Contradistinction to the Standard Web Approach with Cloud Computing

Danko Naydenov
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: sky at eurorisksystems dot com

Abstract: *Every new technology has its supporters and its opponents. The purpose of this paper is to impartially consider two alternative approaches to solve the same task, one is the traditional WEB approach and the other is contemporary and modern approach – cloud computing.*

Keywords: *Cloud computing, WEB programming, Stress testing*

1. INTRODUCTION

When creating a new application, it is very important and at the same time very hard to answer the question: what technology to use?
Using traditional and proven technologies on the one side can ensure high reliability and experience. But the area of computing is very rapidly growing and constantly appearing new tools and technologies. If they are not used and applied in practice, certain applications may become uncompetitive capable.
But on the other side it can by show many examples of developed and promising technology, which however, have not won many supporters or users, and left in countries of the modern trends and commonly used technologies.
On the agenda is the question of whether the new and modern technology called cloud computing will be able to replace the standard, well-proven and well-known WEB technology [4].

2. OBJECTIVES AND TASKS

To be evaluated both approaches are planning an experiment to compare the specific features of both platforms. To achieve this goal, following tasks are formulated:

- develop an application that could operate in both environments;
- carry out a series of tests to demonstrate the operability of this application running on both platforms;
- make a comparison and analysis of the obtained results;

3. ANALYSIS OF THE PROBLEM

Cloud providers offer a lot of services that can be used. From the perspective of software developers, the most appropriate is Platform as a Service (PaaS). The reason for this is the development of PaaS application is largely identical with the development of a standard WEB application. Choosing this cloud service will give the ability to develop an application that can be deployed on both platforms.
For cloud provider service is selected Google and their PaaS – Google App Engine [6]. The reason is that access is available free of charge, for the moment is not limited in time, but there are some limitations in resources and opportunities to configure. Another reason is the fact that one of the supported languages for the Google App Engine is Java. It is widely used language also for developing standard WEB applications.

4. THE TEST APPLICATION

The application, which was chosen to be realized is described in a previous paper [1] and its task is to recover the missing values in the time series.
For the purposes of the experiment the application has a simple interface. It is fed as input file, in which the time series are present. The file format is widely used and common format for spreadsheets-Excel. The first line specifies the names of each of the time series, and the first column respectively is the numbers of the reports. If for some reason an account is not made, it is leaved a blank cell as an indication of a missing value.
The application reads the supplied file and applies the algorithm [1] to recover the missing values. The result is displayed on the screen by a table specifying the number of the report, the name of the magnitude and the restored value.
Due to the fact that Google App Engine does not support the web archive [5], to deploy the cloud application it is loads all the war directory and for the command WEB server the archive is deployed. In practice, for both platforms the same application has been used.

5. FORMULATION OF THE EXPERIMENT

Experimental topology is shown in the following diagram:

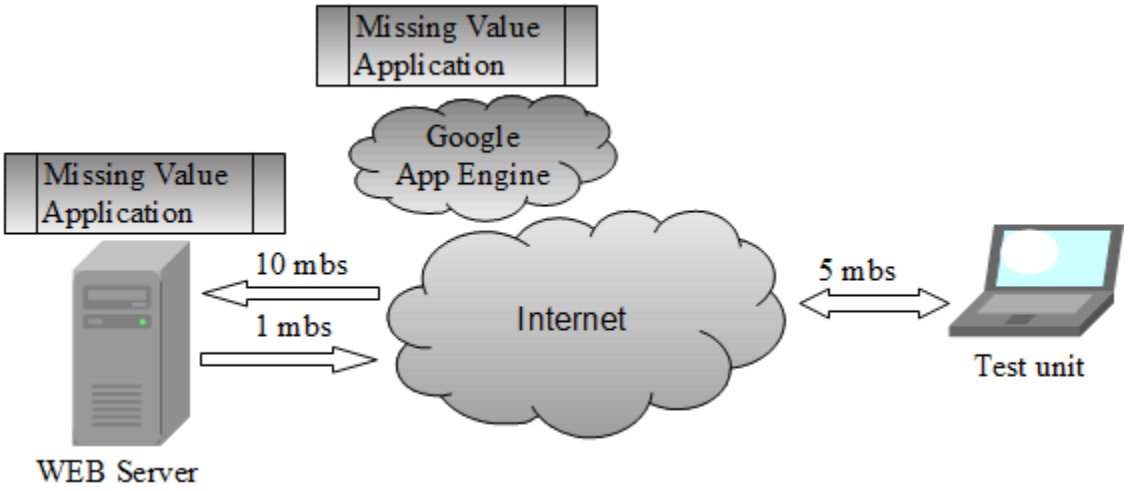

Fig. 1. Experimental topology

The test sequences are done by a computer with a symmetrical speed to and from the Internet— 5mbs. For the purposes of the experiment characteristics of the machine from which the tests are perform is not essential.

For the WEB server was defined architecture with the following parameters:

- operating system: Windows 7 32bit;
- RAM: 3GB;
- CPU: Intel Pentium Dual core CPU T3400 @ 2.16GHz
- download speed: 10mbs
- upload speed: 1mbs
- WEB server software: Jetty-6.1.1

A great detail on the exact architecture and speed to and from the Internet to Google App Engine cannot be given. The reason for this is that, in practice, the cloud architecture is a virtual and dynamic. This means that parameters can be changed. For the experiment two basic configurations of architecture were used, respectively, the minimum and maximum:

- F1: CPU speed - 0.6GHz, RAM - 128MB
- F4: CPU speed - 2.4GHz, RAM - 512MB

It should be noted that these cloud resources are only for the application, which means that unlike the WEB server where there are three gigabytes of RAM, which is shared by the operating system, the other running applications and the test application, in cloud architecture the whole resource is only dedicated for application.

In addition to the specific virtual hardware parameters of the free access plan it has some additional restrictions:

	Free quota per day	Price if the quota is exceeded
Deploying an application	Free	Free
CPU time	28 hours per day	0.08$ for each additional hour
Datastore	1GB	0.24$ per GB per month
Outgoing	1GB	0.12$ per GB per day
Incoming	1GB	free

Table 1.

Some data in the table may look intimidating and therefore will be more fully explained.
A day consists of 24 hours, but the free plan gives 28 free hours of CPU time per day. This is because of two reasons:

- it is time for the F1 plan. If you tune the architecture of F2 (1.2 GHz/256 MB) this time is reduced twice, and for F4 – four times;
- even with F1 plan it is possible more than one request at a time to be sent to application. In this case there will be for example two parallel running requests, with each of them separately from the other consumes CPU time or total consumed time is twice as much as CPU time for work;

For free plan incoming traffic is limited to one gigabyte, but if you do subscribe to a paid plan, only then the incoming traffic is unlimited.

6. CONDUCTING THE EXPERIMENT

To conduct the experiment, it is used specialized software. This is Apache Jmeter version 2.8[3]. Jmeter is an application written in Java open source designed to test the performance of different servers: HTTP, HTTPS, SOAP, Database via JDBC, LDAP and more. It has a very simple and intuitive interface to work. With the help of this software it is possible to simulate multiple simultaneous queries. For the experiment carried out a series of ten successive requests, as the results are average, and parallel requests consistently change respectively in each test one, two, four and eight.

The performance test will be done using several configurations:

- on the WEB server is located both the application itself and the testing program. The goal is to make a control measurement of performance by eliminating the influence of network communication;
- the WEB server is accessed over the Internet from the test computer;

- since Google offer an SDK that locally emulates the work of cloud PaaS service, two more additional tests will be performed corresponding to the previous two tests using that tool;
- performance testing of the configuration of F1 Google App Engine;
- performance testing of the configuration of F4 Google App Engine;
- stress test to F4;

7. SERVER LOCATION

The WEB server and the test machine are geographically located in Varna. Information received from www.iplocation.net page for locating IP addresses for Google App Engine is as follows:

IP Address	Country	Region	City	ISP
173.194.69.141	United States	California	Mountain View	Google Inc.

Table 2.

The response time (ping) to WEB server is:
Reply from 79.100.53.82: bytes=32 time=36ms TTL=244
The response time (ping) to Google App Engine server is:
Reply from 173.194.69.141: bytes=32 time=55ms TTL=40
The greater second time is explained by the greater distance to the second server.

8. RESULTS OF THE EXPERIMENT

The first set of tests was performed on the WEB server:

number of concurrent requests	1	2	4	8
local response time [ms]	592	624	1126	2159
standard deviation [ms]	22	23	149	697
response time for remote requests [ms]	2234	3448	5468	10367
standard deviation [ms]	69	410	807	2884

Table 3.

Results of testing the local instance of the Google SDK are:

number of concurrent requests	1	2	4	8
local response time [ms]	2711	3592	7196	14530
standard deviation [ms]	40	54	461	1276
response time for remote requests [ms]	4342	6515	9474	15890
standard deviation [ms]	206	368	1776	5614

Table 4.

Results when testing the application on Google App Engine are:

number of concurrent requests	1	2	4	8
response time of the F1 instance [ms]	2004	2430	2485	3092
standard deviation [ms]	174	359	638	1379
response time of the F4 instance [ms]	1355	1797	2006	2397
standard deviation [ms]	127	246	491	775

Table 5.

Two stress tests are performed against F4 instance [2]. The first test is the standard as well as the others, but head 64 concurrent requests. The average response time is 12674ms, and the standard deviation 10740ms.
The second stress test also was conducted on F4, but 256 were sent requests. Due to the fact that there is no way they are sent entirely from one computer at the same time, they are sent for a very short period of time – 648ms from 12:19:43.477 to 12:19:44.125. The first answer has arrived after 4614ms, and the last after 97860ms while there was response for all request.
Statistical distributions of response times in testing performance with eight concurrent requests are as follows:

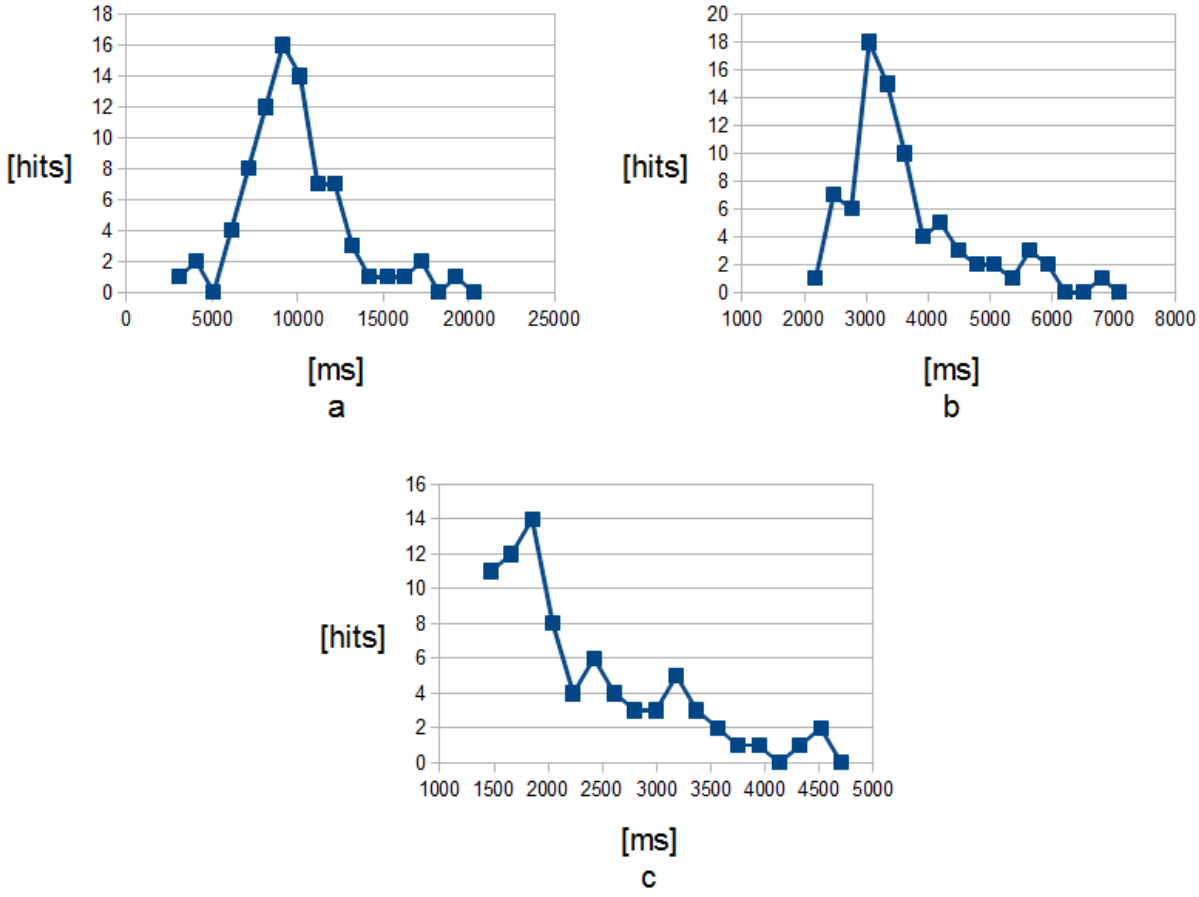

Fig. 2. Statistical distributions

In Figure 2.a is the distribution of the times for a response from the WEB server. In Figure 2.b responses from the F1 instance and 2.c contains distribution from F4.

9. ANALYSIS OF THE EXPERIMENTAL RESULTS

During the experiment it was found that the tests with more than eight concurrent requests may not be processed with appropriately quality by the WEB server. Therefore, the results are limited to eight parallel queries.

When testing the WEB server, both locally and over the Internet shows that the response time of one or two parallel applications is almost the same (Table 3). This can be explained by the fact that the hardware architecture of the server has a dual core processor on which can run two applications simultaneously. As the number of applications grows linearly and response time also grows linearly and response times for four queries is twice greater than the response time of two. Respectively, and response time of eight parallel applications is twice the response time of four. This result is explained by the principle of time slicing of CPU time between processes. In cases when there are more than two active processes, the operating system switches them in time, so that each of processes receives a fair share of the CPU time.

The same results are confirmed by the control test with Google SDK (table 4). In this case, the times are proportionate to those on the WEB server, but the bigger value is determined by two facts: first architecture of the machine is the same; and secondly by the fact that the SDK is not optimized for performance. SDK serves to develop and test the cloud application locally, but not for the actual load.

Greatest interest are the results of testing the performance of real cloud application (Table 5). It shows an interesting dependence. Again, increasing the number of concurrent requests increases linearly and response time, but it is not at the same rate as in the actual physical WEB server. If the previous two tests had increased by a factor proportional to one, which means that if it is doubled the workload it also doubled the response time, in this case the ratio is only 0.2. The following figure shows this result graphically.

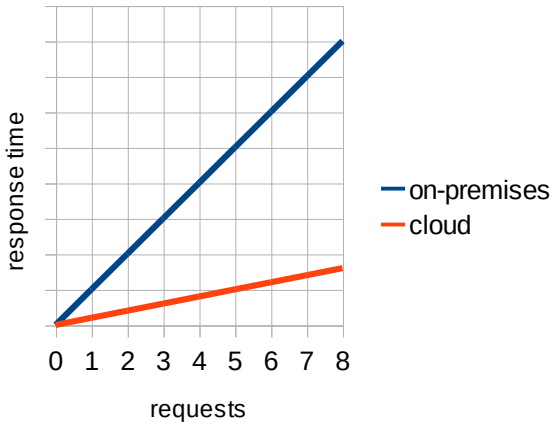

Fig. 3. Rate of response time

It is possible that the increase in response time with increasing the number of concurrent requests for cloud application to be mainly due to increased data traffic.
Another characteristic that is measured during the tests is the standard deviation. It represents a quantitative characteristic of how different are the reports from the average value of the sample. In the actual physical WEB server this indicator grows exponentially. This means that the more loaded the server is, the more requests are waiting too long for an answer. In cloud application the result is similar, but with the main difference that the increase is linear.
The last result, which will be addressed, is the response time of the most powerful free cloud architecture F4 tested against local physical server with eight parallel applications - they are approximately equal. This means that after a certain load, cloud application can provide better performance than a similar locally executed application despite network traffic.

10. CONCLUSION

The results confirm the basic idea of cloud computing:

- low to non-initial investment. To start this application does not need to allocate resources for server designed for this application;
- dynamic scalability. With increasing load, the cloud automatically allocates resources to meet the rising requirements;
- pay only for what you use. In the course of the entire test, which lasted about four hours was spent 26% of daily free CPU limit. This means there is no need to invest in infrastructure to be used only in a brief moment of time and the rest to stay unusable.

There are many examples demonstrating how comfortable and elegant solution can be cloud application.
In 2008 in Sofia applying for kindergarten had become over the Internet, but it failed due to the large number of requests.
By the end of 2011 all companies had to re-register, but again because of the large number of applications to Trade Register the servers were down.
These examples are best fitted with cloud technology. Despite great advantages, cloud computing is not universal technology that can be used for all problems.

REFERENCES

1. Nikolova N., Toneva-Zheynova. D., Naydenov D., Tenekedjiev K. (2012). Imputing Missing Values of Environmental Multi-Dimensional Vectors Using a Modified Roweis Algorithm. Workshop on Dynamics and Control in Agriculture and Food Processing.
2. Ellestad M. (March 20, 2003). *Stress Testing: Principles and Practice*. Oxford University Press
3. Emily H. Halili (June 27, 2008). *Apache JMeter*. Packt Publishing
4. Robbins N. (August 21, 2012). *Learning Web Design*. O'Reilly Media; Fourth Edition
5. Sanderson D. (November 2009.). *Programming Google App Engine*. O'Reilly Media
6. https://developers.google.com/appengine/