A Comparison between EJB and COM+ Business Components, Case Study: Response Time and Scalability

Abedulhaq Abu-Kamel, Raid Zaghal, and Osama Hamed
Dept of Computer Science, Al-Quds University
Software Development Directorate, Hulul Business Solutions
Palestine
{abedhaq}@yahoo.com, {zaghal, ohamed}@science.alquds.edu, {osama.hamed}@hulul.com
http://www.alquds.edu/

Abstract. Most distributed system architectures are designed as a three-tier systems consisting of a thin-client, middleware and a database. The overall performance of such systems depends on the performance of each tier individually and the overhead incurred by the collaboration between these three tiers. Nowadays, the two most popular middleware systems are: Microsoft's .NET platform and Sun's Java 2 Enterprise Edition (J2EE) platform. In J2EE, the middle tier infrastructure is called Enterprise JavaBeans (EJB) and in the .NET framework, it is called Component-Oriented Middle-Tier (COM+). Usually, the middle tier provides the business logic (any code that is not specifically related to storing and retrieving data, or formatting data for display to the user) and the performance of this tier is crucial to the overall performance of the distributed system.

In this paper, we will measure via real experimentation the performance of the middle-tier (business logic) of the two platforms, namely: EJB3 and COM+ 1.5, in terms of response time and scalability. Then we compare and analyze the performance of each technology under different workload scenarios.

1 Introduction

Computer software had rapidly increased and developed specially in Distributed Software Systems, these systems need connectivity, ignoring whether these systems are homogeneous or heterogeneous; a middleware is needed, the performance of the middleware affects the overall system performance. Within just few years, the Windows platform has evolved from personal computing operating system to a complete Enterprise solution.

Distributed applications grew from the traditional client-server approach to three-tier applications, since the client-server approach had some problems such as fat-clients—when most or all of the code is written on the client side, which requires downloading all data to the client to fulfill any business task. Therefore, the traffic on network becomes overloaded when the system has a large number of clients and thus affects the system's performance severely.

The three-tier approach was proposed as a solution for client-server problems. It consists of the presentation tier, the business tier (or business logic), and the database tier. This approach offers more advantages such as performance, scalability, maintenance, security, flexibility and more freedom. Figure (1) shows the three-tier architecture.

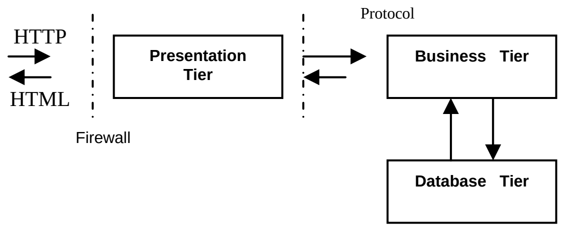

Fig. 1. Distributed architectures are typically based on three tiers

A presentation tier is responsible for working with clients (users), it accepts an HTTP requests from a web browser and returns an HTML page that browser can then display. The business tier is where much of the business logic is implemented, since the business tier lies between the presentation tier and the database tier, it is often called the middle tier.

Business logic often requires expensive resources, such as database connections, threads, TCP/IP connections, and message queue connections. These resource requirements would normally make it difficult to support a large number of clients at any one time, a requirement of most e-Commerce applications.

Both J2EE and the .NET framework business tiers include a sophisticated middle tier infrastructure that supports resource sharing among clients. This infrastructure is critical to supporting large numbers of clients, and thereby achieving high system throughput. In J2EE, this middle tier infrastructure is called Enterprise JavaBeans (EJB). In the .NET framework, it is called COM+. In this study, we consider specific versions of middle tier namely EJB3 and COM+ 1.5.

In this work, an emulation approach will be used to do a comparison among EJB3 server and COM+ 1.5 to identify the performance in terms of response time and scalability for the business tier in both frameworks. This work will help software engineers and developers to choose the right platform for their new distributed applications to fit their business requirements.

2 Background

Performance is defined as an indicator of how well a software system or its component can meet the timeline requirements, and this is usually measured in terms of response time and throughput. There are two important dimensions of software performance timelines: responsiveness and scalability. Responsiveness is the ability of a system to meet its objectives for response time or throughput, while Scalability is its ability to continue to meet these objectives as the demand for the software functions increases.

Meeting performance requirements is not enough for stringent performance application, but also must be able to scale gracefully to changing patterns of use as well as increasing in demand. Studying performance can be estimated or measured by studying the system's architecture. And performance failures can be avoided early in the design process.

Vast numbers of performance modeling/testing studies were done on JAVA EE and .NET technologies, among these [13][14]. Even the field of EJB and its performance has been much researched [2][3][9], but these researches focused on how to improve the performance of EJB systems in general. This work will focus on examining the specific performance aspects of EJB systems and gives quantitative results.

Most of previous researches focused on one particular Application Server [2][7], whereas other works focused on making a comparison between different vendor's products [4][5]. Some researches modified the Application Server configuration parameters, that includes parameters such as the number of server threads and the number of beans deployed, some of these researchers examined the effects of changes in the hardware architecture.

Many researches where conducted in this area with different objectives, such as which EJB system was used, which type of beans were used, whether local or remote calls were made, which transactional or security options were utilized, or changes to any other software specific parameters.

The EJB server manages one or more EJB containers. The container is responsible for providing component pooling and lifecycle management, client session management, database connection pooling, persistence, transaction management, authentication, and access control. COM is a standard; it is platform independent, distributed, object-oriented system for creating binary software components that can interact with each other. COM components can be written in any programming language. COM+ is an extension to COM, and it is possible to use COM objects in COM+, it was developed as a standard for designing distributed, multi-tier applications. In COM+ it is possible to move and copy components. Jeremy Singer [1], conducted a comparative study JVM vs. CLR. He found that CLR and JVM Compilers are approximately the same regarding compilations and executing object oriented programs.

3 Methodology & Testbed

First, we will define the infrastructures of EJBs and COM+ and the similarities and differences between them. COM+ is an extension of Component Object Model (COM), Microsoft's strategic building block approach for developing application programs. COM+ is both an object-oriented programming architecture and a set of operating system services. It adds to COM a new set of system services for application components while they are running, such as notifying them of significant events or ensuring they are authorized to run. COM+ is intended to provide a model that makes it relatively easy to create business applications that work well with the Microsoft Transaction Server (MTS) in a Windows NT or subsequent systems. It is viewed as Microsoft's answer to the Sun Microsystems-IBM-Oracle approach known as Enterprise JavaBeans (EJB).

Enterprise JavaBeans (EJB) technology is the server-side component architecture for Java Platform Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure, and portable applications based on Java technology. To measure the performance (Response Time) of these two middle tiers (COM+ 1.5 & EJB3) we will build two distributed web applications that use the same hardware and software, but differ at the middle tier, first we will use COM+ and then EJB as a middle tier (or business tier).

Hardware Components: Four Computers with the following properties or higher, CPU Pentium 4 (3.00 GHz), RAM 1 GB DDR1, Network card 100Mbps, 8-port switch, and STP wires for network connections. Software Components: Windows Server 2003, Windows XP, Oracle 10g (Server and Client), Java platform, JDeveloper, OC4J, Visual Studio 2005, .NET Framework 2.0, and LoadRunner 8.1 [12].

At this experiment a compatible presentation, database, and middle tiers are used, EJB/JAVA and COM+/.NET platforms are used as an environment to deploy the business logics at each one. For the presentation tier, HTML with Suitable Scripting Language is used. For the Database tier, Oracle10g is used since oracle 10g is compatible with both Java and .NET platforms.

To achieve the comparison two parts of the experiment were developed (these two parts must be equivalent at each tier). In the first part of the experiment, the COM+/.NET platform was used as a middle tier and the experiment was repeated several times. At each attempt, the workload was increased at both the presentation tier (increasing the number of users) and the database tier. A log file keeps track of response times for each attempt. Figure (2) illustrates the configuration.

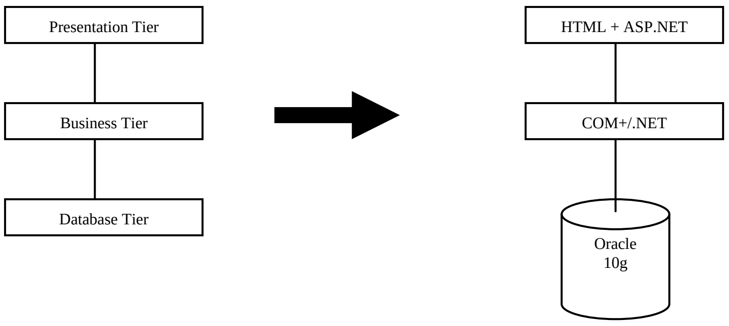

Fig. 2. The architecture of COM+ experiments.

In the second part, the experiment is repeated using the same parameters (Hardware and software) but we replaced the COM+/.NET platform with EJB/J2EE platform. We also used the JSP as the scripting language for the presentation tier since it is compatible with Java (most Web sites built on Java use JSP). This experiment represents a real distributed application as seen in any business installation. The configuration is shown in Figure (3).

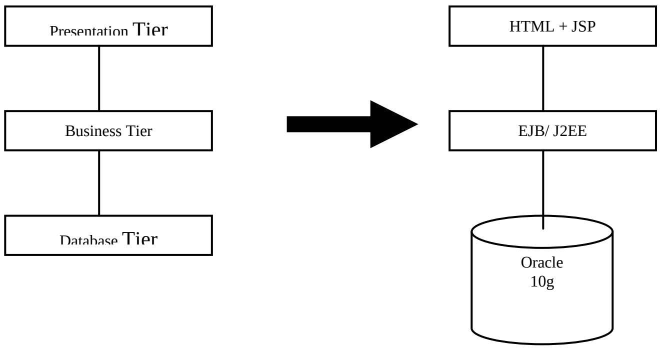

Fig. 3. The architecture of EJB experiments.

After each run, all relevant readings from the experiment were logged and analyzed. Then the differences in performance among these middle tiers were noticed. This experiment represents a simple university registration system, implemented as a distributed web application using both middle tiers. Table (1) summarizes the software that we have used at each tier.

Table 1. Software used at each application in our experiment

	Presentation Tier	**Business Tier**	**Database Tier**
Experiment 1 **First application**	HTML with ASP.Net as scripting language	COM+ / VB.NET	Oracle10g Version 10.2.0.1.0
Experiment 2 **Second application**	HTML with JSP as scripting language	EJB using JDeveloper as integration development environment	Oracle10g Version 10.2.0.1.0

Since we want to make a judgment on the performance of these technologies, the implementation was equivalent in both technologies. All transactions took place at the business tier. In such application, there are many transactions, and the following scenarios describe the transaction model for our experiment.

The user chooses the student ID, department ID, and semester from drop down lists, then he/she edits the year. After that he/she presses the (insert) button. The values that were chosen are inserted into the table (regest). The student will be enrolled into all courses from the selected department. The application gives a random grade for each course for this student, and inserts this data into the (semester_courses) Table.

In the scenario just describes, many transactions took place to complete one process; If the application has 5 departments in the (dept) table, 300 students in the (student) table, and 10 courses for each department in the (courses) table, the application deal with (300*10*5 = 15,000) records to be able to insert into the (semester_courses) table for the student. Therefore, in one process there are a lot of transactions for the CPU to execute. Furthermore, the application must process many transactions for each student, many transactions for the selected departments, and many transactions for the selected semesters. After that they will be inserted into (semester_courses) Table. This is for one user only. Naturally, the workload will be multiplied when more than one user performs this scenario concurrently.

4 Experiment Results

We will take the COM+ experiment as an example, and the same applies for EJB experiment. The experiment consists of three tiers, presentation, business, and database tiers. Thus, the application is distributed on more than one machine (the database server, the COM+ server, the Web server + application proxy from the COM+ components, and the client machine that runs the LoadRunner program).

We open the http page on the client machine; the client requests the web page from the web server, which responses and opens the page on the client. The user chooses from drop-down lists; student, department, and semester. Then, he edits the year of registration and presses the (insert) button. The client sends these commands to the web server which has the web page and to COM+ components as a proxy, the connection among application proxy and COM+ server will be established since the COM+ server has all methods of our application. Also, the COM+ server establishes a connection with the database server whenever the transaction needs to access the database.

Most transactions happen on the COM+ server since it has all the methods (business logic), and it is responsible for the connection to database and web server. Furthermore, if we want to add security to the application, it will be deployed on the COM+ server (as a property of the COM+ components). Since we aim to test performance we will implement more than one scenario.

4.1 Scenario 1 (Loading Page)

In this scenario we will just load an empty page without any transactions (the page doesn't contain any data). We did this scenario to test the response time to establish a connection for both EJB and COM+. The virtual users were loaded simultaneously, the duration of the scenario was 10 seconds, and all virtual users stopped simultaneously too. This scenario was repeated several times for both COM+ and EJB applications and the response time for both cases is shown in table (2).

Table 2. The average response times for scenario1

Number of Virtual users	AVG of Response Time	
	EJB	COM+
10	0.006	0.008
20	0.012	0.016
30	0.018	0.024
50	0.029	0.036
70	0.039	0.052
100	0.044	0.071
130	0.06	0.078
150	0.061	0.107

At this scenario as we saw, the EJB applications had better results than COM+ applications, at the beginning when we had just 10 users the results for both were approximately the same, but when we increased the number of virtual users the EJB applications had a better result than COM+ applications, at 150 virtual users EJB response was at 0.061 seconds, but COM+ response was at 0.107 seconds, that is a big difference, COM+ application needs approximately double the time that was needed for EJB application to respond. Figure (4) shows the results of scenario 1.

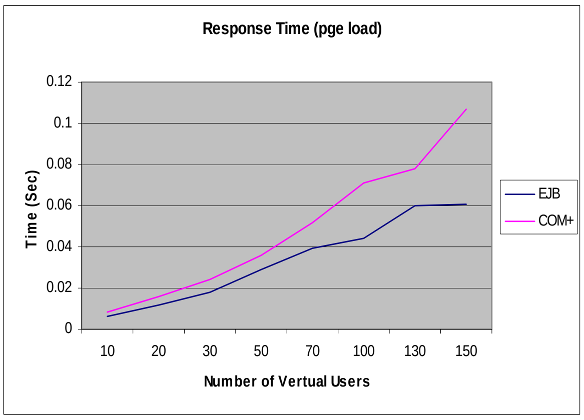

Fig. 4. Connection establishment times for EJB and COM+

4.2 Scenario 2 (Inserting Data)

We built a site to insert registration data into the database. In this scenario we only attempted to insert data into the database and not retrieve any data from the page. The scenario ran as follows: we loaded 15 virtual users for 15 seconds each. The duration of the scenario was 1 minute and the virtual users were stopped simultaneously to exit. The scenario was repeated several times on the COM+ and the EJB applications. At each run we increased the number of virtual users, and then took readings of response times and the number of transactions (i.e. number of record inserted into the registration table). The results are summarized in table (3).

Table 3. Scenario 2 results

Number of Virtual users	AVG of Response Time		Number Of transactions	
	EJB	COM+	EJB	COM+
1	0.008	0.046	6,981	1289
10	0.021	0.061	29865	25984
20	0.033	0.082	41749	52191
30	0.049	0.109	45760	65764
40	0.061	0.164	51317	70832
50	0.066	0.209	59546	80492
60	0.076	0.254	61539	82423
70	0.083	0.293	65519	91523
80	0.089	0.332	69691	96403
90	0.155	0.373	71815	103056
100	0.162	0.414	75188	108978

It can be noted from the table that the response times and the number of transactions increased dramatically as the number of virtual users were increased. Also, we can notice when the scenario has one user, the response time for COM+ was 0.046 seconds and added 1289 records, on the other hand, and the response time for EJB was 0.008 seconds and added 6981 records. This is interesting since even though EJB has incurred much more transactions than COM+, the response time was much better. This trend continues for the rest of the table with multiple users. Figure (5) shows the performance for both COM+ and EJB application for one user.

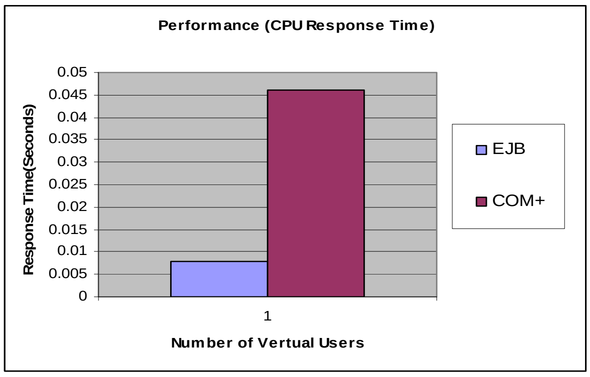

Fig. 5. Average response time for one user for scenario2

Figure (6) shows the relationship between (response time) and the (number of virtual users) for the previous scenario. In this figure, we can see that the COM+ application needed more time than the EJB application in order to complete the same process for all scenarios. Here, EJB delivers better performance than COM+. If we look closely at the figure we can also notice that when the number of virtual users increase, the time needed by COM+ increases faster than that needed by EJB in order to complete the same process for the same number of users which is an indication for the scalability of the system; it is obvious that EJB scales-up better than COM+.

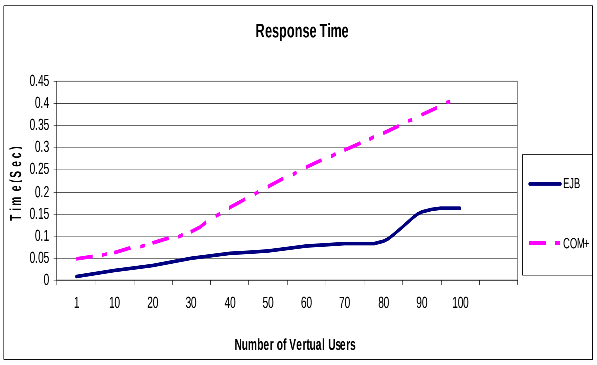

Fig. 6. Average of response times for all users for scenario2

Figure (7) shows the relationship between the number of virtual users and number of transactions that were completed. In this plot we can see that COM+ have added more records than EJB and thus incurred more time to complete the same process, and thus needed more time to complete one transaction. Figure (8) shows the average response time per transaction for one user on both middle tiers, and figure (9) represents the average response time per transaction for 100 users. It took 0.414 and 0.162 seconds for COM+ and EJB respectively, and incurred 108978 and 75188 records on COM+ and EJB respectively. Again EJB wins since it needed less time per transaction and this is another indication that the EJB applications delivers better response time and scales-up better than COM+ applications.

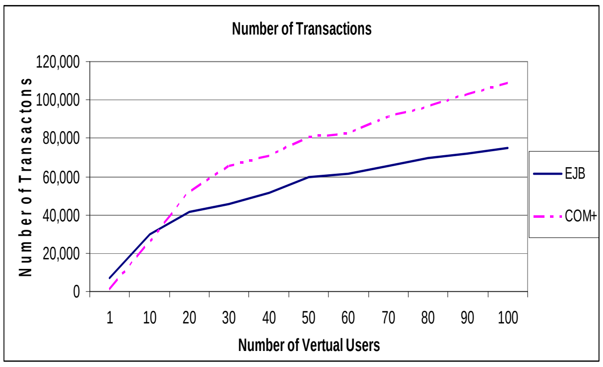

Fig. 7. Number of transaction vs. number of virtual users for scenario2

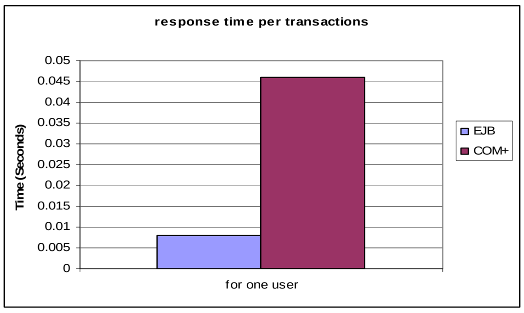

Fig. 8. Average response time per transaction for one user

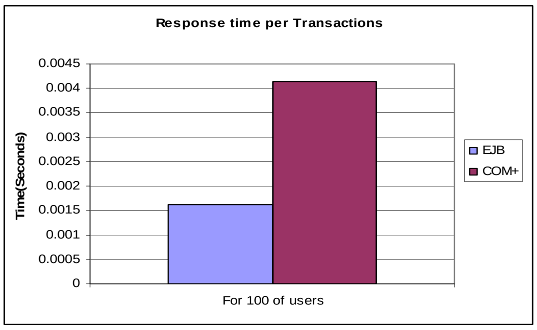

Fig. 9. Average response time per transaction for 100 users

4.3 Scenario 3 (Loading Data)

In this scenario data will be retrieved from the database to fill-in the drop-down lists on the Web page. Here, the scenario will open the page and fill these drop-down lists with values retrieved from the database tables. The duration if the scenario was 3 seconds, and the data was filled into the page without insertions of any new data. The virtual users were loaded simultaneously and stopped simultaneously too. Table (4) summarizes the results of this scenario in terms of response time and standard deviation.

Table 4. Average response time and standard deviation for scenario 3

Number of virtual users	Std deviation of response time		Average response time	
	EJB	COM+	EJB	COM+
1	0.008	0.002	0.014	0.027
10	0.118	0.128	0.084	0.155
20	0.156	0.205	0.164	0.282
30	0.175	0.307	0.247	0.419
40	0.268	0.227	0.315	0.583
50	0.344	0.287	0.408	0.71

In this scenario the maximum number of virtual users was 50. It can be seen that the performance of the EJB application was better than that of the COM+ application; when we had one virtual user the EJB application needed 0.014 seconds to respond to the user (e.g. load the page), while the COM+ application needed 0.027 seconds to load the page. Figure (10) shows the response time while varying the number of virtual users from 1 to 50 on both EJB and COM+.

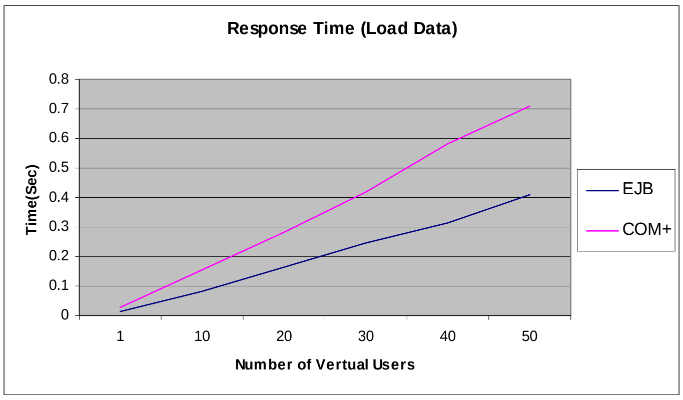

Fig. 20. Average of response times for scenario 3

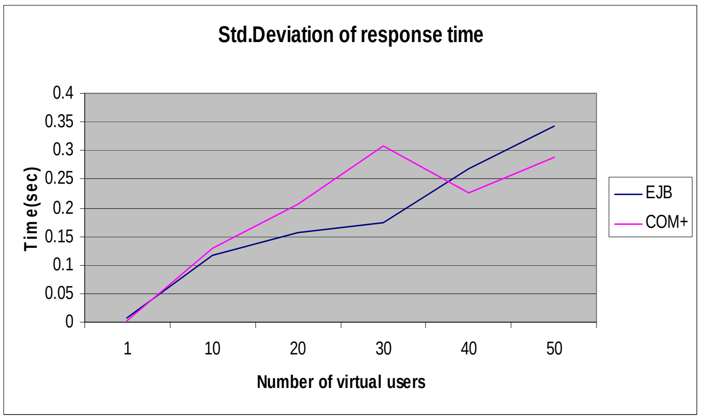

Fig. 31. Standard deviation of response time for scenario 3

For this scenario we take a standard deviation for response time, standard deviation measures the spread of the data about the mean value. It is useful in comparing sets of data which may have the same mean but a different range. Figure (11) plots the standard deviation values that we have presented in Table (4). In this figure cannot see a steady pattern or consistency on the COM+ application. This is reflected in the leap from top to bottom around 30 users. This is not the case on EJB applications where the growth can be characterized as being stable.

5 Conclusion

In this paper we have presented the results of a real experiment that we have conducted in the lab to study the performance of the middle tiers (business logic tiers) of two prominent distributed applications: The EJB technology of Sun's J2EE applications, and the COM+ technology of Microsoft's .NET applications. The experiments aimed at measuring the performance in terms of response-time and scalability of each technology using three scenarios (page load, data insertion, and data loading) and by varying the number of virtual users and the workloads in each scenario. It was observed that in all three scenarios EJB applications have demonstrated better performance in terms of response time, scalability. In our opinion this is due to the fact that COM+ environment is more heavy-weight than the EJB as it needs to load (and re-load) many supportive and system files for each session regardless of the user load.

References

1. Singer, J.: “JVM versus CLR: A Comparative Study”. In Proc. of 2nd International conference on Principles and practice of programming in Java. Kilkenny City, Ireland, (2003)
2. Cecche, E., Marguerite, J., Zwaenepoel W.: Performance and Scalability of EJB Applications. ACM SIGPLAN Notices. (2002)
3. Sessions , R.: Java 2 Enterprise Edition (J2EE) versus The .NET Platform Two Visions. ObjectWatch. (2001)
4. Kárason, R.: Dependability aspects of COM+ and EJB in multi-tiered distributed systems. Springer. (2002)
5. Hirschfeld, R.: Three-Tier Distribution Architecture. (1996)
6. Eddon, G.: COM+: The Evolution of Component Services. IEEE. (1999). ISSN: 0018-9162
7. Voth, G. R., Kindel , C. , Fujioka, J.: Distributed Application Development For Three-Tier Architectures: Microsoft On Windows. IEEE Internet Computing, (1998)
8. Kambhampaty, S., Modali , V.S., : Performance Modeling for Web based J2EE and .NET Applications: In Proc. of world Academy of Science, Engineering and Technology, V 8, Oct. 2005, ISSN 1307-6884.
9. Microsoft: Improving Performance and Scalability. Patterns and Practices. Microsoft Corporation ISBN 0-7356-1851-8
10. J.D. Meier, S. Vasireddy, A. Babbar, A.Mackman, “Improving .NET Application Performance and Scalability, Patterns & Practices”.
11. Kounev, S., Buchmann, A.: Performance Modeling and Evaluation of Large-Scale J2EE Applications. In Proc. of Computer Measurement Group (CMG 2003), Dallas, Texas, (2003)
12. 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
13. Hamed, O., Kafri, N.: Performance Testing for Web Based Applications Architectures (.NET vs. Java EE). In Proc. of Networked Digital Technologies, 2009. NDT '09. Ostrava, Czech Republic. IEEE pp. 218 – 224. ISBN: 978-1-4244-4614-8 (2009)
14. Hamed, O., Kafri, N.: Performance Prediction of Web Based Application Architectures Case Study: .NET vs. Java EE. International Journal of Web Applications (IJWA), Volume 1 (2009).