A Minor Project Report on

Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal

under the guidance of

Dr. Savita Sri

Submitted by

Shreyas Shankar 16IT138

Dashan Jot Singh 16IT216

Dhvanil Parikh 16IT217

VII Sem B. Tech (IT)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

November 2019

Certificate

This is to certify that the project entitled ”Inter Cloud Elastic Load Balancing and AutoScaling for increasing traffic with Service Level Agreements” is a bonafide work carried out by the team, Shreyas Shankar, Dashan Jot Singh and Dhvanil Parikh, students of final year B.Tech (I.T), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the VII Semester of the academic year 2019 – 2020. It is submitted to the Department in partial fulfillment of the requirements of the course IT418, Cloud Computing for the award of the degree of Bachelor of Technology in Information Technology. I certify that the team has carried out the work in their own capacity and have successfully completed the work assigned to them.

Place: Surathkal
Date:

(Signature of the Guide/Mentor)

Declaration

We, hereby declare that the project entitled "Inter Cloud Elastic Load Balancing and AutoScaling for increasing traffic with Service Level Agreements" was carried out by us during VII Sem of the academic year 2019-2020. We declare that this is out original work and has been completed successfully according to the direction of our guide, Savita Sri as per the specifications of NITK Surathkal. We also confirm that, the report is only prepared for our academic requirement not for any other purpose. The information incorporated in this project is true and original to the best of our knowledge.

Shreyas Shankar, 16IT138
Dashan Jot Singh, 16IT216
Dhvanil Parikh, 16IT217

Acknowledgement

We would like to express gratitude towards Ms. Savita Sri for her kind cooperation and encouragement which helped in the completion of this project entitled ”Inter Cloud Elastic Load Balancing and AutoScaling for increasing traffic with Service Level Agreements” submitted to Department of Information Technology, NITK. We would like to thank her for her guidance and constant supervision as well as for providing necessary information regarding the project and also for the support in completing the project.

We would also like to thank Mr. Sanket Salvi, for guiding us throughout the project for every little thing.

Abstract

This project aims at the Inter-cloud elasticity framework, which focuses on cloud load balancing based on dynamic virtual machine re-configuration when variations on load or on user requests volume are observed. The main objective of the project is to create a load-balancer which can accommodate both increasing traffic and increasing resource utilisation as well. The work includes an inter-cloud load-balancer for distributing incoming user HTTP traffic across multiple instances of inter-cloud applications and services. It also consists of inter cloud mediation and monitoring of resource usage. Vertical autoscaling of the resources is also proposed which automatically scales up the resources of the cloud as per incoming traffic and resource availability. Basic Service Level Agreement is also ensured so that a minimum level of service is maintained.

Contents

List of Figures

1 Introduction

Cloud computing presents new business opportunities as an environment to implement applications and services. Fundamentally, it provides an elastic infrastructure to use virtual servers that are available at any time from anywhere. It includes three models, namely, as Infrastructure as a service (IaaS) that includes virtualized resources, Platform as Service (PaaS) that includes an environment for development applications and services and software as a service (SaaS) That includes software on demand and pay per use. Too, includes platforms that offer resources such as hardware (CPU, memory, hard disk), software and custom network conduct. Today, the elasticity of the cloud seems to be a vital cloud. active as it allows users to increase or decrease the ability to virtualized resources on demand, so pay only for what use.

Cloud services also refer to cloud enablers that are part of a generic service-oriented architecture (SOA), in which providers develop applications or services by selecting functionalities from different cloud platforms (i.e. authentication, data storage, data analysis, etc.). There are many providers that offer these services and pretend to offer an ”infinite” vision of virtual resources This refers to scalability as the ability to system to accommodate larger loads and elasticity such as ability to scale with loads dynamically. Cloud platform offer both, however, do not allow automatic VM reconfiguration (which refers to increasing or decreasing VM resources) according to use in real time. Consequently, how to use virtual machine resources in the cloud the requirements change dynamically, the initial configuration could lead to degradation of service performance, for example if demand increases significantly

2 Literature Review

2.1 Background

In ”An adaptive hybrid elasticity controller for cloud infrastructures” the authors present a solution of horizontal elasticity by allowing the assignment of VM (adding or removing VM) in the cloud. They present a simulation analysis that demonstrates a minimization of the SLA violation, focusing on horizontal elasticity. In ”Agile dynamic provisioning of multi-tier internet applications” the authors monitor VM resources and develop an architecture to reduce the provision overhead.

According to ”Advancing research infrastructure using openstack”, the approach used does not offer a scale reduction mechanisms. The authors focus on the use of a private cloud environment to improve high-performance computing (HPC) research infrastructure. Specifically, they have implemented an HPC job scheduler to improve the utilization of resources in the cloud. In ”Automated control for sla-aware elastic clouds” the work focuses on the need for integration of QoS and SLA requirements with the cloud and automated dynamic elasticity of the cloud for SLA management and it is a theoretical discussion of research directions.

The play presented in ”Automatic elasticity in openstack” focuses on elasticity as the capacity of a cloud to add and remove instances automatically. The solution is called Elastack, which is a monitoring and adaptive system. The authors state that the solution is generic and could be applied to existing IaaS frameworks.

In ”Dynamically scaling applications in the cloud” we can see a study on the dynamic scaling of cloud applications. The work shows efforts to the limit of cutting edge technology in the elasticity of the cloud. In ”Cloudscale: Elastic resource scaling for multi-tenant cloud systems” the authors demonstrate an automated system for elastic scaling of multi-user cloud resources called CloudScale.It achieves adaptive resource allocation no need to know a priority.

The authors in ”Kaleidoscope: Cloud micro-elasticity via vm state coloring” present a system that allows VM instance cloning when demand is increased when copying the complete or partial status of the original instance. Kaleidoscope does not launch new instances. In ”A cost-aware elasticity provisioning system for the cloud” the authors present an architecture for an IaaS cloud to allow dynamic allocation of resources. They develop a system called Kingfisher that contains components for replication and migration using an entire linear program to optimize the cost and implement an OpenNebula extension to balance the load when the load is changing.

2.2 Identified Gaps

Here the papers referenced involves using of different techniques to encounter problems like vertical and horizontal scaling, service-level agreement, dispatching of the incoming request. However as a whole one paper did not resolve all this conflicts that involves trade off between performance and availability, focusing on migration or resizing of cloud VMs.

2.3 Problem Statement

Design an elastic framework for efficiently dispatching the increasing number of HTTP requests with appropriate service level agreements for a service distributed on multiple cloud platforms.

2.4 Objectives

- To deploy a service in 3 different clouds
- To design a load balancer which distributes the load among different available instances of all the clouds hosting the service
- To implement different dispatching algorithms for the inter cloud load balancer
- To develop an interface which provides real time monitoring and mediation of all the clouds
- To implement an vertical autoscaling algorithm which automatically scales up the resources of the cloud as per incoming traffic and resource availability
- To employ a SLA to ensure a minimum level of service is maintained

3 Methodology

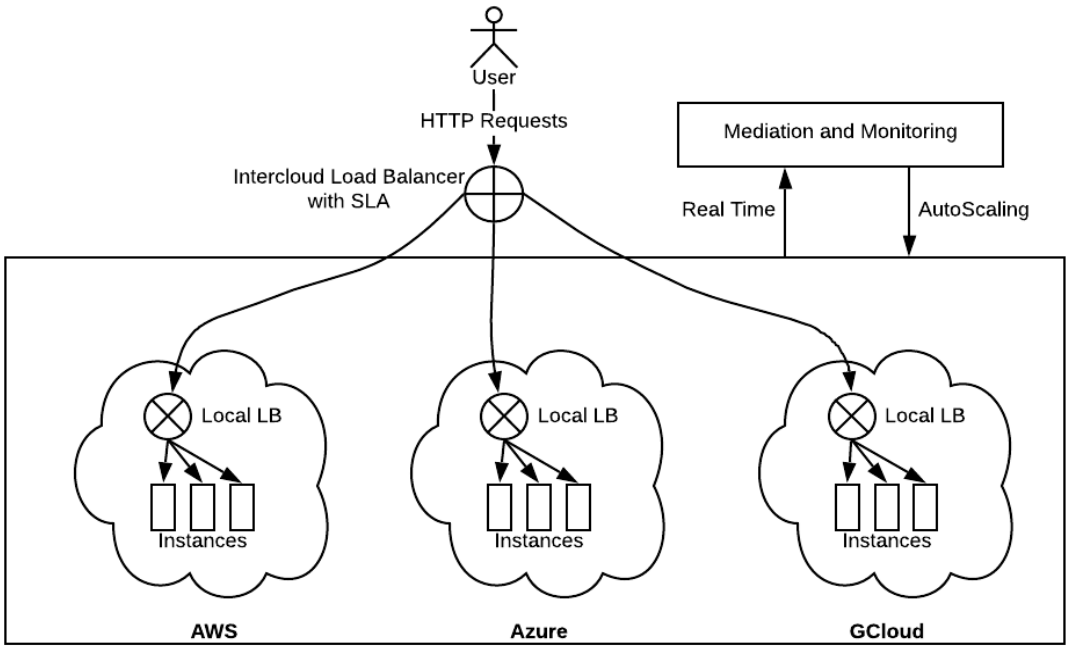

Figure 1: System Architecture

The Inter Cloud Load Balancer framework, demonstrated in Figure 1, targets the automation of the inter-cloud elasticity and performs dynamic VM reconfiguration based on the variation of HTTP traffic.

The **Intercloud Load Balancer** is configured by the ICLB components in order to allow traffic distribution in the deployed App over 3 different clouds. At this point the assumption is that the owner instantiates 2 to 3 VMs that host the App in the same or different clouds. During initialization, the owner configures any local load balancers of the local providers by creating, if needed, replicated instances of the local services. The owner configures the IC load balancer to distribute traffic among service endpoints based on HTTP load balancer algorithms such as round-robin or least-connected. The ICLB also employs a SLA to ensure a minimum level of service is maintained.

Using ICLB, the service owner configures the monitoring aspects of the App/service. In particular, the **Mediation and Monitoring** module allows live data capturing based on an interval (i.e data collection every 5 seconds). The Mediation and Monitoring component requires installation and configuration within the VMs and will offer a centralized control point for storing data including configuration and performance data. The real-time time monitoring will allow vertical autoscaling of the VMs on the clouds as per incoming traffic and resource availability.

4 Implementation

4.1 Work Done

The software was developed in PyCharm Idea Commercial Version which is a Python integrated development environment for developing computer software. Also python virtual environment along with PIP was used which made package management simple. Django, a Python framework is used to implement the app/service using the architecture mentioned in the methodology section.
The software consisted of the following features:

1. Developer can add new clouds and configure their resources that can be used by the system
2. User can select the type of dispatcher according to which his request will be handled
3. User can select the type of request from a pool of requests to be served
4. User can enter the time by which he needs the request to be completed as a Service Level Agreement
5. Developer can monitor the usage of resources across all clouds in real time, updated every 5 seconds
6. The server takes care of vertical autoscaling of the VMs on the clouds as per incoming traffic and resource availability by the help of real-time time monitoring

There are two dispatchers used which employ the following algorithms:

- Round Robin : This algorithm goes down the list of servers in the group, the round-robin load balancer forwards a client request to each server in turn. When it reaches the end of the list, the load balancer loops back and goes down the list.
- Least Connections: This load balancing algorithm uses the number of current connections to each servef in the group(member) to make its load balancing decision. The member with the least number of active connections is chosen.

A service-level agreement (SLA) was implemented where the user who sends the request can specify the time in which he wants the request to be completed. This request is handled by both the types of dispatchers namely round-robin and least-connections. Once the server to handle the request is selected, the existing process is then preempted for the new request so that the request can be completed in the given time by the user. The entire software that was developed was hosted on an amazon-ec2 instance that made it accessible to anyone.

4.2 Result and Analysis

The topology of the App/services is similar to Figure 1 where users send HTTP requests to the Intercloud Load Balancer that in turn it forwards each into the clone VMs, present in various clouds (here, ICLB is acting as a proxy).

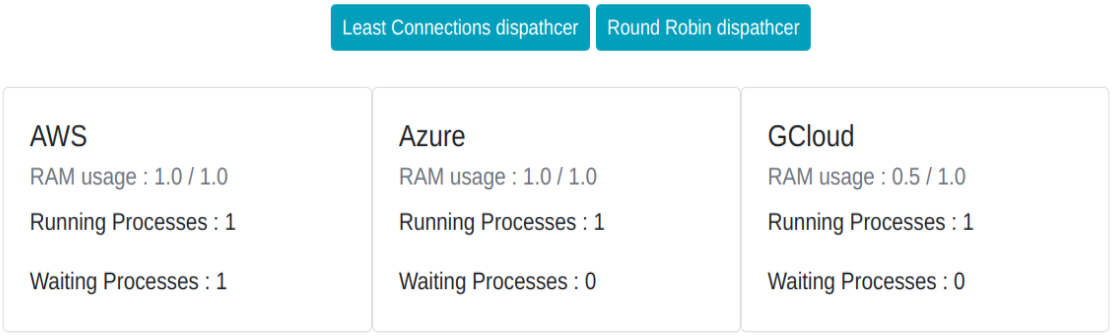

Figure 2: User Interface

Figure 2 depicts the main user interface of the designed framework. It demonstrates real time monitoring and cloud mediation of the resource usages of the different clouds among which the service is distributed. It also allows the requesting user to choose the dispatching algorithm amongst Least Connections and Round Robin. Upon clicking either of the buttons, the user is redirected to another page as shown in Figure 3.

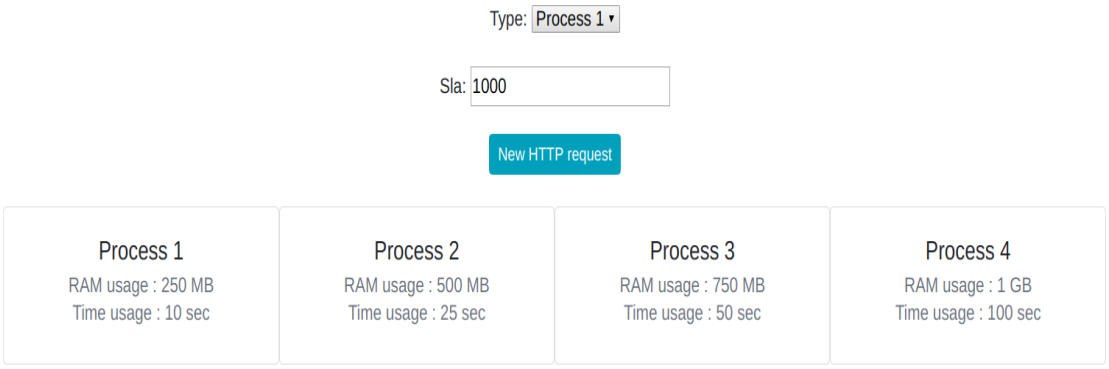

Figure 3: HTTP request selection

Figure 3 displays the interface where the user can select the type of process to generate an HTTP request that is dispatched to the appropriate cloud using algorithm selected earlier. The user sending the request can also specify the time in which he wants the request to be completed to ensure a minimum level of service is maintained.

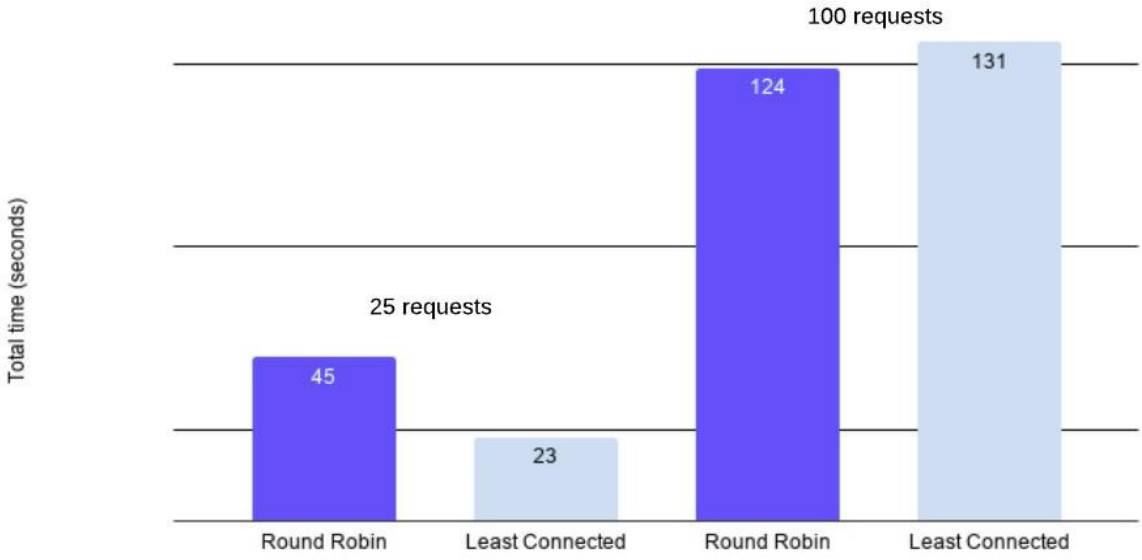

Figure 4: Dispatch Comparison

To compare performance, we present two scenarios in which (a) a traffic of 25 requests is forwarded to the ICLB that executes all HTTP requests and (b) the traffic is distributed among two identical VMs. Figure 2 demonstrates the results of the execution of both scenarios where the numbers demonstrate the total time(seconds) taken to serve the requests. As seen in the figure, least connected dispatching algorithm fairs well when the number of requests is low but the round robin dispatching algorithm triumphs when the number of requests increases.

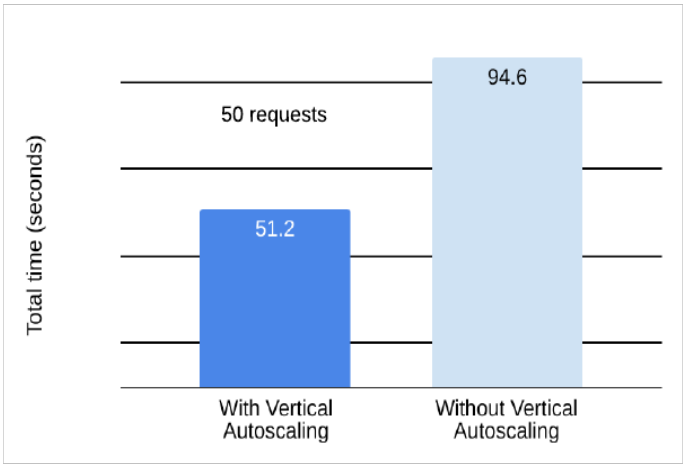

Figure 5: Autoscaling Comparison

Figure 3 demonstrates comparison between total time taken for completion of requests for cases of vertical and non-vertical autoscaling.

4.3 Innovation Work

A service-level agreement (SLA) is a contract between a service provider and its internal or external customers that documents what services the provider will furnish and defines the service standards the provider is obligated to meet. Here we have implemented a service-level agreement (SLA) where the user who sends the request can specify the time in which he wants the request to be completed.

This request is handled by both the types of dispatchers that are being used namely least-connections and round-robin algorithms. The existing process with a higher SLA expiry time is preempted so that the request can be completed in the given time.

4.4 Individual Contribution

5 Conclusion and Future Work

The Inter Cloud Load Balancer, a modular framework that allows load balancing of inter-cloud applications and services that belong to heterogeneous providers was implemented and deployed on a cloud patform. We aimed to improve the elasticity in the IaaS level through auto-scaling of cloud and inter-cloud which was successfully achieved. We have implemented different algorithms for dispatching and also successfully demonstrated that this did not result in any loss of communication or failures.

This dispatching algorithm can be further improved to smartly select the appropriate cloud instance for serving the request. This can be achieved by collecting sufficient statistical information and then selecting the cloud instance with the help of a mathematical model. Also if the statistical data collected is large enough, machine learning models can be used. This models can be trained against the collected data to achieve a better performance from the overall architecture implemented. Also various other Service Level Agreements can be implemented to ensure a minimum level of service is maintained.

References

[1] An adaptive hybrid elasticity controller for cloud infrastructures: https://people.cs.umu.se/ahmeda/06211900.pdf

[2] Agile dynamic provisioning of multi-tier internet applications: http://citeseerx.ist.psu.edu/viewdoc/1.1.pdf

[3] Scaling for multi-tenant cloud systems: https://dl.acm.org/citation.cfm?id=2038921

[4] A cost-aware elasticity provisioning system for the cloud: https://dl.acm.org/citation.cfm?id=2014786

[5] Amazon auto scaling documentation: https://docs.aws.amazon.com/autoscaling/index.html