Comparative Study of Web Services Platforms in a GUISET Environment

Ijeoma N. Mba, Matthew O. Adigun
Department of Computer Science
University of Zululand,
Tel: +27 35 9026067, Fax: +27 35 9026569
email: ijaymba@gmail.com; madigun@pan.uzulu.ac.za

Abstract – GUISET (Grid-based Utility Infrastructure for SMME Enabling Technology) architecture aims at delivering e-services to resource-constrained Small Medium and Micro Enterprises (SMMEs) by taking advantage of advances and trends toward Service Oriented Architecture (SOA) within the software industry. A technology that has received widespread adoption within SOA for implementing services is Web Services (WS). However, Web Services’ success has been facilitated by implementation of core WS-Standards into enterprise application development platforms and tools within both the proprietary and Open Source Communities. Although, for the target group of GUISET (SMMEs), the Open Source platforms seem a natural choice as proprietary tools are often expensive and are out of their reach. But, even within the open source communities are many platforms with different characteristics and features. A major challenge is how to choose the appropriate platform that is suitable for GUISET services scenario. One question that is crucial to addressing this challenge (especially within the context of GUISET): can a comparative study of existing Web Services development platforms be used in recommending one for GUISET services? This is the central goal of this research. Our aim is to test the performance of a few leading open source web service platforms with a view to determine their suitability for developing GUISET services under an e-commerce scenario and use to recommend one for GUISET services.

Keywords - Performance, Web Services, Tools, SOA and SMME

I Introduction

There is growing recognition in the role of Service Oriented Computing as it regards SOA and Web services in the realization of reusability of assets, interoperability and flexibility among platforms and other technologies as well as reduction in both operational and developmental cost. In addition, Web services is a middleware technology for the purposes of integrating enterprise applications over the Internet which is achieved by its support for loosely coupled SOA and messaging, furthermore allowing heterogeneous systems and applications to be developed in different languages and technologies to interoperate with one another [1]. Various applications in these areas such as e-commerce, scientific computing, grid computing, utility computing and finance have been exposed as Web services. Owing to this, many platforms exist for Web services implementation. The question that might arise is what level of performance is shown by most Web service implementations?

There are so many web service development toolkits available today both in the market place and the open source community. The widespread availability of these products has lead product vendors and all other participants to compete against each for market share. Against this background, it is difficult to make a choice on one that suit the requirements of GUISET as an infrastructure or any other infrastructure which is meant to basically serve the needs of Small Medium and Micro Enterprises (SMME).

In this paper, we make a comparative study of some Web Service Platforms which can also be referred to as Web service engines that can be used in deploying web services for GUISET on demand service delivery services. The paper is structured as follows: section 2 gives a background of this work as it concerns Service oriented Computing and section 3, the GUISET project and its need for a web service development platform. Related work is introduced in section 4. Criteria for selection of each web service platform, the choice of benchmark suite and the development environment and overview of the SOAP engines that were considered are discussed in Section 5. The next section then shows the qualitative analysis and section 7 describes the quantitative comparison, its methodology and test environment. Section 8 discusses the results and the paper is then concluded with recommendation for future work in section 9.

II Background

Service Oriented Architectures are getting widely adopted as businesses tend to be more responsive and flexible in their communication and meeting customer’s need through new and existing technologies [2]. SOA enables information sources and software functionality to be delivered as individual business distinct service units, which are distributed over a network and are also combined to create business application which helps in solving complex problems. SOA is implemented with the Web service idea. Web services as well as grid services are attracting significant industry interest as a low cost and flexible technology alternative for delivery of on-demand business processes. As these technologies become well
accepted, so is the platform for its development and deployment and Web Service Platform is one of such tools. Web Service Engines is a framework for constructing SOAP processors such as clients, servers, gateways [3] that transforms SOAP messages into invocations and vice versa. Therefore it is considered a necessary requirement for web service invocation [4]. The SOAP engine layer is a core layer of any Web Service Provider toolkit [5]. Web Service engines and tools provide a way for data compression, data representation efficiency and efficient processing of XML. The source code of an open source SOAP engine facilitates integration with the software product.

SOAP messages are simple XML documents [6]. A SOAP engine has to do more than process SOAP messages. For example, WSDL documents have to be created, requests must be dispatched and the lifecycle of service objects has to be managed. It gets even more complicated because new standards and requirements also apply.

III The GUISET Architecture

GUISET which is short for Grid Based Utility infrastructure for SMME-Enabling Technology is a research project based on the concept of Services (Service Oriented Architecture, web Services Grid Computing and Utility Computing) and e-commerce. According to Kabanda et-al [7], GUISET is motivated by the emerging Service Oriented Architecture and ongoing technological convergence between grid services and web services which is creating trends towards Information technology (IT) service provisioning as utilities and the provision of such a technology that will be affordable for Small Medium and Micro Enterprises. The GUISET research project envisaged a future where small businesses (SMME) will act as service providers and provide services at a cost to its clients based on quality of service requirements [8].

The main idea of GUISET is to provide an e-infrastructure which would enable SMMEs to pool their resources and expertise together for the sharing and collaboration among themselves and their partners. The proposed GUISET architecture categorically stated the need for a concrete application building and deployment infrastructure. However for GUISET to effectively function in meeting the business needs of the target SMMEs, it must be built deployed upon a development environment that would fulfill the needs of SMMEs. In fulfilling the needs of SMMEs and eliminating the barriers to successful implementation of IT for small businesses comes the GUISET Architecture.

GUISET architecture as shown in Figure 1 is a typical Service Oriented Architecture with three layers that uses resources from a pool to grant service requests. These layers are multimodal interfaces layer, Middleware layer and Grid infrastructure layer.

IV Related Work

Performance evaluation of Web service tools and other software tools has been compared in different ways. Most of the work done on this performance is basically to identify areas that affect the execution time between the web service client and server.

Msimanga and Chadwick [9] presented an evaluation of Free and open source e-commerce web application technology with regard to SMMEs was carried out to help SMME looking for a means on Internet-enabling business. They found out that software’s evaluated were all good though most of them are meant for highly skilled person.

Padmanabhuni et-al [10] compared of XML Binding Frameworks in the Context of Service-Oriented Architecture. The aim of their evaluation was to come up with the role XML binding frameworks play in the context of Service Oriented Architecture (SOA) platforms and provide an objective evaluation of popular XML binding frameworks in J2EE environment. They concluded with it will enable practitioners to make an informed choice about the appropriate XML binding platform.

Kunti et-al [11], did a performance analysis of three different Web Services Engines/platforms combined with several binding frameworks based on First, Second and third generation Web Service Engines. The performance analysis studies was limited to binding performance, round trip performance, scalability speed and throughput which are directly affected by factors such as the binding framework used and the parsing model of the SOAP engine. Their aim was to highlight the relative merits in using specific platforms for specific scenarios.

Suzumura et-al [12] did a performance comparison of Web Service Engines in three different programming Languages which are PHP, Java and C. Their results show that PHP is reasonably high performing compared to Java and C-based implementations while providing users with high software productivity.

This study by Govindaraju et-al [13] compared the performance of some widely used SOAP toolkits for various workloads commonly used in scientific computing like varying array sizes of doubles, integers and strings. According to the study, it was stated that performance study for scientific data structures will aid in the design and development of new toolkits and also guide users in choosing appropriate toolkits for their current application requirement. Their objective is based on identifying the features of SOAP that affects Web Service performance.

![Figure 1: GUISET Architecture [8]](image-url)
The performance testing approach applied by Tripathi et al [14] is based on testing both the simulated environment and the actual hosted environment. This was done by comparing with performance metrics like response time, throughput for web services which they argued that it will help developers in early life cycle of web services and also in tuning the applications before putting it out for proper use. The performance measurements suggested that from modelling perspective web service can be stimulated first and tested for various performance metrics, which would also give results close to the original one

V Selection Criteria
The criteria for inclusion of any particular platform for comparison are as follows: User Friendliness and Ease Of Installation; In identifying the user friendliness features of these Web services stacks, the configuration, deployment and installation components and patterns were observed. Availability Of Technical Information; Availability refers, unsurprisingly, to the availability of detailed information resources. Continued Development; how much it support continuous technology change. Commercial Relevance which examined the extent of its usage by both individuals and businesses. Moreover, it must be Free and Open Source.

There were a lot of contending web service platforms but based on the above mentioned criteria’s, these three were chosen for evaluation; Axis2 [15], CXF[16] and Metro[17].

A Checklist and Summary of Selection Criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Axis</th>
<th>CXF</th>
<th>Metro</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Friendliness</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Availability Of Technical</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continued Development</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Free and Open Source/Latest</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commercial Relevance</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

D Overview of Web Services Stack under investigation
Axis2 is an open source project that is a core engine for Web services whose main design criteria is flexibility and extensibility. It is a complete re-designs and re-write of the widely used Apache Axis SOAP stack. Apache Axis2 is a flexible and powerful third generation web services stack which was rebuilt to more performance awareness and flexibility to support an assortment of Web Services standards as well as asynchronous web services. It also has support for REST. Its unique server platform provides service monitoring functions and allows easy deployment and management of services. Implementations of Axis2 are available in Java and C. Axis2 not only provides the capability to add Web services interfaces to Web applications, but can also function as a standalone server application.

CFX is an open source services framework which helps in building and developing services by using frontend programming APIs, like JAX-WS and JAX-RS and also provides an easy to use, standard-based programming model for developing web services. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI. It actually includes a broad feature set, but it is primarily focused on Web Services Standards Support and Frontends. Apache CXF builds on a highly-configurable architecture to support a range of transports, data bindings, and extension technologies. Its use of widely-used Spring Framework
configuration options makes it an especially appealing choice to organizations using Spring for their application development, and CXF combines this with great support for JAX-WS service configuration.

**Metro** which is also referred as JAXWS Reference Implementation develops and evolves the code base for the reference implementation of the Java API for XML Web Services (JAX-WS) specification. The current code base supports JAX-WS 2.x Web Services Standards and JAX 2.x data binding but it also uses additional components to provide features beyond the basic support as defined by JAX-WS. Unlike the other toolkits, the only XML binding mechanism supported by Metro is JAXB, although others could be added.

Sun/Oracle Metro is the open source stack based on the reference implementations of the JAXB 2.x and JAX-WS 2.x Java standards. It doesn’t provide the flexibility of Axis2 or CXF, but its close ties to the developers supporting the official version of these standards and focus on .Net interoperability makes it a strong choice for many organizations.

### VI Qualitative Comparison

In view of performing qualitative benchmarking of the selected platforms, the word quality is extensively looked at to properly denote its meaning to the concept of this work. Therefore as was defined by Crosby, it is a conformance to requirement and its measurement is defined in terms of measuring products against a set of defined requirements and determining the level of conformance [20]. In order to effectively report on this aspect of comparison, the requirements of Web service platform that this work is considering are: the software architecture, XML Parser and Processing Model, Programming Model and Deployment Model, the standard support and other features.

### VII. Quantitative Comparison

This research work takes the approach of measuring the time required to execute a particular sequence of requests when both the client and server are running on a single a machine which gave room to eliminate impact of network latencies and overhead from timing results. Each test case was put through standard procedure to quantitatively measure its performance and give a fair basis for a comparative analysis.

#### A Testing Methodology

In performing these tests, a simple e-commerce application was designed and deployed using each of the Web service engine. This e-commerce application consisted of a web service that took an array of object as parameter and return the same to the client. The schema used is representing a collection of three products with different price tags.

The procedure was designed to measure the performance of each case with different product value ranging from one product to up-to one million products where a quotation service is obtained from a database of prices of three different products.

In this work, there are sets of request sequences were used for performance tests, The first set used requests with query parameters adjusted to match the information on the database. A single client was used to invoke the Web service and the time taken to unmarshal the incoming message into objects, marshal objects into message, and complete a roundtrip were determined.
B Testing Environment

This section describes the test environment that was used for the performance measurements. The benchmarking suite used is SOAPUI which is an open source functional testing tool for SOA and Web Service testing. The test system consisted of an integration of the benchmark suite SOAPUI which is an open source functional testing tool for SOA and Web services with a development environment Netbeans and this integration consisted of operations in WSDL files along with bindings for SOAP calls and a driver that reads trace data from local files. Each of the selected SOAP engine is made to implement the operations defined in the WSDL document for benchmark. It was running on Windows XP Professional as the host operating system. The tests were carried out on a Desktop workstation with 2.79GHz Pentium 4 processor, 400MHz front-size bus and 1.24 GB of RAM. The Java code was run on tomcat. The Web service stack versions were Metro with JAXWS as the data binding framework, Axis2 with its native data binding and CXF also using JAXWS as data binding framework.

To obtain optimum results, we assumed that there is no overhead cost and network latency.

VIII Results and Discussion

Since XML parsing, de-serialization and serialization are the three most time-consuming stages in the process of handling SOAP messages, they also determine the Web Services performance to a large degree, especially when effective load is increasing. Finding out the effects of XML parsing, serialization and de-serialization are three key factors that would help in improving the performance of Web Services.

Marshalling Performance

Figure 2 shows the marshalling performance. It can be seen from Figure 2 that Metro has the least marshalling time thus the best marshalling performance. Axis and CXF match marshalling performance closely among themselves for the time taken to complete marshalling, when one product and two products were requested. Consequently, for the time taken when three different products were requested, Axis fared better than CXF.

Unmarshalling Performance

Figure 3 depicts the unmarshalling performance of the three tested web services platforms. The figure indicates that Metro outperforms CXF and Axis in all product requests while CXF follows suit by outperforming Axis also in all request. The difference in Axis and CXF when one product was requested is minimal. All three web service engines uses pull parser model. The pull parser model has been optimized for speed and performance and also allows for validation to be easily built into the generated marshalling code thus the implication of level of speed on unmarshalling performance due to the parsing model is eliminated. Therefore other factors could then cause either one to be better than the other. One attribute could be the

Round Trip Performance

In figure 3, the chart clearly shows that metro is the fastest of the three as it provides the best round trip performance result. Axis and CXF provide similar round-trip latency results.
IX Conclusion and Future Work

In this paper, performance evaluations of three commonly used and available web service platforms were presented both qualitatively and quantitatively. From the qualitative point of view, one would say that none of the Web Services framework is in general superior to the other, even though one can be better in some aspect than the other. Axii2 is structured modularly, has many features and can be used as an application server for Web services. One of its special features is its support of removable data binding frameworks. Its disadvantages come in its complexity and insufficient JAXWS support. CXF and Metro comes complete with JAXWS support which allow for seamless integration with spring framework. CXF is slim and easy to use. It is said to be the tool of choice if a SOAP engine has to be embedded into existing software. The Metro Web Services stack delivers secure, reliable, transactional interoperability to help you build, deploy, and maintain Composite Applications for your Service Oriented Architecture. Metro provides ease-of-development features, support for W3C and WS-I standards as SOAP and WSDL, asynchronous client and server, and data binding through JAXB 2.0. The quantitative performance attributes a high performance rate to Metro than any other of three which is followed closely by CXF and then Axis which could be because of its use of JAXWS as data binding. Since there is a possibility of an effect on performance due to the data binding framework used, we plan to extend this research further by testing these web services stack with different flavors of pluggable data bindings.

X Reference


Mba Noella Ijeoma received her undergraduate degree in 2004 from the University of Calabar, Nigeria and is presently studying towards her Master of Science degree at the University of Zululand, South Africa. Her research interests include Service Oriented Computing, Software Tools, Web Services and Grid Computing