An Empirical Performance Metrics Measurement and Analysis of Software Platforms for Implementation of Web Services

K.Velmurugan
Assistant Professor
Department of CSE, AAMEC
Anna University, Trichy, India.
E-mail: tnkvel@yahoo.co.in

M.A.Maluk Mohamed
Professor
Department of CSE, MAMCE
Anna University, Trichy, India.
E-mail: malukmd@gmail.com

Abstract

Web services play a vital role in the paradigms of distributed computing and grid computing. Web services can be implemented in different platforms, but the most commonly used platforms are .NET and J2EE as they provide a wide variety of tools for creation and integration of web services to any existing business application. The selection of appropriate platform for implementation is purely based on performance offered by the particular platform. This paper focuses on analyzing the performance of a generic model composed of three tiers for implementing and consuming web services. This generic model is implemented with four different combinations of .NET and J2EE in different tiers. The analysis of performance is carried out with a novel set of four performance metrics proposed and the metrics are based on time spent for actual response and network traffic involved between the different tiers. The performance metrics measurement is observed on the model to study performance of platforms deployed for implementation of web services.

1. Introduction

One of the leading programming techniques for the development of distributed applications is the use of web services (WS) [1]. Web services are viewed in different aspects and defined in different ways based on their characteristics. Web services (WSs) are modular, self-describing, loosely-coupled, platform and programming language-agnostic software applications that can be advertised, located and used across the internet using a set of standards such as SOAP, WSDL and UDDI [2]. The web services paradigm provides organizations with an environment to enhance B2B communications [3]. Web services implement capabilities that are available to other applications or even other web services via industry standard networks and application interfaces and protocols [4].

Recent business applications involve the maximum contribution of web services, because of their nature of being flexible, interoperable and other such features. Since importance and applications of web services in computing thrive in large scale, research in web services, particularly performance metrics analysis of platforms for web services, has become indispensable.

Web services and SOAP-based connections do not currently have the key building blocks for industrial-strength web-based e-business [5]. Messaging, transactions, security and identity are key ingredients for web commerce [5]. .Net and J2EE add capability for messaging, security, identity and transactions to loosely-coupled networks based on SOAP [5]. Moreover .NET and J2EE are the two leading technologies in enterprise-level application development [6]. They are also the platform of choice for developing web services [6].

Performance is one of the key requirements of any software product and it is the foremost factor considered in the arena of computing and also in user satisfaction. The design of systems with better performance and scalability is a real need to fulfill user demands and generate profitable web services [7]. Software Performance Engineering practices have shown that quantitative performance analysis both in scientific (GRID systems) and business-oriented (Web-based systems) environments is essential to satisfy performance requisites [1]. As far as the performance of web service based systems is concerned, the support provided by the web service framework (the software library that facilitates the conversion of services into web services) is of vital importance [8].

A performance study based on a model, instead of looking at performance of particular aspect of a system, is a better way to evaluate performance of platforms for implementation of web services as a whole that is system's approach.

Based on the above considerations, this paper focuses on performance aspects of the platforms J2EE and .Net, by deploying them in different combinations, for realizing and consuming web services.

The rest of this paper is organized as follows: Section 2 provides related works in analyzing performance of web services. Section 3 describes the model, realizations and specifications of systems used for experimentation. Section 4 presents a set of metrics proposed, their significance and results of experimentation. Section 5 presents conclusions and future work.

2. Related works

In the context of performance of web services, several works had already been carried out with respect to different aspects of web services. In this work, web service is implemented and consumed by deploying the platforms in different combinations and performance is analyzed with a novel set of metrics, when compared to works discussed below.

Considerable efforts had been spent on studying performance of implementation of simple object access protocol (SOAP) messages [8][9][10][11]. [8] identified that the two factors, complexity of SOAP message and size of payload, affects the RTT of SOAP messages. Marin Litoiu [9] show results about the latency and scalability of Apache's implementation of SOAP, compare it with the performance of middleware such as RMI and look at end-to-end performance of web services built on top of existing EJB applications. The paper [10] proposes a standard benchmark suite for quantifying, comparing and contrasting the performance of SOAP implementations under a wide range of representative use cases. A new approach was proposed by Lei Li et al., to increase web services' performance by incorporating SOAP processor into SOAP engine [11]. The above works concentrated on performance of SOAP messages whereas this work approaches performance of web services as a whole.

Another key aspect is the platform in which web service is implemented. It is already discussed that .Net and J2EE are the two platforms which provides better support for web services. Some works concentrated on comparing these two platforms for web services implementation [6][12][13][14].

Sun micro systems Inc. [12] considered the performance of web service technologies in the two primary middleware platforms J2EE and .Net and concluded that J2EE outperformed .NET. [13] implements a web-based system with four architectures using J2EE and web services and analyze their performance. Sandeep Kachru et al. compare the web services development process in IBM's websphere (for J2EE) and Microsoft's visual studio .Net and find them remarkably similar [6]. The results from Sanjay P.Ahuja et al. provide an unbiased comparison of the two platforms(J2EE and .Net) based on their features and services offered from the viewpoint of developers in the context of building an enterprise or web application from design right through to deployment [14]. In contrast to the above works, this work deploys J2EE and .NET in different permutations for implementing and consuming web services.

Some other works have been done to explore other aspects of web services' performance like scheduling, semantic web etc [15][16][17][18]. [15] evaluated different scheduling policies for improving composite web service (CWS) performance in overload situations. Shiping Chen et al. give a picture of the current web services performance behaviors and developed a simple performance model that can be used to estimate web service latencies [16]. SangJeong Lee et al. present a novel deployment-time binding selection framework for Web services to improve the performance [17]. Semantic web concepts and ontologies were used in the process of web services matchmaking to gain performance by using knowledge acquisition algorithm [18].

Attempts were made to increase web services' performance by incorporating use of protocols, data structures etc [19][20]. File Transfer protocol was used to transfer data to optimize web services' performance [19]. Zhumin chen et al. introduce an approach based on a data structure called Double Parameter Inverted File to discover and compose web services according to input and output of service request and ensured performance of web services composition [20].

Few papers evaluated use of web services for other applications for increasing performance [21][22]. [21] evaluates the performance of web services and SNMP based notifications considering network usage and delivery delay. The paper [22] investigates the performance of web services compositions for network management considering response time and network traffic.

None of the above works assessed the performance of platforms deployed for web services in the lower level, as it is done in this work. Besides, this work focuses on a model implemented with different software platforms and a unique set of metrics for performance measurement of web services in terms of time and network traffic.

3. Experimentation

3.1 Proposed novel metric measurement

The proposed novel metric measurement concentrates on measurement at the level of packets in the network, so it exactly reflects the performance of software platforms used in the model for realizing and consuming web services. Also the classical performance metrics mostly focus on response time measurement only, which may not reveal actual internal communication activities involved in consuming web services.

3.2 Model used for experimentation

3.2.1 Multi-tier design

The multi-tier system design is better suitable for any business application. It involves the logical separation of the tasks that formulates the application. In the model shown in figure 1, the web service server acts as a service provider which provides many standard generic web services. The application server is simply a web server that contains the web applications which in turn consumes the web services provided by the web service server.

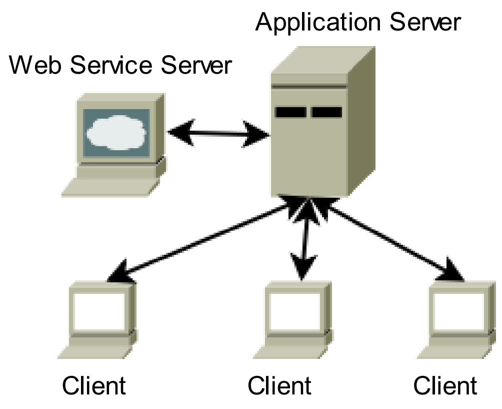

Figure 1. System model

The following three components serve as the various tiers for the model given in figure 1.

- **Web Service Server:** Web service server acts as the container for web services. It provides WSDL document for the services rendered by the web service server.
- **Application Server:** It implements user application which consumes web services described by WSDL document (provided in the web service server) through client proxy. The application server includes the logic which uses the web service. The application server provides a generic http based web application which can be accessed from any other systems connected in the network.
- **Clients:** Clients are the systems connected to this network through which user can access the application served by the application server.

The generic model is realized with different platforms in different tiers as shown in Table 1. Each approach is tested with single common web service application for studying its performance.

Table 1. Realizations of model

Realizations of model using combinations of platforms	Web Service Server	Application Server
RI	Apache Axis (J2EE)	Glass Fish (J2EE)
RII	Apache Axis (J2EE)	IIS 6.0 (.NET 2.0)
RIII	IIS 6.0 (.NET 2.0)	IIS 6.0 (.NET 2.0)
RIV	IIS 6.0 (.NET 2.0)	Glass Fish (J2EE)

3.3 Test case

In this work, an application is developed for storing and retrieving circular messages for different departments of an engineering college. Web service server implements and provides a web method which returns a text message for a request. It contains WSDL document for the method stated above. The application server provides an application which consumes this web service.

3.4 Deployment environment

Deployment environment consists of two machines, one for web services provider and other for application server. Thin clients access the application from the application server. These systems are connected by means of 100 MB Ethernet LAN. The configurations of the systems stated above are listed in Table 2.

Table 2. Deployment environment

System	Hardware	Software
Web Service Server, Application server, Client	Intel Core 2 Duo Processor E6550 2.33 GHz , 4 MB L2 Cache, 1.33 Mhz FSB, 1 GB RAM	Windows XP SP2., IIS 6.0 Netbeans 6.0 (Apache Axis , Glass Fish)

Wireshark, network protocol analyzer, is used for analyzing performance based on network traffic involved in communication between different tiers for materializing web services in this model.

4. Results of experimentation

4.1 Metrics proposed for performance measurement

The performance of web services is analyzed through the following metrics:

1. Actual response time (ART): time spent between requests originated from client, reaches the web service server through application server, for which response being delivered to Application server and acknowledgement reaches web service server.
2. Number of packets, HTTP requests and responses, from client to application server for a single request (NOPR).
3. Number of packets, HTTP requests and responses, for one complete request - response cycle on a web service from an application server (NOPRAR).
4. Number of packets, HTTP requests and responses, from application server to web service server for acknowledgment (NOPA).

The reason for choosing these four metrics is that they cover complete life cycle of accessing web services (i.e) from origination of request from client, consuming web service and till acknowledgement reaches the server.

4.2 Significance of the metrics used

- The Actual response time measured here involves only with the time elapsed for web service request, web service response and its acknowledgement in the network. The time is measured in the link layer of the OSI model which provides valid time spent for communication, independent of the delays by the transformation in the other layers. So this metric provides the real performance of the platform for web service.
- The second metric measures effectiveness of technique (number of packets) used by the platforms (J2EE and .NET) for handling a request call.
- The third metric deals with the packets used for communicating the request from application server and response of the web service from the web service server.
- The fourth metric deals with the packets transmitted for acknowledgement, as it serves as the termination point of transaction.

4.3 Actual response time (ART)

The Actual response time includes the activities shown by the solid line arrows in the figure 2. The application specific transactions (indicated by dotted line box shown in the figure 2) have not been included in the measurement of the actual response time, because it is based on user interface created for client and does not depend on either platform deployed for web service or realization and consumption of web service.

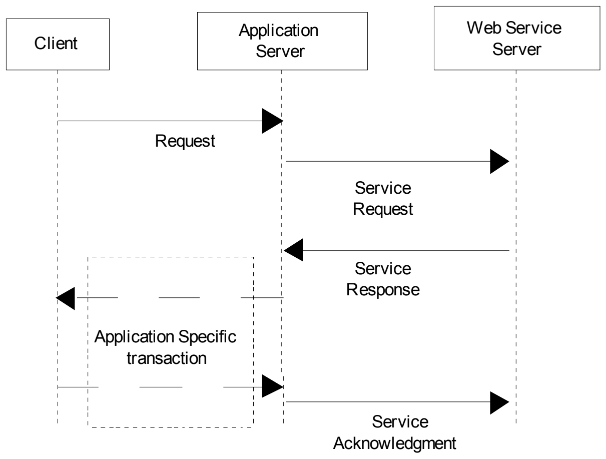

Figure 2. Sequence diagram for actual response time.

Table 3. Actual response time (ART) metric values for different realizations

Realization	RI	RII	RIII	RIV
Time Taken (in Sec)	0.013	0.004	0.002	0.055

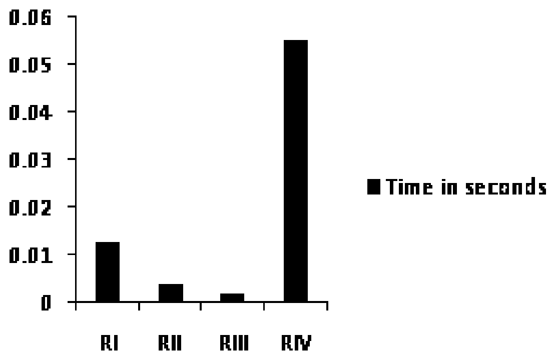

Figure 3. Actual response time (ART) metric values for different realizations

The performance analysis based on the metric actual response time (ART) shows that the Realization type III (RIII) using .Net for both web service server and application server consumes optimum time.

The Realization RI and RII consume time slightly more than RIII but RIV takes more time compared to other realizations.

4.4 Request from client (NOPR)

This metric evaluates performance by measuring number of packets used for initiating request from client to the application server. Whenever the application is loaded in the client, it receives a copy of the user interface in the html/script format.

Whenever user fires an action, which depends on a web service, this action/request should be communicated to the application server and this communication is measured by this metric.

Table 4. NOPR metric values for different realizations

Realization	RI	RII	RIII	RIV
TCP	3	0	2	10
HTTP	1	2	2	1

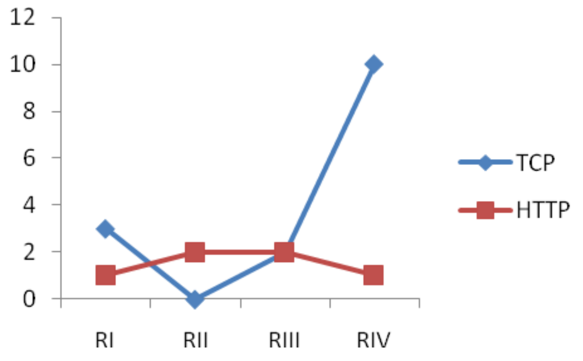

Figure 4. NOPR metric values for different realizations.

In case of metric, request from client (NOPR), the RII consumes optimum number of packets whereas RI and RIII take more than RII but RIV takes maximum number of packets.

4.5 Service request - response cycle (NOPRAR)

Here the application server initiates a transaction with the service server to get the response from the actual service from the port specified by the WSDL document. This takes place after the client's request reached the application server. The number of packets used in this communication is measured by this metric.

Table 5. NOPRAR metric values for different realizations

Realization	RI	RII	RIII	RIV
TCP	7	3	1	6
HTTP	2	3	3	3

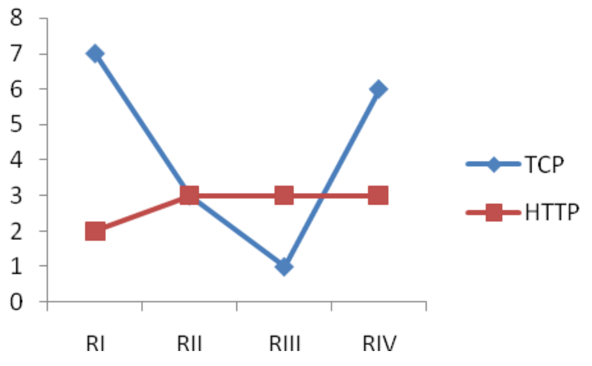

Figure 5. NOPRAR metric values for different realizations.

With respect to the metric number of packets consumed for service request - response cycle (NOPRAR), RIII consumes less number of packets whereas RII deviates slightly higher from RIII, but RI and RIV consumes more number of packets than RIII.

4.6 Acknowledgement from the application (NOPA)

This is the termination point of the entire transaction. Here the application server acknowledges the web service server and number of packets involved in this communication are measured by this metric.

Table 6. NOPA metric values for different realizations

Realization	RI	RII	RIII	RIV
TCP	4	2	1	5
HTTP	0	0	0	0

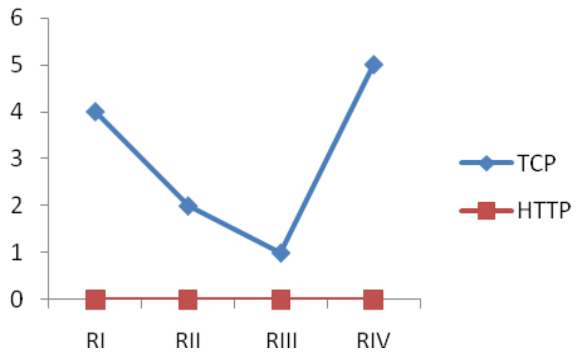

Figure 6. NOPA metric values for different realizations.

In case of the metric, number of packets consumed for Acknowledgement (NOPA), the performance is very similar to the results obtained in metric service request- response cycle (NOPRAR).

5. Conclusions and future work

The experimentations carried out on the model, by using the metrics introduced in this work, revealed that .NET in both application and web service server uses less number of packets with minimum amount of actual response time for realizing and consuming web services. This helps in lowering traffic load for consuming web services. The reduction in traffic load not only causes reduction in time and also helps in preventing congestion, to certain extent, in the network. In total, .NET in both the servers provides optimum performance compared to other realizations of web services discussed in this work.

This work may be extended both based on software platforms deployed and metrics. Based on the software platforms used in the model, platform-oriented performance enhancement techniques may be incorporated in each realization to provide better performance. The reasons for each result may further be analyzed in the context of performance. A new set of metrics may be added to evaluate the platforms deployed in the model and its performance. Further studies may be carried out based on more Quality of Service (QoS) parameters for web services.

6. Acknowledgement

The authors express thanks to Mr.S.Thiagarajan for his invaluable assistance in this work.

7. References

[1] Emilio Mancini et al., "Performance-driven Development of a web services Application using MetaPL/HeSSE", Proceedings of the 13th Euromicro conference on Parallel, Distributed and Network-Based Processing (Euromicro-PDP'05), IEEE, 9-11 Feb. 2005, pages 12-19.
[2] Kyriakos Kritikos, Dimitris Plexousakis, "Semantic QoS Metric Matching", Proceedings of the European Conference on Web Services(ECOWS'06), IEEE, Dec. 2006, pages 265-274.
[3] Carolyn McGrogor , Josef Schiefer, "A Framework for Analyzing and Measuring Business Performance with Web Services", Proceedings of the IEEE International Conference on E-Commerce(CEC'03), IEEE, 24-27 June 2003, pages 405-412

[4] Sandeep chatterjee and James webber, "Developing enterprise web services, An Architect's guide", Pearson education, First Impression 2007 pages 2-3

[5] Frank P.coyle, "XML,Web services, and the Data Revolution", Pearson Education, Fourth Impression 2007, pages 12-13.

[6] Sandeep kachru and Edward F.Gehringer, "A Comparison of J2EE and .NET as platforms for Teaching web services", 34th ASEE/IEEE Frontiers in Education conference, october 20-23, 2004, vol.3, pages S3B-12-17.

[7] Adriano Pereira, Leonardo Silva, Wagner Meira Jr., Walter Santos, "Assessing the Impact of Reactive Workloads on the Performance of Web Applications", IEEE International Symposium on performance analysis of systems and software, 19-21 March 2006, pages 211-220

[8] Narada Wickramage, Sanjiva Weerawarana, "A benchmark for web service frameworks", Proceedings of the 2005 IEEE International conference on services computing(SCC'05) IEEE 2005, 11-15 July 2005, vol.1, pages 233-240.

[9] Marin Litoiu, "Migrating to Web Services – Latency and Scalability", Proceedings of the Fourth International workshop on web site Evolution(WSE'02) IEEE 2002, 2 oct. 2002, pages 13-20

[10] Michael R.Head et al., "A Benchmark suite for SOAP-based communication in Grid Web services", Proceedings of the 2005 ACM/IEEE conference on supercomputing (SC'05), 12-18 Nov. 2005, pages 19-19

[11] Lei Li et al., "High Performance Web services Based on Service-Specific SOAP Processor", Proceedings of IEEE International Conference on Web Services(ICWS'06), Sept. 2006, pages 603-610.

[12] Web Services Performance - Comparing Java 2 Enterprise Edition (J2EE platform) and .NET framework http://java.sun.com/performance/reference/whitepapers/WS_Test-1_0.pdf

[13] Yan Liu, Ian Gorton, "An Empirical Evaluation of Architectural Alternatives for J2EE and Web services", Proceedings of the 11th Asia-Pacific Engineering conference(APSEC'04) IEEE, 30 Nov - 3 Dec 2004 , pages 10-17

[14] Sanjay P.Ahuja, Raquel clark, "Comparison of Web Services Technologies from a Developer's Perspective", Proceedings of the international conference on Information Technology: Coding and Computing (ITCC'05) IEEE, vol.2 4-6 April 2005, pages 791-792

[15] Dmytro Dyachuk, Ralph Deters, "Optimizing Performance of Web Service Providers", 21st International Conference on Advanced Networking and Applications(AINA'07) IEEE, 21-23 May 2007, pages 46-53

[16] Shiping chen et al., "Evaluation and Modeling of Web Services Performance", International conference on Web Services, 2006, Sept.2006, pages 437-444

[17] SangJeong Lee et al., "Improving the Performance of Web Services Using Deployment-Time Binding Selection", IEEE International Conference on Web Services, 2007, ICWS 2007, 9-13 July 2007, Pages 159 – 167

[18] Chaitlali Gupta et al. , "Improving performance of web services query Matchmaking with automated Knowledge acquisition", IEEE/WIC/ACM International conference on Web Intelligence, 2-5 nov. 2007 pages 559-563.

[19] Tanakorn Wichaiwong, Chuleerat Jaruskulchai, "A Simple Approach to Optimize Web Services' Performance" IEEE, Third International Conference on Next Generation Web Services 2007, NWeSp 2007, 29-31 Oct. 2007, pages 43-48

[20] Zhumin Chen, Jun Ma, Ling Song, Li Lian, " An Efficient Approach to Web Services Discovery and Composition when Large Scale Services are Available", Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing(APSCC'06) IEEE, Dec. 2006, pages 34-41

[21] Weldson Queiroz de Lima et al., "Evaluating the Performance of SNMP and Web Services Notifications", IEEE, 10th IEEE/IFIP/Network operations and Management Symposium 2006. pages 546-556

[22] Ricardo Lemos Vianna et al., "Evaluating the Performance of Web Services Composition for Network Management", IEEE International conference on Communications 1007(ICC'07), 24-28 June 2007, pages 1943-1948