SENRA Academic Publishers, Burnaby, British Columbia
Vol. 5, No. 1, pp. 1449-1453, Feb 2011
Online ISSN: 1920-3853; Print ISSN: 1715-9997

Canadian Journal of pure&applied sciences

ARCHITECTURE FOR SCALABLE WEB SERVICES SOLUTION

*Godspower Ekuobase and Emmanuel Onibere
Department of Computer Science, University of Benin, Benin City, Nigeria

ABSTRACT

The fresh ambition to make computers seamlessly interoperable in dynamic and heterogeneous systems triggered the rush on Web Services by the computing community. We established that this rush may lead us into shambles unless we address the issue of poor scalability of Web Services. We have examined the technology to unveil the root cause of its poor scalability and figure out a key to improving it. Web Service support for replication was identified as one such key. We examined available replication schemes with a view to determining the one most suited to Web Services. The three tier replication scheme appears to have stood out. We have explored the possibility of making Web Service support this replication scheme in building scalable Web Services solution by application programmers. The result is proposed Replication Oriented Architecture (ROA) for building Web Services solution. ROA is unique in its application of replication to Web service in that it employs a non conventional replication technique; it is server side oriented and transparent to consumer applications. ROA solutions therefore free Web Services consumers from issues of Web Services server selection schemes. This proposed architecture however has some critical issues which were also exposed.

Keywords: Web services, software architecture, scalability, replication, web services solutions.

INTRODUCTION

The integration and interoperability problem of heterogeneous systems is not only a threat to globalization but also erodes the value we get from Information Technology (IT) as investments on IT increases (Williams, 2003; Weerawarana *et al*., 2005). Already, 60% of IT cost is absorbed by effort directed at making computers seamlessly interoperable (Williams, 2003) and some companies now receive less benefit from IT, even as they increase their spending on it, because of interoperability problem (Weerawarana *et al*., 2005).

There is however some hope as Web Service promises to provide unparalleled solution to this problem (Bichler and Lin, 2006; Weerawarana *et al*., 2005; Birman, 2005a, Hogg *et al*., 2004; Chung *et al*., 2003; Vawter and Roman, 2001). With its promises which include:

a. Reduced intra and inter-enterprise application integration effort,
b. Heterogeneous access enablement,
c. Flexibility and reusability of software components,
d. Capability for evolutionary and incremental deployment requiring no changes to current infrastructures and applications,
e. Extensibility of existing functionality,
f. Relative ease and low cost of adoption,
g. Uniformity in service description, discovery and access within a network, it is obvious that Web service is the ubiquitous platform technology for next-generation computing systems and therefore cannot avoid the massive deployment ahead of it. Can Web Service survive this unprecedented deployment?

This question is particularly interesting considering the fact that the very technology that holds the greatest promise will become the Achilles' heel of future generation systems, if it cannot survive deployment on a large scale (Birman *et al*., 2001). Besides, the Web bedeviled with similar Web Service limitations as security, reliability, guaranteed responsiveness, and self-administration (Birman, 2005a), has not only survived but has remained a toast of Broadway for about three decades now because it is scalable (Krummenacher *et al*., 2005).
Web Service could also survive unprecedented deployment as the Web, if it is scalable. Unfortunately, Web Service is inherently poorly scalable (Birman, 2005a; Birman, 2006; Ciganek *et al*. 2006). It is therefore not ready for the massive deployment its promises will attract and hence will become the Achilles' heel of future generation systems if its scalability defect is not addressed.

This paper intends to check this imminent catastrophe by proposing server-side Software Architecture that will enable Application Programmers builds scalable Web Services solution. The imperativeness of such an architecture has been stressed by Birman, (2005a, 2005b), and although he believes that it is possible, no such architecture exists to our knowledge. We are however aware of architectures that are either client side oriented or not consumer application transparent or both (Azevedo

*Corresponding author email: godspower.ekuobase@gmail.com

et al., 2003; Keidl and Kemper, 2004; Mendonca and Silva, 2005) and are basically aimed at providing alternatives for service consumer. We are also aware of the Abraham's hub function (Abraham *et al*., 2005), aimed at monitoring various alternative services and allocate request (balance load) appropriately. Although server side oriented and consumer application transparent, it cannot be extended to a set of replicated services since it does not guarantee consistency – a necessary attribute of replication (Coulouris *et al*., 2001) which is key to scalable Web Services solution.

MATERIALS AND METHODS

The basic materials for this research were published literatures. We surveyed relevant literatures in the domain of distributed computing; service oriented computing in particular and was particular about the nature and scalability of Web Services, diagnoses of the root cause of the poor scalability of Web Services, properties of the root cause, opportunities for the root cause in realizing software architecture for building scalable Web services solution as well as the intelligibility of the architecture. First, we exposed the cause of the poor scalability of Web Services with a view to figure out a key to improving its scalability. There after, efforts were made to realize an architecture, that incorporates this key into Web Services solution at the server side and, that when used by application programmers could realize scalable Web Services solution. The architecture was further examined to appreciate its potentials and flaws.

The architecture was built on the principle of replication, the third party replication mechanism in particular. Replication is a technique that helps make exact copies (replicas) of a given functionality or data simultaneously accessible and available in possibly different locations with a view to enhancing overall system performance, and ensuring service availability and fault tolerance. A replicated data or computation is expected to be transparent and consistent (Coulouris *et al*., 2001). Conventionally, replication can be passive or active (Baldoni *et al*., 2002).

Recently, Baldoni *et al*. (2002) came with the idea of separating clients, servers and replication logic of replication technique resulting in a three-tier approach to replication as shown in figure 1. They argued that the conventional (two-tier) replication techniques can not support the deployment of server replicas implementing a state full service over an asynchronous distributed system such as the internet. This argument is consistent with Birman's (2005b). Baldoni *et al*. (2002) went ahead to show that two necessary and sufficient properties for replication in state full asynchronous distributed application services are (i) Client/Server-Asynchrony and (ii) Client-Autonomy; and established that only the three-tier replication technique can satisfy these properties. The feasibility of this technique was also sufficiently demonstrated in Baldoni *et al*. (2002).

RESULTS AND DISCUSSION

From the study, it became evident that replication an unfortunately missing feature of Web service (Birman, 2004, 2005a ; Birman *et al*., 2004) is key to improving the scalability of service solutions built on the technology and can best be incorporated into such solutions by interposing replication logic as an autonomous in-direction layer to services; as depicted in figure 2.

The Proposed Web Services Third Party Replication Logic (WSTPRL) has the following merits:

a. It is in line with, besides being a better alternative to, the third party philosophy of Krummenacher *et al*. (2005).
b. Supports implementation over standard technologies based on TCP such as Internet Inter ORB Protocol (IIOP) and SOAP (Baldoni *et al*., 2002).
c. On-the-fly maintenance (adaptive, corrective or perfective) of replication logic.
d. Consistent with the loose coupling principle of service orientation (Erl, 2008).
e. The "fractal" scenario feared by Bussler, (2007) is adequately accommodated.

Baldoni *et al*. (2002) says this idea, though unique, is not strange and has been very successful in applications using gateways, say. A little wonder why Abraham *et al*. (2005) came with the excellent idea of increasing Web Services availability via central hub or gateway. This Abraham's hub function is basically a fraction of typical replication logic – to monitor various alternative services and allocate request (balance load) appropriately. However, with the distributed replication logic, no alternate services but service replicas and no single point of failure as the hub exist. Also, with this separation, the third party entity can be intrinsically equipped with diagnostic and other utility features.

The proposed WSTPRL as depicted in figure 2, consist basically of distributed replication logic and sets of interacting Web Services which may not necessarily be replicas. With this replication logic in place, a sender Web Service sends a request to a particular Web Service but this is intercepted by the WSTPRL (an autonomous distributed system) which forwards such request to the appropriate receiver Web Service (replica) according to its logic. The receiver Web Service (replica) executes requests computing the results and sends them to the WSTPRL, which finally returns them to the sender Web Service; and if a change of state of replica results, it will simultaneously update sibling replicas. This will greatly

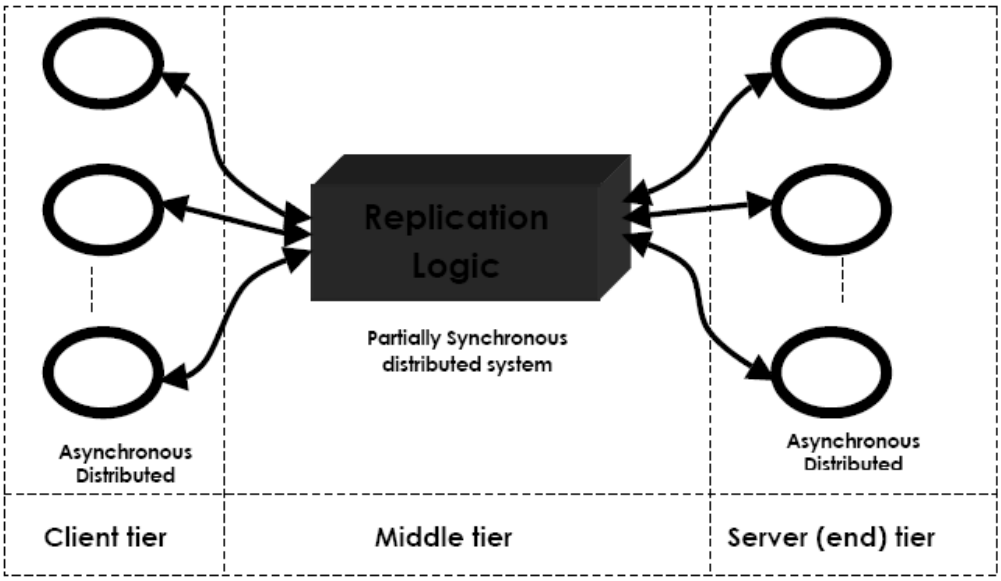

Fig. 1. Three tier Replication Architecture (Baldoni *et al*., 2002).

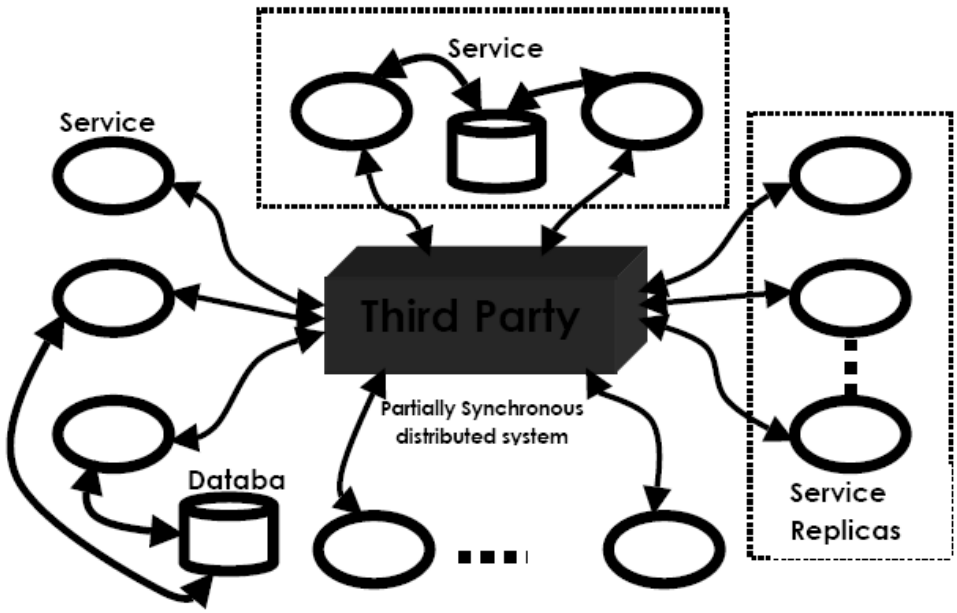

Fig. 2. Proposed Web Services Third Party Replication Logic (WSTPRL).

improve scalability of Web Services solutions and is quit flexible in view of the following promise:

a. A Web Service (respective service replica) may communicate with the WSTPRL on a point to point basis following a simple request/response asynchronous message pattern.
b. Maintains the loose coupling principle of service orientation: WSTPRL is black box to Web service (respective atomic service replica) which cannot exchange message directly among themselves.
c. Enforces services composition since composite services can only be formed from atomic services.
d. No distributed (agreement) protocol is run at the Web services end but instead by a specialized autonomous distributed WSTPRL running appropriate protocol for accuracy and efficiency of interaction, and consistency of replica states.
e. No single point of failure: in case of crash of a WSTPRL entity responsible for a given interaction, another WSTPRL entity will conclude the process.
f. Support for replication of Web Services – a necessary requirement for its large scale deployment.

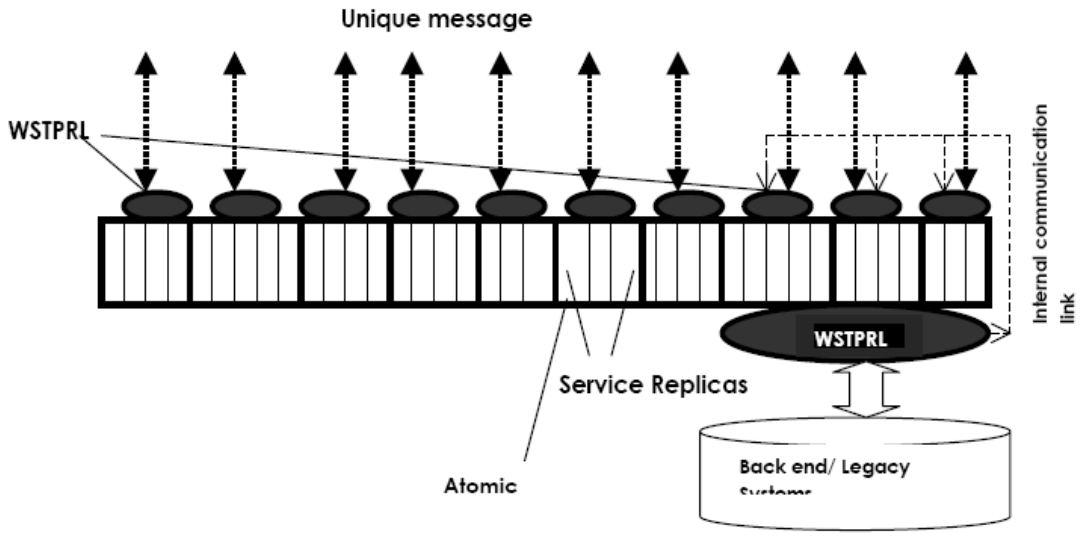

Fig. 3. Replication Oriented Architecture for Building Scalable Web Services solution.

g. Continuous and autonomic monitoring, reporting and advising, detection and/or repair of faulty component, Web services state and resource by WSTPRL.
h. Provide other utility services; say payment mechanism for service calls, where needed.
i. The WSTPRL can function also as a proxy server or front end to the Web service backend, and we know the beauty of this capability.
j. No extension or reduction of the Web services standards is required.
k. Ensure high availability and reliability: this is made possible due to its support for replication and possible autonomic management of Web service state and resource.

The WSTPRL later became a key component of the main architecture this paper is proposing – Replication Oriented Architecture (ROA) for Web Services solution. ROA consists of arrays of partitioned atomic services with each of its element being an array of service replicas. Each atomic service is assigned a light weight WSTPRL that manages its service replica and a heavy weight WSTPRL for a set of atomic services that have the same back end or legacy system, if any, as shown in figure 3. It is important to note that the list of atomic services need not be ordered or resident in the same server but the location of a Web service must be unique (Weerawarana *et al*., 2005). This is guaranteed by the deceptive stance of WSTPRL as the Web service which actually are service replicas within the same LAN or network system it manages.

The ROA architecture is unique particularly in its application of replication to Web service in the following respects:

a. It employs a non conventional replication technique.
b. It is server side oriented and transparent to consumer (applications).
c. Frees Web Services consumers from issues of Web Services server selection algorithms/policies

This fact is particularly interesting, if we observe that though we know that some important quality of service attributes can be provided for internet applications through the use of replication (Berners-Lee *et al*., 1996), efforts so far at taking advantage of this understanding in the domain of Web Services is not only rare but the few like Azevedo *et al*. (2003), Keidl and Kemper (2004), that dared are either client side oriented or not consumer (application) transparent or both; and hence the proliferation of Client-side server selection policies, and subsequent evaluation of some of these policies for accessing replicated Web Services by Mendonca and Silva (2005).

One may argue that the overhead required by WSTPRL may encumber the scalability of service solutions built on ROA particularly for services with little or no computational capability. Besides, the empiricists will want ROA implemented and have its scalability claim authenticated particularly with Web services solution that can expose possible flaws in ROA. This issue of feasibility and authenticity were addressed but reported separately.

CONCLUSIONS

Server-side Software Architecture that will enable Application Programmers build scalable Web Services

solution now exists. With this development, there is renewed hope that Web Service will survive the unprecedented deployment its promises will attract.

REFERENCES

Abraham, S., Thomas, M. and Thomas, J. 2005. Enhancing Web Services Availability. Proceedings of IEEE International Conference on e-Business Engineering. IEEE Computer. 352-355.

Azevedo, V., Pires, PF. and Mattoso, M. 2003. WebTransact-EM: A Model for Dynamic Execution of Semantically Equivalent Web Services. Brazilian Symposium on Multimedia Systems and Web. pp 9.

Baldoni, R., Marchetti, C. and Termini, A. 2002. Active Software Replication through a Three-tier Approach. Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS). IEEE Computer. pp 10.

Berners-Lee, T., Gettys, J. and Nielsen, HF. 1996. Replication and Caching Position Statement. available online at http://www.w3.org/Propagation/Activity.html.

Bichler, M. and Lin, K. 2006. Service-Oriented Computing. IEEE Computer. 39(3):99-101.

Birman, K. 2005[a]. Can Web Services Scale Up? IEEE Computer. 38(10):107-110.

Birman, K. 2006. The Untrustworthy Web Services Revolution. IEEE Computer. 39(2):98-100.

Birman, KP. 2004. Like it or not, Web Services are Distributed Objects. Communications of the ACM. 47(12):60-61.

Birman, KP. 2005[b]. Reliable Distributed Systems: Technology, Web Services, and Applications. USA: Springer-Media. pp 668.

Birman, KP., van Renesse, R. and Vogels, W. 2001. Spinglass: Secure and Scalable Communication Tools for Mission-Critical Computing. Proceedings of the DARPA Information Survivability Conference and Exposition. pp 15.

Birman, K., van Renesse, R. and Vogels, W. 2004. Adding High Availability and Autonomic Behaviour to Web Services. Proceedings of 26th International Conference on Software Engineering (ICSE). pp 10.

Bussler, C. 2007. The Fractal Nature of Web Services. IEEE Computer. 40(3):93-95.

Chung, J., Lin, K. and Mathieu, RG. 2003. Web Services Computing: Advancing Software Interoperability. IEEE Computer. 36(10):35-37.

Ciganek, AP., Haines, MN. and Haseman, W. 2006. Horizontal and Vertical Factors Influencing the Adoption of Web Services. Proceedings of the 39th International Conference on System Sciences. pp 10.

Coulouris, G., Dollimore, J. and Kindberg, T. 2001. Distributed Systems: Concepts and Design. USA. Pearson education. pp 772.

Erl, T. 2008. SOA: Principles of Service Design. USA: Prentice Hall. pp 573.

Hogg, K., Chilcott, P., Nolan, M. and Srinivasan, B. 2004. An Evaluation of Web Services in the Design of B2B Application. Proceedings of Conferences in Research and Practice in Information Technology. Australian Computer Society. 26:331-340.

Keidl, M. and Kemper, A. 2004. A Framework for Context-Aware Adaptable Web Services. Proceedings of 9th International Conference on Extending Database Technology. 288-293.

Krummenacher, R., Hepp, M., Pollere, A., Bussler, C. and Fensel, D. 2005. WWW or What is Wrong with Web Services. Proceedings of the 3rd European Conference on Web services (ECOWS'05). IEEE Computer. pp 9.

Mendonca, NC. and Silva, AF. 2005. An Empirical Evaluation of Client-side Server Selection Policies for Accessing Replicated Web Services. ACM Symposium on Applied Computing. 1704-1708.

Vawter, C. and Roman, E. 2001. J2EE vs. Microsoft.NET: A comparison of building XML-based web services. USA. Sun Microsystems. pp 28.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T. and Ferguson, DF. 2005. Web Services Platform Architecture. USA: Pearson Education. pp 416.

Williams, J. 2003. The Web Services Debate: J2EE vs. .NET. Communications of the ACM. 46(6):59-63.

Received: Jan 17, 2010; Accepted: Dec 17, 2010