INTERNATIONAL JOURNAL OF RESEARCH CULTURE SOCIETY ISSN: 2456-6683 Volume - 4, Issue - 5, May – 2020
Monthly, Peer-Reviewed, Refereed, Indexed Journal Scientific Journal Impact Factor: 5.245
Received on : 26/04/2020 Accepted on : 09/05/2020 Publication Date: 31/05/2020

Evaluation of the performance of Enterprise Service Bus in Applications developed using Service Oriented Architecture

Dr. Kamlendu Kumar Pandey
Assistant Professor
Department of Information and Communication Technology,
Veer Narmad South Gujarat University, Surat, India
Email – kspandey@vnsgu.ac.in

Abstract: *"ESB" popularly called Enterprise Service Bus are the major breakthrough as a message routers, data transformers, protocol negotiator and service discoverer. Several vendors like Oracle and IBM provide this as integrated product with their application servers while open source efforts like WSO2, Mule etc are also in the completion. It is therefore very important to judge the performances of such systems before their adoption in real time. They become the essential components where Service Oriented Architecture is being used to develop Business Applications to achieve complete end to end solution in a heterogeneous environment. The performance of ESB becomes of the whole system. In this paper there is an effort to develop a model for assessing the performance of ESB for closed system and a queue model. The model is validated by by modern stress testing tools. Such models will help us to strategize the technological complexity in Inter- service communication. The model has been tested with the varying load, routing combinations and traffic situations.*

Key Words : *ESB, cloud, SOA, performance testing.*

1. INTRODUCTION:

In the era of On-Demand-Business and Software Engineering model like Scrum or Micro Profile where a complete dynamic solution is expected for heterogeneous environments SOA emerges as a promising solution architecture. SOA abstracts the complexities implementation of language based business logic and protocol based communications and gives a very easy to use plug and play end points to work within the application. Using the loos coupling and a Decomposed Business model it solves many teething problems of current enterprise applications [1]. The best thing is that that this architecture is well standardized by W3C and is a part of reference implementation in almost all major players of software industry. The standardization solves the compatibility issues with in the vendors and standards so developers have to focus mainly on the the orchestration part rather than peeking into the code and fine tuning it as per new standards. The orchestration tools and the Business Process Execution Language for Web Services (BPEL4WS) makes it most attractive option to work with. It facilitate point to point, multi point communication with sufficient flags and fault handlers creating a composite application involving services abstracting the business logic written in various languages. Almost all the vendors are providing the XML interface or GUI based wizards . Not only business logic but even system based events like signals, messaging, authentication and authorization repositories can also be abstracted as a service.As far as direct communication is concerned the standard SOAP based or REST based protocols are sufficient but as the complexity increases it becomes difficult to manage and record the communication. Apart from that the diversity of data structures, protocols and diverse standards of communication made it too difficult to sustain [2]. That is where ESB comes in the picture as independent service manager and a message router which takes care of all the issues related to web service execution, data negotiation, content based and conditional routing. As standards of ESB are already defined by W3C now all the vendors are releasing it as a product. Some of them are propriety while many of them are open source. ESB must provide a distributed, message-oriented architecture, supporting traditional EAI features such as message routing and transformation, within the context of integration based upon 'business services' and XML messages [9]. It is the technology that adds 'intelligence' to the integration infrastructure, and thus enables the creation of applications that automate business processes and interactions between multiple systems and organizations. In many projects this service flow must be dynamically altered based on the content of the message, or the results of a previous step in the process. An ESB must provide this ability if it is to be of any use in addressing complex integration challenges. Clearly there must be support for asynchronous messaging, publish and subscribe, store and forward and similar messaging models [3].

Looking to various aspects of ESB , it becomes quite clear that the performance of entire software systems is now highly depending on the performance of Enterprise Service Bus. It becomes now more important to examine the performance of ESB in various stress situations before actually using them for commercial or production level deployment[4]. There are no such standards till now as regards to ESB . Although various vendors like Oracle , IBM have published their own performance charts and and some research on them. As there are wide ranges of products we needed to develop a common strategy to test all of them in quite a transparent manner using the open source tools in a fully unbiased manner. The performance falls in two categories . One the individual services which are part of one composite application and the other which are either to a different composite application or third party services deployed on remote locations [10]. Services in single composites can be assessed for the performance much easily as their logic , structure too chain, deployment containers are well known and can be operated in a controlled scenario. The difficulty lies with the remote services or cross domain services where we generally have no control over their working and environment. This requires a preparation of some standard way which can be trusted for performance testing. The paper is arranged as follows: section 2 deals with the functioning of the Enterprise Service Bus, Section 3 depicts a real world banking problem which can be tested on ESB, Section 4 is about the assessment of the problem and working out an analytical solution of the load / stress on the system and thereby its performance under various constraints, Section 5 is real world stress testing of the application using popular web testing tools and thereby validating the developed model, Section 6 is about discussion and future work while Section 7 is the conclusion of the paper.

2. ABOUT ENTERPRISE SERVICE BUS:

As the paper is dealing with performance testing model of Enterprise Service Bus it is imperative to know its structure , components, functioning , interaction, events and transactional aspects so that all the factors are wisely considers in the underlying example application.

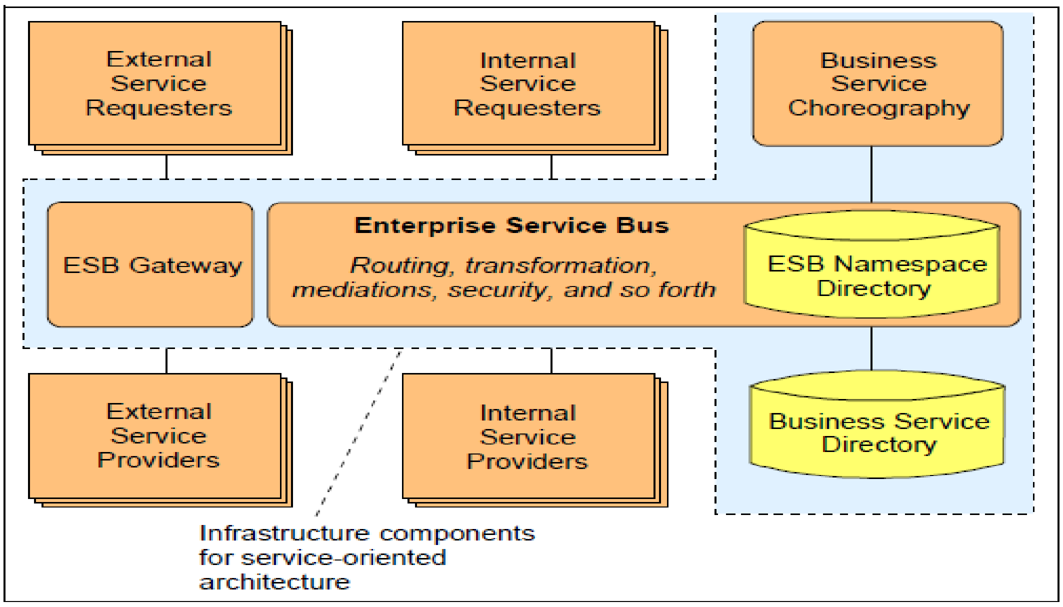
External Service Requesters
Internal Service Requesters
Business Service Choreography
ESB Gateway
Enterprise Service Bus
Routing, transformation, mediations, security, and so forth
ESB Namespace Directory
External Service Providers
Internal Service Providers
Business Service Directory
Infrastructure components for service-oriented architecture

Figure 1: A schematic model of Enterprise Service Bus (courtesy: IBM Red Book, Implementing SOA using an ESB)

The ESB [3] is broadly has the following components which we shall describe one by one.

- Core ESB Module
- Business Service Registry
- Internal Service Providers
- External Service Providers
- Business Service Choreography
- ESB gateways
- Internal Service Requester
- External Service Requester
- Logging and Metering Services

2.1. Core ESB Module: This module carries out all the essential function of ESB. The function included

- ***Call validation*** **:** This checks the structure of service call coming from the external or internal requester, if the desired standard like SOAP or REST [7] is not followed the call is rejected

- *Authentication/Authorization* with conventional login-password , token based and Public-Private Key Asymmetric Encryption
- ***Message Transformation*:** The message or input-output format of calling service does not match with the target service than there is facility in the ESB module to apply transformation with input and output pipe lining. The mismatch of name space and complex types are taken care of in this. Apart from that one can do a service call out to include or exclude a additional data. The message transformation happens while routing the call to the target service.
- **Message Routing:** The SOAP-RPC and now the REST based message routing is one main features of ESB. The routing can be conditional and content based. The calls made by the proxy services are routed to the target business services the target is based on the content of the call. ESB presents a elegant IF-ELSE based and pipeline pair based routing of the calls
- **Service Discovery:** In line with discovery standards like UDDI the ESB too has its own discovery mechanism for the called service.

2.2. Business Service Registry: The service end point abstracting and encapsulating the core business logic written in the native/advanced programming languages. The ESB framework maintains the registry of such services and these services are hooked to the ESB with a name. The ESB uses this name for the purpose of discovery.

2.3. Business Service Choreography: The orchestration of business services are one of the main features of SOA and ESB executes the orchestration planned by developers in the composite applications. The routing pallets and the routines are given to developers to orchestrate the business services.

2.4. External Service providers: They are the out of core organization and third party business services hooked to ESB.

2.5. Internal Service Providers: They are the In-organization business services developed as part of business process.

2.6. ESB Gateway: The outside world clients can given access to the services hooked to ESB using an ESB gateway. The URL mapping and access controls are defined in the Gateways to facilitate a discovery and execution of the services. This is not the core part of ESB but some vendors use them. For normal ESBs the proxy services play the role of gateways. The firewalls are also adopted for traffic filtering.

2.7. Internal/External Requesters: ESB achieves this through proxy services. The Proxy Services act as entry point to execute a business process which is the orchestration of several business services

2.8. Logging and Metering: This is to constantly maintain the log and meter the usage to facilitate dash boarding applications to check stress and health of services.

Enterprise Service Bus generally support the three major styles of Enterprise Integration:

- **Service-oriented architectures** in which applications communicate through reusable services with well-defined, explicit interfaces.
- **Message-driven architectures** in which applications send messages through the ESB to receiving applications.
- ***Event-driven architectures*** In which applications generate and consume messages independently of one another.

In this paper main focus is done on using the maximum components of ESB which are crucial for performance point of view.

3. THE APPLICATION FOR PERFORMANCE EVALUATION:

Scenario: The application for testing the performance of ESB is a mortgage application where a bank classifies the customer' s eligibility for the application of Loan. The bank provides a client interface for applying bank loan and the client applies by providing the data asked by the bank. The bank later on classifies the applicants. There are two criteria of classification 1) on the basis of desired interest rate and secondly on the basis of the amount. The bank have three sanctioning authorities for loans based on the gradation of the customers. If the customer applies for a loan of an amount under a certain threshold limit and as per the interest rate charged by the bank, such customers are subjected to a Normal loan processing service where decision can be taken by the loan officer. The customers who have demanded a lower charge of interest rate are subjected to the approval of Manager who depends upon the past experience has the right to approve or disapprove. If the customer applies for a larger loan above a threshold limit then the approval process is complex. In that case a their party financial service is utilized to obtain the credit rating of the customer. If the credit rating of the customer is above certain grade then the loan for that customer can be approved else rejected.

ALGORITHM :

Process :

```
•   Bank Threshold values :
    IR : prevailing interest rate of bank
    TA: Threshold loan amount
    CR : Minimum credit rating on Large Loan amount
    CCR : Customer Credit Rating
    VA : Valid Application (True, False)
•   Client applies for loan
    var LoanApp  : properties : name, address, loanAmount, deisiredIntrestRate
•   Bank Receives the Applications
    call ApplicationValidationLogic
    if VA is True  then
            if loanAmount LESS THAN  TA
                    If deisiredIntrestRate LESS THAN IR
                            Call NormalLoanProcessing Service
                    Else
                            Call ManagerLoanProcessing Service
            Else
                    Call LargeLoanProcessing Service and
                    CCR = call populateCCR
                    If  CCR More than CR Then
                            Goto loanaccept
                    Else Go to loanreject
    End If
Business Logic :
    : NormalLoanProcessing
    : ManagerLoanProcessing
    : largeLoanProcessing
    : populateCCR
    : ValidateApplicationLogic
Messages :
    :loanaccept : "Your loan is approved:
    :loanreject : "Your loan is rejected"
```

The above scenario and the algorithm clarifies the problem. This scenario gives us following services to constructed to abstract the business logic

Business Services :

• NormalLoanService	: Abstracting the logic of NormalLoanProcessing
• ManagerLoanService	: Abstracting the logic of ManagerLoanProcessing
• LargeloanService	: Abstracting the logic of LargeLoanProcessing
• CreditRatingService	: Abstracting the logic of populateCCR (Third party)
• LoanvalidationService	: Abstracting the logic of ValidateApplicationLoic

Proxy Services

- ValidationProxy : Abstracting loanApp in the algorithm for validation
- LoanAppProxy   : Proxy service leading to MangerLoanService
- LargeLoanProxy : Proxy service leading to Normal or large loan processing

The Services are correctly identified in this model . The application was developed in Oracle Fusion Middleware and Oracle Service Bus which developed over Weblogic Server. The IDE used for developing the composite application is Jdeveloper. The ESB console is a web based application which facillitates the developers to do all the functionalities like transformation, routing, pipe lining, altering the message structures and service call out. The application was tested for its correctness and functioning with test data. The input and output formats were XML artifacts.

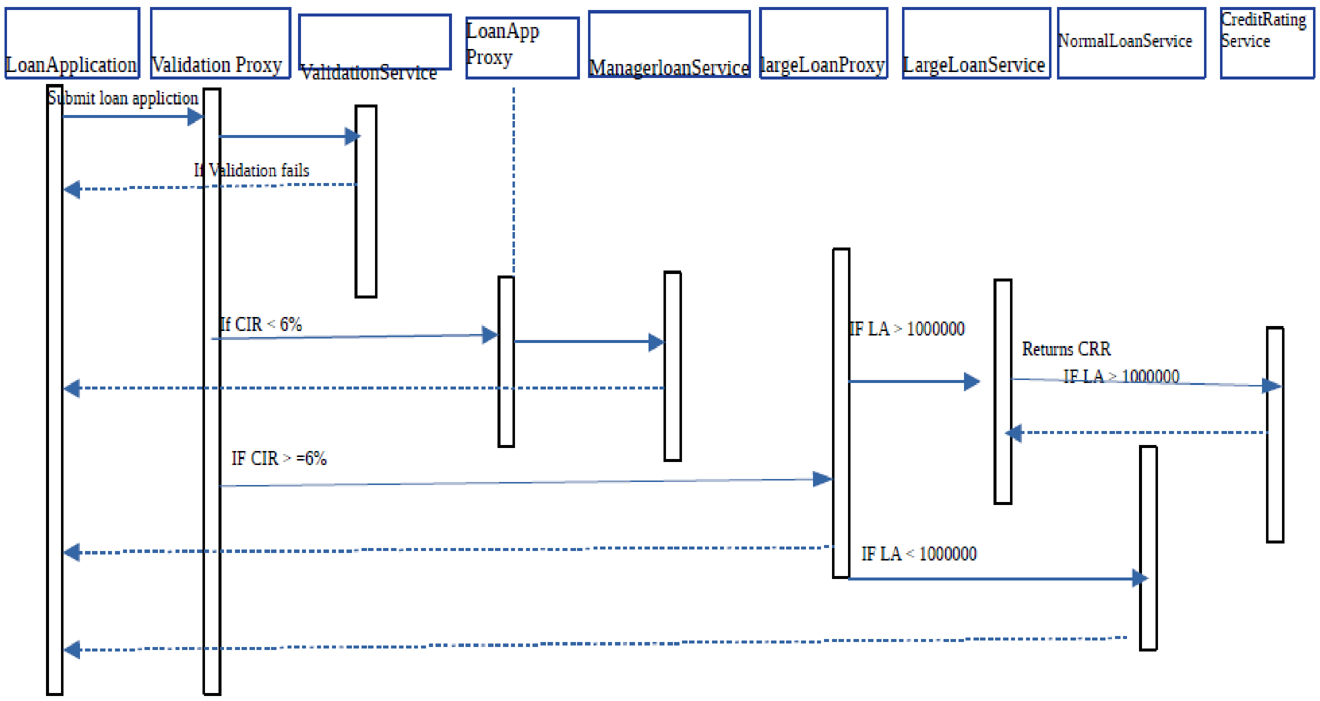

Figure 2: Sequence diagram of the application.

4. THE TESTING STRATEGY:

Now as we have already identified the crucial service and steps to test the applications we need to find out the factors responsible for the performance of the ESB. Two types of systems exist. The close system and the open system. The close system highly synchronous and a predicatble corelation can be esatblished by the participating logic. Various testing tools are devised for that. The Open System has a quasi way of execution as many request are arriving in concurrent and asysnchronous matter [5]. The our application is a open system and thus is very difficult by predicting it through the analytic s of the closed model[11]. Several researchers have tried to use closed model for this application and compared the result with the real testing tool. There is still no reliable analytical solutions for open system. Lets identify the time parameters for various services and their branches of execution.

Time Parameters:

Business Services :

NormalLoanService	: TNR
ManagerLoanService	: TML
LargeloanService	: TLL
CreditRatingService	: TCR
LoanvalidationService	: TVA

Proxy Services

ValidationProxy	: TVP
LoanAppProxy	: TNP
LargeLoanProxy	: TLP

Any additional time in sending request and receiving response :

Table 1 : Branch Time Equations

No.	Process	Time Estimate
1	Loan Application with validation failure	TVP = TVA + Ta
2	Loan Application with validation success and IR> 6	TVP = TVA+ TML+Ta
3	Loan Application with validation success and IR> 6 and Amount less than TA	TVP = TVA+ TNR+Ta
4	Loan Application with validation success and IR> 6 and Amount less than TA	TVP = TVA+ TLL+ TCR +Ta

In all the above equations the uncertainty is in the value of Ta which may vary as per the location of requester. The test results can be put into the equation and iteratively we can work out the load account on various services under various conditions of the request. This will give a reliable estimate for preparing a reasonable analytical solution for an open ended system.
The test results can be compared with the estimated demand from the literature

Table 2. Estimated demand for the services from literature (for the closed system) chapter 9 [6]

Service name	Service type	Load (ms)
TVP	Proxy	2.81
TLN	Proxy	3.46
TLP	Proxy	6.71
TML	Business	120.34
TVL	Business	78.92
TNR	Business	120.82
TLL	Business	177.10

5. RESULTS AND DISCUSSION:

5.1 The Test Setup:

To conduct the testing of various services and branch operation the application was developed on the following platform

Table 3: Test Setup

Hardware	Dell Server with 8 GB RAM quad core CPU 1.2 GHz, 1 Work station client on LAN
Software Development	Oracle Service Bus, Weblogic Server, Jdeveloper, OSB console
Programming language	Java Enterprise Edition 7
Testing Tool	*Apache Jmeter – A open source tool* . Apache JMeter ise used to judge performance both kind of resources (dynamic and static), Web dynamic applications. It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load

The test was done on the application with varied parameters and no of concurrent request by increasing the number of clients. The throughput of the test are recovered from the logs and compared with the analytical throughput.

Table 4. Error of performance modeling

	10 clients			50 clients			100 clients		
	Modeled	Measured	Error (%)	Modeled	Measured	Error (%)	Modeled	Measured	Error (%)
TVAL (15%)	13.73	13.46	2.01	13.81	14.5	4.76	13.81	15.5	10.9
TML (30%)	9.85	9.56	3.03	9.58	9.23	3.79	9.58	9.03	6.09
TLL (20%)	6.33	6.2	2.1	6.22	5.99	3.84	6.28	5.9	6.44
TNR (35%)	8.45	8.1	4.32	8.24	8.66	4.85	8.44	8	5.5

Above are the results obtained by running the test over various branches which seems to be quite in agreement with the analytical modeled results. The error is ranging between 1 to 10 % . Interesting results are obtained from the error matrix. We can see the error is under control when no of users are 50 but increasing the load somewhat deteriorates the performance and error goes up to 10% in case of Validation Service. The Normal loan Service are quite in control between the range of 4 to 5% while manage loan Service is ranging between 3 to 6%. These results can be used for creating a solution for open end system. We see that the time is not predictably in the proxy services as they are not executed in system environment but depends on lot many external factor. Substituting the time consumed by various branch outs and no of clients the analytical equations can be iteratively solved to obtain the right solution for open ended systems. Several other factors are not considered here and can be done in the future work, The factors are Authorization and Authentication load . This is typical because now a days we have several ways of security mechanism from login-password to json web tokens or OAuth and OpenId connect. Such aspects are not yet covered in the analytical models. The other factor is the network firewall and devices like routers which carry your request to the target service or proxy services.

6. CONCLUSION:

Enterprise Service Bus is a crucial part of Enterprise Service Bus Service Oriented Architecture which is the current model of development in the industry. A model was developed for the publish-subscribe architecture with SOAP based access to various services abstracting the business logic . The orchestration , transformation and routing is done by ESB. An application was developed for the purpose of testing. This application was tested for the stress or load in the LAN based architecture and a comparison was done to understand the analytical load calculation and the test results. Moreover the test validates the analytical model but in case of proxy services the error increases as the no of current user increases. This paper is an effort as how to validate a open systems like SOA applications. Using this one can work on developing a reliable model for open systems like publish subscribe SOAP and REST based applications.

REFERENCES:

1. Booth D., et. al. "W3C Working Group Note 11: Web Services Architecture", World Wide Web Consortium (W3C), February 2004, http://www.w3.org/TR/ws-arch/#stakeholder Consortium (W3C), February 2004, http://www.w3.org/TR/ws-arch/#stakeholder
2. Hashimi, S., "Service-Oriented Architecture Explained", O'Reilly ONDotnet.com, August- 2003,
3. Booth, D., et. al. (editors), "W3C Working Group Note 11: Web Services Architecture", World Wide Web
4. IBM red Book , Patterns Implementing SOA using Enterprise Service Bus
5. Almeida, V. A. and Menascé, D. A. 2002. Capacity Planning: An Essential Tool for Managing Web Services. *IT Professional* 4, 4 (Jul. 2002), 33-38. DOI=http://dx.doi.org/10.1109/MITP.2002.1046642
6. Bianca Schroeder, Adam Wierman and Mor Harchol- Balter. Open vs closed: a cautionary tale, Networked System Design and Implementation NSDI,2006.
7. Menascé, D. A and Almeida, V. A., Capacity Planning for Web Services: metrics, models and methods,PrenticeHall,2001,ISBN0-13-065903-7.
8. Chen, S., Yan, B., Zic, J., Liu, R., and Ng, A. 2006. Evaluation and Modeling of Web services Performance. In *Proceedings of the IEEE international Conference on Web Services (Icws'06) - Volume 00* (September 18 - 22, 2006). ICWS. IEEE Computer Society, Washington, DC, 437-444. DOI=http://dx.doi.org/10.1109/ICWS.2006.59
9. Davis, D. and Parashar, M. Latency Performance of SOAP based Implementation. In Proceedings of the IEEE Cluster Computing and the GRID 2002 (CCGRID'02), Berlin, Germany, IEEE,2002
10. Gregor Hohpe, Bobby Woolf, Enterprise Integration Pattern: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley Professional (October 10, 2003), ISBN-13:978-0321200686
11. Kohlhoff, C. and Steele, R. Evaluating SOAP for High Performance Business Applications: Real–Time Trading Systems. In Proceedings of theWWW2003, Budapest, Hungary, 2003
12. Menascé, D. A. 2002. QoS Issues in Web Services. *IEEE Internet Computing*6, 6 (Nov. 2002), 72-75. DOI=http://dx.doi.org/10.1109/MIC.2002.1067740
13. Menascé, D. A and Almeida, V. A., Capacity Planning for Web Services: metrics, models and methods, Prentice Hall,2001,ISBN0-13-065903-7.