize the same and similar experiments on different environments and clouds.

References

1. Agarwal, D., Prasad, S.K.: AzureBench: Benchmarking the storage services of the Azure cloud platform. In: Proc. of the IEEE 26th Int. Parallel and Distributed Processing Symp. Workshops & PhD Forum. pp. 1048–1057. IPDPSW '12 (2012)

2. Brebner, P., Liu, A.: Performance and cost assessment of cloud services. In: Proc. of the 2010 Int. conf. on Service-oriented computing. pp. 39–50. ICSOC'10, Springer-Verlag, Berlin, Heidelberg (2011)
3. Gao, J., Pattabhiraman, P., Bai, X., Tsai, W.: SaaS performance and scalability evaluation in clouds. In: Service Oriented System Engineering (SOSE), 2011 IEEE 6th International Symposium on. pp. 61–71 (2011)
4. Gaster, B.: PhluffyFotos on Windows Azure (October 2012), http://www.bradygaster.com/post/phluffyfotos-on-windows-azure
5. Gusev, M., Ristov, S.: Superlinear speedup in Windows Azure cloud. In: Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on. pp. 173–175. Paris, France (2012)
6. Gusev, M., Ristov, S.: A superlinear speedup region for matrix multiplication. Concurrency and Computation: Practice and Experience (2013), http://dx.doi.org/10.1002/cpe.3102
7. Gusev, M., Ristov, S.: Resource scaling performance for cache intensive algorithms in Windows Azure. In: Zavoral, F., Jung, J.J., Badica, C. (eds.) Intelligent Distributed Computing VII, SCI, vol. 511, pp. 77–86. Springer Int. Publishing (2014)
8. Gusev, M., Ristov, S., Velkoski, G., Simjanoska, M.: Optimal resource allocation to host web services in cloud. In: Proceedings of the 2013 IEEE 6th International Conference on Cloud Computing. pp. 948–949. CLOUD '13, CA, USA (June 2013)
9. Gusev, P., Ristov, S., Gusev, M.: Performance analysis of SaaS ticket management systems. In: Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, (SCoDiS-LaSCoG'14 Workshop). p. in press (Sep 2014)
10. Gustafson, J.L.: Reevaluating Amdahl's law. Communication of ACM 31(5), 532–533 (May 1988)
11. Hill, Z., Li, J., Mao, M., Ruiz-Alvarez, A., Humphrey, M.: Early observations on the performance of Windows Azure. In: Proc. of the 19th ACM International Symposium on High Performance Distributed Computing. pp. 367–376. HPDC '10 (2010)
12. Kondo, D., Javadi, B., Malecot, P., Cappello, F., Anderson, D.: Cost-benefit analysis of cloud computing versus desktop grids. In: Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. pp. 1–12 (2009)
13. Lu, W., Jackson, J., Ekanayake, J., Barga, R.S., Araujo, N.: Performing large science experiments on Azure: Pitfalls and solutions. In: CloudCom'10. pp. 209–217 (2010)
14. Mao, M., Li, J., Humphrey, M.: Cloud auto-scaling with deadline and budget constraints. In: Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on. pp. 41–48 (2010)
15. Microsoft: Picture gallery service (Apr 2008), http://phluffyfotos.codeplex.com/
16. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: A performance analysis of EC2 cloud computing services for scientific computing. In: Avresky, D., et al. (eds.) Cloud Computing, LNICST, vol. 34, pp. 115–131. Springer Berlin Heidelberg (2010)
17. Tudoran, R., Costan, A., Antoniu, G., Bougé, L.: A performance evaluation of Azure and Nimbus clouds for scientific applications. In: Proc. of the 2nd Int. Workshop on Cloud Computing Platforms. pp. 4:1–4:6. CloudCP '12, ACM (2012)
18. Zhang, L., Ma, X., Lu, J., Xie, T., Tillmann, N., de Halleux, P.: Environmental modeling for automated cloud application testing. Software, IEEE 29(2), 30–35 (March 2012)