SENRA Academic Publishers, British Columbia
Vol. 7, No. 1, pp. 2251-2270, Feb 2013
Online ISSN: 1920-3853; Print ISSN: 1715-9997

Canadian Journal of pure&applied sciences

SCALABILITY OF WEB SERVICES SOLUTION BUILT ON ROA

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

ABSTRACT

ROA an acronym for Replication Oriented Architecture is speculated as capable of attenuating the scalability defect of Web Services and help Application Programmers build Scalable Web Services Solution. The essence of this paper is to authenticate the scalability of Web Services solution built on ROA. We have selected a test problem that enabled the test of the architecture in its worst state scenario. We have also developed ten different Web Services solution for this problem but with different number of replicas ranging from one to ten using Java technologies. We have subjected these systems to load tests using Apache JMeter under varying load stress vis-à-vis the solutions built on conventional Web Service Java technologies, void of any solution architecture, for the same problem and platform. The data from Apache JMeter (throughput and response time) underwent mathematical transformation to realize relative scalability estimates at the varying load points. These estimates were subjected to statistical analysis and the result is that ROA can enhance the scalability of Web Service by about 32%. This scalability is guaranteed 90% of the time. We also exposed areas where we think ROA can be improved.

Keywords: ROA, scalability, web services, web services solution, apache JMeter, java EE.

INTRODUCTION

Ekuobase and Onibere (2011) asserted that the poor scalability of Web Services, if not properly addressed, may not only pose a serious threat to its survivability but may also lead the Information Technology (IT) community into shambles as Web Services has the potential of becoming the ubiquitous platform technology for next-generation computing systems, having an unprecedented number of active systems dependent on it, and hence introduced ROA and claimed that ROA can attenuate the scalability defect of Web Services and help application programmers build scalable Web Services solution. It is however imperative that the scalability claim on ROA be authenticated, for this claim to be given any consideration. This paper authenticates the scalability of Web Services solution built on ROA. We were careful to use a test (small-to-medium sized) software development project of an application with little or no computational capability because of the fears raised in Ekuobase and Onibere (2011).

The Automated Teller Machine (ATM) System is the application system of choice. The ATM system was selected because it requires less computational capability, easy to appreciate, conversational, and incorporates the use of back end system (database). Besides, the ATM system is in reality of high activity and great economic importance. Also, with the proliferation of hand held devices, emergence of ubiquitous systems and the move towards cashless society, the ATM system reflects two important domains – financial transaction (be it commerce or governance) and ubiquitous systems domains – that will be fully dependent on Web services.

MATERIALS AND METHODS

We began the software development project by first selecting an appropriate software development methodology (SDM) for it. We observed that no SDM is a failure and none is a silver bullet; they all have their strengths and weaknesses and project domain or nature of software projects they suite most (Ekuobase, 2006) as summarized in table 1. From table 1, it became obvious that the agile software development approach (Ekuobase, 2006; Agile Alliance, 2001) is most appropriate for the software development task. The tailored agile approach, depicted in figure 1, was adopted as the software development process. The software development tools used were basically software and hardware tools. These tools and the roles they played in the research are highlighted as follows:

Hardware Tools: the tools in this category were the computer and its peripherals. In particular, Dell notebook (Intel® CPU T2080 @ 1.73GHz x 2, Duo Core; 794MHz, 2.0GB RAM and 150GB Hard Disk) was used for developing the prototype system. It also served as host to the software tools used in this research.

Software Tools: we choose to discuss software tools under Operating System, Development Platform, Language, Integrated Development Environment (IDE), and Packages.

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

❖ **Operating System:** We settled for Microsoft Windows XP Professional Version 2002 Service Pack 2 which work seamlessly with the other software tools used in the research and for which our institution holds a multi-user license. The Operating System enabled our applications and other software tools used interact with the machine and tap its computational and peripheral resources.

❖ **Development Platform:** The choice of Java EE (Jendrock *et al*., 2006) as our Development platform for building the Web services solution, is not only because we were biased towards the platform as a result of our priori comfortable Java programming experience but also because as it stands today, Java EE and Microsoft. NET platforms remain the most dominant application developer's platform for enterprise applications in general and Web Services solutions in particular (Birman, 2005; Williams, 2003; Vawter and Roman, 2001) but Microsoft .NET platform is proprietary.

❖ **Language:** Here, we mean programming language, modelling language and Database Management System (DBMS). Obviously, our core programming language of implementation is Java 5.0 (van der Linden, 2004; Horstmann and Cornell, 2005a, 2005b; Deitel and Deitel, 2010) since it is the only language our choice platform supports. Java was used for coding. The Structured Query Language (SQL) being the de facto language for interacting with DBMS was adopted. Our choice of Java DB as our DBMS for building and managing the databases was based on seamless compatibility with both Java and NetBeans – our Integrated Development Environment of choice. The Unified Modelling Language (UML) was used for the systems design but not without some modifications since object oriented analysis and design which UML stands for, fall short of the concept of service orientation which requires additional elements for it to be realized (Zimmermann *et al*., 2004; Arsanjani, 2004).

❖ **Integrated Development Environment (IDE):** We have several IDEs that support Java. They include JBuilder, JCreator, Eclipse, and NetBeans. We choose NetBeans (NetBeans 6.0) on the ground of familiarity though it is not in any way less powerful than the others. Normally, IDE makes programming easy, nimble and interesting.

❖ **Packages:** the following software packages were handy. Argo UML (Ramirez *et al*., 2006; Tolke and Klink, 2006) – for the UML design, Apache JMeter (http://jakarta.apache.org/) – request generation, load testing the Web Services application and capturing of useful data as response time and throughput which were vital to this research, and Microsoft Excel – for data manipulation and computation. We added Apache JMeter as plug-in to NetBeans, to ease its use with the IDE.

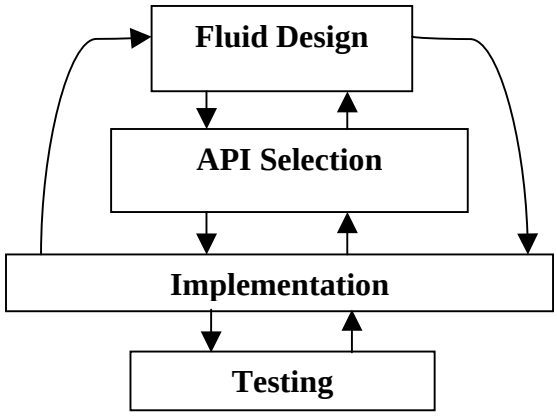

Fig. 1. An Agile Implementation Model.

Table 1. Software Projects and the basic Software Development Approaches.

Agile Software Development Approach	Traditional Software Development Approach
Visible systems	Legacy/embedded systems
Low risk projects	High risk projects
Blurred and unstable requirements	Explicit and fairly stable requirements
Small to medium sized projects	Large and complex projects
Time~to~market driven	Product quality driven

The following assumptions about the ATM system under construction were made:

a. The ATM system does not concern itself with the working and the nature of the ATM terminals (screen, card reader, keypad, cash dispenser and deposit slot).
b. The ATM system serves as a base station for several ATM terminals with the sole duty of interfacing with the user.
c. The ATM system does not include security features sufficient for real life deployment.
d. Although ATM system provides four basic services as depicted in figure 2 and 3, only the Fund Transfer service was built since it encompasses the other basic ATM services.

The service equivalent of the ATM system is modeled in figure 4. Observe that the ATM was replaced in the service model by two SOAP routers - Web Services client and Web services solution that communicate in a SOAP request-response manner. These in turn communicate with backend (logic) objects directly as in the object model and thus same internal state as its object equivalent

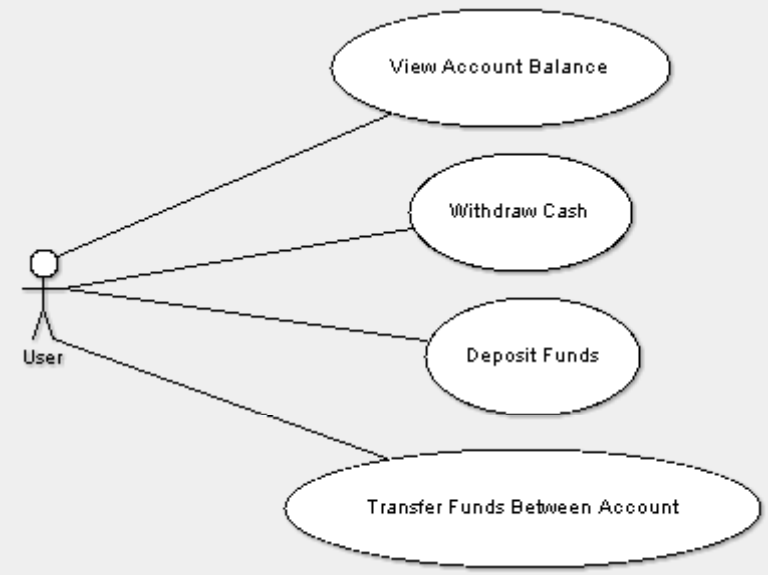

Fig. 2. Use Case Diagram for the ATM System.

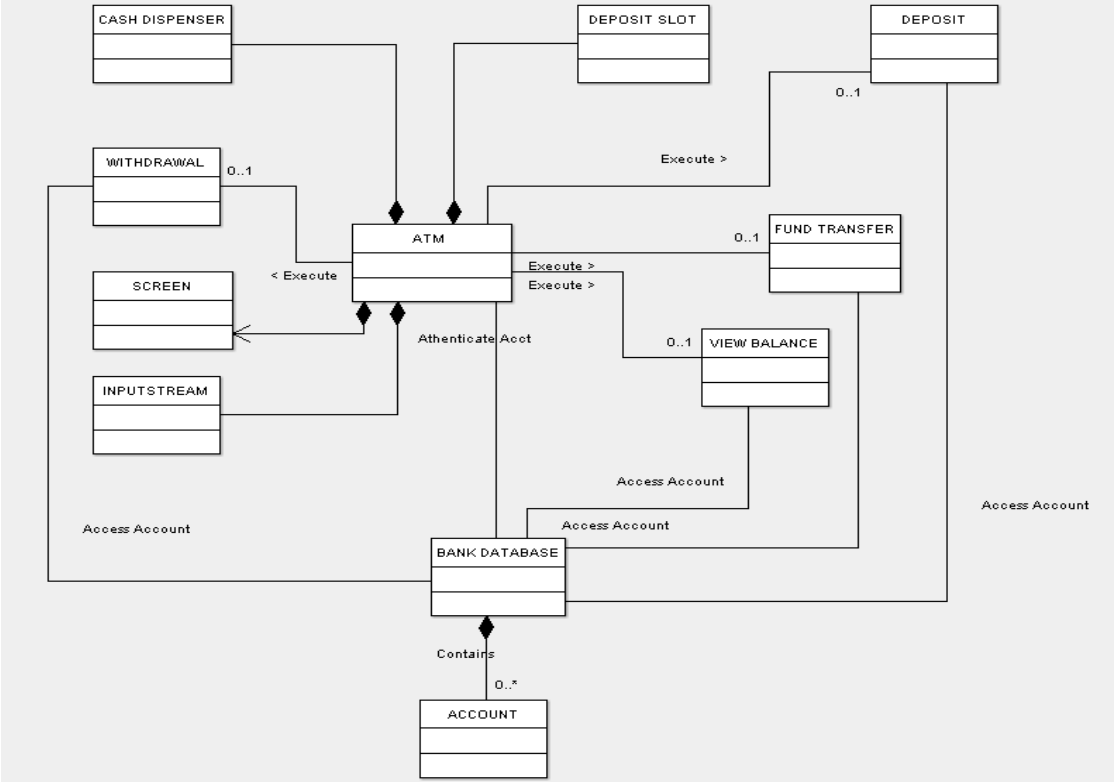

Fig. 4. Class Diagram for the ATM system.

is maintained. The implication of this is that the logic state of a Web Services solution is basically the same as that of its object counterpart and can be modeled in the same manner. A significant difference is however evident during communication with external (Web Services) systems, when it has to perform the ritual of invocation, serialization/deserialization and deployment (Hansen, 2007). At this stage UML modeling symbols proved inadequate and we had to adopt the SOAP router class symbol in Hansen (2007), which is reproduced in figure 5. This compound symbol aggregates UML symbols with the exception of the SOAP symbol which emphasizes the basic transportation mode of Web Services. The UML symbol usage is consistent with the original UML

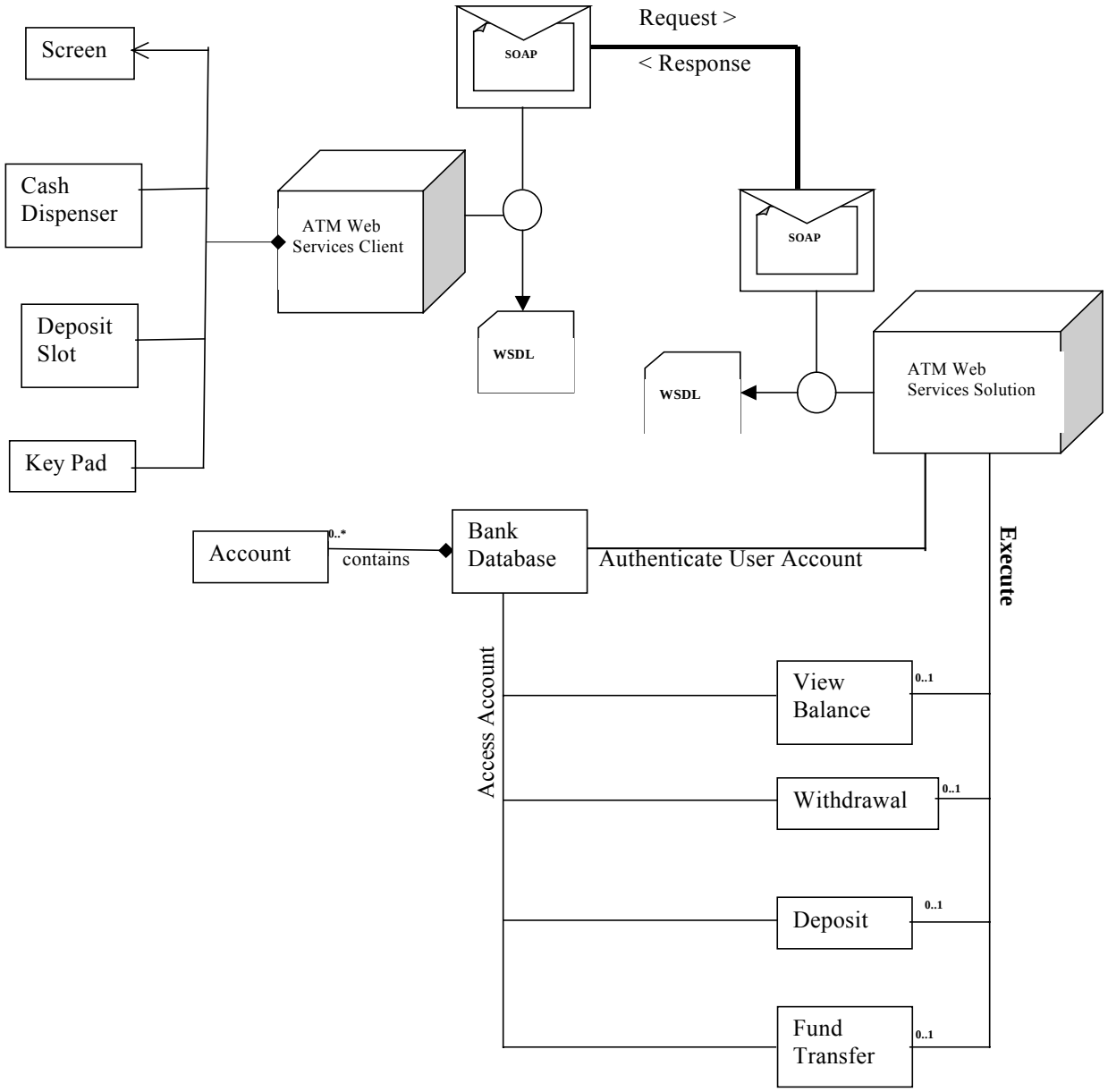

Fig. 4. Equivalent Service Model for the ATM system using Modified UML.

definitions (Rumbaugh *et al*., 1999). Figure 6 captures the sequence diagram for the Fund Transfer ATM service.

Java Application Programming Interfaces (APIs) were selected to design the implementation equivalent of ROA, depicted in figure7, for a single set of service replica. The Java EE programming models selected were Servlet, Enterprise Java Bean (EJB), Java Messaging Service (JMS) and SOAP with Attachment API for Java (SAAJ) (Crawford and Farley, 2006; Jendrock *et al*., 2006).

The EJB-endpoint is a service replica – the Web Services endpoint. As figure 7 shows, the replication logic of ROA is implemented as Java Messaging Service (JMS) on Java Servlet. Besides strength of realization, these APIs ensures portability, loose coupling, and very low overhead of WSTPRL – the virtual WS-Server (Crawford and Farley, 2006). In particular, a Servlet is persistent i.e. instead of shutting down at the end of each request the servlet can remain loaded, ready to handle subsequent requests (Crawford and Farley, 2006; Jendrock *et al*., 2006) and this guarantees a continuous active state of the WSTPRL. As depicted in figure 7, servlet accepts requests from the WS-Client and passes it (as JMS message producer) to the JMS messaging domain (Queue) from where the EJB-endpoints, which also serves as a

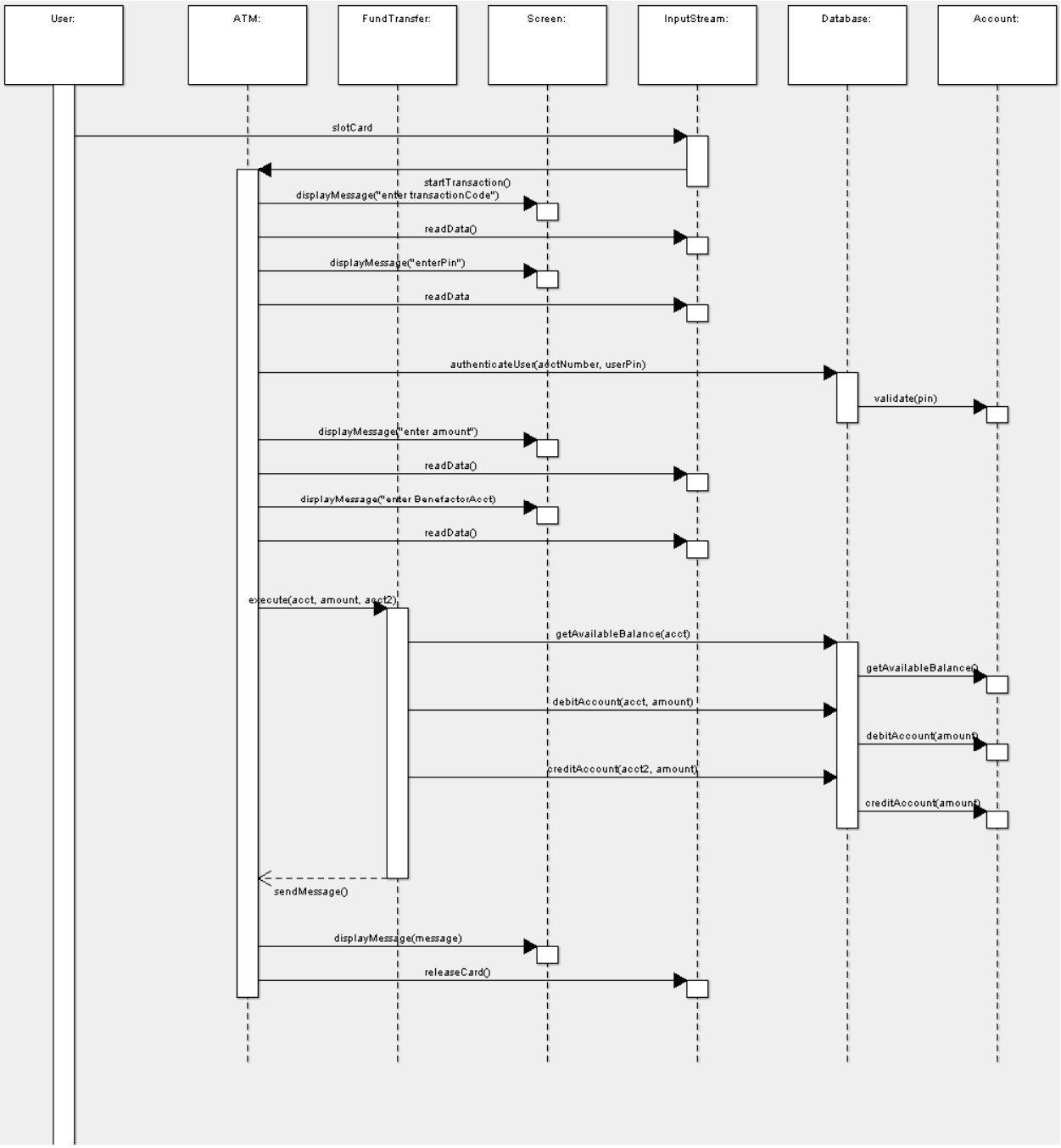

Fig. 5. Sequence Diagram for Fund Transfer ATM Service.

JMS message consumer, fetches messages asynchronously. This fetching by an EJB-endpoint is done as soon as it has no "job on hand". Clearly, the JMS provider can handle any (increasing) number of service replicas as though it is just one i.e. seamlessly accommodates replica growth.

We also made use of a small sized database built using Java DB – an open source Relational Database Management System (RDBMS) for 20 account holders with a varying fictitious amount in the accounts; as captured in figure 8. Each account has three fields: account number, account pin and available amount.

In all, eleven Web services solutions were built on the API's of the Java platform using NetBeans 6.0. Ten of these solutions were built using the Java equivalent of ROA (Fig. 7) with each differing in their number of endpoint replicas in the range of one to ten while the remaining one was not built on ROA but using JAX-WS

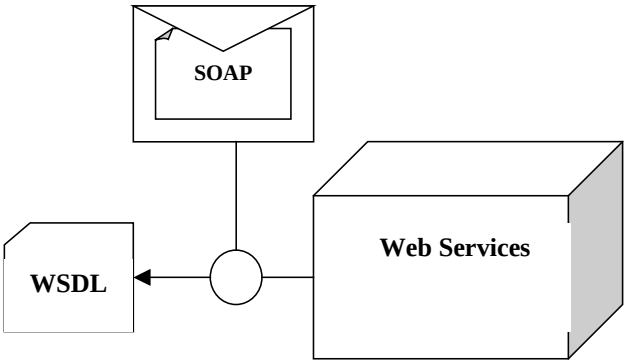

Fig. 6. Symbol for modeling SOAP Router Class.

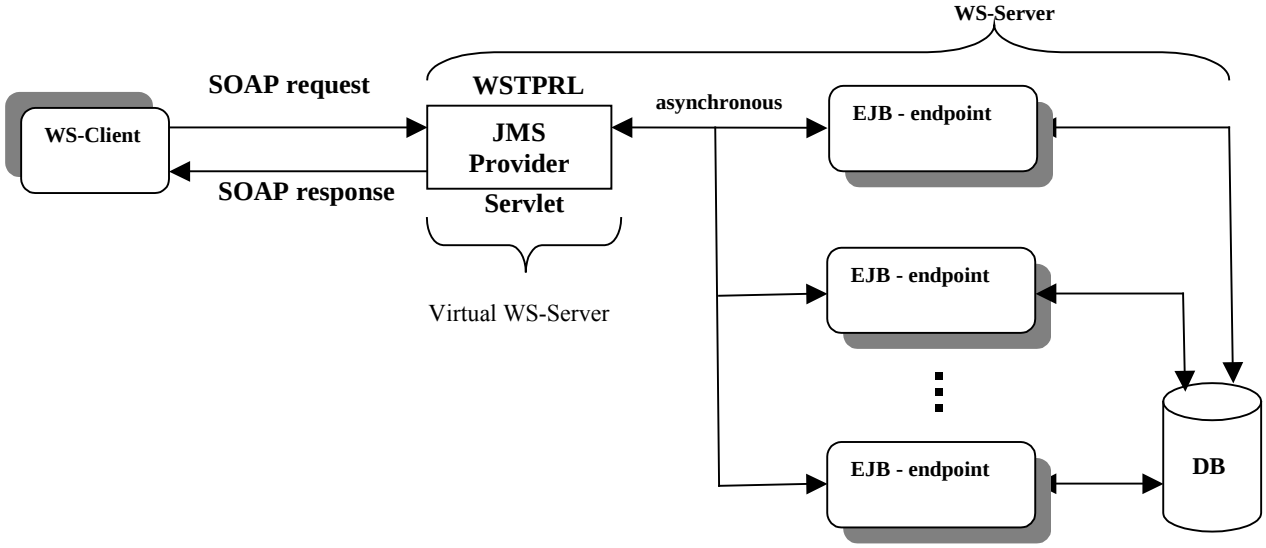

Fig. 7. ROA Implementation Equivalent using Java Technology.

API for Java (Deitel and Deitel, 2010; Crawford and Farley, 2006; Jendrock *et al*., 2006). This last solution is hereafter referred to as the conventional solution while the others will be referred to as ROA solutions.

These systems are all server side applications and we therefore need a client to consume them. Apache JMeter will play this role. Apache JMeter is not only a load generator but also load testing tool. It can handle any type of request from HTTP requests to SOAP request depending on how its test plan is prepared.

We subjected the solutions to load performance test under varying loads (sample size) ranging from one to 2000 requests per five seconds using Apache JMeter. The resultant data sample throughput and response time for each of the applications were collected. We then entered this data into Microsoft Excel for appropriate manipulation and computation of scalability. The manipulation and computation were based on the mathematical model for estimating the computational strength of applications with increasing requests (Ekuobase, 2009), given by:

$$\text{if } |S_{ij} - S_{ij+1}| < \xi \quad \text{for all j in J,} \tag{1}$$

where ξ is the user degradation tolerance, we say application i is scalable. Where S_{ij} is the computational strength of an application i for j request per square unit time and is given by:

$$S_{ij} = K * (T_{ij}/R_{ij}) \tag{2}$$

Where k is a constant denoting server and hardware strengths, T_{ij} = application throughput per unit request; T_{ij} is converted to per millisecond (ms), R_{ij} = application mid response time for a given set of request (sample) in millisecond (ms), i represents the application; in our case, i = 0(1) 10 ; where i = 0 is for the application realized by the conventional approach otherwise i denotes application realized by ROA and in particular denotes the number of endpoint replicas in the application. j = sample size per unit time; in our case, we defined j to be in the set J = {1, 5, 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750,800,850,900,950,1000, 1100,

Table 2. Scalability Data for Web Services Solution built using the Conventional Approach.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.3301	14.3301	0.003981	4187	9.50701E-07	
2	5	37.9459	7.58918	0.002108	3969	5.31143E-07	
3	10	71.908	7.1908	0.001997	4047	4.93562E-07	3.7581E-08
4	50	335.0832	6.701664	0.001862	4110	4.52938E-07	4.06242E-08
5	100	660.9385	6.609385	0.001836	4125	4.45076E-07	7.86112E-09
6	150	971.2929	6.475286	0.001799	4016	4.47881E-07	2.80468E-09
7	200	1261.034	6.30517	0.001751	4062	4.31176E-07	1.67053E-08
8	250	1283.4774	5.13391	0.001426	4266	3.34291E-07	9.68847E-08
9	300	1178.7427	3.929142	0.001091	4187	2.60671E-07	7.36204E-08
10	350	2232.3462	6.378132	0.001772	4079	4.34347E-07	1.73677E-07
11	400	1737.4937	4.343734	0.001207	4063	2.96971E-07	1.37377E-07
12	450	1410.6583	3.134796	0.000871	4157	2.09472E-07	8.74985E-08
13	500	1330.554	2.661108	0.000739	4391	1.68344E-07	4.11288E-08
14	550	1311.7621	2.385022	0.000663	4125	1.60608E-07	7.73604E-09
15	600	1506.8436	2.511406	0.000698	4203	1.6598E-07	5.37218E-09
16	650	1381.6294	2.125584	0.00059	4297	1.37407E-07	2.85722E-08
17	700	815.1955	1.164565	0.000323	4297	7.52828E-08	6.21247E-08
18	750	1088.4288	1.451238	0.000403	10406	3.87394E-08	3.65435E-08
19	800	1344.9899	1.681237	0.000467	4203	1.11114E-07	7.23742E-08
20	850	2194.9645	2.582311	0.000717	4125	1.73893E-07	6.27794E-08
21	900	1210.1408	1.344601	0.000374	4594	8.13018E-08	9.25913E-08
22	950	1355.1102	1.426432	0.000396	4343	9.12344E-08	9.93266E-09
23	1000	1339.854	1.339854	0.000372	4421	8.4185E-08	7.04946E-09
24	1100	1335.8701	1.214427	0.000337	4344	7.76568E-08	6.5282E-09
25	1200	1400.1789	1.166816	0.000324	4578	7.07985E-08	6.85826E-09
26	1300	1239.287	0.953298	0.000265	4515	5.865E-08	1.21485E-08
27	1400	1534.4622	1.096044	0.000304	4515	6.74323E-08	8.78225E-09
28	1500	1371.1152	0.914077	0.000254	4610	5.50781E-08	1.23541E-08
29	1600	998.2115	0.623882	0.000173	11016	1.57317E-08	3.93464E-08
30	1700	1133.3726	0.66669	0.000185	5266	3.51674E-08	1.94357E-08
31	1800	1319.3418	0.732968	0.000204	6906	2.94819E-08	5.6855E-09
32	1900	1341.9184	0.706273	0.000196	6438	3.04733E-08	9.9135E-10
33	2000	1405.3003	0.70265	0.000195	5828	3.34902E-08	3.01688E-09
							$\bar{x}$ = 3.92253E-08

1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000} in order. Since we will be comparing S_{ij} with each other using the same hardware and software, we let k = 1. The resultant data were then subjected to statistical analysis and interpretations.

In particular, we ascertained the scalability significance of the Web Services solution built using ROA over that built using the conventional approach. Since the two classes of Web Services solution were built on the same problem and platforms but with different development approaches, the student t-distribution for the difference of two means was found appropriate and adopted. The samples x and y are the computational strength (performance) degradation at varying but increasing request rates for the conventional and ROA solutions respectively. Let x and y be normally distributed with means μ_x and μ_y, and variance σ_x and σ_y respectively. The problem was to decide whether or not the use of ROA will improve the scalability of the Web Services solution.

Consequently, we tested the hypothesis H_0: $\mu_x = \mu_y$ (no scalability significance between conventional and ROA solutions), H_1: $\mu_x > \mu_y$ (conventional solution significantly scalable) and H_2: $\mu_x < \mu_y$ (ROA solution significantly scalable).

RESULTS AND DISCUSSION

After configuring the JMeter, we executed the package for varying number of threads (sample size) for each of the eleven applications and the valuable data: throughput and response time were collected. Table 2 to 12 contain these data and the resultant results including their S_{ij}'s, performance degradation and the mean performance degradation for each solution. Each of the table is for a solution. These data were subjected to statistical analysis.

Table 3. Scalability Data for Web Services Solution with one replica built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.2214	14.2214	0.00395	4219	9.36333E-07	
2	5	38.4764	7.69528	0.002138	4188	5.10405E-07	
3	10	70.3317	7.03317	0.001954	3985	4.90253E-07	2.01524E-08
4	50	338.6387	6.772774	0.001881	4078	4.61335E-07	2.89175E-08
5	100	661.1018	6.611018	0.001836	4157	4.41759E-07	1.95761E-08
6	150	897.2186	5.9814573	0.001662	4062	4.09039E-07	3.27205E-08
7	200	1284.3638	6.421819	0.001784	4063	4.39045E-07	3.00058E-08
8	250	1313.3701	5.2534804	0.001459	4391	3.32339E-07	1.06706E-07
9	300	1298.7013	4.3290043	0.001203	4218	2.85088E-07	4.72509E-08
10	350	1176.8662	3.3624749	0.000934	4328	2.15809E-07	6.92791E-08
11	400	2473.4618	6.1836545	0.001718	4079	4.21104E-07	2.05295E-07
12	450	2095.8037	4.6573416	0.001294	5031	2.57147E-07	1.63957E-07
13	500	1609.954	3.219908	0.000894	4516	1.98056E-07	5.90913E-08
14	550	1685.5654	3.0646644	0.000851	4188	2.0327E-07	5.21466E-09
15	600	2241.3149	3.7355248	0.001038	4390	2.36366E-07	3.30956E-08
16	650	1844.4898	2.8376766	0.000788	4235	1.86126E-07	5.02398E-08
17	700	1989.5784	2.8422549	0.00079	4203	1.87846E-07	1.71967E-09
18	750	1189.9592	1.5866123	0.000441	6515	6.76478E-08	1.20198E-07
19	800	1349.1484	1.6864355	0.000468	4281	1.09426E-07	4.17785E-08
20	850	1437.8753	1.691618	0.00047	4265	1.10174E-07	7.48043E-10
21	900	1573.0599	1.7478443	0.000486	4485	1.08252E-07	1.92195E-09
22	950	1126.9722	1.1862865	0.00033	4406	7.47898E-08	3.34626E-08
23	1000	1484.928	1.484928	0.000412	4250	9.70541E-08	2.22643E-08
24	1100	1211.3848	1.1012589	0.000306	16672	1.83484E-08	7.87057E-08
25	1200	1352.1711	1.1268093	0.000313	4547	6.88372E-08	5.04887E-08
26	1300	1221.0341	0.939257	0.000261	4828	5.40399E-08	1.47972E-08
27	1400	1409.9164	1.0070831	0.00028	4782	5.84996E-08	4.45973E-09
28	1500	1284.2832	0.8561888	0.000238	4859	4.89463E-08	9.55332E-09
29	1600	1360.8142	0.8505089	0.000236	6266	3.77039E-08	1.12425E-08
30	1700	1466.2967	0.8625275	0.00024	6000	3.99318E-08	2.22795E-09
31	1800	1106.1041	0.6145023	0.000171	5891	2.89756E-08	1.09563E-08
32	1900	1331.6201	0.7008527	0.000195	6437	3.02441E-08	1.26854E-09
33	2000	1253.6696	0.6268348	0.000174	4844	3.59457E-08	5.70156E-09
							$\bar{x}$=4.13869E-08

The performance degradation defined by equation (1) is stored in the S_{ij+1} row in each table. Observe that we began computing performance degradation from the 2nd run (5 request load). This is because Apache JMeter was configured to load test in 5 seconds i.e. its ramp-up time. Apache JMeter, (http://jakarta.apache.org/), demands that ramp-up time should not be too small or too large.

It will be noted that the two sets of Web services solution were built on the same platform, using the same technology except that the ROA's solutions were built on unique development architecture – ROA. It is also important to note that the performance load test and request generator was handled by the same package – Apache JMeter, and configured the same way.

The samples x and y are the performance degradation at varying but increasing request rates for the conventional and ROA solutions respectively. Let x and y be normally distributed with means μ_x and μ_y, and variance σ_x and σ_y respectively. The problem is to decide whether or not the use of ROA will improve the scalability of the Web Services solution. Consequently, we tested the hypothesis H_0: $\mu_x = \mu_y$ (no scalability significance between conventional and ROA systems), H_1: $\mu_x > \mu_y$ (conventional system significantly scalable) and H_2: $\mu_x < \mu_y$ (ROA system significantly scalable).

It is safe to assume $\sigma_x = \sigma_y$, and then apply the formula below (Hoel, 1962):

Table 4. Scalability Data for Web Services Solution with two replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.7674	14.7674	0.004102	4063	1.00961E-06	
2	5	37.7977	7.55954	0.0021	4078	5.14927E-07	
3	10	70.0689	7.00689	0.001946	4141	4.70021E-07	4.49056E-08
4	50	335.6831	6.713662	0.001865	4156	4.48726E-07	2.12951E-08
5	100	614.4393	6.144393	0.001707	4140	4.12265E-07	3.64615E-08
6	150	972.973	6.4864867	0.001802	4031	4.46986E-07	3.47216E-08
7	200	1151.4105	5.7570525	0.001599	4032	3.96622E-07	5.0364E-08
8	250	1559.8246	6.2392984	0.001733	4218	4.10891E-07	1.42687E-08
9	300	1363.2233	4.5440777	0.001262	4172	3.02551E-07	1.0834E-07
10	350	1885.0987	5.3859963	0.001496	4078	3.66873E-07	6.43222E-08
11	400	1184.3079	2.9607698	0.000822	4141	1.98608E-07	1.68265E-07
12	450	1441.1529	3.202562	0.00089	4125	2.15661E-07	1.70527E-08
13	500	1441.4068	2.8828136	0.000801	4375	1.83036E-07	3.2625E-08
14	550	1317.7548	2.3959178	0.000666	4156	1.60138E-07	2.2898E-08
15	600	3674.4709	6.1241182	0.001701	4125	4.12399E-07	2.52261E-07
16	650	1539.7979	2.3689198	0.000658	4234	1.55416E-07	2.56982E-07
17	700	1664.421	2.3777443	0.00066	4406	1.49906E-07	5.51076E-09
18	750	1947.2933	2.5963911	0.000721	4265	1.69102E-07	1.91962E-08
19	800	1410.4787	1.7630984	0.00049	4297	1.13975E-07	5.51272E-08
20	850	1153.5024	1.3570616	0.000377	12687	2.97124E-08	8.42623E-08
21	900	1549.1216	1.7212462	0.000478	4235	1.12898E-07	8.31858E-08
22	950	1435.6598	1.5112208	0.00042	4437	9.46098E-08	1.82884E-08
23	1000	1459.4674	1.4594674	0.000405	4297	9.43467E-08	2.63111E-10
24	1100	1432.8203	1.3025639	0.000362	4610	7.84866E-08	1.586E-08
25	1200	1340.3079	1.1169233	0.00031	4406	7.04168E-08	8.06981E-09
26	1300	1435.301	1.1040777	0.000307	4781	6.41473E-08	6.26951E-09
27	1400	1195.2021	0.8537158	0.000237	4953	4.78787E-08	1.62686E-08
28	1500	1184.6938	0.7897959	0.000219	5406	4.05823E-08	7.29645E-09
29	1600	1231.7483	0.7698427	0.000214	5375	3.97852E-08	7.97116E-10
30	1700	1152.5548	0.6779734	0.000188	6485	2.90402E-08	1.07449E-08
31	1800	1282.3742	0.7124301	0.000198	5188	3.81452E-08	9.10495E-09
32	1900	1275.0822	0.6710959	0.000186	6422	2.90276E-08	9.11755E-09
33	2000	1176.2976	0.5881488	0.000163	7859	2.07882E-08	8.23942E-09
							$\bar{x}$=4.78182E-08

$$t = \frac{(\bar{x}-\bar{y})-(\mu_x-\mu_y)}{\sqrt{n_x s_x^2+n_y s_y^2}}\sqrt{\frac{n_x n_y (n_x+n_y-2)}{n_x+n_y}} \quad (3)$$

Where, t is the student's t distribution for difference of two means and every other elements of equation (3) assume their conventional statistical use.

In our case, sample sizes are equal and equal to 31 i.e. $n_x = n_y = 31$; therefore equation (3) can be rewritten as:

$$t = \frac{(\bar{x}-\bar{y})-(\mu_x-\mu_y)}{\sqrt{n_x s_x^2+n_y s_y^2}}\sqrt{930} \quad (4)$$

Adopting the null hypothesis H_o, we can rewrite equation (4) as (5) below:

$$t = \frac{(\bar{x}-\bar{y})}{\sqrt{n_x s_x^2+n_y s_y^2}}\sqrt{930} \quad (5)$$

but, $$ns^2 = \sum_{i=1}^{n}(x-\bar{x})^2 \quad (6)$$

Table 13 shows the computation of ns^2 for each solution: conventional solution followed by solutions built on ROA with their number of replica denoted by R1..R10. Rn means replica size is n while table 14 shows the calculation details of t, for each Web Services solution as associated.

From the student's t table (Hoel, 1962) the 0.005 critical value of t is 2.576. Observe that none of the calculated t

Table 5. Scalability Data for Web Services Solution with three replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	13.9147	13.9147	0.003865	4312	8.96381E-07	
2	5	38.018	7.6036	0.002112	3890	5.42959E-07	
3	10	71.5052	7.15052	0.001986	4125	4.81516E-07	6.14427E-08
4	50	332.7418	6.654836	0.001849	4110	4.49773E-07	3.17439E-08
5	100	653.0968	6.530968	0.001814	4156	4.36515E-07	1.32573E-08
6	150	763.871	5.092473	0.001415	4157	3.40288E-07	9.62277E-08
7	200	1154.8455	5.774228	0.001604	4079	3.93222E-07	5.29342E-08
8	250	1602.7353	6.410941	0.001781	4203	4.23701E-07	3.04795E-08
9	300	1046.3291	3.487764	0.000969	6969	1.39019E-07	2.84682E-07
10	350	1611.5417	4.604405	0.001279	3985	3.20954E-07	1.81935E-07
11	400	1485.5162	3.713791	0.001032	4125	2.50087E-07	7.0867E-08
12	450	1114.8732	2.477496	0.000688	4156	1.6559E-07	8.44966E-08
13	500	1712.7198	3.42544	0.000952	4640	2.05067E-07	3.94767E-08
14	550	1558.6624	2.833932	0.000787	4188	1.87966E-07	1.71006E-08
15	600	1764.187	2.940312	0.000817	4250	1.92177E-07	4.21085E-09
16	650	1028.5898	1.582446	0.00044	7344	5.98541E-08	1.32323E-07
17	700	1852.587	2.646553	0.000735	4156	1.7689E-07	1.17036E-07
18	750	1535.5703	2.047427	0.000569	4375	1.29995E-07	4.68943E-08
19	800	1384.055	1.730069	0.000481	4250	1.13076E-07	1.6919E-08
20	850	2022.365	2.379253	0.000661	4250	1.55507E-07	4.24303E-08
21	900	1310.5842	1.456205	0.000405	4218	9.58988E-08	5.96079E-08
22	950	1503.2212	1.582338	0.00044	6172	7.12149E-08	2.46839E-08
23	1000	2019.0656	2.019066	0.000561	4390	1.27757E-07	5.65417E-08
24	1100	1535.4551	1.395868	0.000388	4328	8.9589E-08	3.81676E-08
25	1200	1621.7313	1.351443	0.000375	4547	8.25601E-08	7.0289E-09
26	1300	2714.2409	2.087878	0.00058	4797	1.20902E-07	3.83417E-08
27	1400	1230.7377	0.879098	0.000244	5547	4.40227E-08	7.68791E-08
28	1500	1037.0814	0.691388	0.000192	5985	3.20889E-08	1.19338E-08
29	1600	1226.5863	0.766616	0.000213	5438	3.91594E-08	7.07053E-09
30	1700	1147.4044	0.674944	0.000187	5110	3.66897E-08	2.46973E-09
31	1800	1029.1889	0.571772	0.000159	5500	2.88774E-08	7.81235E-09
32	1900	1055.712	0.555638	0.000154	5141	3.00221E-08	1.14479E-09
33	2000	1255.5191	0.62776	0.000174	7047	2.47449E-08	5.2772E-09
							$\bar{x}$ =5.35941E-08

value is greater than 2.576 that is the hypothesis H_1: $\mu_x > \mu_y$ is not valid and is hereby rejected. Thus, scalability conventional Web services solution is not significantly better than its equivalent solution built on ROA. This does not in any way mean that the scalability of Web services solutions built on the proposed ROA architecture is significantly better than its equivalent conventional solution.

To ascertain this latter assertion, we verified alternate hypothesis, H2:$\mu_x < \mu_y$ using the left tail t-test this time as against the right tail t-test. From the same student's t table the 0.45 critical value of t is 0.126 and only Web services solutions with five, six, seven and ten replicas have their computed t value greater than 0.126 and could be said to have significantly better scalability than its equivalent conventional Web services solution. That is H_2: $\mu_x < \mu_y$ is valid in these instances and H_0: $\mu_x = \mu_y$ holds in every other instances. Also, observe that the calculated t value for the replica five ROA's solution is highest.

We therefore assert that the scalability of Web Services solution built on ROA can be significantly better than its equivalent conventional solution depending on the replica size; an optimal replica size does exist.

Table 6. Scalability Data for Web Services Solution with four replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	13.9114	13.9114	0.003864	4313	8.95961E-07	
2	5	37.6459	7.52918	0.002091	4031	5.18839E-07	
3	10	70.713	7.0713	0.001964	4078	4.8167E-07	3.71688E-08
4	50	338.6001	6.772002	0.001881	4125	4.56027E-07	2.56429E-08
5	100	676.0563	6.760563	0.001878	4157	4.51752E-07	4.27481E-09
6	150	986.7979	6.578653	0.001827	4125	4.43007E-07	8.74535E-09
7	200	1254.8363	6.274182	0.001743	4047	4.30647E-07	1.236E-08
8	250	1041.0959	4.164384	0.001157	4313	2.68206E-07	1.62441E-07
9	300	1237.3685	4.124562	0.001146	4063	2.81987E-07	1.37804E-08
10	350	1982.2541	5.663583	0.001573	4094	3.84274E-07	1.02287E-07
11	400	1708.6715	4.271679	0.001187	4172	2.84415E-07	9.98594E-08
12	450	1647.319	3.660709	0.001017	4047	2.51264E-07	3.3151E-08
13	500	3189.4535	6.378907	0.001772	4343	4.07994E-07	1.56731E-07
14	550	1117.0814	2.031057	0.000564	8218	6.8652E-08	3.39342E-07
15	600	2400	4	0.001111	4250	2.61438E-07	1.92786E-07
16	650	1799.557	2.768549	0.000769	4188	1.8363E-07	7.78081E-08
17	700	1409.5379	2.013626	0.000559	4172	1.3407E-07	4.95597E-08
18	750	1697.0886	2.262785	0.000629	4203	1.49548E-07	1.54782E-08
19	800	1650.7325	2.063416	0.000573	4250	1.34864E-07	1.46845E-08
20	850	1061.4607	1.248777	0.000347	4406	7.87296E-08	5.61342E-08
21	900	1538.7246	1.709694	0.000475	4313	1.10112E-07	3.13829E-08
22	950	1443.038	1.518987	0.000422	4235	9.96319E-08	1.04806E-08
23	1000	1448.7977	1.448798	0.000402	4438	9.06813E-08	8.95052E-09
24	1100	1453.0404	1.320946	0.000367	4453	8.24005E-08	8.28085E-09
25	1200	1677.4614	1.397885	0.000388	4484	8.65971E-08	4.19657E-09
26	1300	1407.3833	1.082603	0.000301	4844	6.20815E-08	2.45155E-08
27	1400	1147.7318	0.819808	0.000228	4750	4.7942E-08	1.41395E-08
28	1500	1087.7163	0.725144	0.000201	5844	3.44676E-08	1.34744E-08
29	1600	1535.6069	0.959754	0.000267	7063	3.77458E-08	3.27813E-09
30	1700	1629.499	0.958529	0.000266	5203	5.11739E-08	1.34282E-08
31	1800	1591.1602	0.883978	0.000246	4593	5.34617E-08	2.28772E-09
32	1900	1433.6918	0.754575	0.00021	5297	3.95703E-08	1.38913E-08
33	2000	1134.7518	0.567376	0.000158	5734	2.74859E-08	1.20844E-08
							=5.04072E-08

Table 7. Scalability Data for Web Services Solution with five replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.6021	14.6021	0.004056	4109	9.87135E-07	
2	5	37.3506	7.47012	0.002075	4078	5.08836E-07	
3	10	71.1069	7.11069	0.001975	4094	4.8246E-07	2.63759E-08
4	50	335.6831	6.713662	0.001865	4094	4.55522E-07	2.69383E-08
5	100	651.9613	6.519613	0.001811	4156	4.35756E-07	1.97654E-08
6	150	981.6361	6.544241	0.001818	4062	4.47525E-07	1.17681E-08
7	200	992.2276	4.961138	0.001378	4078	3.37934E-07	1.09591E-07
8	250	1040.0777	4.160311	0.001156	4375	2.64147E-07	7.3787E-08
9	300	1333.9321	4.44644	0.001235	4141	2.98267E-07	3.412E-08
10	350	1286.1342	3.674669	0.001021	4094	2.49326E-07	4.89405E-08

Continued…

Table 7 continue…

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
11	400	1171.6462	2.929116	0.000814	4188	1.9428E-07	5.50465E-08
12	450	1394.6884	3.099308	0.000861	4219	2.04058E-07	9.77789E-09
13	500	1606.4214	3.212843	0.000892	4500	1.98324E-07	5.73391E-09
14	550	1909.6117	3.472021	0.000964	4204	2.29413E-07	3.10889E-08
15	600	1343.434	2.239057	0.000622	4219	1.47419E-07	8.19937E-08
16	650	1396.7568	2.148857	0.000597	4218	1.41514E-07	5.9052E-09
17	700	1451.7134	2.073876	0.000576	4265	1.35071E-07	6.44291E-09
18	750	1389.9614	1.853282	0.000515	4172	1.23394E-07	1.16766E-08
19	800	2019.777	2.524721	0.000701	4516	1.55295E-07	3.19007E-08
20	850	2101.7927	2.472697	0.000687	4250	1.61614E-07	6.31937E-09
21	900	1855.0964	2.061218	0.000573	4203	1.36227E-07	2.53876E-08
22	950	1706.2803	1.796085	0.000499	4281	1.16541E-07	1.96856E-08
23	1000	1437.1257	1.437126	0.000399	4375	9.12461E-08	2.5295E-08
24	1100	1381.3022	1.255729	0.000349	4422	7.88814E-08	1.23646E-08
25	1200	1666.5124	1.38876	0.000386	4375	8.81753E-08	9.29383E-09
26	1300	1516.8605	1.166816	0.000324	4438	7.30319E-08	1.51434E-08
27	1400	1090.4701	0.778907	0.000216	4687	4.61624E-08	2.68695E-08
28	1500	1230.4704	0.820314	0.000228	6688	3.40707E-08	1.20917E-08
29	1600	1140.0884	0.712555	0.000198	4782	4.13911E-08	7.32035E-09
30	1700	1055.2779	0.620752	0.000172	5922	2.9117E-08	1.2274E-08
31	1800	1246.9836	0.692769	0.000192	4765	4.03853E-08	1.12682E-08
32	1900	1104.7796	0.581463	0.000162	6906	2.3388E-08	1.69973E-08
33	2000	1166.4642	0.583232	0.000162	5875	2.7576E-08	4.18799E-09
							=2.56565E-08

Table 8. Scalability Data for Web Services Solution with six replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	13.9147	13.9147	0.003865	4312	8.96381E-07	
2	5	36.5052	7.30104	0.002028	4125	4.91653E-07	
3	10	69.8162	6.98162	0.001939	4141	4.68326E-07	2.33263E-08
4	50	341.0253	6.820506	0.001895	4079	4.64473E-07	3.85331E-09
5	100	655.308	6.55308	0.00182	4094	4.44626E-07	1.98466E-08
6	150	976.3506	6.509004	0.001808	4047	4.46765E-07	2.1384E-09
7	200	1141.1183	5.705592	0.001585	4125	3.84215E-07	6.25498E-08
8	250	1533.5855	6.134342	0.001704	4234	4.02453E-07	1.82376E-08
9	300	1336.5013	4.455004	0.001238	4828	2.56318E-07	1.46135E-07
10	350	1437.4701	4.107057	0.001141	4031	2.83019E-07	2.67014E-08
11	400	1964.1542	4.910386	0.001364	4875	2.79794E-07	3.22488E-09
12	450	1807.5919	4.016871	0.001116	4282	2.60579E-07	1.92155E-08
13	500	1505.8729	3.011746	0.000837	4328	1.93299E-07	6.728E-08
14	550	1547.2618	2.813203	0.000781	6687	1.1686E-07	7.64382E-08
15	600	1331.9474	2.219912	0.000617	4203	1.46715E-07	2.98544E-08
16	650	1411.406	2.171394	0.000603	4219	1.42964E-07	3.75084E-09

Continued…

Table 8 continue…

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
17	700	1031.0544	1.472935	0.000409	4281	9.55731E-08	4.73908E-08
18	750	1991.6792	2.655572	0.000738	4313	1.71032E-07	7.54584E-08
19	800	1389.6911	1.737114	0.000483	4218	1.14398E-07	5.66333E-08
20	850	2238.6094	2.633658	0.000732	4250	1.72135E-07	5.77363E-08
21	900	1565.2174	1.73913	0.000483	4187	1.15379E-07	5.67555E-08
22	950	2162.2857	2.27609	0.000632	4140	1.52717E-07	3.73378E-08
23	1000	1212.1212	1.212121	0.000337	4359	7.72426E-08	7.54742E-08
24	1100	1192.8645	1.084422	0.000301	4390	6.86169E-08	8.62561E-09
25	1200	1302.0815	1.085068	0.000301	4313	6.98836E-08	1.2666E-09
26	1300	1222.628	0.940483	0.000261	4672	5.59172E-08	1.39663E-08
27	1400	1310.8819	0.936344	0.00026	4859	5.35286E-08	2.3886E-09
28	1500	1397.8879	0.931925	0.000259	4344	5.95921E-08	6.06348E-09
29	1600	1639.7083	1.024818	0.000285	4688	6.07235E-08	1.13135E-09
30	1700	1207.7581	0.710446	0.000197	5453	3.61904E-08	2.45331E-08
31	1800	1195.7991	0.664333	0.000185	8484	2.17512E-08	1.44392E-08
32	1900	1116.6931	0.587733	0.000163	5469	2.98518E-08	8.10058E-09
33	2000	1051.3405	0.52567	0.000146	5219	2.79784E-08	1.8733E-09
							$\bar{x}$ =3.19912E-08

Table 9. Scalability Data for Web Services Solution with seven replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.171	14.171	0.00393639	4234	9.29709E-07	
2	5	38.0228	7.60456	0.00211238	4172	5.06323E-07	
3	10	70.2	7.02	0.00195	4047	4.81838E-07	2.44842E-08
4	50	331.6016	6.632032	0.00184223	4094	4.49983E-07	3.18552E-08
5	100	666.6667	6.666667	0.00185185	4125	4.48934E-07	1.04937E-09
6	150	963.2281	6.421521	0.00178376	4078	4.37409E-07	1.15244E-08
7	200	1036.4484	5.182242	0.00143951	4297	3.35004E-07	1.02406E-07
8	250	1538.4615	6.153846	0.0017094	4266	4.00704E-07	6.56997E-08
9	300	1932.7821	6.442607	0.00178961	4047	4.42207E-07	4.15037E-08
10	350	1404.4005	4.012573	0.0011146	4172	2.67163E-07	1.75044E-07
11	400	1526.8147	3.817037	0.00106029	4218	2.51372E-07	1.57907E-08
12	450	1648.3113	3.662914	0.00101748	4078	2.49504E-07	1.86851E-09
13	500	1259.023	2.518046	0.00069946	4453	1.57076E-07	9.24282E-08
14	550	1452.5287	2.640961	0.0007336	4172	1.75839E-07	1.87635E-08
15	600	1502.9433	2.504906	0.00069581	4875	1.4273E-07	3.31094E-08
16	650	2008.0321	3.08928	0.00085813	4110	2.08792E-07	6.60619E-08
17	700	1917.1954	2.738851	0.00076079	4750	1.60167E-07	4.86249E-08
18	750	2004.1865	2.672249	0.00074229	4265	1.74043E-07	1.38758E-08
19	800	1516.3002	1.895375	0.00052649	4375	1.20341E-07	5.37012E-08
20	850	1791.4852	2.10763	0.00058545	4343	1.34804E-07	1.44625E-08
21	900	1526.5186	1.696132	0.00047115	4375	1.07691E-07	2.71128E-08
22	950	1367.8249	1.439816	0.00039995	4375	9.14169E-08	1.6274E-08
23	1000	1289.4482	1.289448	0.00035818	4312	8.30659E-08	8.35099E-09
24	1100	1317.5231	1.197748	0.00033271	4563	7.29143E-08	1.01516E-08
25	1200	1242.0432	1.035036	0.00028751	4750	6.05284E-08	1.23859E-08

Continued…

Table 9 continue…

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
26	1300	1128.3928	0.867994	0.00024111	4657	5.17736E-08	8.75484E-09
27	1400	1106.6319	0.790451	0.00021957	4781	4.59255E-08	5.84808E-09
28	1500	1084.7425	0.723162	0.00020088	5515	3.6424E-08	9.50152E-09
29	1600	1237.2091	0.773256	0.00021479	4328	4.96288E-08	1.32048E-08
30	1700	956.6172	0.562716	0.00015631	5312	2.94258E-08	2.02029E-08
31	1800	1028.1034	0.571169	0.00015866	5062	3.13429E-08	1.91711E-09
32	1900	1159.5618	0.610296	0.00016953	5313	3.19079E-08	5.64948E-10
33	2000	1133.9047	0.566952	0.00015749	7250	2.17223E-08	1.01856E-08
							$\bar{x}$ = 3.0861

Table 10. Scalability Data for Web Services Solution with eight replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.6592	14.6592	0.004072	4093	9.94869E-07	
2	5	36.8505	7.3701	0.002047	4016	5.09773E-07	
3	10	70.3235	7.03235	0.001953	4125	4.73559E-07	3.62145E-08
4	50	331.6383	6.632766	0.001842	4172	4.41619E-07	3.19398E-08
5	100	651.0907	6.510907	0.001809	4141	4.36751E-07	4.86828E-09
6	150	825.2338	5.501559	0.001528	4047	3.77616E-07	5.91352E-08
7	200	1275.7814	6.378907	0.001772	4047	4.37835E-07	6.02194E-08
8	250	1456.735	5.82694	0.001619	4312	3.7537E-07	6.24653E-08
9	300	1099.2366	3.664122	0.001018	4375	2.32643E-07	1.42727E-07
10	350	1367.1875	3.90625	0.001085	4110	2.64007E-07	3.13645E-08
11	400	1319.5514	3.298879	0.000916	4141	2.21288E-07	4.27188E-08
12	450	1848.1758	4.107057	0.001141	4172	2.73454E-07	5.21654E-08
13	500	2670.227	5.340454	0.001483	4391	3.37841E-07	6.43871E-08
14	550	1422.2299	2.585873	0.000718	4266	1.68377E-07	1.69464E-07
15	600	2249.7055	3.749509	0.001042	4203	2.47806E-07	7.9429E-08
16	650	1264.159	1.94486	0.00054	4172	1.29492E-07	1.18315E-07
17	700	1631.068	2.330097	0.000647	4141	1.56303E-07	2.6811E-08
18	750	1648.5328	2.198044	0.000611	4187	1.45825E-07	1.0478E-08
19	800	2022.4151	2.528019	0.000702	4234	1.65854E-07	2.00298E-08
20	850	1356.1578	1.59548	0.000443	4312	1.0278E-07	6.3074E-08
21	900	1463.1767	1.625752	0.000452	4406	1.02496E-07	2.84255E-10
22	950	1432.8448	1.508258	0.000419	4438	9.4403E-08	8.09309E-09
23	1000	2024.0767	2.024077	0.000562	4203	1.33772E-07	3.9369E-08
24	1100	1337.3465	1.21577	0.000338	5219	6.47085E-08	6.90634E-08
25	1200	1256.6103	1.047175	0.000291	4594	6.33178E-08	1.39071E-09
26	1300	1216.3301	0.935639	0.00026	4844	5.36539E-08	9.66389E-09
27	1400	1170.9928	0.836423	0.000232	5031	4.61816E-08	7.47228E-09
28	1500	1502.73	1.00182	0.000278	4719	5.89708E-08	1.27892E-08
29	1600	1348.2578	0.842661	0.000234	4812	4.86435E-08	1.03273E-08
30	1700	1416.3716	0.83316	0.000231	5344	4.33071E-08	5.33638E-09
31	1800	1517.7706	0.843206	0.000234	6328	3.70139E-08	6.29324E-09
32	1900	1243.1436	0.654286	0.000182	6891	2.63744E-08	1.06395E-08
33	2000	1505.8918	0.752946	0.000209	5078	4.11878E-08	1.48134E-08
							$\bar{x}$ = 4.1011E-08

Table 11. Scalability Data for Web Services Solution with nine replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.4335	14.4335	0.004009	4157	9.64471E-07	
2	5	37.281	7.4562	0.002071	4094	5.05903E-07	
3	10	71.242	7.1242	0.001979	4078	4.85273E-07	2.06297E-08
4	50	333.3333	6.666666	0.001852	4094	4.52333E-07	3.29402E-08
5	100	651.8905	6.518905	0.001811	4125	4.38984E-07	1.33496E-08
6	150	953.3267	6.355511	0.001765	4047	4.36229E-07	2.75424E-09
7	200	1321.8771	6.609386	0.001836	4109	4.4681E-07	1.05803E-08
8	250	1130.7953	4.523181	0.001256	4407	2.85101E-07	1.61709E-07
9	300	1955.884	6.519613	0.001811	4062	4.4584E-07	1.6074E-07
10	350	1410.3425	4.02955	0.001119	4250	2.63369E-07	1.82471E-07
11	400	1699.115	4.247788	0.00118	4047	2.91559E-07	2.81901E-08
12	450	1453.3319	3.229626	0.000897	4203	2.13447E-07	7.81122E-08
13	500	2935.708	5.871416	0.001631	4328	3.76837E-07	1.63389E-07
14	550	1434.3372	2.607886	0.000724	5766	1.25635E-07	2.51201E-07
15	600	1656.3909	2.760652	0.000767	4312	1.7784E-07	5.22051E-08
16	650	2296.2789	3.532737	0.000981	4156	2.3612E-07	5.82799E-08
17	700	1326.0924	1.894418	0.000526	4141	1.27077E-07	1.09043E-07
18	750	1433.3934	1.911191	0.000531	4265	1.24475E-07	2.60218E-09
19	800	1541.0526	1.926316	0.000535	4141	1.29217E-07	4.74189E-09
20	850	1712.2689	2.014434	0.00056	4203	1.33135E-07	3.91764E-09
21	900	1317.0732	1.463415	0.000407	4734	8.5869E-08	4.72656E-08
22	950	2217.64	2.334358	0.000648	4281	1.51468E-07	6.55985E-08
23	1000	1462.8794	1.462879	0.000406	4625	8.78606E-08	6.3607E-08
24	1100	1758.5463	1.598678	0.000444	4390	1.01157E-07	1.32959E-08
25	1200	1278.2275	1.06519	0.000296	4672	6.33318E-08	3.78248E-08
26	1300	2080	1.6	0.000444	4454	9.97855E-08	3.64537E-08
27	1400	1326.0924	0.947209	0.000263	4828	5.44974E-08	4.5288E-08
28	1500	1564.782	1.043188	0.00029	4765	6.08131E-08	6.31568E-09
29	1600	1323.8458	0.827404	0.00023	6187	3.71479E-08	2.36652E-08
30	1700	1482.9387	0.872317	0.000242	177765	1.36309E-09	3.57849E-08
31	1800	1241.3793	0.689655	0.000192	6375	3.00503E-08	2.86872E-08
32	1900	1452.803	0.764633	0.000212	6109	3.47681E-08	4.71773E-09
33	2000	1713.4286	0.856714	0.000238	13859	1.71712E-08	1.75968E-08
							=5.68696E-08

Table 12. Scalability Data for Web Services Solution with ten replicas built on ROA.

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
1	1	14.2755	14.2755	0.003965	4203	9.43473E-07	
2	5	38.4763	7.69526	0.002138	4078	5.24172E-07	
3	10	71.242	7.1242	0.001979	4031	4.90931E-07	3.32403E-08
4	50	332.1891	6.643782	0.001845	4094	4.5078E-07	4.0151E-08
5	100	649.7726	6.497726	0.001805	4234	4.26293E-07	2.44876E-08
6	150	971.2929	6.475286	0.001799	4140	4.34466E-07	8.17348E-09
7	200	1101.8272	5.509136	0.00153	4219	3.6272E-07	7.17463E-08
8	250	1046.8281	4.187312	0.001163	4344	2.67758E-07	9.49616E-08

Table 12 continued…

Runs Number	Number of Request (sample size)	Throughput in per minute	Per request Throughput in per minutes	Per request Throughput in per ms (T_{ij})	Mid-response of sample size (R_{ij}) in ms	Computational strength per unit request (S_{ij}) in $(ms)^{-2}$	Performance degradation in $(ms)^{-2}$
9	300	1531.9149	5.106383	0.001418	4078	3.47827E-07	8.00689E-08
10	350	2021.1742	5.774783	0.001604	4109	3.90389E-07	4.25612E-08
11	400	1266.2903	3.165726	0.000879	4110	2.13958E-07	1.7643E-07
12	450	1339.5515	2.976781	0.000827	4172	1.98198E-07	1.57598E-08
13	500	1638.27	3.27654	0.00091	4375	2.08034E-07	9.83591E-09
14	550	1346.9388	2.44898	0.00068	4203	1.61854E-07	4.61803E-08
15	600	1691.6498	2.819416	0.000783	4218	1.85674E-07	2.38196E-08
16	650	1386.6667	2.133333	0.000593	4234	1.3996E-07	4.57131E-08
17	700	1559.1937	2.22742	0.000619	4235	1.46099E-07	6.13816E-09
18	750	1525.4237	2.033898	0.000565	4328	1.30539E-07	1.55599E-08
19	800	1252.8385	1.566048	0.000435	4187	1.03896E-07	2.66426E-08
20	850	1523.7981	1.792704	0.000498	4140	1.20283E-07	1.63872E-08
21	900	2002.8853	2.225428	0.000618	4219	1.46522E-07	2.62382E-08
22	950	1945.6581	2.048061	0.000569	4312	1.31936E-07	1.45861E-08
23	1000	1421.6662	1.421666	0.000395	4438	8.89832E-08	4.29523E-08
24	1100	1405.7903	1.277991	0.000355	4422	8.02799E-08	8.7033E-09
25	1200	1372.2388	1.143532	0.000318	4344	7.31234E-08	7.1565E-09
26	1300	1163.9012	0.895309	0.000249	4547	5.46947E-08	1.84287E-08
27	1400	1733.0665	1.237905	0.000344	5125	6.70951E-08	1.24004E-08
28	1500	1152	0.768	0.000213	5422	3.93459E-08	2.77492E-08
29	1600	1334.4825	0.834052	0.000232	6125	3.78255E-08	1.52041E-09
30	1700	1154.1726	0.678925	0.000189	6250	3.01744E-08	7.65102E-09
31	1800	1072.8809	0.596045	0.000166	4718	3.50928E-08	4.9184E-09
32	1900	1253.6014	0.65979	0.000183	5187	3.53335E-08	2.40693E-10
33	2000	1140.8254	0.570413	0.000158	5891	2.68966E-08	8.43692E-09
							$\bar{x}$ =3.09303E-08

It is also important we calculate the confidence limit for $\mu_x - \mu_y$ for those instances where ROA's solutions show significantly better scalability over their conventional equivalent. Here, equation (4) comes handy and in our case, the 90 per cent confidence limits are given by:

$$|t| < 0.126 \quad (7)$$

Substituting (4) in (7), reduces (7) to:

$$\alpha < \mu_x - \mu_y < \beta \quad (8)$$

Where α and β are the lower and upper limits respectively and are given by:

$$\alpha = (\bar{x} - \bar{y}) - 0.126\left(\frac{\sqrt{n_x s_x^2 + n_y s_y^2}}{\sqrt{930}}\right) \quad (9)$$

$$\beta = (\bar{x} - \bar{y}) + 0.126\left(\frac{\sqrt{n_x s_x^2 + n_y s_y^2}}{\sqrt{930}}\right) \quad (10)$$

Table 14 also shows these calculated values for the ROA's solutions.

Consequently, we can only guarantee α unit of increase i.e. $\alpha/\bar{x}$ per cent in scalability if ROA is used to build Web services solution. These results show that ROA can improve the scalability of a Web Services solution by 31.73% with 90% confidence.

We also wanted to know the impact of replication on the scalability of the ROA's solutions. Hence, we let ROA solution with just one replica (i.e. null replication) be the base solution and modified equations (5) and (8) into equations (11) and (12) respectively:

$$t = \frac{(\bar{y}_1 - \bar{y}_i)}{\sqrt{n_{y1} s_{y1}^2 + n_{yi} s_{yi}^2}}\sqrt{930}, \quad i = 2(1)10 \quad (11)$$

$$\alpha < \mu_{y1} - \mu_{yi} < \beta, \quad i = 2(1)10 \quad (12)$$

Where α and β are the lower and upper limits respectively and are given by:

Table 13. Table showing the computation of ns^2 for each web services solution.

$(x_i - \bar{x})^2$	Conventional	ROA, R1	ROA, R2	ROA, R3	ROA, R4	ROA, R5	ROA, R6	ROA, R7	ROA, R8	ROA, R9	ROA, R10
1	2.70382E-18	4.50908E-16	8.48304E-18	6.16004E-17	1.75257E-16	5.17614E-19	7.50801E-17	4.0671E-17	2.30066E-17	1.31333E-15	5.33613E-18
2	1.95691E-18	1.55486E-16	7.03474E-16	4.77432E-16	6.13274E-16	1.64318E-18	7.9174E-16	9.87381E-19	8.22869E-17	5.72616E-16	8.50209E-17
3	9.83713E-16	4.75714E-16	1.28975E-16	1.62706E-15	2.1282E-15	3.47052E-17	1.4749E-16	8.88766E-16	1.3063E-15	1.89399E-15	4.15089E-17
4	1.32646E-15	7.51067E-17	1.71521E-16	1.81762E-15	1.73571E-15	1.92886E-16	8.91188E-16	3.73927E-16	3.28485E-16	2.92847E-15	5.17873E-16
5	5.07151E-16	1.2953E-16	6.48095E-18	4.35426E-19	1.4476E-15	7.04497E-15	9.33827E-16	5.11855E-15	3.68962E-16	2.1427E-15	1.66595E-15
6	3.3246E-15	4.26655E-15	1.12557E-15	5.34282E-16	1.25515E-14	2.31655E-15	1.89161E-16	1.2137E-15	4.60288E-16	1.09912E-14	4.10001E-15
7	1.18302E-15	3.43859E-17	3.66286E-15	5.34018E-14	1.34152E-15	7.16306E-17	1.30288E-14	1.13255E-16	1.03462E-14	1.0789E-14	2.4146E-15
8	1.80772E-14	7.77974E-16	2.72383E-16	1.64714E-14	2.69154E-15	5.42145E-16	2.79822E-17	2.07887E-14	9.30551E-17	1.57757E-14	1.35279E-16
9	9.63367E-15	2.68658E-14	1.45075E-14	2.98355E-16	2.44552E-15	8.63776E-16	8.275E-16	2.27131E-16	2.91654E-18	8.22511E-16	2.11703E-14
10	2.3303E-15	1.50234E-14	9.46519E-16	9.54966E-16	2.97778E-16	2.5213E-16	1.63219E-16	8.40597E-16	1.24421E-16	4.5125E-16	2.30143E-16
11	3.62329E-18	3.13445E-16	2.30835E-16	1.993E-16	1.13047E-14	3.96908E-16	1.2453E-15	3.79045E-15	5.46442E-16	1.13465E-14	4.44974E-16
12	9.91574E-16	1.30843E-15	6.21018E-16	1.33177E-15	8.34834E-14	2.95115E-17	1.97553E-15	1.46363E-16	1.65E-14	3.77648E-14	2.32564E-16
13	1.14603E-15	6.8747E-17	4.17967E-14	2.4387E-15	2.02717E-14	3.17388E-15	4.56573E-18	5.05259E-18	1.47594E-15	2.17572E-17	5.05615E-17
14	1.13488E-16	7.83731E-17	4.37495E-14	6.19827E-15	7.50809E-16	3.90113E-16	7.97517E-16	1.23907E-15	5.97588E-15	1.98894E-18	2.18532E-16
15	5.2438E-16	1.57349E-15	1.78992E-15	4.02483E-15	7.18378E-19	3.69161E-16	2.37149E-16	3.15536E-16	2.01639E-16	2.72206E-15	6.14651E-16
16	7.19234E-18	6.21115E-15	8.19218E-16	4.48865E-17	1.22004E-15	1.95438E-16	1.8894E-15	2.88516E-16	9.32264E-16	2.94495E-15	2.36251E-16
17	1.09885E-15	1.53344E-19	5.34209E-17	1.34506E-15	1.27611E-15	3.89898E-17	6.07235E-16	5.2165E-16	4.40213E-16	2.7173E-15	1.83848E-17
18	5.54796E-16	1.65152E-15	1.32817E-15	1.24629E-16	3.27978E-17	3.73924E-16	6.62812E-16	2.6893E-16	4.86778E-16	2.80391E-15	2.11502E-16
19	2.84792E-15	1.55749E-15	1.25087E-15	3.61661E-17	3.61927E-16	7.23193E-20	6.13273E-16	1.40528E-17	1.65867E-15	9.22364E-17	2.20161E-17
20	8.5806E-16	6.27949E-17	8.72007E-16	8.35796E-16	1.59414E-15	3.56517E-17	2.85859E-17	2.12796E-16	1.08359E-15	7.61945E-17	2.67134E-16
21	1.03529E-15	3.65677E-16	2.26149E-15	8.68865E-18	1.71866E-15	1.30659E-19	1.89077E-15	5.06725E-16	2.6963E-18	4.53921E-17	1.44529E-16
22	1.0691E-15	1.39269E-15	1.02132E-15	2.37975E-16	1.77463E-15	1.76673E-16	5.4595E-16	4.28902E-16	7.86939E-16	1.89866E-15	4.9404E-16
23	1.04763E-15	8.28422E-17	1.57994E-15	2.16831E-15	2.13543E-15	2.67736E-16	9.44E-16	3.41351E-16	1.56977E-15	3.62704E-16	5.65194E-16
24	7.33156E-16	7.07012E-16	1.7263E-15	2.32634E-16	6.7038E-16	1.10525E-16	3.24896E-16	4.88707E-16	9.82642E-16	4.16809E-16	1.56291E-16
25	9.2678E-16	1.36362E-15	9.95379E-16	5.42193E-16	1.31535E-15	1.47142E-18	8.76313E-16	6.25674E-16	1.12485E-15	1.34133E-16	3.43358E-16
26	7.22059E-16	1.01338E-15	1.64201E-15	1.73558E-15	1.36404E-15	1.84004E-16	6.72246E-16	4.56251E-16	7.96471E-16	2.5557E-15	1.01193E-17
27	1.46668E-20	9.0869E-16	2.21098E-15	2.16444E-15	2.22115E-15	3.36214E-16	9.52329E-16	3.11762E-16	9.41488E-16	1.10253E-15	8.64942E-16
28	3.91629E-16	1.53343E-15	1.37443E-15	2.6137E-15	1.36745E-15	1.7909E-16	5.56231E-17	1.13606E-16	1.27268E-15	4.44566E-16	5.41925E-16
29	1.12492E-15	9.26027E-16	1.49872E-15	2.09597E-15	2.31549E-15	2.07022E-16	3.08072E-16	8.37781E-16	1.20532E-15	7.94245E-16	6.76619E-16
30	1.46184E-15	1.60949E-15	1.49774E-15	2.75093E-15	1.33341E-15	7.49819E-17	5.70761E-16	9.17884E-16	9.22431E-16	2.71982E-15	9.41852E-16
31	1.31105E-15	1.27345E-15	1.56648E-15	2.33452E-15	1.46864E-15	4.60896E-16	9.07086E-16	4.27496E-16	6.86316E-16	1.54235E-15	5.05952E-16
ns^2	5.53401E-14	7.22567E-14	1.3142E-13	1.09109E-13	1.63409E-13	1.83233E-14	3.31854E-14	4.18648E-14	5.27289E-14	1.20189E-13	3.79274E-14

Table 14. Table showing the computed t value and confidence limits for Conventional vs ROA's solutions.

Computed values	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
$\overline{x}-\overline{y}$	-2.16164E-09	-8.5929E-09	-1.43688E-08	-1.11819E-08	1.35688E-08	7.23414E-09	8.36376E-09	-1.78569E-09	-1.76443E-08	8.29501E-09
$n_x s_x^2+n_y s_y^2$	1.27597E-13	1.8676E-13	1.64449E-13	2.18749E-13	7.36635E-14	8.85255E-14	9.7205E-14	1.08069E-13	1.7553E-13	9.32675E-14
$\sqrt{n_x s_x^2+n_y s_y^2}$	3.57207E-07	4.32158E-07	4.05524E-07	4.67706E-07	2.7141E-07	2.97532E-07	3.11777E-07	3.28739E-07	4.18963E-07	3.05397E-07
$(\overline{x}-\overline{y})/\sqrt{n_x s_x^2+n_y s_y^2}$	-0.006051492	-0.019883704	-0.035432587	-0.023908024	0.049993858	0.02431377	0.026826084	-0.005431949	-0.042114219	0.027161365
t	-0.184545704	-0.606371487	-1.080548674	-0.729096752	1.524607762	0.741470345	0.818085603	-0.165652174	-1.284311071	0.828310308
$\left(\frac{\sqrt{n_x s_x^2+n_y s_y^2}}{\sqrt{930}}\right)$	1.17133E-08	1.4171E-08	1.32976E-08	1.53367E-08	8.89989E-09	9.75647E-09	1.02236E-08	1.07798E-08	1.37383E-08	1.00144E-08
$0.126\left(\frac{\sqrt{n_x s_x^2+n_y s_y^2}}{\sqrt{930}}\right)$	1.47587E-09	1.78555E-09	1.6755E-09	1.93242E-09	1.12139E-09	1.22932E-09	1.28817E-09	1.35825E-09	1.73103E-09	1.26181E-09
α	-3.63751E-09	-1.03784E-08	-1.60443E-08	-1.31143E-08	1.24475E-08	6.00482E-09	7.07559E-09	-3.14394E-09	-1.93753E-08	7.0332E-09
β	-6.85762E-10	-6.80735E-09	-1.26932E-08	-9.2495E-09	1.46902E-08	8.46345E-09	9.65193E-09	-4.27441E-10	-1.59133E-08	9.55682E-09
(α/$\overline{x}$) per cent	-9.273369391	-26.45854525	-40.90280964	-33.43338158	31.73322018	15.30853113	18.03832303	-8.015083425	-49.39491067	17.93024895

Table 15. Table showing the computed t value and confidence limits for the null replication ROA's solution vs other ROA's solutions.

Computed values	R2	R3	R4	R5	R6	R7	R8	R9	R10
$\overline{y_1}-\overline{y_i}$	-6.43126E-09	-1.22071E-08	-9.02029E-09	1.57305E-08	9.39577E-09	1.05254E-08	3.75944E-10	-1.54826E-08	1.04566E-08
$n_{y1}s_{y1}^2+n_{yi}s_{yi}^2$	2.03677E-13	1.81366E-13	2.35665E-13	9.058E-14	1.05442E-13	1.14122E-13	1.24986E-13	1.92446E-13	1.10184E-13
$\sqrt{n_{y1}s_{y1}^2+n_{yi}s_{yi}^2}$	4.51306E-07	4.25871E-07	4.85454E-07	3.00965E-07	3.24718E-07	3.37819E-07	3.53533E-07	4.38687E-07	3.3194E-07
$(\overline{y_1}-\overline{y_i})/\sqrt{n_{y1}s_{y1}^2+n_{yi}s_{yi}^2}$	-0.014250345	-0.028663894	-0.01858115	0.052266768	0.028935126	0.031156925	0.001063391	-0.035293159	0.031501619
t	-0.434577117	-0.874131271	-0.56664893	1.593922195	0.882402737	0.950158512	0.032429062	-1.076296682	0.96067027
$\left(\frac{\sqrt{n_{y1}s_{y1}^2+n_{yi}s_{yi}^2}}{\sqrt{930}}\right)$	1.47989E-08	1.39649E-08	1.59187E-08	9.86904E-09	1.06479E-08	1.10775E-08	1.15928E-08	1.43851E-08	1.08847E-08
$0.126\left(\frac{\sqrt{n_{y1}s_{y1}^2+n_{yi}s_{yi}^2}}{\sqrt{930}}\right)$	1.86466E-09	1.75957E-09	2.00575E-09	1.2435E-09	1.34164E-09	1.39577E-09	1.46069E-09	1.81252E-09	1.37148E-09
α	-8.29593E-09	-1.39667E-08	-1.1026E-08	1.4487E-08	8.05413E-09	9.12963E-09	-1.08475E-09	-1.72952E-08	9.08517E-09
β	-4.5666E-09	-1.04475E-08	-7.01454E-09	1.6974E-08	1.07374E-08	1.19212E-08	1.83664E-09	-1.36701E-08	1.18281E-08
(α/$\overline{y_1}$) per cent	-20.0447876	-33.74660175	-26.64135152	35.00373489	19.46055463	22.05919459	-2.620994733	-41.78894312	21.95176472

$$\alpha = (\overline{y_1} - \overline{y_i}) - 0.126\left(\frac{\sqrt{n_{y1}s_x^2 + n_{yi}s_{yi}^2}}{\sqrt{930}}\right), \text{ i =2(1)10} \quad (13)$$

$$\beta = (\overline{y_1} - \overline{y_i}) + 0.126\left(\frac{\sqrt{n_{y1}s_x^2 + n_{yi}s_{yi}^2}}{\sqrt{930}}\right), i = 2(1)10 \quad (14)$$

Similar calculations as in the cases of equations (5) and (8) were also carried out for equations (11) and (12) as shown in table 15.

This statistical result further establish the fact that an optimal replica size for maximum scalability of ROA solution exist i.e. the relationship between the replica size of ROA's solution and scalability is not linear.

CONCLUSION

ROA is capable of enhancing the scalability of Web Services solution by 32%, 90% of the time. With this development, there is renewed hope that Web Service will survive the unprecedented deployment its promises will attract. Implementers of ROA should note however that optimum replica size will vary from one solution to another and therefore will have to get the optimum replica for their solution by simply load testing the solution prior to deployment and analyzing the resultant data using the instruments as described in this paper.

Opportunities for further improvement of the scalability of Web Services solution exist:

- Strengthening the traffic handling capacity of WSTPRL.
- Support for dynamic service replication.
- The possibility of using the concept of group multicast, communication and membership to make Web Service support replication. Though a middleware approach, this may yield more scalable Web services solution.

Other areas of this research requiring further effort include:

- Determining user (client) degradation tolerance for various service domains.
- Deploying this novel Web Service architecture in real life situations.

REFERENCES

Agile Alliance. 2001. Manifesto for Agile Software Development. available online at URL: http:// www.agilemanifesto.org.

Arsanjani, A. 2004. Service-Oriented Modelling and Architecture. available online at URL: http://www.ibm.com/developerworks/library/ws-soa-design1/

Birman, KP. 2005. Reliable Distributed Systems: Technology, Web Services, and Applications. USA: Springer-Media. pp668.

Crawford, W. and Farley, J. 2006. Java Enterprise in a Nutshell, (3rd edi.). O'Reilly Media, USA. pp892.

Deitel, P. and Deitel, H. 2010. Java: How to Program, (8th edi.). Pearson Education, USA. pp1506.

Ekuobase, GO. 2006. Software Creative Milestones. Proceedings of International Conference on Advances in Engineering and Technology. Entebbe, Uganda. pp8.

Ekuobase, GO. 2009. Towards Scalable Web Services Solution. Ph.D. Thesis. University of Benin, Nigeria. pp126.

Ekuobase, G. and Onibere, E. 2011. Architecture for Scalable Web Services Solution. Canadian Journal of Pure and Applied Sciences. 5(1):1449-1453.

Hansen, MK. 2007. SOA Using Java Web Services. USA: Pearson Education. pp574.

Hoel, PG. 1966. Introduction to Mathematical Statistics, (3rd edi.). John Wiley & Sons, USA. pp427.

Horstmann, CS. and Cornel, G. 2005[a]. Core Java 2, (7th edi.). Sun Microsystems, USA. 1:762.

Horstmann, CS. and Cornel, G. 2005b. Core Java 2, (7th edi.). Sun Microsystems, USA. 2:1002.

Jendrock, E., Ball, J., Carson, D., Evans, I., Fordin, S. and Haase, K. 2006. The Java™ EE 5 Tutorial, (3rd edi.). Sun Microsystems, USA. pp1304.

van der Linden, P. 2004. Just Java 2, (6th edi.). Sun Microsystems, USA. pp816.

Ramirez, A., Vanpeperstraete, P., Rueckert, A., Odutola, K., Bennett, J., Tolke, L. and van der Wulp, M. 2006. ArgoUML User Manual: A tutorial and reference description. USA: Open Content. Also available online at http://www.opencontent.org/openpub/. pp386.

Rumbaugh, J., Jacobson, I. and Booch, G. 1999. The Unified Modelling Language Reference Manual. Addison Wesley Longman, USA. pp23-95.

Tolke, L. and Klink, M. 2006. Cook book for Developers of ArgoUML: An Introduction to Developing ArgoUML. University of California, USA. pp157.

Vawter, C. and Roman, E. 2001. J2EE vs. Microsoft.NET: A comparison of building XML-based web services. Sun Microsystems, USA. pp28.

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

Zimmermann, O., Krogdahl, P. and Gee, C. 2004. Elements of Service-Oriented Analysis and Design. available online at URL:http://www.ibm.com/ developworks/ library/wssoad1/

Received: May 4, 2011; Accepted: Dec 24, 2012