Performance Prediction of Web Based Application Architectures Case Study: .NET vs. Java EE

Osama Hamed[1], Nedal Kafri[2]
[1]Hulul Business Solutions
P.O.Box:4167, AI-Bireh, Palestine
Osama.Hamed@Hulul.com

[2]Department of Computer Science and IT
AI-Quds University
Main Campus, Abu-Dies
P.O. Box: 20002, Jerusalem
Palestine
nkafri(S)@science.alquds.edu

ABSTRACT: *Efficient web application is a challenge that we need to achieve when architecting web applications. This research follows a performance testing approach that aims to utilize load testing tools to give ideas about performance issues early in the development life cycle for applications implemented using Java Enterprise Edition (Java EE) or .NETplatform. Thus, it helps system architects to choose amongst competitive frameworks. To achieve this aim, the applications are subjected to artificial workload. Direct measurements are obtained on the specified application scenarios using different testing tools. Parasoft WebKing and Hewlett-Packard LoadRunner were used for this purpose. Later on, the obtained results were compared in order to verify the results.*

Keywords: Web application architectures, performance testing, load testing, response time, memory utilization

Received: 8 April 2009, Revised 2 June 2009, Accepted 8 June 2009

1. Introduction

In addition to the functional requirements, the web based applications developers are considering many other key issues to be facilitated and granted in the final stage of the developing process. Amongst the key non-functional requirements influenced by the architecture of the application and the utilized technology are performance, scalability, availability, and security [3]. Performance of the application may not be looked at until there is a concern [7]. In other words, as the access and requests on demand increases, performance is the most likely concern. Therefore, the web application has to achieve the high performance demand. If the system does not meet its performance objectives, the application will unlikely be accepted by the stakeholders. The application performance objectives are specified in terms of Response time, Throughput, CPU Utilization, Network I/O, Disk I/O and Memory Utilization [3].

Identifying that the web based system performance is properly met cannot be postponed until the system delivery phase. Therefore, it is recommended to begin and continue performance testing, starting from the early stages of system development process repeatedly during its life cycle [3] [7]. Thus, Enterprises can predict the performance issues before full application development and the system architects are able to choose among alternative competitive architectures and/or implementation technologies. The evaluation of the obtained performance results could also help in redesigning the application to achieve the required performance objectives.

Analyzing the performance of the web based system can be achieved using load testing and/or performance modeling approaches [4]. The load testing can be used to identify and isolate system bottlenecks, tune application components, predict system scalability, and make judgments on system architecture or design; while performance models are used in analyzing the performance and scalability characteristics of the system under study. Vast number of platforms was launched aiming at addressing these requirements including .NET and Java EE/J2EE (Java Enterprise Edition) platforms. When architecting an enterprise web application, many design alternatives with their corresponding technologies are in use. Among these are the Model View Controller (MVC) and 3-tier architectures and their corresponding implementations in Java EE (Sun Microsystems) and .NET frameworks (Microsoft) [6].

The aim of this work is to test and compare the performance of these two well-known web applications technologies (Java EE and .NET) in implementing MVC and 3-tier architecture in terms of memory utilization and response time. Seeking reliability and fairness in measurement, the response times were carried out using two load testing tools: Parasoft WebKing™ [13] and HP LoadRunner7 [14], which is a performance and load testing product by Hewlett-Packard (since it acquired Mercury Interactive in November 2006), were used for this purpose. The experiments in this work follows the suggested modeling approaches in [3] and [6] for performance evaluation of web based applications.

The paper starts by giving a background on performance evaluation issues, motivation, and overview of the different used approaches and techniques. Then it introduces the methodology used in this work. Finally, results obtained in this research are discussed.

2. Background and Motivation

In a typical enterprise web application testing, some areas require the following types of testing such as integration, functional, acceptance, and load testing [10]:

- Integration Testing: Starting from the simplest components, classes, the developers need to program unit tests to ensure that the application’s smallest units behave correctly - as part of a subsystem, and as part of the whole application.
- Load Testing: This type of testing is appropriate to help us in choosing among architectural and design alternatives. In other words, for benchmarking .NET and Java EE applications they are subjected to artificial workloads in terms of the notion of virtual users [8].
- Functional Testing: During application development, quality assurance engineers perform automated and manual functional tests to evaluate the application’s behavior from the user’s viewpoint.
- Acceptance Testing: When a development project nearly completes a specific milestone, acceptance tests can be performed to verify that the application fulfilled the requirements.

Evaluation techniques are the methods by which performance evaluation indices are obtained, these can be classified into two categories: empirical (load testing) techniques and performance modeling techniques [1] [2] [3] [4]. Empirical techniques require that the system or network to be evaluated exists and direct measurements of the evaluation target have to be taken. However, they can evaluate the performance of application by simulating virtual users interacting with the system [4]. On the other hand, when talking about performance models, they fall into two categories: simulation and analytical. The simulation models simulate the behavior of the application. In spite of their generality and ease of use, simulation models may fail or produce non-accurate results. The analytical models capture the key aspects of a computer system and relate them to each other by mathematical formulas with computer algorithms [4] [9].

Vast number of performance modeling studies -belong to these two- were done on web applications using different approaches and techniques. These techniques and approaches were introduced in [3] [4].

Up to our knowledge, there are few research studies that have evaluated performance of application architectures/ technologies and best practices to achieve the desired performance [3] [4] [5] [6] [11] [12]. However, no work has been conducted to evaluate and compare the performance of the mentioned architectures and technologies. For instant, in [3] the authors introduced an approach for performance modeling for .Net and Java EE platforms based on identified framework. For this purpose, they developed a Proof-of-Concept (PoC) to address the requirement of an application in the financial domain for both platforms. They tested the performance of .NET and Java EE PoC by means of throughput, response time, %CPU utilization, and %Disk Idle Time. It should be noted that, benchmarking .NET and Java EE was not the objective of the research in [3]. Hence, the performance experiments were not running on machines with similar configuration and with same testing tools. In [4] they discussed the different approaches to performance analysis of large J2EE applications, focusing on analytical modeling. Furthermore, in [5] the authors studied the performance of JBOSS application server (the most popular open-source Java EE application server). They used SPECjAppServer2004 -an industry standard benchmark for measuring the performance and scalability of Java EE application servers.

This research presents a work similar to that in [3], but for the purpose of benchmarking .NET and Java EE in implementing MVC and 3-tier architecture, respectively. Therefore, the performance experiments were running on machines with exactly similar configuration and with same testing tools. A simple identical application was developed and developed using Java EE and .NET for this purpose. Herein, a load testing process was followed to get ideas about the performance issues in terms of response time and memory utilization.

2.1. Performance modeling

Performance modeling is a structured and repeatable approach to model the performance of software applications. It begins during the early phases of application design and continues throughout the application development life cycle. The performance model presented has two parts:

- An information structure to help us capture performance-related information.
- A process that helps us incrementally to define and capture the information that helps working team to focus on using, capturing, and sharing the appropriate information.

A performance model provides a path to discover what we do not know. The benefits of performance modeling include the following:

- Performance becomes a feature of our development process and not an afterthought.
- We can evaluate our tradeoffs earlier in the development life cycle based on measurements.
- Test cases usually show us whether we are trending toward or away from the performance objectives throughout our application life cycle.

Modeling allows us to evaluate the application design before investing time and resources to implement a flawed design. Having the processing steps for the performance scenarios laid out enables us to understand the nature of our application's work. By knowing the nature of this work and the constraints affecting that work, we can make more informed decisions.

2.2. Performance testing

Performance testing is used to verify that an application is able to perform under expected and peak load conditions, and that it can scale sufficiently to handle increased capacity. Performance testing is the process of capturing performance measures by subjecting the application to a specified set of conditions and input [3]. For the purposes of performance testing, the application should be deployed on infrastructure that is similar to production environment. There are three types of performance tests that share similarities yet accomplish different goals [6]:

- **Load testing:** We usually use load testing to verify application behavior under normal and peak load conditions. This allows us to verify that the application can meet the desired performance objectives; these performance objectives are often specified in a service level agreement or requirements gathering document. Load testing enables us to measure response times, throughput rates, resource utilization levels (CPU, Memory, Disk I/O, and Network I/O), and to identify our application's breaking point, assuming that breaking point occurs below the peak load condition.
- **Stress testing:** we usually use stress testing to evaluate our application's behavior when it is pushed beyond the normal or peak load conditions. The goal of stress testing is to uncover application bugs that appear only under high load conditions.
- **Capacity testing:** capacity testing is complementary to load testing and it determines our server's ultimate failure point, whereas load testing monitors results at various levels of load and traffic patterns

We can define performance testing as the process of identifying how an application responds to a specified set of conditions and input. It usually aims to identify how well our application performs in relation to the specified performance objectives like response time, thus identifying the suitability of architecture and design. We may not be able to identify all the characteristics by running a single type of performance test. Some of the application characteristics that performance testing helps us identify are response time, throughput, maximum concurrent user support, and resource utilization in term of CPU, RAM, network I/O, and disk I/O resources that our application consumes during the test [6].

To simulate workloads, performance testing is usually accomplished using the notion of virtual users [8]. There are many tools can be use to simulate the load in terms of users, connections and capture data related to Response Time, Throughput, and Resource Utilization. Among these tools are HP LoadRunner, Microsoft Application Center Test (ACT), Microsoft Web Application Stress tool, Parasoft WebKing, Compuware's QA load and Rational Performance Tester. Furthermore, there are

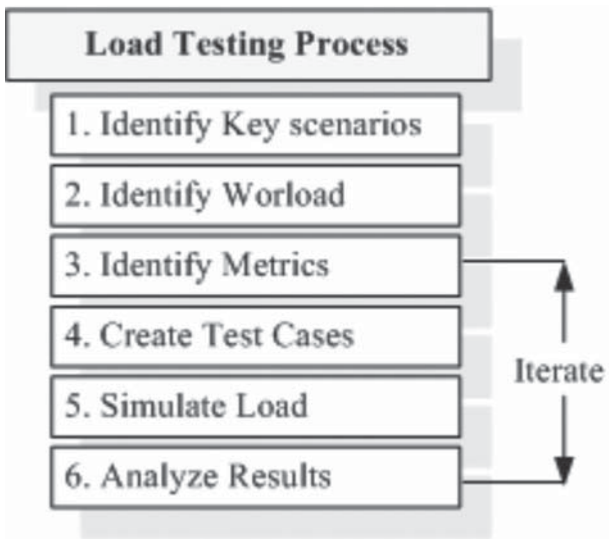

Figure 1. Load testing process

organizations such as the System Performance Evaluation Corporation
(SEPC) that offers benchmarks and develops standardized performance tests and publishes review results [15], below is an overview of the used two tools.

- **LoadRunner Overview:** It works by creating virtual users who take the place of real users operating client software, such as Internet Explorer sending requests using the HTTP protocol to web servers.
- **WebKing Overview:** It involves exercising your application to load testing with virtual users and verifying whether it supports the tested traffic levels, patterns, and combinations. It is typically used to verify whether the application will perform adequately under the expected loads and surges, determine system capacity, and identify the source of any bottlenecks.

In this work, we followed a performance testing approach that used these two tools to identify which application architecture suits for full application development.

3. Methodology

Herein, we use the load testing process by using different workloads to choose among the different design alternatives and their corresponding implementations. Figure 1 below shows the load testing process as a six steps process [6].

This work follows a performance modeling approach similar to that used by Microsoft teams; it conducts a performance testing scenarios on specific use cases in the application to give ideas about the performance issues. Measurements are to be obtained using two available load testing tools: Parasoft WebKing M[13] and Hewlett-Packard LoadRunner™ [14]. In this research two identical simple web based applications that describe the same business functions were implemented using Microsoft .NET and Java EE technologies, these applications conforms to 3-tier and MVC architectures respectively. Then they are deployed on the same environment to benchmark both technologies with the help of mentioned load testing tools.

4. Case study

This research aims at measuring the performance in terms of response time for two web application architectures; in other word it is used to know which architecture suites for full application development or meets the performance objectives specified in the requirements. It also gives a brief introduction to web application architectures as will as shows some implementation details for both architectures.

4.1 Web application architectures

Architecting web application means how to connect the components that comprise the application together; these are the view or presentation, the business logic and the data access.

Application architectures have evolved increasingly, in the first generation where all things are assembled in a single tier; in the second generation where a move to the first phase of distributed tiers took place i.e. the client server architecture; in the third generation the three-tier architecture took place

4.1.1 Three-tiers architecture

- Presentation Layer (PL): through which the users interacts with system
- Business Logic Layer (BLL): that serves as an intermediary for data exchange between the presentation layer and the DAL.
- Data Access Layer: The Data Access Layer (DAL) cleanly separates the data access details from the presentation layer, it does not enforce any business rules that may apply

4.1.2 Model View Controller architecture

The MVC is an alternative to three-tiers; it separates responsibilities into three layers of functionality.

- **Model:** The data and business logic; it typically has some means to interact with persistent storage such databases.
- **View:** The presentation; it is responsible for displaying data from the Model to the user. This layer also sends user data to the Controller
- **Controller:** The flow control; it handles all requests from the user and selects the view to return. The struts action acts as controller.

4.2 Technologies Matrix

Architecting the web applications means separating the functionalities among tiers/layers as will as giving flexibility for extensibility in the application. It also means providing services to architecture above tiers. This means that, data access tier gives services to business logic and the business logic gives services to GUI tier. The implementation technologies and framework components for both technologies are shown below.

4.2.1 ASP.NET application

Technologies Matrix for ASP.NET Implementation	
GUI (PL) as Web Project (Common Layout)	ASP.NET 2.0 (AJAX enabled)
	Master Page for common layout
	Login Control
	Client validation using validation controls
	The same images and CSS
	Data Grids for table view
BLL as Class Library Project	Checking for business roles like user privileges goes here, it also acts as bridge between the GUI and the Data Access Layer
DAL as Class Library Project	SQLs are handled using SQL Data Adapter
	Dataset as mapping for DB schema
	User table as VB.NET class
	Ms SQL 2005 as DBMS

4.2.2 JAVA EE application

Technologies Matrix for JAVA EE Implementation	
GUI (PT) as Web Project (Common Layout)	Struts 2.0.6 (AJAX enabled)
	Tiles for common layout
	Login Action
	Client validation using validation frame work
	The same images and CSS
	Display tag 1.0.1 for table view

BLT as Business Tier Project	Checking for business roles like user privileges goes here, it also acts as bridge between the GUI and the Data Access Tier
DAT as Business Tier	SQLs are handled JPA using Entity Manager on top of EJB3
	Oracle Toplink as ORM tool
	User table as Entity Bean using annotations
	Oracle lOg as DBMS

However, the performance goals in our research are based on comparing the response time and memory utilization for the two implemented applications. To achieve this, a virtual application was implemented using .NET and Java EE to be considered as a case study. After that, the two load testing tools were used to conduct the performance testing process on the key scenarios specified.
Performance modeling is to be performed for a Proof-of-Concept (PoC); in our case it represents the Tasks Management System (TMS) that obey application architectures for two competitive frameworks, the .NET and Java EE platforms. The target application described is a Task Management System (TMS) in which the administrator creates simple tasks for the normal users in the system. The major use cases in the TMS application are shown in Figure 2.

It should be noted that the applications servers and testing tools were deployed on Intel(R) machine with the specifications Core™ 2 Duo processor 1.86 GHZ, 2 GB RAM, and the database servers is in remote machine connected via 100 Mbps network.

A. .NET version of the PoC, TMS

The .NET version of the TMS is discussed in this section; the architect for the PoC is shown in Figure 3 (a). The user of the application uses the browser to access the presentation layer of the application which implements the 3-tier [7]. The request for business functionality is passed to the business layer which in turn may pass the request to data access components in the data layer. The .NET version of TMS was developed using VB.NET running on Windows XP and the database is SQL server 2005. The IIS (Internet Information Service) web server and the data base server were running on separate machines. The pseudo code for the ASP.NET application is as below:

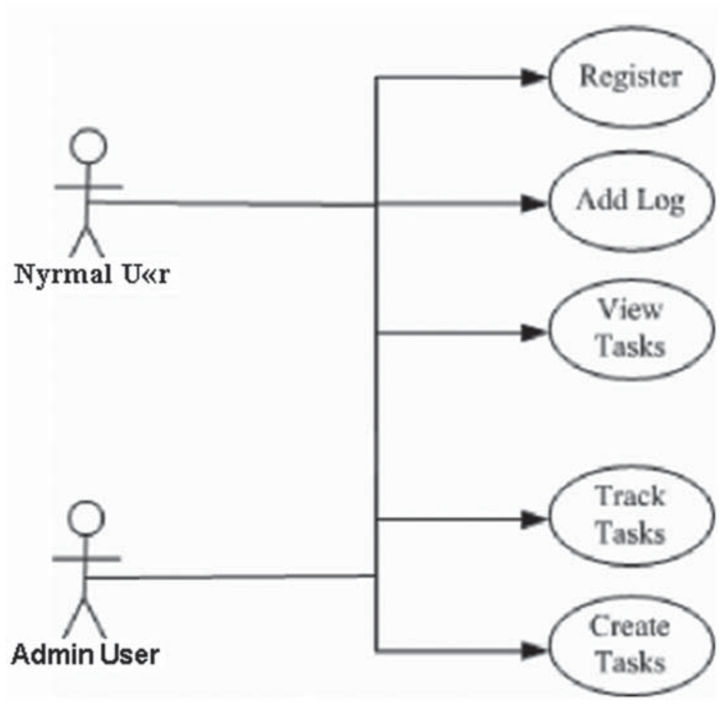

Figure 2. TMS Use Cases

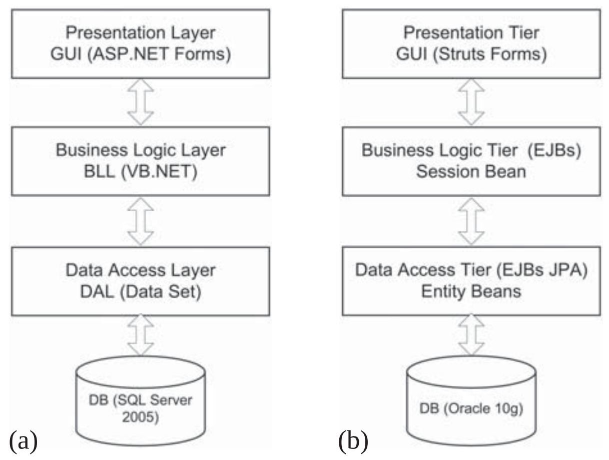

Figure 3. (a) Architecture of .NET version (b) Architecture of Java EE version

- In the code behind of the web project, we usually define instances of business logic classes.
- We use this instance to call the methods of business logic; the later creates instances for one of the classes in DAL project. The later will call methods to execute database operations relying on SQL specific class. For example SQLConnection, SQLCommand and SQLDataAdapter.

B. Java EE version of the PoC, TMS

The Java EE the TMS is discussed in this section, the architect for the PoC is shown in Figure 3(b). The user of TMS uses the browser to access the application. The request from the browser passes, on (over HTTP) to the presentation tier which implements the MVC design pattern as described in [7]. The presentation tier is implemented with Struts framework. The request for business functionality is passed to the business tier. The business tier implements business processes for different modules as Enterprise Java Beans (EJBs). The data tier implements the Java Persistence (JPA) with Enterprise Java Beans (EJBs). The pseudo code for the JAVA EE application is as below:

- In the struts action of the web project, we usually define instances of remote interface implemented by the session bean exists in the database and business logic projects.
- Use this instance to call the methods in session bean, the later uses the EntityManger method to execute database operations.

The application is developed in Java running on Windows XP and the database is Oracle 10g. The OC4J (Oracle Container-forJava) application server and the database were running on separate machine.

5. Experiment results

In this context, it should be noted that the purpose of the exercise was to determine the Performance Models of the Case Study applications that were developed based on identified architectures (3-tier & MVC) and their corresponding implementations .NET and Java EE. Benchmarking both .NET as well as Java EE applications were the objective of this exercise and hence the performance tests were run on machines with exactly similar configuration using different testing tools. Performance testing was conducted using Parasoft WebKing™ and Hewlett-Packard LoadRunner™ using two samples of virtual users. The two samples carried out are as follows:

- Sample I: 1,5, 10, 15, and 20 users; the results were obtained using WebKing and LoadRunner.
- Sample II: 1, 20, 40, 60, 80, and 100 users; the results were obtained using WebKing only due to license limitations in LoadRunner.

In addition to the response time, the memory utilization of the evaluated technologies was also considered.
First of all it is important to know that there is a performance risk because we are considering web based applications: thus Performance modeling is critical for the case study application since it is web based that could be accessed from different locations. Suppose, for example that the critical use cases were “Login” or “View Tasks” or “View Users”.

Step 1: Identify the Key Scenarios or Critical Use Cases: The key scenarios are those involving execution of the use cases by a specific number of concurrent users for example 100 concurrent users.

Step 2: Identify the Workload: for example 100 concurrent users.

Step 3: Identify the Metrics: Suppose that the performance objective is to meet 2 second response time with the specified number of concurrent users, in our case 100 concurrent users.

Steps 4, 5, and 6: Identify the Workload: Performance tests were conducted using Paarsoft WebKing™ and Hewlett-Packard LoadRunner™ which simulates the workload specified in terms of virtual users, results were measured and the bottlenecks can be identified. Although many performance measures can be identified through performance testing, among these are:

- Response Time, we considered the response time for the purpose of benchmarking of .NET & Java EE applications.
- Memory Utilization, we also considered the memory usage for the purpose of benchmarking of .NET & Java EE applications.

5.1. Response time performance

The performance testing for the ASP.NET and the Java EE case study applications was conducted using WebKing™ and LoadRunner™ testing tools. The obtained results for Login used case for sample I. using both tools are provided and summarized in Figure 4(a) and Figure 4(b). The figures show the difference in the increase of the response time as the numbers of users increase. It is clear that the response time is always shorter in the case of Java EE application for different loads.

Similarly, Table 1. and Figure 5. below show the response time obtained in the case of .Net and Java EE implementation for Sample II. using WebKing tool. It can be clearly seen from the figures that implementation using Java EE behaves better than .NET for the PoC application in terms of response time. The .NET graph increases more rapidly than the relevant graph of Java EE as the numbers of virtual users grow.

Virtual Users	**Response time (msec)**	
	ASP.NET	**Java EE**
1	3.0680	1 .09407
20	3.4610	3.32000
40	53.521	3.48200
60	56.975	3.36100
80	71.741	3.85200
100	88.415	5.76800

Table 1. Performance testing of ASP.NET (Application, using WebKing)

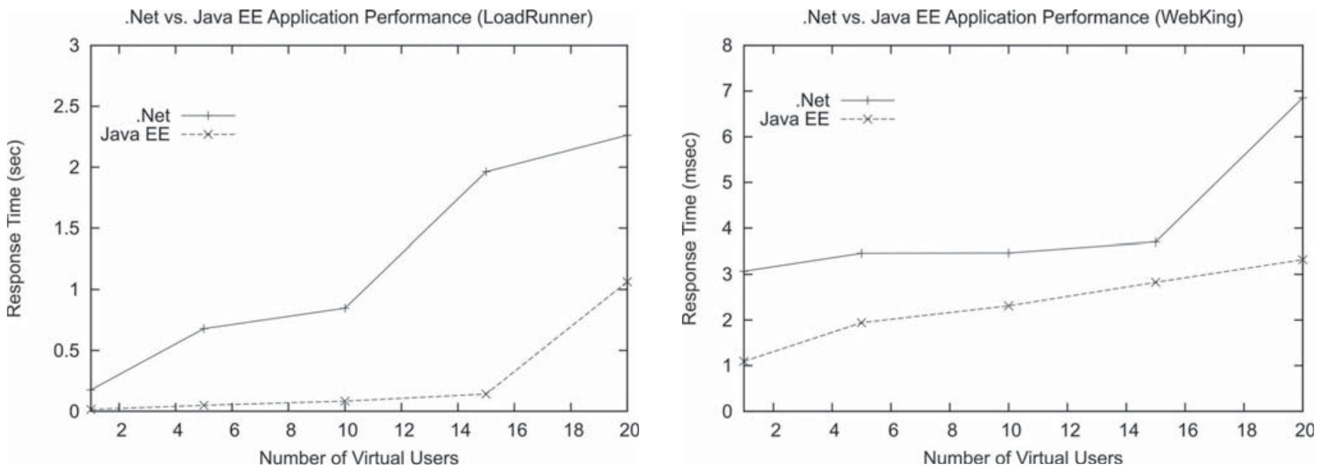

Figure 4. (a) .NET vs. Java EE response time (b) .NET vs. Java EE response time

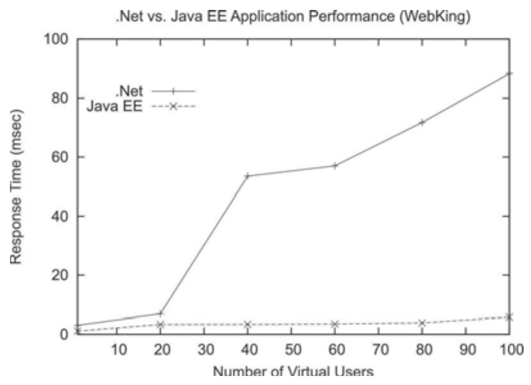

Figure 5. .NET vs. Java EE response time

5.2. Memory utilization

Moreover, the memory utilization for both applications was obtained with the help of Windows Task Manage and the maximum values were recorded. Performance tab as shown in Table 2. and Figure 6. for sample II.

Use case: Login		
Users	**Memory utilization (MB)**	
	ASP.NET	**Java EE**
1	1031	655
20	1185	695
40	1231	748
60	1290	853
80	1325	957
100	1374	1085

Table 2. Memory utilization of ASP.NET and Java EE Applications

Thus, in term of memory utilization, it can be clearly seen that Java EE implementation requires less memory than .NET for the PoC application.

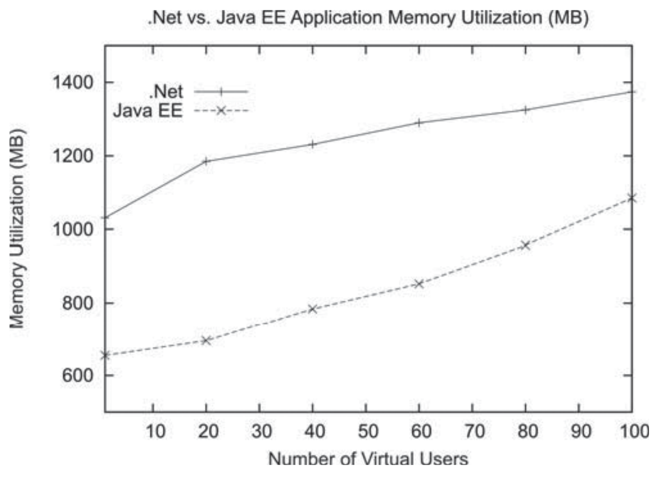

Figure 6. .NET vs. Java EE memory utilization

6. Conclusion

In this research we proposed and implemented a performance testing approach for web based applications early in the development process. Our approach has potentials to help system architects in choosing among competitive frameworks; in our case we considered the world's most popular frameworks .NET and Java EE.

Performance testing measurements for two versions of PoC (Tasks Management System} were carried out using .NET and Java EE technologies. Moreover, two different tools were utilized (i.e. WebKing and LoadRunner™) in order to verify the results.

Finally, obtained results of using both load testing tools show that Java EE performs better than .NET by means of response time and memory utilization.

We are looking forward to make our results general. Therefore, in the coming future we are going to make some investigations about the performance measures of business components supported in both technologies as well as in other technologies.

References

[1] Bertoli, M. Casale, G ,. Serazzi, G (2007). An Overview of the JMT Queueing Network Simulator. Politecnico di Milano - DEI, TR 2007.2,. Available at: http://jmt.sourceforge.net/Papers/tech07jmt.pdf.

[2] Bertoli, M. Casale, G ,. Serazzi, G, (2007). The JMT Simulator for Performance Evaluation of Non-Product-Form Queueing Networks," ANSS, pp.3-10, 40th Annual Simulation Symposium (ANSS'07).

[3] Kambhampaty, S, Modali, V.S, Bertoli, M. Casale, G , Serazzi, G (2005). Performance Modeling for Web based J2EE and .NET Applications, In Proc. of world Academy of Science, Engineering and Technology, V 8, Oct.

[4] Kounev, S., Buchmann, A (2003). Performance Modeling and Evaluation of Large-Scale J2EE Applications, n Proc. 2003 Computer Measurement Group conference (CMG-2003), Dallas, Texas, December 7-12.

[5] Kounev, S, Weis, B. Buchmann, A.. Buchmann, A (2004). Performance Tuning and Optimization of J2EE Applications on the JBOSS Platform," In Journal of Computer Resource Management, Issue 113, 2004. Available at: http://www.dvs. tu-armstadt.de/publications/#section2004

[6] Meier, J. D., Vasireddy, S., Babbar, A., Mackman, A (2004) Improving .NET Application Performance and Scalability, Patterns & Practices". Microsoft Corporation, 2004, ISBN 0-7356-1851-8

[7] Mitchell, S. (2008). The Official Microsoft ASP.NET Site, Data Access Tutorials, Microsoft Corporation, 2008. Available at: http://www.asp.net/learn/data-access/

[8] Menasce, D. A. (2002). Load Testing, Benchmarking, and Application Performance Management for the Web, in Proc. Computer Management Group Conference, pp., 271-281, Reno, Nevada, December 2002. Available at: http://www.cmg.org

[9] Menasce, D. A., Almeida, V. A. F. Dowdy, L. W (2004). Performance by Design: Computer Capacity Planning by Example. Prentice Hall, 2004

[10] Sidat, M (2005). Automated Stress Testing of Web Applications using User Session Data, Department of Computer Science, University of London. Available at: http://www.dcs.kcl.ac.uk/staff/mark/PastMScProjects2004/MohamedSidat.doc.

[11] Smith, C. U., Williams, L. G (2002). Performance Solutions: A Practical Guide to Creating Responsive, Scalable Softwar,. Boston, MA: Addison-Wesley.

[12] Smith, C. U. Williams, L. G (2003). Best Practices for Software Performance Engineering", in Proc. CMG, Dallas, Dec.

[13] Parasoft WebKing, Available at: http://www.parasoft.com/webking

[14] HP LoadRunner software, Available at: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17%5E8_4000_100

[15] System Performance Evaluation Corporation, Available at: http://www.spec.org/

Authors bibliography

OSAMA A. HAMED was born in Anabta, Palestine and received his B. Sc. and MSc. degrees in computer science from AI-Quds University, Jerusalem in 2003 and 2009 respectively. He is interested in performance modeling for business components and software architectures. He worked at AI-Quds University from 2003 to 2004 as teaching assistant, after that he worked as Web Developer, now he is working as a senior developer (JAVA EE) in HULUL Business Solutions IT Company in Palestine. His formal email is Osama.Hamed@hulul.com and his alternative email is Sam.Hamed@qmail.com. HULUL website is www.hulul.com and AI-Quds University website is www.alquds.edu

NEDAL M. S. KAFRI was born in Attil, Palestine and received his education at the Technical University of Plzen in Czech Republic, where he obtained his M Sc. degree in Control Systems and Installation Management in 1982. He worked at AI-Quds University as a lecturer at the Computer Science Department for fifteen years. Then he moved to the Czech Technical University in Prague in 1997, where he obtained his Ph.D. degree in Distributed Systems in 2002. Now he is a member of the academic staff of the Department of Computer Science of AI-Quds University in Palestine. His research interest is in Distributed, Parallel Computing and Security. His email is nkafri@science.alquds.edu and his Webpage is http://www.alquds.edu/staff/kafri.