International Journal of Engineering &Technology, 7 (3.12) (2018) 740 -743

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Dynamic High Availability Architecture Framework for SOA Computing

S.Kumaresan[1*], Sumithra Devi.K.A[2]

[1]Bharathiyar University, Tamil Nadu
[2]DayanandSagar Engineering & Technology,Bangalore
**Correponding Author Email: [1]skumaresan_77@yahoo.com, [2]sumithraka@gmail.com*

Abstract

In Software technology stackCloud services provides easy coupling implementation to enhance encapsulation data between multiple platform data exchanges. My finding towards introducing High Availability Architecture for cloud environment which covers Load Balancing, Failover, High Availability Resources. To achieve thisfeatures it's identified framework architecture which is called as Dynamic High Availability Architecture Framework for SOA Computing which increase cloud services standard inhigh witheasy adaptable security. Even though cloud service supports loose coupling and isolation business logics. At current cloud service provide wants to launch new web service request on fly same service will notnotified into client in real-time scenario. To overcome this complicated situation we have introduced (GHAFC) Generic Architecture Framework in Cloud Computing. Which will support data exchanges between producer and consumer onthe fly with real time scenario.

1. Introduction

This paper provides Generic Architecture Framework for CloudComputing by which we achieve High Availability, Load Balancing, and Failover. Access the applications as utilities over the internet. It allows us to create, configure, and customize the business applications online.Cloud Computing can be defined as delivering SAAS,PAAS,IAAS (CPU, RAM, Network Speeds, Storage OS software) a service over a network (usually on the internet) rather than dedicated data center having the computing resources at the customer location and hosted remotely. Given above architecture applications performance increases and CPU processor

Dynamic High Availability Architecture Framework suit for below environment.

- Public Cloud Model
- Private Cloud Model
- Hybrid Cloud
- Virtualization
- Hardware Virtualization
- Software Virtualization
- Server Virtualization
- Storage Virtualization
- Supports High Availability
- Supports Fail Over
- Supports Load balancing

High Level Component Architecture captured in Figure-1, contains 3 Major section which is High Availability, Failover, Software Load balancing from Application Server

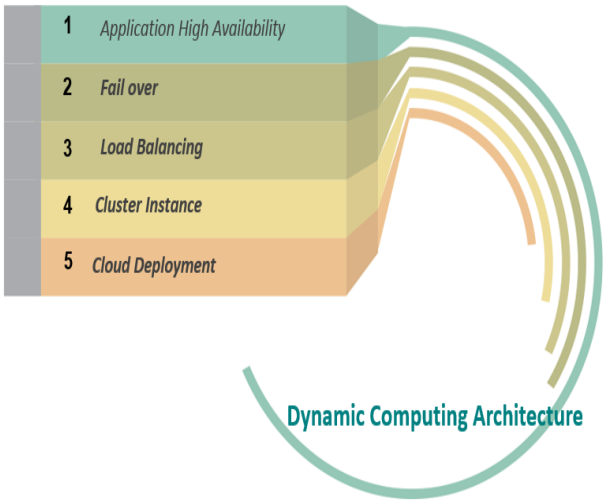


Figure 1 -HAGFC High Level Components Diagram

1. 1 Web Servcie Data Isoloation Layered

Interact with Multiple System or Platform communication we can uses Web services are long livable mainstream technology in common use throughoutmultiple domain business industries, and allow widely used in enterprise boundary and in B2B and B2C stream. Business domain essentially encapsulate a confidential data into simple form and exposed as functional capabilities for invocation by client program via a web service. Web service will bridge the multiple system and stitching the sharing multiple complex business data over the internet protocol.

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

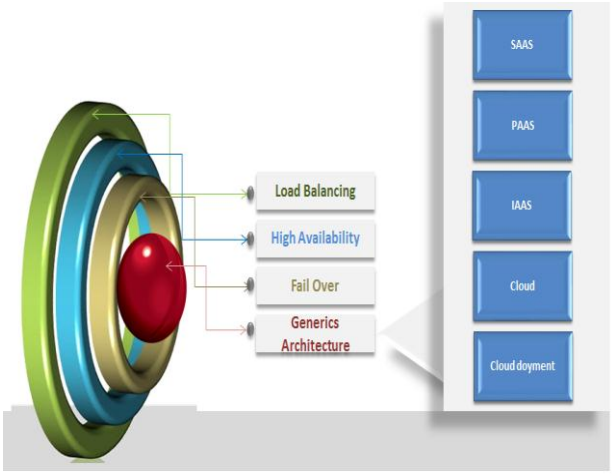


Fig. 2: HAGFC Framework Layered Diagram

Above figure described into HAGFC layered architecture which is used for cloud computing as a type of framework that relies on support high availability , load balancing , failover computing resources, having different services delivered to an cloud hosting and devices through the Internet rather than having local data center or personal computer to deploy and handle applications.
In order to archive 99.9999 application availability in cloud computing uses networks of large groups of hardware servers connected together to divide the resource and data-processing across nodes.Generic Architecture Framework techniques are used to maximize the power of cloud computing.
The goal of Generic Architecture Framework cloud computing is to apply high performance computing power in Infrastructure, Platform and Application which served to consumer applications in order to save time and financial resources to purchase, deploy and maintain an infrastructure in dedicated data center.

1. 2 Hybrid Data Service Technology

Information Industries started adopting Cloud Computing and also changed the way companies looking into their digital Infrastructure now a days. Generic High availability cloud computing with its unique architecture brings in new opportunities and challenges implementation. Generic Framework unique curriculum content which will bring beginners easy implementation with Cloud technologies.
The Generic Framework will start with basic introduction to cloud concepts like SAAS, PAAS and IAAS. You will also implement with Linux systems or window and changing the Infrastructure landscape in cloud.

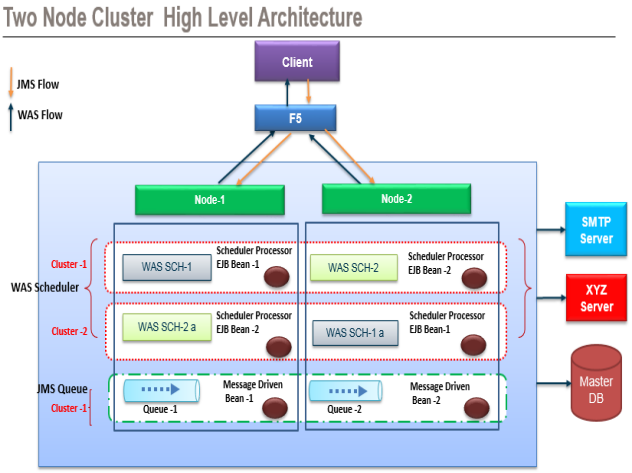


Fig. 3: HAGFC Framework Data flow Diagram

1.3 Generic Web Servcie Data Coupling

Above figure described into HAGFC data flow architecturediagram contain below

- Single Web Admin application used to configure and control all the application.
- Single Application Scheduler Instance used for all the Domain
- Single Web Instance used for all the Web application.
- Single JMS Instance used for all the messaging in form of MDB's.
- One Generics. Jar file used for Web/JMS/E-mail Scheduler applications.
- One Add-ons. Jar file used for Web/JMS/E-mail Scheduler applications.
- Each addition of new application deployment required.
- One time deployment for Web/JMS/E-mail Scheduler in case of maintenance.
- One dedicated EJB configuration for Each train.
- Less Server Down time window period required for maintenance.
- Deployment System Changed.

2. Proposed Solution

Using Elastic Architecture frame work we could able to achieve Runtime data mapping with Manager Node and Data node. One Manager Node will connected with multiple data node which will store data in to 3 backup.
And also used Docker tools to moved application build into cloud on the fly for that no server down time required. Web service support both SOAP & Restful services caching infrastructure over HTTP GET method (for most servers). This can improve the performance, if the data the Web service returns is not altered frequently and not dynamic in nature.
Given below figure High Availability Architecture explain consumer and provider data exchange between application server. Server configured with clustered mode witch scheduler based JMS architecture. Which will provide high performance fail over application infrastructure.

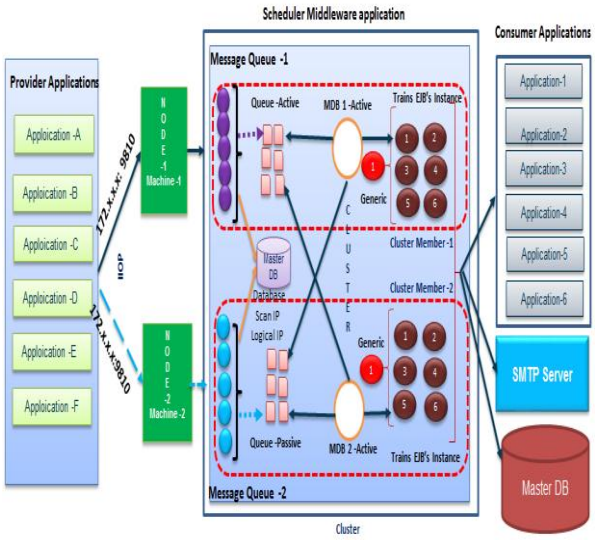


Fig. 4: High Availability Architecture

2.1 WEB Service Data Coupling

Static Resource Mapping Web service Architecture used in both SOAP and RESTFUL Web service, Here Server Data Resource Provider will be having Static and Dynamic Data which will be given to Consumer to Mapping and Binding Consume the Service. Consumer has to do configuration according to provider Resource

for this consumer will spend more time in his application to consume provider service.
If any changes done in provider side consumer may not aware of the changes and its lead to the error or unable to find the Resource from the provider end. And provider also need to inform to Broker and Broker will inform to new consumer and existing consumer whenever resource changes done in provider same time consumer also need to do changes otherwise Consumer, Broke and Provider will not be in Sync.
Even REST Web service also consumes static Resource from the Provider, its totally tightly coupled with Resource in Consumer and Provider. Restful web service are not having Broker Module in between Provider and Consumer. Restful always directly tightly coupled with Provider and consumes static data Figure 5-blow shows the architecture of the Static Resource Mapping Web service that consists of the following components

2.2 Agent Messaging Framework

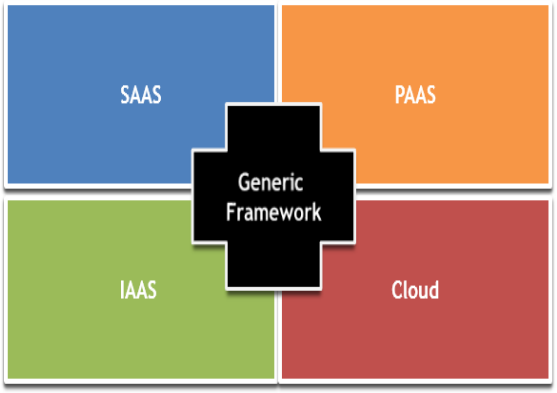


Fig. 5: HAGFC building block diagram

Below diagram Generic HA architecture for Cloud computing Dynamic Resource Mapping Web service that consists of the following components:

2.3 Web Servcie Load Balancing

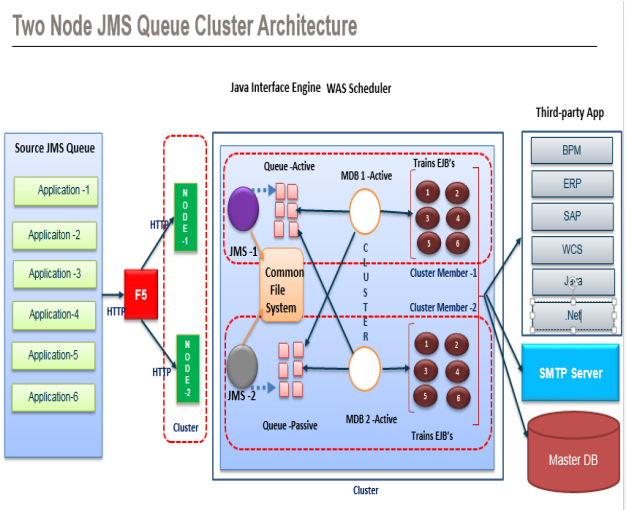


Fig. 6: HAGFC Web Service Load Balancing

WAS Scheduler in Cluster Environment:

Advantages:
- Generic WAS scheduler
- High Availability
- Load Balancing
- Fail over
- Scalability

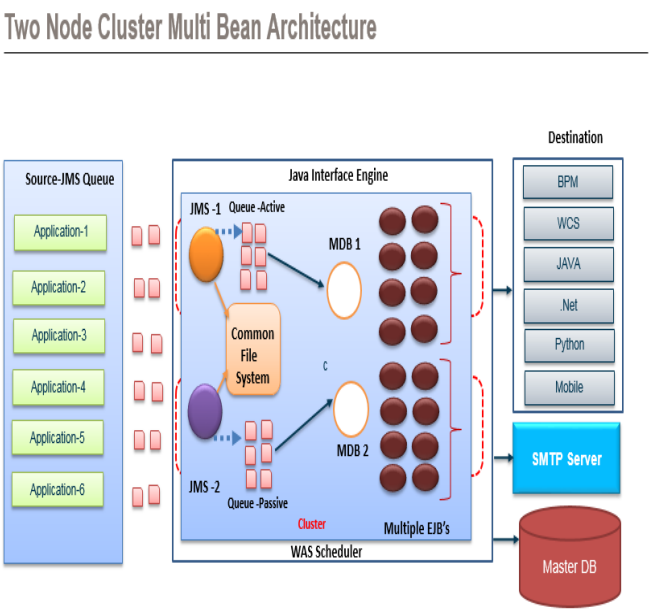
3. Results

The HA Generics tested with Hybrid Elastic Data Architecture forBidirectional Dynamic Resource Mapping using Data Grid is Implemented and Tested and also captured result. As mentioned, both End Service Provider and Service Consumer achieves Loose Coupling Pattern and Resource Access Dependency has been removed.
The system Architecture Design "Phase One" was implemented in testing phase as previously described. As mentioned above, both Client Service Consumer and End Service Provider are connected through Data Grid. And uses Java programming language. A prototype as part of the suggested system has been built and all the above mentioned features have been implemented.

Implementation:

- Used Java Technology Communicate with Data Grid.
- Used Eclipse Editor for Coding
- Used WXS 7,0 frame work to store Data Grid Key and Value
- Used Restful Web service and **Json** Key Value stored in to Data Grid
- Deployed Java application in to web server Tomcat

a) HA Generics Elastic Data Grid Dynamic Resource Mapping Implementation Low level Class Design Diagram:



Methodology Used:

Used SOA Web service Methodology to achieve this Result and followed Enterprise Architecture with B2B business required specification

Strengths

- Hybrid provide Loose coupling Resource Mapping
- It support Dynamic Resource Mapping, Key & Value can be changed on fly
- This Architecture More suitable for Cloud Based SAAS Environment
- URI Resource file Storage can elastic because of using Data Grid Storage concepts
- No server Down time required

- No need to restart Consumer and Provider Server for changes flection
- High volume of Key and Value can be stored
- Based on Data side Storage will be enhanced
- No Data failure and high through put
- Its support for archiving
- Generics frame work supports High availability and failover, load balancing in application server its self, not required physical load balancer.

Weaknesses

- Required Little effort to configure Container Server and Data Sever and storage partition
- Frame work up gradation required based on release
- Little more expensive

4. Conclusion

UsingGeneric Hybrid Elastic Data Grid Architecture, will be achieved Bidirectional Dynamic Resource mapping in web service. And also called as Centralized Resources sharing and configuration between Consumer and Provider.

The specifications for Generics Hybrid Elastic Architecture supports Bidirectional Dynamic Resource Mapping web services with Loose Coupling between Provider/Consumer. This Architecture improves the interoperability among Cloud in SAAS. The GHAFC architectures will allow Consumer and Provider users to work together in new ways and deliver more Flexible and no resource dependency.

Generics Hybrid Elastic Architecture Bidirectional Dynamic Resource Mapping supports both SOAP & Restful Web service with very less Effort and one time configuration required in Consumer and End Service Provider.

Generics Hybrid Architecture provides Elasticity which adopts key elements, Map Sets of both Distributed and Non-distributed System features. The Main advantage of using Data Grid is flexibility to create partitioned and store Data Object.

The current Cloud Web service SOA Architecture information has several disadvantages such as tightly coupling, Static Resource Mapping, Consumer totally depends on Provider Resource path. If any changes taken place in Provider side it has to inform to Client in very high priority basis otherwise client and server connectivity will be totally disconnected.

To overcome these existing problem we have introduced Generics Hybrid Architectureweb service for both SOAP and Restful webservice, which will provide loose coupling and also dynamic Resource data mapping which will help both provide and consumer to avoid inter dependency.

References

[1] BHARGAVAN, K., FOURNET, C., AND GORDON, A. D, "Verifyingpolicy-based security for web services", In CCS '04: Proceedingsof the 11th ACM conference on Computer and communicationssecurity , pp. 268–277, 2004.

[2] BHARGAVAN, K., FOURNET, C., GORDON, A. D., ANDO'SHEA, G, "An advisor for web services security policies", InSWS '05: Proceedings of the 2005 workshop on Secure web services(New York, NY, USA, ACM, pp. 1–9, 2005.

[3] CANTOR, S., MOREH, J., PHILPOTT, R., AND MALER, E, "Metadata for the OASIS Security Assertion MarkupLanguage (SAML) V2.0", OASIS Standard, 15.03.2005, 2005. http://docs.oasis-org/security/saml/v2.0/saml-metadata-2.0-os.pdf.

[4] D. Richards, S. van Splunter, F. M. T. Brazier, and M. Sabou, "Composing Web Services using an Agent Factory", In Proceedings of the 1st International Workshop on Web Services and Agent Based Engineering, Sydney, July 2003.

[5] M. O. Shafiq, A. Ali, H. F. Ahmad, and H. Suguri, "AgentWeb Gateway - a Middleware for Dynamic Integration of Multi Agent System and Web Services Framework", In Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pages 267–270, Washington, DC, IEEE Computer Society, 2005.

[6] L. Z. Varga and ´A. Hajnal, "Engineering Web Service Invocations from Agent Systems", In Proceedings of the 3rd International Central and Eastern European Conference on Multi Agent Systems, pages 626–635, Prague, Czech Republic, June 2003.

[7] BENAMEUR, A., KADIR, F. A., AND FENET, S, "XML RewritingAttacks: Existing Solutions and their Limitations", In IADISApplied Computing 2008 (Apr. 2008), IADIS Press.

[8] CHAN, Y.-Y, "Weakest link attack on single sign-on and its casein saml v2.0 web sso", In Computational Science and Its Applications- ICCSA 2006.

[9] M. Gavrilova, O. Gervasi, V. Kumar, C. Tan,D. Taniar, A. Lagan, Y. Mun, and H. Choo, Eds., vol. 3982 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 507–516. 10.1007/11751595 54, 2006.

[10] LUTZ, D., AND STILLER, B, "Combining identity federation withpayment: The saml-based payment protocol", In Network Operationsand Management Symposium (NOMS), 2010 IEEE, pp. 495 –502, April 2010.

[11] RAHAMAN, M. A., MARTEN, R., AND SCHAAD, A, "An inlineapproach for secure soap requests and early validation", OWASPAppSec Europe, 2006.

[12] EASTLAKE, D., REAGLE, J., SOLO, D., HIRSCH, F., ANDROESSLER, T, "XML Signature Syntax and Processing ",(SecondEdition), http://www.w3.org/TR/xmldsig-core/., 2008.

[13] GAJEK, S., LIAO, L., AND SCHWENK, J, "Breaking and fixingthe inline approach", In SWS '07: Proceedings of the 2007 ACMworkshop on Secure web services (New York, NY, USA), ACM, pp. 37–43, 2007.

[14] GROSS, T, "Security Analysis of the SAML SSO Browser/ArtifactProfile", In ACSAC, IEEE Computer Society, pp. 298–307, 2003.

[15] GRUSCHKA, N., AND IACONO, L. L, "Vulnerable cloud: Soapmessage security validation revisited", In ICWS [1], pp. 625–631.

[16] GUDGIN, M., HADLEY, M., MENDELSOHN, N., MOREAU, J.-J., AND NIELSEN, H. F. SOAP Version 1.2 Part 1: MessagingFramework. W3C Recommendation, 2003.