Volume 1, No. 4, Nov-Dec 2010

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Adaptive SLA enforcement for SOA Applications Using a Middleware

Prof. P. Karthikeyan*[1], Prof. E.Sathya Moorthy[2] and S.Senthil Ganesh[3]
School of Information Technology and Engineering, VIT University
Vellore, India
[1]mailme_kk@yahoo.com
[3]sesas.senthil@gmail.com

Abstract: Service Level Agreement becomes the prevailing business model for specifying the QoS requirements of the web services in a complex service based environments. Such QoS requirements include service availability, service efficiency, service throughput, etc.., The Service Level Agreement for the web services are difficult to maintain since the usage of the service varies widely over time. We present a middleware architecture which dynamically allocates the clustered resources to the services. The middleware generates the SLA document on the fly based on the popularity of services deployed in the cluster. This feature of adaptive SLA generation is not done in any application server. Thus our middleware architecture eliminates the need for the service provider to manually provide the Service Level Agreement to the Application Service Provider (which needs to be replaced often). This paper discusses the middleware architecture, its implementation and sample scenario which illustrates the usage of the middleware in the real case.

Keywords: Service Level Agreement (SLA), Service Oriented Architecture (SOA), Middleware, Quality of Service (QoS), Application Service Provider (ASP).

I. INTRODUCTION

Service Oriented Architecture (SOA) is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services. Thus, the services may be distributed across an enterprise or even across the globe. There may be thousands of services defined for the same purpose and so the customer needs to select the service which suits his needs. To ease the work of the customer the Web Service providers has come up with the QoS (Quality of Service) requirements for their services which will help the customers to select appropriate service for their need.

The services are often given to the Application Service Providers (ASP) to deploy and maintain the service in a clustered (or even grid) computing environment. Thus it's the responsibility of the ASP to assign clustered resources (the number of nodes in the cluster to service the request) to the services deployed in their cluster.

All services are not treated alike by the service providers. So the ASP maintains a SLA with the service provider which specifies certain QoS factors to be satisfied by the ASP. The ASP reads the SLA document and configures the cluster to accommodate the QoS requirements of the services deployed in the cluster. The service providers can assess the popularity of the service only after its being deployed and used by the client. Thus the service provider needs to update the SLA often to accommodate the different QoS requirements of the service over time. The specification of wrong SLA by the service provider will be unprofitable to both the ASP and the service provider.

Thus we are in need of an adaptive SLA enforcement mechanism for the SOA based applications. The addition of new mechanism should not bring down the performance of the application server. The implementation of the mechanism should also work cooperatively with the other services. Thus, we are in need of middleware services to the application server. It allows the ASP to configure the existing application server only to the minimum and the feature can be enabled/ disabled by doing few changes to the configuration of the application server.

The reminder of this paper is organized as follows; section 2 presents the literature reviews of various concepts involved in our paper. Section 3 presents the proposed middleware architecture and its interactions with various components in it. Section 4 discusses generation of SLA based on the runtime configuration of the deployed services. Section 5 discusses the implementation feasibility of the middleware in JBoss Application Server; Section 6 presents a scenario that corresponds to the real case. Section 7 is the conclusion.

II. RELATED WORK

The QoS requirements are specified using a SLA document which can be a simple XML file. There are various SLA Specification Languages such as WSLA and SLAng. Mostly the specification of the SLA can be satisfied by configuring the Application Server cluster, network routers, database management systems, middleware and so on.

The Web Service Level Agreement (WSLA) addresses the service level management issues and challenges in a Web Services environment on SLA Specification, creation and monitoring [1]. The WSLA framework consists of flexible and extensible language based on XML Schema and a runtime architecture comprising several SLA monitoring services, which may be outsourced to third party monitoring services. However, a WSLA only covers the agreed common view of a service between the parties involved. To actually act as a participant in a WSLA, parties have various degrees of freedom to define an implementation policy for a service and its supervision. Typically the obligations of a WSLA must be translated into system-level configuration information, which can be proprietary to each party involved.

The SLAng is a language for defining Service Level Agreements which falls under the three primary services: network services, Storage Services and the middleware services

[2]. SLAng besides specifying the QoS requirements of the services at the application layer level, it specifies the contractual agreements that are necessary when different ISO /OSI layers of a deployment are spread across multiple organizations.

III. MIDDLEWARE ARCHITECTURE

The middleware addresses the following issues in the SLA driven maintenance of SOA Applications.

a. The service provider often needs to specify the SLA document with changing requirements of the services.
b. The service provider may not be accurate in specifying the QoS requirements for the deployed services.
c. Optimized resource utilization in assigning the clustered resource to process the service request.

To address these issues we proposed QoS middleware architecture which incorporates the three principle services: SLA Generation and Monitoring Service, Cluster Configuration Service, and the Load Balancing Service. This architecture will be designed to be deployed in a cluster of Application Servers. The cluster will contain a set of application server instance. Each node (Application Server Instance) will contain the copy of these services running in it.

The principle responsibilities of these services are as follows:

SLA Generator and Monitoring Service is responsible for generating the SLA documents for the services deployed in the cluster. The SLA is generated whenever the popularity (usage) of the services in the cluster gets changed. This service computes the popularity Factor based on its usage and the maximum resource that can be allocated to that particular service. The percentage of clustered resource that can be assigned to a service depends on the above mentioned factors of the remaining services in the cluster. Thus change in configuration of one service will reflect the change in SLA parameters for the remaining services also. A *logger service* maintains the history of SLA specification for each service deployed in the cluster. This log can be sent to the service provider for his reference.

The change in SLA document will trigger the configuration service to reconfigure the cluster configuration if necessary.

Cluster Configuration Service is responsible for configuring the cluster in order to honour the SLA of the services deployed in the cluster. The main activities performed by the cluster configuration service is configuring the cluster at the web service deployment time and possibly reconfiguring the cluster at the run time.

The cluster configuration process takes the default SLA parameters which is generated by the SLA Generator and Monitoring Service and forms a initial cluster from a minimum set of available nodes (application server instance) to ensure that the default SLA requirements of the deployed service is met.

The cluster re configuration process takes place to account for the dynamically increasing load on the cluster and in case a clustered node fails and needs to be replaced by an operational one (or more than one); for this purpose a pool of spare nodes is maintained.

The nodes may also be released from the cluster in order to optimize the resource utilization. If the load on the hosted web service significantly decreases, some of the nodes allocated to that application may be dynamically de allocated and included in the pool of spare nodes for future use.

Load Balancing Service is implemented at the middleware level and balances the load of SOAP client requests among the clustered nodes. It is aware of the runtime operational conditions of the nodes in the cluster. Thus it helps in avoiding overloading any particular node or sending requests to nodes which may be unavailable at that instant. Thus Load Balancing Service provides high availability and as well as it contributes in honouring the SLA of the deployed services.

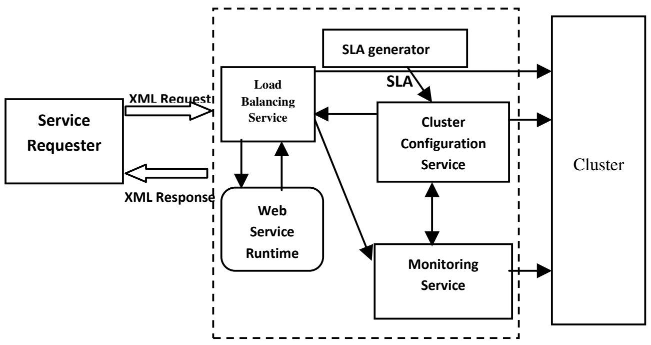

Figure 1: Middleware service interactions

The Load Balancing service uses a WSDL (Web Service Definition Language) file to extract technical interface details about a Web Service and uses information returned to create an actual SOAP request to that Web Service. The Load Balancer Service then emulates a real Web Service Client making a request. The SOAP request can be used to confirm that the Web Service is serving the expected response data and in a timely manner in correspondence with the SLA.

A. *Middleware interactions*

Our middleware services co-operate with each other to dynamically generate the SLA and honour the generated SLA by configuring the cluster at runtime.

Fig 1 shows how the services interact. In Fig 1, the service requester sends the XML request to call a web service in the cluster. The request is intercepted by the load balancer. When a request for a particular service is received the popularity of the service gets changed. The SLA monitoring services notifies this change and calls the SLA generator to generate the new SLA documents for the services. The configuration service takes the generated SLA and sends the SLA parameters to the SLA monitoring service. The SLA Monitoring Service cooperates with the load balancing service to obtain the QoS delivered by the cluster. i.e., the maximum service the cluster can provide at this instant. The monitoring service then computes the monitoring parameters such as the popularity Factor and efficiency index which serve to check whether cluster operational conditions are close to violate the generated SLA for the requested service.

SLA Monitoring service firsts checks the client responsibilities of the generated SLA. If clients are sending more number of requests than the allowable limit then an application level exception is raised and no other action is taken by the cluster. Next the SLA monitoring service checks the server responsibilities of the generated SLA. If the cluster operational condition is close to breach the SLA parameters then the monitoring service calls the configuration service to reconfigure the cluster. The configuration service acts upon the cluster by adding new nodes to the cluster up to a predefined limit. Adding nodes to the cluster causes the performance to be increased which can ultimately satisfy the generating SLA.

Node failures and voluntary disconnections are detected by Monitoring Service, which then raises an exception to the cluster Configuration Service. In both cases the configuration service re configures the cluster;

IV. SERVICE LEVEL AGREEMENT

A service level agreement is an arrangement between a customer and a provider, describing technical and non-technical characteristics of a service, including QoS requirements and the related set of metrics with which provision of these requirements is being measured [1].

The content of an SLA varies depending on the service offered and incorporates the elements and attributes required for the particular negotiation. The SLA specifies the following responsibilities which are as follows:

Client Responsibilities – This must be stated by the client of the service. In our middleware it is obtained only once from the client.

Service Responsibilities – It includes the responsibilities which must be satisfied by the application server cluster in delivering the service. It is not obtained from the service provider. Our middleware will generate this part of the SLA based on the popularity of the service.

Mutual Responsibilities – It specifies clearly the charges and benefits of the two parties. Some of them might overlap [1]. These requirements are compiled by both the client and service provider. This part of the SLA document is also obtained once from the client and service provider.

A snapshot of the partial SLA which is based on the SLAng Specification language is as follows:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <SLAng xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance

xs1:noNamespaceSchemaLocation="abc/QaASCluster/SLAng1_1/SLA.xsd">
         <Vertical>
           <Hosting>
            <Client>
             <Name> ABC private Lmt </Name>
             <Place>Texas</Place>
             <Availability>95%</Availability>
           <Client>
           <Server>
             <Name>MyQaASCluster</Name>
               <Place>Chennai</Place>
             <HighAvailability>99.5</HighAvailability>
             <Performance response time="2.6"/>
             <Cluster_throughput containers="9" active clients ="310"
                                  method_invocation="53.141" />
               ....
            </Server>
          <Mutual>
                  <service_schedule start="2009-12-12" end = "2010-12-12" />
                  ....
           </Mutual>
         </Hosting>
       </Vertical>
     </SLAng>
```

Fig -2 SLAng Specification of SLA document

In fig -2 the tags which are in bold face are generated by the SLA generator module of the middleware based on the runtime popularity of the deployed service while the rest of the portion of the document is obtained from the client and the service provider only once.

The SLA documents are enforced by the SLA Enforcement engine which co-ordinates with the Web Service runtime, manipulates the WSDL file accordingly and deploys the file to the web service runtime.

V. JBOSS IMPLEMENTATION

We conducted a feasibility study on implementing our middleware architecture on JBoss v 4.0.4.GA Application Server. JBoss consists of a collection of middleware services for communication, persistence, transactions and security. [5] These services interact by means of a microkernel based on the Java Management Extension (JMX) Specifications [6]. JMX defines a common software bus that allows the java developers to integrate components such as modules, containers and plugins. These components are declared as Managed Bean (MBean) services; they are loaded into JBoss and administered by the JMX software bus.

A number of JBoss Application Server instances can be started so that it forms a cluster. The nodes in the cluster communicate with each other by the underlying group communication mechanism, namely, JGroups, included in the JBoss Application Server. JGroups provides the clustered nodes with reliability properties such as lossless message transmission, message ordering and atomicity.

Each service will be serviced by a subset of nodes (referred as partition). The number of nodes in the partition depends on the runtime SLA of the service. A node (application server instance) may be present in more than one partition. The service is deployed with the binding addresses of all the nodes in its partition. Thus whenever a web service requests comes then the load balancer module selects a node from its partition to service the request. The selection of node is based on the load balancing algorithms such as Round Robin algorithm, Random Selection, Queue Selection and so on.

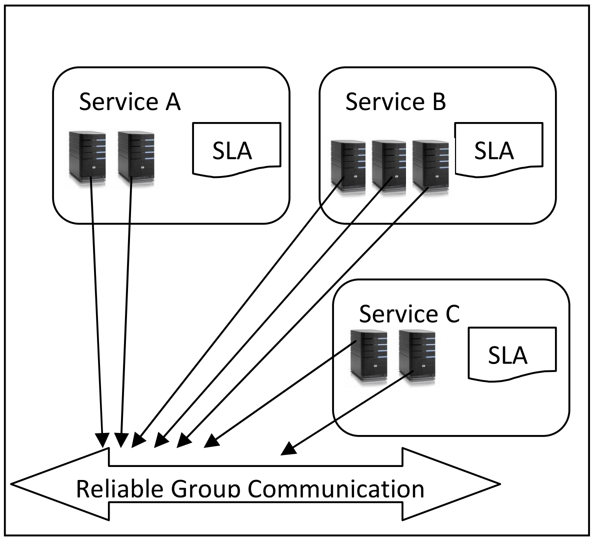

Figure 3. Logical representation of services and their partitions.

The cluster configuration service starts up the additional server nodes by means of an external program which starts or stops the application server in the network. The external program uses RMI to start/stop the server instance. The cluster configuration also maintains the partition information for each service. The SLA monitoring service evaluates the partition members of each and every service and notifies the cluster configuration service in case of adding/removing nodes from the sub partition. Adding new nodes to the partition is required if the generated SLA requires new nodes to be added to the partition to honour the SLA. The removal of nodes assigned for a partition of any service is required if honouring the SLA of the service doesn't require the extra node to be present in that partition. Thus the removed node can be used by other partitions in the cluster. In case of situations where there is no exclusive node added to the partition to service the web service request then the node from any other partition may be shared with this partition. The impact of sharing the node from one partition to the other is assessed by the SLA monitoring service and the partition may be re configured if necessary.

VI. SAMPLE SCENARIO

Consider a sample scenario of Book Shop application which is deployed as a service to the Application Server cluster. The client of the Book Shop application gives the client responsibilities of the SLA document. The service provider and the client of the service co-operatively provide the mutual responsibilities of the SLA document. The partial SLA document is given to the SLA generator Module of the middleware. The SLA generator Module initially fills in the server responsibilities of the SLA document with the default values for the Performance response time and cluster throughput which is based on the assessment tests carried out in the cluster environment. The assessment tests are out of scope of this paper.

For each client invocation of the service the client responsibilities are checked by the SLA monitoring service. If the client breaches the client responsibilities specified in the SLA then the application level exception is raised and the client is blocked from using the service.

The popularity of the service is updated for each successful invocation of the service. The SLA monitor evaluates the updated popularity against the SLA parameters specified in the document along with the current cluster configuration and cluster efficiency. If the change in popularity of the service needs to update the SLA, then the SLA Monitor notifies the SLA generator to generate a new SLA for the service. The SLA generator generates the new SLA document based on the popularity of the service for which the document is going to be generated and the popularity of other services in the cluster. The SLA generator notifies the cluster configuration service to modify the cluster configuration to accommodate the changes made to the SLA document.

Thus the SLA document is generated dynamically which is monitored and honoured by the middleware services.

VII. CONCLUSION AND FUTURE WORK

In this paper we discussed a middleware architecture which facilitates the adaptive SLA generation for the services deployed in the application server cluster. The paper also discusses the various components of the middleware and their interactions. A sample scenario is presented which clearly explains how the services interact and generates the SLA at runtime.

What we do next is to take advantage of the design idea of the middleware discussed in this paper and implement the prototype of the middleware services using JBOSS Application Server which is a fully complaint J2EE application Server with robust EJB Support.

VII. REFERENCES

[1] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P.King, Richard Franck, "Web Service Level Agreement (WSLA) Language Specification".

[2] D.Davide Lamanna, James Skene, Wolfgang Emmerich, "SLAng: A Language for Defining Service Level Agreements"

[3] "JBoss Enterprise Middleware System," JBoss, http://www.jboss.org, 2006.

[4] "SLA for Application Service Provisioning," ASP Industry Consortium White Papers, http://www.allaboutasp.org, 2006.

[5] M. Fleuryy and F. Reverbel, "The JBoss Extensible Server," Proc. ACM/IFIP/USENIX Int'l Middleware Conf., June 2003

[6] G.Lodi and F.Panzieri, "QoS-Aware Clustering of Application Servers," Proc. First IEEE Int'l Workshop Quality of Service in Application Servers/23rd Int'l Symp. Reliable Distributed Systems (SRDS '04), Oct. 2004.