Gajaba: Dynamic Rule Based Load Balancing Framework

Y. Pandithawattha, K. Perera, M. Perera, M. Miniruwan, M. Walpola, and A. Azeez

***Abstract*—Cloud computing has become the norm of today's heavily used computer science applications. Load balancing is the key to efficient cloud based deployment architectures. It is an essential component in the deployment architecture when it comes to cloud native attributes of multi-tenancy, elasticity, distributed and dynamic wiring, and incremental deployment and testability. A load balancer that can base its traffic routing decisions on multiple cloud services is called a service-aware load balancer. We are introducing a novel implementation of a flexible load balancing framework which can be customized using a domain specific scripting language. Using this approach the user can customize the framework to take into account the different services running on each cluster (service-awareness) as well as the dynamically changing tenants in each cluster (tenant-awareness) before making the load balancing decisions. This scripting language lets users to define rules and configure message routing decisions. This methodology is more light weight and expressive than products already available, making the cluster based load balancing more efficient and productive..**

***Index Terms*—Load balancing, tenant-awareness, cloud computing, customizable framework.**

I. Introduction

Computer networks have grown from small scale intranets that spread across a single room to worldwide networks that interconnect every region around the globe. The Internet can surely be identified as the largest network, which is composed of other networks such as corporate networks, campus networks, factory networks and home networks around the world.

With the emergence of internet based services such as the World Wide Web and Electronic Mail, the information flow between two nodes in a network has increased vastly over the last decade. Because of this, network congestion has become a major problem. And because some nodes receive higher number of requests than others in a network, those nodes are overloaded and the overall performance of the network degrades. It is unacceptable for a network to go down or exhibit poor performance as it can literally shut down a business in a networked economy. The main logic behind load balancing servers and networks is to even out the network information flow among nodes to boost performance and reduce network congestion.

As the Internet and the intranets that it is composed of have become the operational backbone of businesses, two types of equipment can be identified as the business IT infrastructure. They are computing devices that function as a client and/or a server, and switches and routers that connect these devices [1]. Load balancers act as a bridge between the servers and the network. On one hand they must have knowledge of higher level properties of servers in order to communicate with them intelligently and on the other hand they must understand network protocols to integrate with them effectively [1] Simple input load distribution is not the only functionality that is expected from a typical load balancer. Server health monitoring, keeping session persistence, fault tolerance and changing the load distribution scheme according to various conditions are some of the many capabilities of today's load balancing products.

A lot of research has been conducted on load balancing algorithms and on how to achieve the other additional requirements. The core algorithms can be broadly divided in to categories such as Client based, DNS based, Dispatcher based and Server based algorithms [2] Many existing load balancers can switch between these algorithms dynamically based on the availability and congestion of nodes (knowledge from server health monitoring) and the services running on them (WWW, SMTP etc.).

But today, networks have evolved from simple interconnected information nodes to complex interconnected service clusters. Users in a network have become service consumers rather than simple information requesters. In this environment, the demands for new services as well as existing services increase in an exponential rate. To serve this growing demand, the IT infrastructure of corporate service providers must take advantage of new approaches like Web Services, Service Oriented Architecture [3] and Software as a Service (SaaS) [4].

Because of this, load balancers must take into account these factors in order to provide better functionality. They must have knowledge of high level services (here, high level services means application level services such as order processing services, credit transaction services etc.) a cluster of servers provide as opposed to the low level services individual servers provide. This knowledge is then used to make intelligent load balancing decisions.

The term "Cloud Computing" refers to the technology that allows consumers to use applications and services without having to install them or deploy them and access their personalized data and services from anywhere in the world with the internet access. One such example is Google, providing Web search engine services, emailing, document sharing, application sharing and many other facilities. According to Rosenberg and Mateos [5], the five main principles behind cloud computing are:

- Pooled computing resources available to any subscribing users.
- Virtualized computing resources to maximize hardware

Manuscript received March 17, 2013; revised May 22, 2013.

Y. Pandithawattha, K. Perera, M. Perera, M. Miniruwan, and M. Walpola are with Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. (e-mail: malakajw@uom.lk).

A. Azeez is with WSO2 Inc, Mountain View, CA, USA. (e-mail: afkham@gmail.com).

utilization.
- Elastic scaling up or down according to need.
- Automatic creation of new virtual machines or deletion of existing ones.
- Resource usage billed only as used.

The key to cloud based deployment architectures is load balancing. It is an essential component in the deployment architecture when it comes to Cloud native attributes of multi-tenancy, elasticity, distributed and dynamic wiring, and incremental deployment and testability. In cloud deployments, a cluster of nodes that performs a single collaborative service is called a "Cloud service". A load balancer that can base its traffic routing decisions on multiple cloud services is called a service-aware load balancer. The most important difference between a typical load balancer and a service-aware load balancer is that a service-aware load balancer can route traffic to the correct cluster and balance the load according to the algorithm specified by each cluster.

II. Existing Load Balancing Systems

Load balancing systems can be divided into two main categories: Network and server load balancers and Content based load balancers. But today, the margin between these two types is blurred and almost every existing system can be identified as both network/server and content based load balancer. Existing load balancing products such as Citrix NetScaler [6], F5 BIG-IP Load Traffic Manager [7], CoyotePoint Equalizer [8] and A10 Networks AX Series [9] were studied to identify common functionality. Some of the most common features of load balancing systems are listed and explained below.

A. Layer 4-7 Switching

Load balancers today has the ability to inspect message header information of not just Layer 2 or 3 in the OSI reference stack, but also Layer 4 to 7 header information. Their load balancing algorithms take this information into account and this enables them to make intelligent routing decisions based on application level information in the servers. This can vastly increase the overall system performance, security, availability and scalability. This functionality is provided by many existing systems, such as Citrix NetScaler, CoyotePoint Equalizer E250GX and F5 BIG-IP Load Traffic Manager.

Compared to the above products, A10 Networks AX Series of load balancers provide a unique feature called aFlex technology [10]. Although AX products support almost all the Layer 4-7 protocol based switching, aFlex tool allows users to write their own scripts using the TCL scripting language to inspect packets and traffic manipulation. This gives much flexibility as the user is given the ability to define the application awareness to address any type of application.

B. Session Affinity (Persistence)

This is the ability to identify packets from a client within the boundary of a session, and route them to the same node until the session is closed. One way to ensure session affinity is by routing packets to the same server based on the client's IP address. But this alone will not solve the problem as IP addresses are shared by many clients in environments with proxy servers. Some other schemes to achieve this are cookie based policies, server based policies, group based policies and JSESSIONID based policies. All the products used in this study provided mechanisms to assure session affinity.

C. Server Health Monitoring

Load balancers must be aware of the availability and current working conditions of each server connected to the network. This is attained by performing health checks on servers from time to time. Routing decisions are effected based on the results of these health checks. Some health monitoring schemes are as simple as pinging the server and others are more complex schemes such as software agents in servers monitoring the health and reporting to the load balancer.

A10 Networks AX Series supports TCL scriptable health checks. CoyotePoint Equalizer E250GX uses a technology called Active Content Verification (ACV) [11] which uses a server side agent to perform health checks and send details to the load balancer. Other products also use scriptable health check support.

D. Fault Tolerance

This is to ensure that the load balancer would not become a single point of failure. Load balancing products today has the ability to interconnect two or more of them in different configurations. As one system fails the other/others can take its place and continue serving the system. All products used in the study were able to configure to handle fault tolerance.

E. Multi-tenancy

Multi-tenancy is the principle of single instance of software running on a server, serving multiple clients. This is one of the core concepts in Cloud Computing, as applications are designed to virtually partition data and configurations (known as tenants) to serve multiple clients. Clients can work with a customized virtual application instance. This improves the scalability of the system to a great extent.

In this environment, the load balancer must have knowledge about the individual tenants as well as applications in order to distribute load efficiently. This is the main scope of Gajaba framework.

III. Load Balancing with Gajaba Framework

Gajaba is a software based load balancing framework written in Java programming language. The main goal of Gajaba project is to be able to customize the framework easily to achieve a large variety of load balancing needs. Rather than implementing every existing load balancing feature, Gajaba provides an easy to learn scripting language the users can use to implement their own needs. This is different from existing load balancing products and how Gajaba achieves this flexibility is purely based on the frameworks unique architecture and the domain specific scripting language

A. Abstract Architecture

Gajaba framework can be logically divided into three main components. They are the server, client and the distributed data storage. The server component contains the main load

balancing mechanism and message routing mechanism. It is the public access point of the overall distributed system. The client component sits on top of each worker node/server. It provides third party applications running on each node/server an interface to communicate with the Gajaba framework. Shared by all the client components and the server component, there is a distributed data storage which is used to by clients to communicate with the server component.

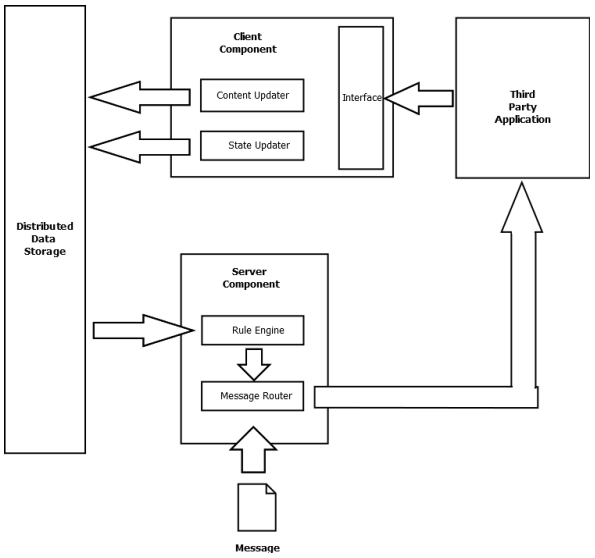

Fig. 1. Abstract Architecture of Gajaba.

Here the third party application can be any process running on the node. It can be either a cloud application or a web service. All it needs to do is use the client interface and publish the data needed by the load balancer to make its routing decisions. To interpret these data values, the user must write rules using the Gajaba scripting language. They are written once and used on the data published dynamically by the client components to make load balancing decisions. Even though these rules do not change over time, the behavior of the load balancer can dynamically change as the data values published by client components change. Which data are needed by these rules depends on the applications running and the nature of rules written by the user. Data items can be anything as long as they are published as key, value pairs as the load balancer expects the data in this format.

With this methodology the users can define the data that their custom application should publish and how to interpret them to achieve their load balancing requirements, giving them a greater flexibility. Using this method, users can implement simple load balancing algorithms such as round robin to advanced content-aware message routing algorithms, making Gajaba framework applicable to a wide range of distributed systems.

B. Rule Engine

The highly customizable nature of Gajaba framework is achieved by the Rule Engine module. This module is designed not only to evaluate rules written in Gajaba scripting language, but also to optimize them for faster execution. The rule evaluation mechanics are bound to the grammar rules of the scripting language. The optimization of rule execution is achieved by converting each rule in to a Java class using source-to-source compilation and compiling these classes into Java bytecode using the system's Java compiler before execution. This way, the rules will be executed in the speed of Java bytecode making use of the optimization techniques employed by the Java Virtual Machine.

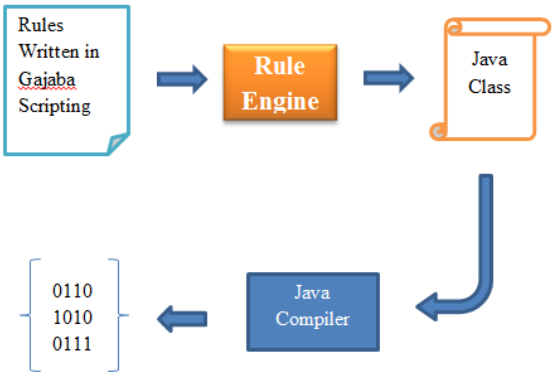

Fig. 2. Gajaba Rule Compilation Process.

C. Gajaba Scripting Language

Gajaba Scripting Language is the domain specific language (DSL) used by the framework to define rules. A Gajaba rule written using this language consists of three parts, a left variable, an operator, and a right variable. All the rules must be written in this format.

Ex: *@ip=#"acceptingIP"*

Left variable: *@ip*
Operator: '='
Right variable: *#"acceptingIP"*

1) Variables

Variables in Gajaba Scripting Language (in both left and right side) can be either a string constant or a field in the message request. A string constant is a series of characters enclosed within single quotes. For instance, in the above example "*acceptingIP*" is a string constant. A string constant in Gajaba Scripting language is exactly similar to a String literal used in Java language (except for single quotes rather than double quotes). It can contain any combination of characters (with the escape character '\' for escaping special characters like quotation marks). Even if these strings are constants, they can be mapped to other values by using the '#' symbol before them. (This will be explained in detail later)

A field in the message request is an identifier that has one or more alphabetic characters proceeded by an '@' symbol. These variables can directly be mapped to the values of the incoming message request. For instance, in the above example *"@ip*" is mapped to the IP address of the sending node.

2) Operators

The current version supports the following operators

- '=' operator

This is the "equals" operator. It must be specially noted that this is not an assignment operator as in many programming languages like C and Java.

- '#' symbol

Even if this symbol is not specified as an operator in Gajaba Scripting Language grammar, this symbol does a special operation on the string constant right to it. It will replace the string constant with a value from the distributed cache that is mapped with a key similar to the string.

Ex: #"*acceptingIP*" in the above example will be mapped to a value in the distributed cache which has a key value equals to "*acceptingIP*".

Key	Value
acceptingIP	175.157.251.176

In the distributed cache, this mapping value is 175.157.251.176. Therefore this value will be mapped. According to this example rule, the message will be routed to all the clients that have published the above key "acceptingIP" with a value of their server IP addresses.

- '[ ]' operator

This is the Regex operator. Whichever expression written within these square brackets, the Rule Engine will try to evaluate them as a standard regular expression

IV. Implementation

Gajaba framework is divided into several key modules. These modules are separated based on their functionality and how each should contribute to the overall framework.

A. Gajaba Rule Module

This module contains the rule engine. It provides two main functionalities. The first is to translate rules written in Gajaba Scripting Language into raw Java source code. The second is to acquire the translated Java code and compile them using the JVM into bytecode for better performance. The main focus of this module was to implement the rule engine without making it a performance bottleneck. Evaluating the rules written in Gajaba Scripting Language as-is is much slower than making them more compact and convert them into an optimized intermediate representation. Therefore the module was designed to compile the rules into working set of Java classes (source-to-source compilation) so that the JVM can in turn compile those classes into Java bytecode. These Java bytecode bundles are compiled once and used many times so that the overhead of doing this type of compilation is quite less, making it a high performance boost.

Gajaba scripting language grammar was defined and its parser was generated using Antlr3 [12] parser generator. Using this generated parser, an abstract syntax tree (AST) is generated for each rule and using this AST the relevant Java code is generated. The whole scripting language integration is implemented as a JSR223 [13] compliant scripting module.

B. Gajaba Server Module

This module can be considered as the heart of the framework. It has two subcomponents, the Proxy which handles the actual message routing and the Server which manages other modules such as Group Module and updates the distributed cache based on the input of the Rule Module. The Proxy component uses the advanced functionalities such as "AsynchronousSocketChannels" of NIO2 API [14], released with Java SE 7.

The main focus was given to make the I/O operations as fast as possible. Another focus was to make the module scalable by allowing it to handle many threads without adding a significant overhead to the operating system. These ideas led to the use of asynchronous I/O mechanisms as opposed to synchronous I/O. The way it is handled in Gajaba framework was by using the new asynchronous channel I/O API defined in Java 7, known as NIO2 API.

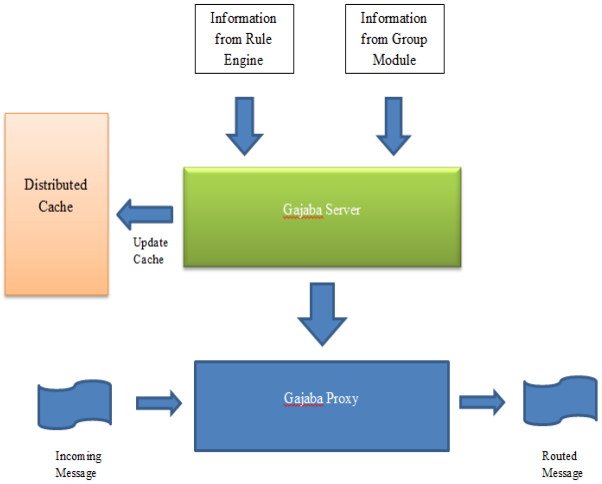

Fig. 3. Gajaba Server Module.

C. Gajaba Group Module

This module handles the group management tasks and cluster management within the framework. It uses a peer-to-peer network protocol to communicate and manages each client in groups specified by the client itself. It also contains the shared distributed cache implementation where clients can update to publish key value pairs.

The main focus given for this module was to achieve the group management functionality with dependencies to a minimum number of third party libraries. The reason for this is to make the module light-weight and get better performance by limiting the communication between sub-component libraries. Therefore Gajaba Group Module uses Shoal Clustering Framework [15], the clustering component used in the Glassfish Project [16]. Shoal uses JXTA [17] peer-to-peer networking protocol and contains a distributed data storage shared among the group members.

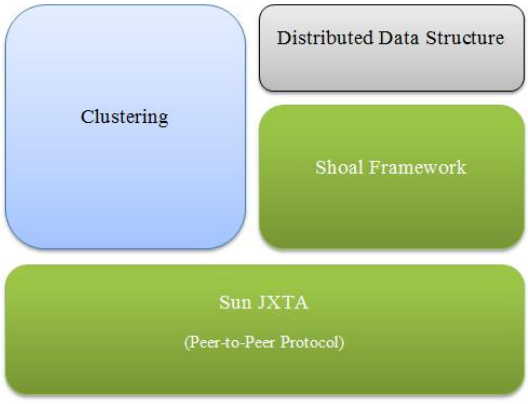

Fig. 4. Gajaba Group Module.

This module keeps track of each member joined and sends out notifications to each other. It can also respond to member failures and rejoins so the users can take into account this information when they write rules to respond to these types of events.

D. Gajaba Agent Module

This module is the agent library that operates within the client server. It publishes client level information to the load balancer via the distributed cache.

The main purpose of this module is to sit on each client and publish key, value pairs describing the clients various details to the load balancer. The mechanisms and desired functionality to support this is already implemented in other modules, mainly in the Group Module. What the Agent Module really does is that use the functionality via the API provided with other modules and publishes information to the system.

It is up to the user to define the services and other contents running on their server using the Gajaba Scripting Language and publish them using the Agent's *publish()* method. This gives the user much flexibility to apply the load balancing to various types of contents.

E. Gajaba Simulator Module

This module represents the framework in a graphical form. It consists of a web based system visualizer and a simulator. It displays each client's information, contents of the distributed cache and message logs. The simulator can be used to pump dummy messages and observe the framework in action.

This module uses the Jetty HTTP Server [18] to run the web based visualizer and the data visualization is rendered using D3 [19] JavaScript library.

The Simulator Module's GUI displays the content of the Distributed Shared Cache in a table format and the server organization in a tree structure. The main functionality of this module is that it can pump dummy messages into the system and the GUI will display the updated context of the system.

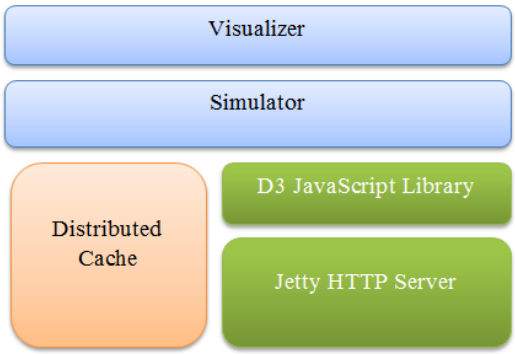

Fig. 5. Gajaba Simulator Module.

V. Conclusion

This paper introduces a dynamic, rule based load balancing framework suited for a wide variety of environments including cloud computing environments. The product can be easily configured and added to most of the existing systems. The project uses the latest technologies for implementing framework. These technologies provide more performance gain to the framework making it competitive among the existing load balancers.

Gajaba Scripting Language which is designed to facilitate to the load balancer, is a simple, yet powerful domain-specific language. Gajaba Rule Engine provides high performance rule evaluation which results in an efficient load balancer.

The load balancing decisions are based on variety of factors. It can collect information from simple server level to application level, service level and cluster level providing the capability of working as a simple load balancer to a complex content-aware load balancer.

Therefore it could be concluded that Gajaba is a combination of latest load balancing features for efficient routing in most environments, including Cloud based and Service Oriented Architectures.

Acknowledgment

The authors of this paper would like to thank Dr. Malaka Walpola for coordinating this module as well as providing guidance as the project supervisor. Special thanks go out to Mr. Afkham Azeez for providing this wonderful idea and for the immense guidance. The authors would also like to thank the staff and the support staff members of Department of Computer Science and Engineering, University of Moratuwa and all the fellow batch mates for their support.

References

[1] C. Kopparapu, *Load Balancing Servers, Firewalls, and Caches*, Wiley, Feb. 2002.
[2] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic load balancing on web-server systems," *IEEE Internet Computing*, vol. 3, pp. 28--39, June 1999.
[3] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*, Upper Saddle River, NJ , USA : Prentice Hall PTR , 2005.
[4] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service," *Computer*, vol. 36, pp. 38- 44, Oct. 2003.
[5] J. Rosenberg and A. Mateos, *The Cloud at Your Service*, Manning Publications, Pap/Psc ed., Nov. 2010.
[6] WSO2 Lean Enterprise Middleware. [Online] Available at: http://wso2.com.
[7] Apache Axis2 - Apache Axis2/Java - Next Generation Web Services. [Online]. Available http://axis.apache.org/axis2/java/core
[8] Apache. [Online] Available: http://axis.apache.org/axis2/java/rampart.
[9] Apache Synapse - The Lightweight ESB. [Online]. Available: http://synapse.apache.org
[10] aFleX Advanced Scripting for Layer 4-7 Deep-Packet Inspection (DPI). [Online]. Available: http://www.a10networks.com/products/axseries-aflex_advanced_scripting.php
[11] Coyote Point Equalizer E250GX | Application Traffic. [Online] Available: http://www.coyotepointworks.com/E250GX.asp
[12] ANTLR 3 Wiki Home - ANTLR 3 - ANTLR Project. [Online] Available: http://www.antlr.org/wiki/display/ANTLR3/ANTLR+3+Wiki+Home
[13] Scripting for the Java Platform. [Online]. Available: http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/
[14] File I/O (Featuring NIO.2) (The Java™ Tutorials > Essential Classes > Basic I/O. [Online]. Available: http://docs.oracle.com/javase/tutorial/essential/io/fileio.html
[15] Project Shoal - A Dynamic Java Clustering. [Online]. Available: http://shoal.java.net/
[16] GlassFish - Open Source Application. [Online]. Available: http://glassfish.java.net/
[17] JXTA™ - The Language and Platform Independent Protocol for P2P Networking. [Online]. Available: http://jxta.kenai.com/
[18] Jetty WebServer. [Online]. Available: http://jetty.codehaus.org/jetty/
[19] D3.js - Data-Driven Documents. [Online]. Available: http://d3js.org/

Y. Pandithawattha is 25 years old, he is now a student at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

M. Miniruwan is 25 years old, he is now a student at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

K. Perera is 25 years old, he is now a student at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

M. Walpola is 28 years old, he is now a student at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

M. Perera is 25 years old, he is now a student at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

A. Azeez is 25 years old, he is a director, Architecture, WSO2 Inc, Mountain View, CA, USA.