Cloud Hopper: A Unified Cloud Solution to Manage Heterogeneous Clouds

by

SHRADDHA JAIN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2016

To my family.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt thanks to Mr. David Levine, my supervisor who guided and motivated me throughout my thesis. I got a lot of exposure to the latest and exciting technologies under his mentorship. Without his invaluable support and encouragement, this would not have been possible.

My heartfelt thanks to my committee members Dr. Ramez Elmasri, and Dr. Farhad Kamangar, and the computer science department, UTA for their invaluable cooperation, and support.

I would like to extend my sincere thanks to the members of our Cloud Team, Mr. Mayank Jain and Mr. Samvaran Kashyap for being my pillars of support.

I am grateful to my family, my mother, my father, and my brother who were always there with me in my good and the bad times. Finally, I would like to thank all my friends who helped me through this journey.

November 15, 2016

ABSTRACT

Cloud Hopper: A Unified Cloud Solution to Manage Heterogeneous Clouds

SHRADDHA JAIN, MS

The University of Texas at Arlington, 2016

Supervising Professor: David Levine

Cloud environments are built on virtualization platforms which offer scalability, on-demand pricing, high performance, elasticity, easy accessibility of the resources and cost efficient services. Most of the small and large businesses use cloud computing to take advantage of these features. The usage of the cloud resources depends on the requirements of the organizations. With the advent of cloud computing, the traditional way of handling machines by the IT professionals has decreased. However, it leads to wastage of resources due to inadequate monitoring and improper management of resources. Often it happens that the cloud resources once deployed are forgotten, and they stay running until someone manually intervenes to shut them down. This results in continuous consumption of the resources and incurs costs which is known as Cloud Sprawling. Many organizations use resources provided by multiple cloud providers and maintains multiple accounts on them. The problem of cloud sprawling proliferates when there are multiple accounts on different cloud providers are not managed properly.

In this thesis, a solution to overcome the problem of cloud sprawling is presented. A unified console to monitor and manage all the resources such as compute instances, and storage deployed on multiple cloud providers is provided. This console provides the details of the resources in use and ability to manage them

without logging into the different accounts they belong to. Moreover, a provision to schedule multiple tasks is provided from the scheduling tasks panel. In this way the resources can be queued to run at a specific time and can also be torn down at a scheduled time, thus the resources are not left unattended. Before terminating, a facility to archive files, and directories on virtual machines is also provided using storage services offered by both IaaS and SaaS providers.Further, a notification system helps in notifying the user about the statuses of the scheduled tasks thus helping enterprises in saving on the costs.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

AWS – Amazon Web Services

EC2 – Elastic Cloud Compute

GCE – Google Cloud Engine

GCP – Google Cloud Platform

IaaS – Infrastructure as a Service

MVT – Model View Template

PaaS – Platform as a Service

S3 – Simple Storage Service

SaaS – Software as a Service

SMTP – Simple Mail Transfer Protocol

VM – Virtual Machine

CHAPTER 1

INTRODUCTION

1.1 Introduction and background

Cloud Computing enables user to access services such as compute, storage, database, networks, and applications over the Internet. It is a model for storing and accessing data and programs over the Internet instead of doing it on a local computer's hard drive. Small and large enterprises, as well as institutions of higher learning, are embracing this technology due to economic advantage, improved manageability, less maintenance, high performance, accessibility, availability, scalability, pay-per-use model, and low carbon footprint thus bringing about a smooth facilitation of organizational operations [1].

The Cloud model comprised of three primary service models as mentioned below:

Infrastructure-as-a-Service (IaaS): Infrastructure as a service provides companies with computing resources including servers, networking, storage, and data center space on a pay-per-use basis [2].

Platform-as-a-Service (PaaS): Platform as a service provides a cloud-based environment with everything required to support the complete lifecycle of building and delivering applications without the cost and complexity of buying and managing the underlying hardware, software, provisioning, and hosting [2].

Software-as-a-Service (SaaS): SaaS allows the use of applications that run on the cloud infrastructure and is accessible from various client devices. Cloud providers install and

operate application software in the cloud and cloud users access the software from cloud clients. Examples of SaaS offerings are Microsoft Office 365 [3], and Dropbox [4].

The Cloud model is also composed of the following main deployment models:

Private cloud: It is a cloud infrastructure that is operated only for an organization. The organization can manage it internally or by a third-party and can be hosted either internally or externally [5].

Public Cloud: Public cloud infrastructures are owned and operated by companies that offer affordable computing resources [2]. Users do not have to purchase hardware, software, or supporting infrastructure rather they will use the third-party's resources as per the demand on a pay-for-use basis. The leading public cloud providers are Amazon Web Services, Google Cloud Platform, and Microsoft Azure [6].

Hybrid cloud: It is a composition of two or more clouds (private, or public) from different service providers that remain distinct entities but are bound together. The companies with hybrid clouds manage their workloads across data centers, private clouds, and public clouds [5].

There are a few other deployment models such as Community Cloud that shares infrastructure between several organizations from a specific community with common concerns, a distributed cloud platform created by assembling distributed set of machines connected to a single network in different locations. An Intercloud in which interoperability happens between public cloud service providers and a multi-cloud model which can be defined as the use of multiple cloud computing services in a single heterogeneous architecture to reduce reliance on single vendors, increase flexibility through choice, and mitigate against disasters [8].

This thesis focuses on the use of multi-Cloud approach in a web application framework, which can be used for management of resources across the different clouds from a single console. A modular approach is used to design the framework in such a way that the services of any cloud provider can be easily plugged into the framework and used. The motive of providing a unified console to all the service providers is to provide an easy to use management console which will help in managing the costs of services by the enterprises using the multi-cloud framework. A common key store is designed to manage the access keys of multiple cloud providers securely. Further, an ability to set lifetime to the resources and a multi-cloud archiving provision along with the monitoring features is provided to manage the resources even in a better way. An analysis of pricing on different cloud services offered by several cloud providers is done. Also, a detailed performance analysis of various cloud operations is presented in the thesis.

1.1.1 Multi-Cloud

Multi-Cloud or Heterogeneous Cloud is a mix of IaaS (Infrastructure as a Service), SaaS (Software as a Service) and PaaS (Platform as a Service) offerings. It is a combination of many best-of-breed cloud services which forms the optimal solution. Not every service provided by a cloud provider is perfect. So, these days companies want to take benefit of using the best services provided by different cloud providers. For instance, there may be a scenario where a company might be running workloads that need a large amount of storage and networking on a private cloud such as OpenStack [7]. On the other hand, it might also have an enormous workload that requires powerful

instances to run applications which can be scaled up or down depending on the demands. In this case, public clouds such as AWS, IBM Bluemix or Google Cloud Platform can be used to fulfill the requirements.

The main advantages of opting for a multi-cloud ecosystem are:

- Better Performance: Since in a multi-cloud environment an organization chooses the best of services across the various cloud providers, we get better performance.
- Cost Efficiency: The services can also be chosen based on a company's budget on different clouds as per the best prices offered by them. In this way, we can break down our requirements and can find the best cloud provider for a service with the affordable prices accordingly.
- Lower Risks: The data can be replicated on multiple clouds so that it reduces the risk of downtime or data loss due to failure in some hardware, software, or infrastructure. For example, On August 7, 2011, Amazon experienced an outage at its cloud computing hub located in Dublin, Ireland, apparently caused by an electrical transformer malfunction [9].

On February 29, 2012, Microsoft's Azure cloud management system experienced an outage that adversely affected users in parts of the United States and Europe for several hours [10]. The loss from these outages could have been handled by using multi-cloud approach. In this way, we can mitigate against disasters.

- Reducing reliance on any single vendor: Since apps and workloads work better in particular type of cloud whether it is public, private or hybrid, so using multiple clouds to handle them can help in reducing the reliance on a single vendor.
- Many choices: A user or a company has a plethora of services to choose from.

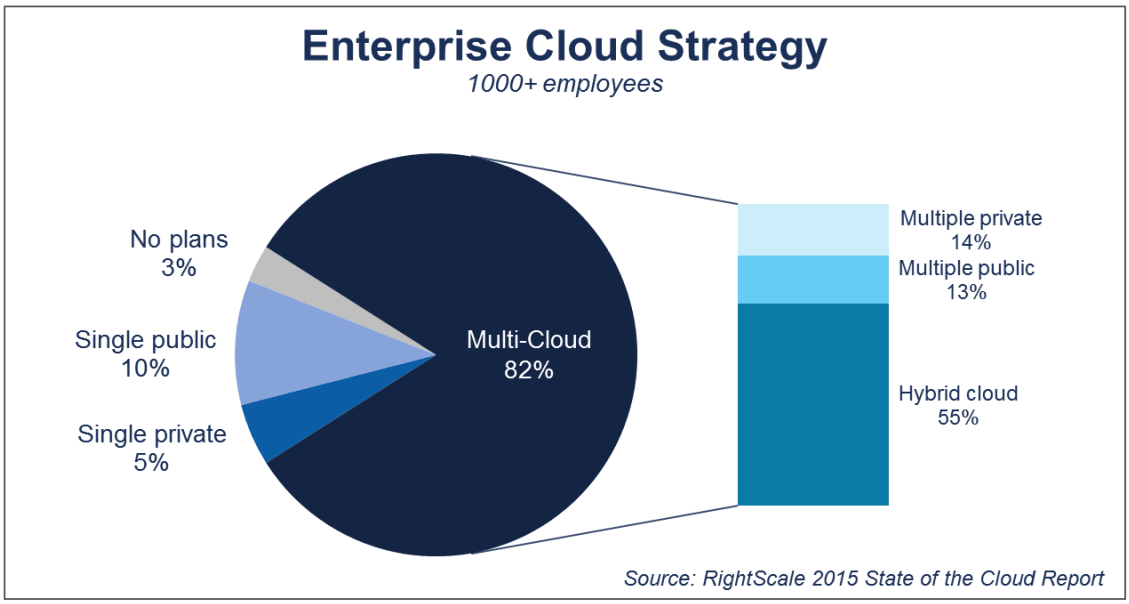

Figure 1.1 Enterprise Cloud Strategy [11]

Since multi-cloud provides lots of business benefits, many enterprises are moving towards multi-cloud approach. A survey conducted by RightScale (Fig. 1.1) on more than 1000 employees stated that 82% companies are opting for the multi-cloud approach. Among them, 14% are opting for multiple private clouds, 13% are going for multiple public clouds while rest of the 55% are choosing hybrid cloud framework.

Another survey conducted by Equinix on 650 IT decision makers across the globe shows that almost 77% of IT decision makers are willing to implement multi-cloud architectures. Almost 91% are planning to deploy cloud-based services in the next 12 months. Out of them, 48% reported that they would deploy around 6-10+ cloud services during that period. The topmost applications that these enterprises are planning to migrate to multiple clouds are storage/backup, disaster recovery, business intelligence, project management and content management [12].

However, managing the multiple clouds is complex, an enterprise might need a team of in-house experts for every cloud or a vendor for every cloud. Users might have to juggle between various login account on different providers and many other issues. Cloud Hopper resolves most of the problems faced while using multi-clouds.

1.2 Motivation behind the thesis

Although multi-cloud approach provides ample reasons which give an edge over single-cloud or hybrid cloud approach, management of resources has always been a challenging task to the companies opting for the multi-cloud environment. Users juggle between multiple logins of different cloud providers to access their services. Managing the access keys of those providers and storing them securely is another overhead. When the compute services offered by cloud service providers such as AWS, Google Cloud Platform, and Microsoft Azure are over-provisioned and are not managed properly that incurs a cost to the company, it leads to a problem known as cloud sprawling. Moreover, if the users want to manage the over-provisioned virtual machines and want to archive the useful documents before terminating them, there is no cost-effective, faster, and efficient way to do it. Most of the times, employees in a company are dependent on system admins to manage the resources, thus wasting a lot of time in communicating with the admin department. There is no provision where the employees with minimal learning can handle the resources themselves.

1.3 Goals of the thesis

The goal of this thesis is to provide a unified cloud solution to manage heterogeneous clouds from a single pane of glass. This will be helpful for enterprises who are using the portfolio of clouds to save the cost of resources. By leveraging the provision of scheduling and monitoring of services provided by different cloud providers, this thesis aims to provide users to do management at a low level. The research on the prices offered by the cloud providers will help in deciding on the best deployment strategy. An analysis of various storage options will help in providing an archiving feature for valuable documents in the fastest and economical way. A key-store will provide the solution to store access keys in a secure way.

1.4 Organization of the Thesis

This thesis starts with an introduction and background, motivation and goals of the thesis. Chapter 2 gives an overview of related work. Chapter 3 defines the problem statement. Chapter 4 explains the framework description which is the backbone of this thesis. Chapter 5 provides a detailed description of the implementation of the application. Chapter 6 gives a detailed description of the price analysis and case studies. Chapter 7 details the experiments, analyses, and results. This thesis is then concluded in Chapter 8 with future work outlined in Chapter 9.

CHAPTER 2

RELATED WORK

Multi-cloud can be defined as the usage of multiple cloud computing services in a single heterogeneous architecture with the motive of using the best of services from various cloud providers. In this thesis, the focus is on to manage these multi-clouds through one platform to save the money on the unmanaged resources and to provide a faster way to access services of different clouds without juggling through the multiple logins.

Previous work related to our goals and processes can be found in the literature. A similar work to schedule the cloud resources is presented. Chen et al. presents an architecture-based integrated management of diverse cloud resources. They performed an experiment on a real-world cloud that demonstrates the feasibility, effectiveness, and benefits of the new approach to integrated management of Cloud resources [14]. Geng et al. presented a heterogeneous solution heterogeneous DBMS and cloud storage system, and the heterogeneous data model for cloud computing [15]. Do et al. showed a framework for the design and understanding of the heterogeneous market cloud computing. They formulated a price competition between cloud broker and public provider [16]. Chen et al. shows system architecture of cloud-based agricultural resource scheduling management. They took an agricultural resource scheduling management for example and built the system architecture of cloud-based agricultural resource scheduling management [13]. Mist.io is an application that provides a mobile management, monitoring, and automation for servers across clouds through a touch-

friendly interface. It helps users to manage and monitor their virtual machines, across different clouds [17]. Gorilla stack is an AWS cost management tools. It provides a solution to take control over the AWS cost optimization [18]. It only works for AWS EC2 instances. Another web-based application Multcloud is used to manage, migrate, transfer, copy, and move files between any cloud storage services [19].

Most of these works are done on single cloud providers and are very different to the context of this thesis. This thesis provides a solution to the problem of cloud sprawling which incurs a cost to the organizations. These costs sometimes exceed the budgets of the organizations and hence faces losses. The resources if managed properly will help in saving an enormous amount of money to the large and small-scale businesses.

CHAPTER 3

PROBLEM STATEMENT

The problem this thesis addresses are cloud sprawling and mismanagement of resources over the multiple clouds which if left unnoticed and unmanaged they costs a hefty amount to the organizations. Often it happens that organizations uses compute instances, and storage and owns multiple accounts on multiple cloud providers. Then they have to go through multiple logins to monitor the resources that are in use, how much cost they are incurring and how much resources they are using. The more cloud providers they use and the more logins they have, the crazier it gets. Below are the issues that originate due to mismanaged services and no standard solution to handle them:

1. Too many choices

Businesses running in multi-cloud environments need to decide what all resources are required for their applications and what all providers should those resources reside. These days there are large choices of providers and the resources for users which makes it difficult to decide on them. For instance, an organization may conclude that the compute instances should be on Amazon Web Services (AWS), the storage on Google cloud storage and database on SQL Azure.

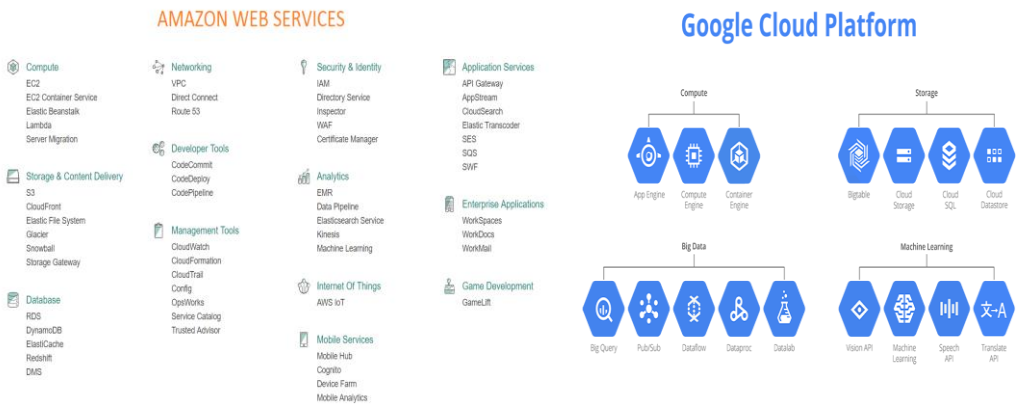

Figure 3.1 Services offered by Amazon Web Services and Google Cloud Platform

Fig. 3.1 shows the services provided by Amazon Web Services and Google Cloud Platform. Amazon Web Services provide 57 services, and Google Cloud Platform offers 23 services (as on 11/22/2016). Sometimes the user is not aware of the most relevant options to choose from, and it becomes difficult for the users to decide on which services to be used from which service provider.

2. Too much to manage

When the options of services are wide in multi-cloud environments, and users take the liberty of using multiple accounts, then the issue of over-provisioning of resources starts. Due to inaccurate estimates of the resources which would be required for an application or allocate more resources than needed so that the applications do not starve, it becomes difficult to manage the services in a cost-effective manner. Fig. 3.2 shows example of an application that is built on multi-cloud framework

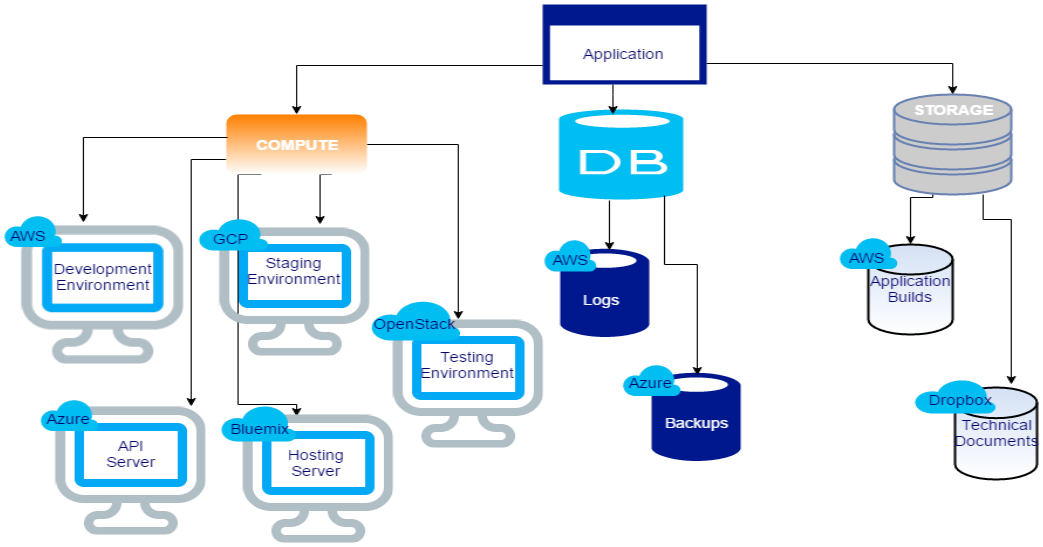

Figure 3.2 An example of a web application using services from different cloud providers

3. Too many logins

When it is multi-cloud, then undoubtedly there are multiple logins and then starts the trouble of handling multiple ‘access-keys’ and ‘tokens’. It becomes an overhead to store all the ‘keys’ to a safer place and with minimum management. Fig. 3.3 depicts the management of multiple logins for an application built on multi-cloud architecture.

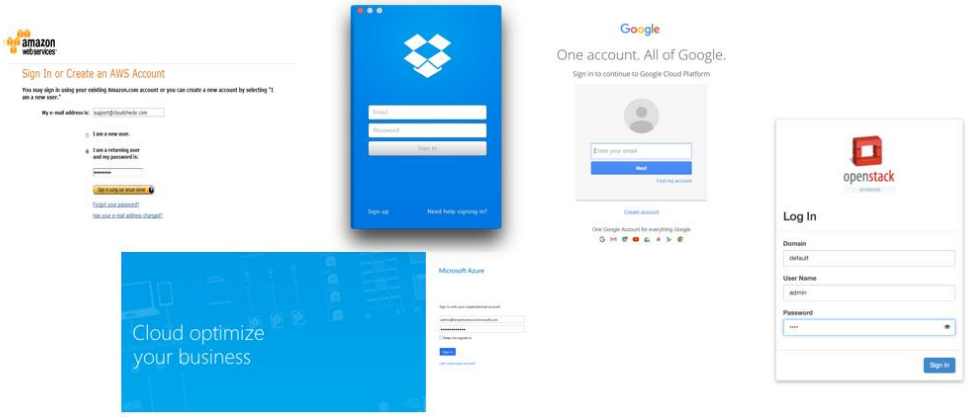

Figure 3.3 Depiction of managing multiple logins on different cloud providers

4. Cloud Sprawling

Cloud Sprawl is a situation created by over-provisioning and ever growing use of cloud services such as virtual machines, and cloud storage by the employees of a company. When each user is free to pick the cloud services of her/his choice for conducting business and managing data, the result can be unmanageable, and inherently insecure which leads to the proliferation of sharing and collaboration services [20]. This uncontrolled cloud sprawl is effecting the company budgets.

As per one of the Forbes article, more than half of U.S. Businesses now use Cloud Computing, and many of them are facing the problem of cloud sprawling [21]. A report released by Avanade, an Accenture, and Microsoft joint IT consultancy firm, shows 61 percent of companies across the globe blame cloud sprawl for causing inefficiencies in their business. That number falls to 52 per cent in the UK, but it rises to 71 percent among those using both public and private clouds. This survey involved 750

IT decision makers in United States, United Kingdom, Germany, China, France, Sweden, Brazil, Japan, and Australia [22].

5. No standard way to handle Multiple clouds

So far there has not been any standardised solution to manage multiple cloud provider offerings through a single panel.

6. No single console to handle AWS VMs in various regions

AWS EC2 console is built in such a way that at a time user can see the instances in a single region. Every time user toggles between the regions to keep an eye on their instances. There is no platform where one can see all the instances of all the regions at one single panel.

7. No monitoring for multi-cloud

There is no notification system to monitor and notify about the status and activities of the services in use by the users on multiple clouds.

8. No provision to take backups before terminating a machine

A company's system admin department might want to shut down some machines which are no more of use but contains lots of important documents and application folders which need to be moved to some safe storage and archived. So far there has

been no instance where the files and folders would have been thought to be saved on faster, reliable, and cheaper storage options such as Dropbox, Google drive, and box.

9. Dependency on IT Admins

In large and small enterprises, to use the resources from a cloud provider, team members contact the system admin department to create them and assign them to the team. This way, there is a layer of admin department between the cloud providers and the team members which increases the timelines of a project. Often the team members are not aware of how to provision the services on a cloud provider. So, there is no user-friendly provision where a team member with minimal knowledge can handle most of the admin stuff which can cut down in extra time used in projects.

CHAPTER 4

FRAMEWORK DESCRIPTION

4.1 Introduction

Cloud Hopper is a web-based application that tries to solve the existing problems mentioned in the previous chapter. It is built on Python libraries [23], Django [24] and various other plugins. The whole framework of Cloud Hopper is designed in such a manner that it will handle the heterogeneous clouds. The packages designed for Cloud Hopper are modular, and any new module, or a package can be easily integrated with minimal efforts.

4.2 Features

Following is the list of features provided by Cloud Hopper:

Common key-store – A key-store is designed to manage the access keys of multiple cloud providers in a secure manner. Before storing these credentials, they are being encrypted by AES 256 bit and 32-bit block algorithm. Further, when the credentials are retrieved from the database, they are decrypted and used.

- Ability to set lifetime to a resource – A resource can be scheduled to expire on a preset date and time which helps the user to manage the tasks efficiently. The unused resources can be removed in an automated manner which will help in reducing cloud sprawling and thus saving money of the companies.

- Multi-Cloud archiving – When the unused virtual machines are planned to schedule to terminate, the important folders and documents can be archived to the various cloud storage options which are cost efficient and the transfer is fast. Using archiving feature, one can easily move the files, and folders from any compute machine such as AWS EC2 [25] instance to either IaaS storage services such as AWS S3 [26] and GCP storage [27] or SaaS providers such as Dropbox [28] and Google Drive [29].
- Dashboard to view the usage of resources – A dashboard is designed to show the usage of different services across multiple clouds, thus giving a provision to present the statistics of the services in use on one single platform.
- Resources management console – The resource management console presents the details of the resources in use across multiple clouds thus saving the time of the user in logging into multiple consoles of the service providers. The provision of documents archiving and tasks scheduling is also given at this console.
- Interactive bot based operations - Slack Bot is an external Slack [30] interface script interacting with REST services provided by Cloud Hopper. It offers interactive operations such as listing, creation, and deletion of resources using Slack Channel interface.
- Usage alerts – Alerts are configured with an external SMTP service such as Mailgun [31] to send timely alerts to the intended user regarding the status of the scheduled tasks and the resource usage.

4.3 Architecture

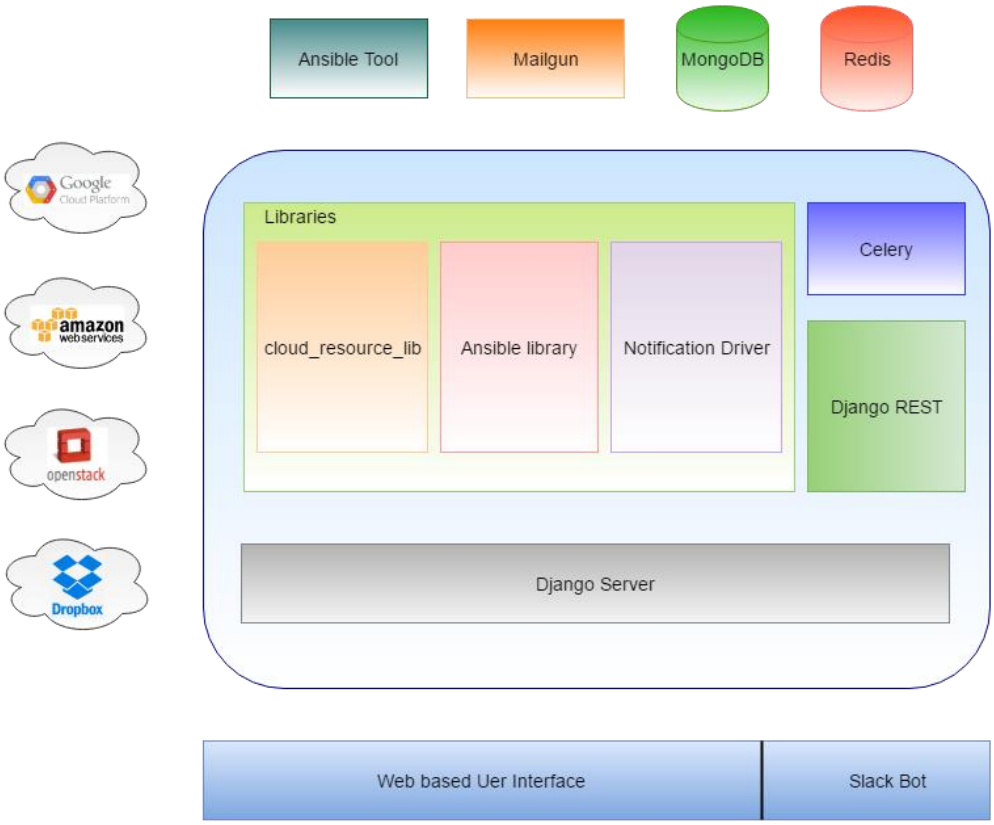

Figure 4.1 Cloud Hopper Framework

The architecture of Cloud Hopper is a three-layered Model, View, and Template (MVT). Fig. 4.1 illustrates the framework and the related functional components of cloud hopper. Cloud Hopper is comprised of the following components:

4.3.1 Web-based user interface

The web-based user interface accepts inputs from the user, makes a REST request to the Django REST API for processing and provides the output on the dynamically rendered web pages. In Cloud Hopper, jQuery AJAX calls are made to interact with the Django REST APIs. Validations are also done on the front end to avoid the incorrect inputs from the user.

4.3.2 Slack Bot

The slack bot is a user-friendly interface built on Slack APIs to interact with the REST services offered by the Django REST. Although, it provides limited operations as compared to the web-based user interface, it is very useful and requires very less learning curve. It is controlled programmatically through some scripts for specific tasks which interacts with the user inputs.

4.3.4 Django Server

Django server is the application server for Cloud Hopper. It provides APIs using Django REST to the front-end and responds to the request and delivers the content of the page back to the user.

4.3.5 Django REST

Django REST framework is a powerful and flexible toolkit for building Web APIs.

Following are the features provided by Django REST framework:

- It provides API explorer for easier management of REST calls.
- It has built in support for authentication packages such as OAuth1a and OAuth2.
- It also offers Serialization that supports both ORM and non-ORM data sources.
- Highly customizable function-based views help developers in easier design of REST API.
- It provides extensive documentation, and great community support.
- It is been used and trusted by internationally recognized companies including Mozilla, Red Hat, Heroku, and Eventbrite [45].

4.3.6 Celery

Celery is an asynchronous task/job queue that is built on message passing system and can be used for scheduling as well. By implementing celery, one can offload tasks to it and continue to execute and function responsively. In this way, the responsiveness of system can be increased and not get locked up while performing long-running tasks. Cloud Hopper uses celery for scheduling of the tasks which user want to run at a particular point in time.

It is tightly coupled with the Django REST APIs that passes the values or the parameters for the tasks or the execution units that are to be scheduled at a time and date. It can also be used to design workflows. Cloud Hopper uses chaining functionality of Celery for workflows that can be used to link tasks also referred as adding a callback task [32]. The linked task runs with the results of the parent task as its first parameter. In

the case of Cloud Hopper, a workflow can be designed with a series of prioritized tasks. Once the execution of the first task is successful, then only the next linked task runs with the output of the parent task. For instance, if a user wants to schedule termination of an EC2 instance which is no more in use which incurring a cost to the user. If it has some important files and folders stored in it which the user wants to archive and store them in his/her Dropbox account before deleting the instance. In that case, the user can design a workflow in the following manner:

Compress the required files and folder on the instance. Create a folder or use an existing location on Dropbox. Move the compressed files to Dropbox. And finally, terminate the EC2 instance. Celery Chain is used to implement all the tasks mentioned above in a series.

4.3.6.1 Celery Architecture

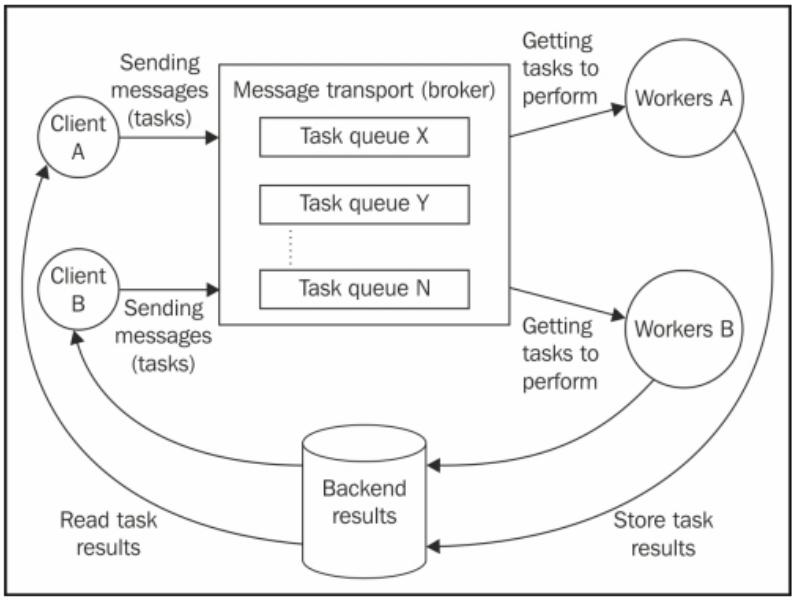

Figure 4.2 Celery Architecture [33]

Celery architecture is designed to scale and distribute tasks across several servers. The client components, as presented in the Fig. 4.2, have the function of creating and dispatching tasks to the brokers.

4.3.7 Libraries

The main libraries make the backbone of Cloud Hopper are explained below:

4.3.7.1 cloud_resource_lib

cloud_resource_lib is a major component of Cloud Hopper. It contains modules that provide access to the functionalities of various services provided by multiple cloud providers by providing an abstraction layer to all the cloud providers.

4.3.7.2 Ansible Library

Ansible library is a group of various Ansible modules. Ansible gives a provision to custom design modules which can be used to leverage the functionalities of several services [34]. Each module is a standalone and reusable script used by Ansible API and playbook programs.

For Cloud Hopper, multiple modules are written to interact with the services like Dropbox, Google Cloud Storage and Google Drive.

4.3.7.3 Notification Driver

This driver is used to send the usage alerts to the users through emails. The tasks statuses are emailed to the users promptly so that users can take appropriate actions.

4.3.8 Redis

Redis [35] is being used as a message broker by celery and is tightly coupled with it. It has been used as a communication system in Cloud Hopper. The tasks created

to be scheduled by celery are queued in the Redis database which are picked up by the brokers for their execution.

4.3.9 MongoDB

MongoDB [36] is a NoSQL database that uses key/value pairs approach to store the data. Cloud Hopper uses this primarily to log the tasks that were being scheduled to run at some particular time. Each task has a lifecycle of four states: accepted, scheduled, active and completed. When the Django REST API receives the task, the status is set as ‘Accepted’ and is sent to celery for scheduling. Once it reaches celery, it is scheduled to run at a particular time, and the status is changed to ‘Scheduled’. The moment task starts running; the status changes to ‘Active’. Finally, when the task is complete, status becomes ‘Completed’. Apart from this, when the task does not run correctly or stops abruptly due to some known or unknown reasons, the status changes to ‘Error’.

All these transitions are saved in the MongoDB collection along with many other details to keep track of the tasks.

4.3.10 Ansible Tool

Ansible is a powerful IT automation tool that automates application deployment, configuration management, cloud provisioning, intra-service orchestration. It can also be used for multi-cloud provisioning. Ansible provides Playbooks that are Ansible’s configuration, deployment, and orchestration language [37]. The modules in the Ansible library are executed directly on remote or through Playbooks [34].

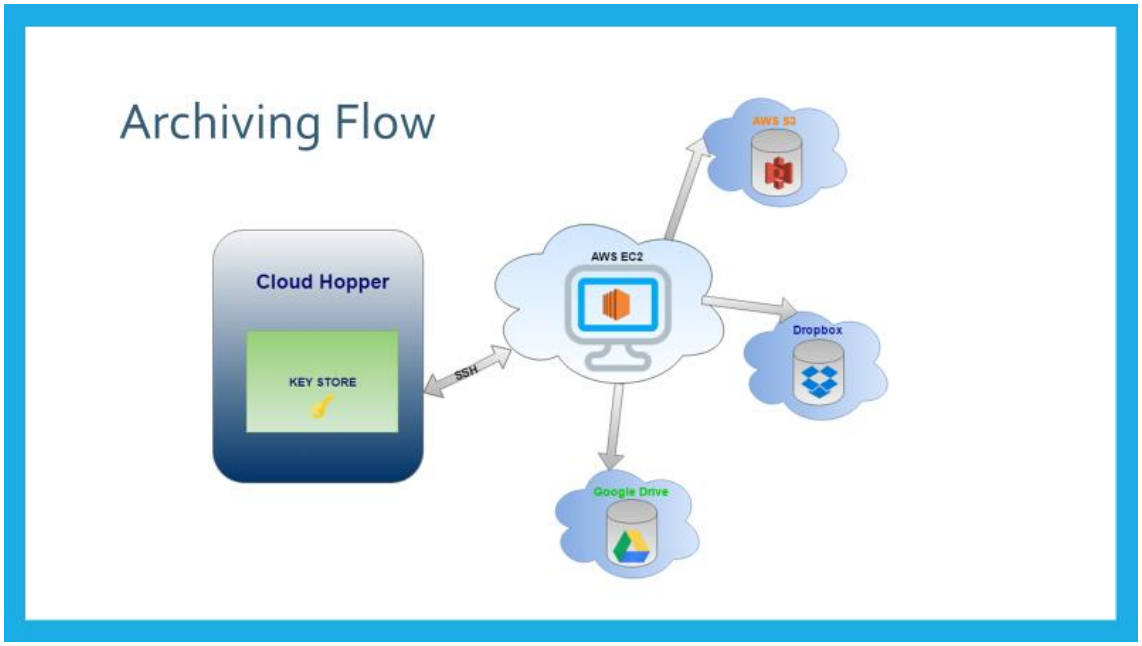

Figure 4.3 Archiving Flow Diagram

Cloud Hopper uses Ansible for archiving the files and folders from a compute instance such as AWS EC2, and Google Compute instance to storage services such as AWS S3, Google cloud storage, Dropbox & Google Drive. During the archiving process, Ansible API is being triggered with the SSH key of the virtual machine fetched from the key store. Ansible connects to the machine and compress the files and push them to the selected storage services. An illustration of the flow of archiving is shown in Fig. 4.3. In this way, the documents can be moved around from one cloud provider to another cloud provider or any other storage service provider with the help of Ansible.

4.3.11 Mailgun

Mailgun [39] is an SMTP [38] service which is used to send alerts through the emails. The notification driver interacts with Mailgun which fetches the statuses from the server and transfer them to the Mailgun which are finally sent to the intended users.

4.4 Packages

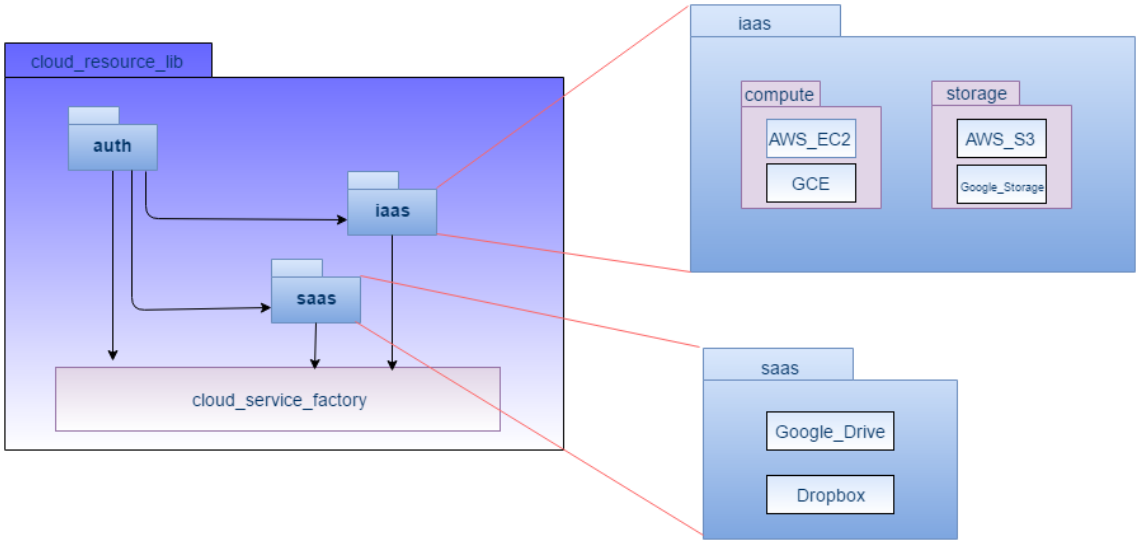

Figure 4.4 cloud_resource_lib package

The cloud_resource_lib package as shown in Fig. 4.4 provides an abstraction layer to the cloud service providers. The package contains sub-packages and modules that provide access to the functionalities of services offered by different cloud service providers.

This package has auth, iaas and saas sub-packages and their modules:

4.4.1 iaas

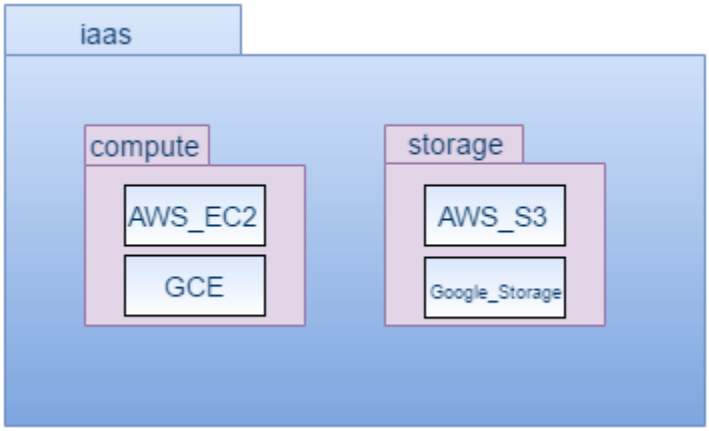

Figure 4.5 iaas sub-package

This module provides the features of Infrastructure as a Service and provides the infrastructure components such as compute and storage. The module further comprised of sub-packages named ‘compute’ and ‘storage’ as shown in Fig. 4.5.

- compute: This package has modules of compute services such as AWS EC2 and Google Cloud Compute offered by cloud providers such as AWS and Google respectively that inherit and override the functions of compute from their abstract classes.

- storage: This package has modules of storage such as AWS S3 and Google Cloud Storage offered by IaaS cloud providers which inherits and override the functions of storage from their abstract classes.

4.4.2 saas

Figure 4.6 saas sub-package

This package provides the features of Software as a Service and further contains modules for each type of SaaS service providers such as Dropbox and Google Drive as depicted in Fig. 4.6. These modules inherit and override the functions of an abstract class of saas module.

4.4.3 auth

This package has modules which are used to handle the access key and tokens of the cloud service providers. The modules of each cloud provider inherit the functions of the abstract class defined in the ‘auth’ module. These functions provide easy access to manage the access key and tokens for the various cloud providers and multiple accounts.

4.4.4 cloud_service_factory

This class is an implementation of factory pattern. It introduces an abstraction layer over the modules of iaas, saas, and auth packages and provides cloud specific API objects on request.

4.5 Classes

This section discusses the classes that are being used in the Cloud Hopper framework with the help of class diagrams.

4.5.1 vm_service class

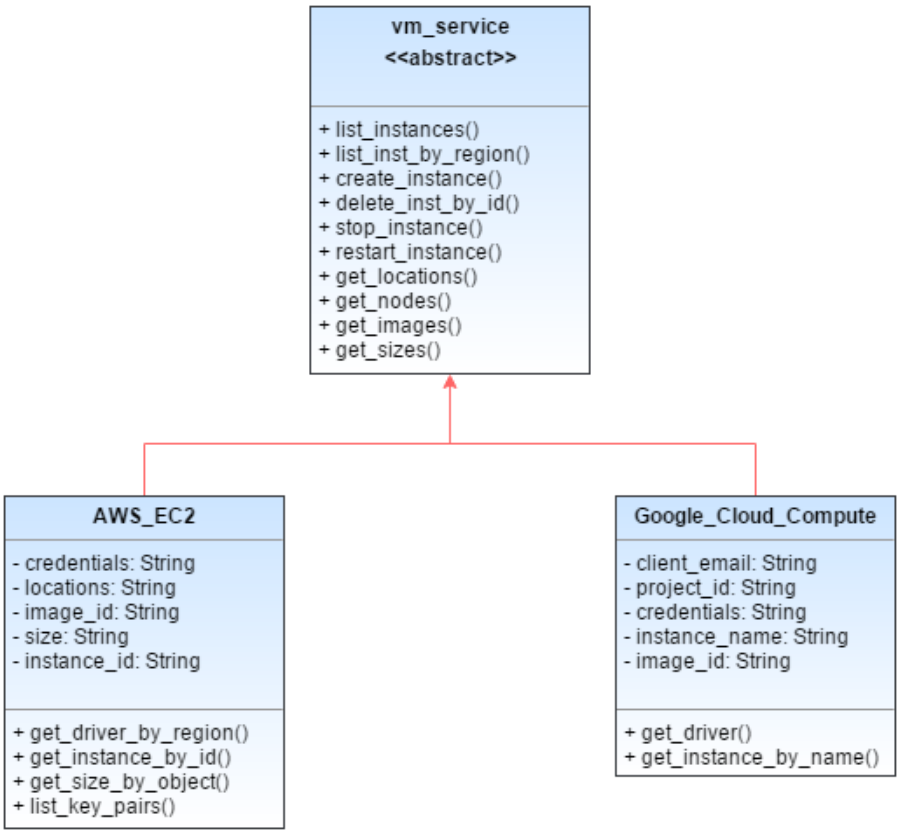

Figure 4.6 vm_service class diagram

vm_service is an abstract class for compute services which indicate that the methods mentioned in the UML representation of the vm_service class in Fig. 4.6 are abstract methods. The two child classes of AWS_EC2 and Google_Cloud_Compute each implement their own version of the abstract methods. Moreover, the inherited classes have their own methods and attributes.

4.5.2 iaas_storage_service class

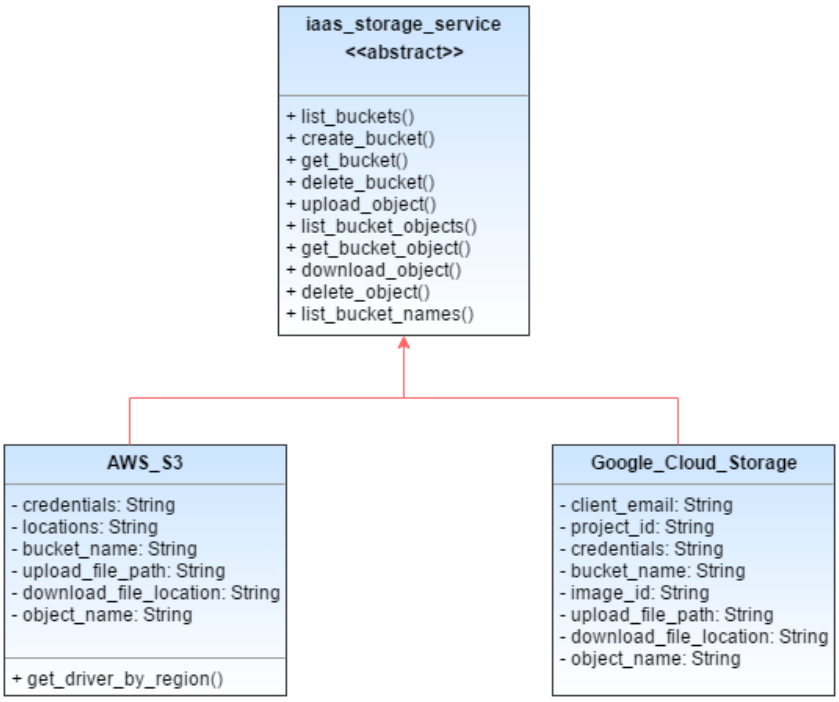

Figure 4.7 iaas_storage_service class diagram

iaas_storage_service is an abstract class for storage services offered by IaaS service providers. The UML representation of the class in Fig. 4.7 indicates the abstract methods. The two child classes of AWS_S3 and Google_Cloud_Storage each implement their own version of the abstract methods. These inherited classes have their own additional methods and attributes.

4.5.3 saas_storage class

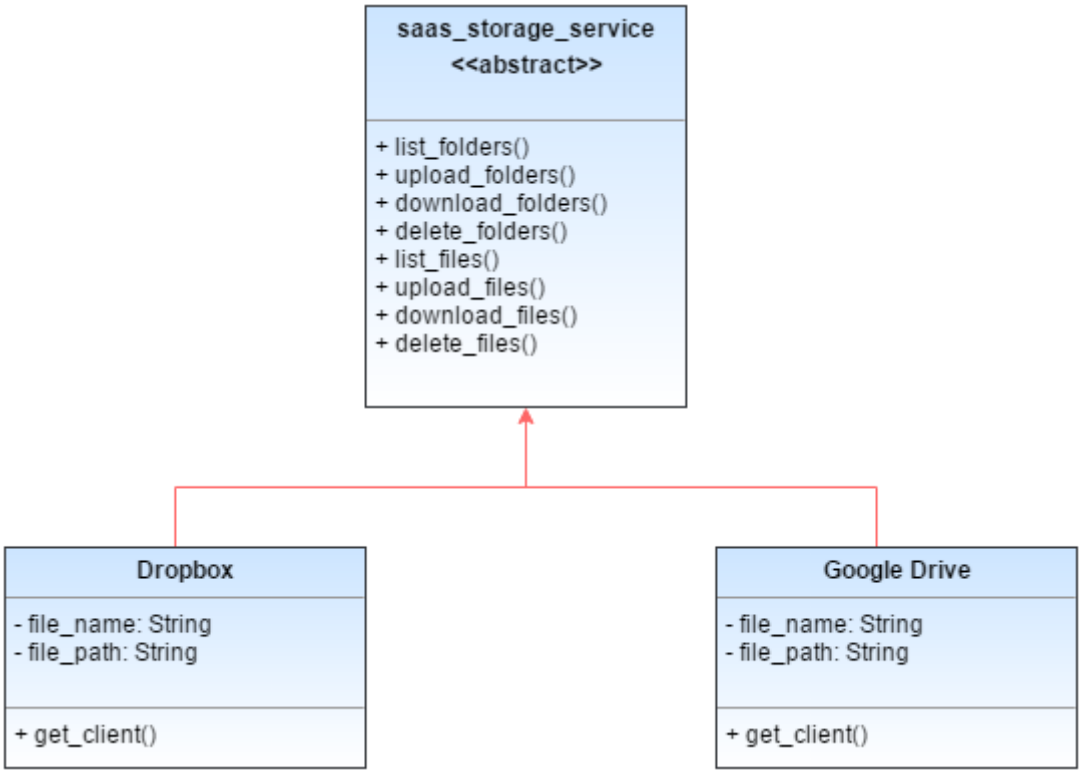

Figure 4.8 saas_storage_service class diagram

saas_storage_service is an abstract class for storage services offered by SaaS service providers such as Dropbox and Google Drive. Fig. 4.8 shows the UML representation of the saas_storage_class and its abstract methods. The two child classes of Dropbox and Google_Drive each implement their own version of the abstract methods. Moreover, these inherited classes have their own attributes and methods.

4.6 Database Structure

This section depicts the database design of the Cloud Hopper. The NoSQL database, MongoDB has been used to store and maintain the data. The reason behind using MongoDB as the database is that it provides high performance, high availability, and automatic scaling [40]. Data in MongoDB has a flexible schema. It is stored in a JSON-like object in field and value format data model. Three main collections are being used to store the credentials, data of tasks and the workflow data. These collections are authentication, scheduled_tasks, and workflow respectively. Each of the collections is explained in the following sections:

4.6.1 authentication Collection

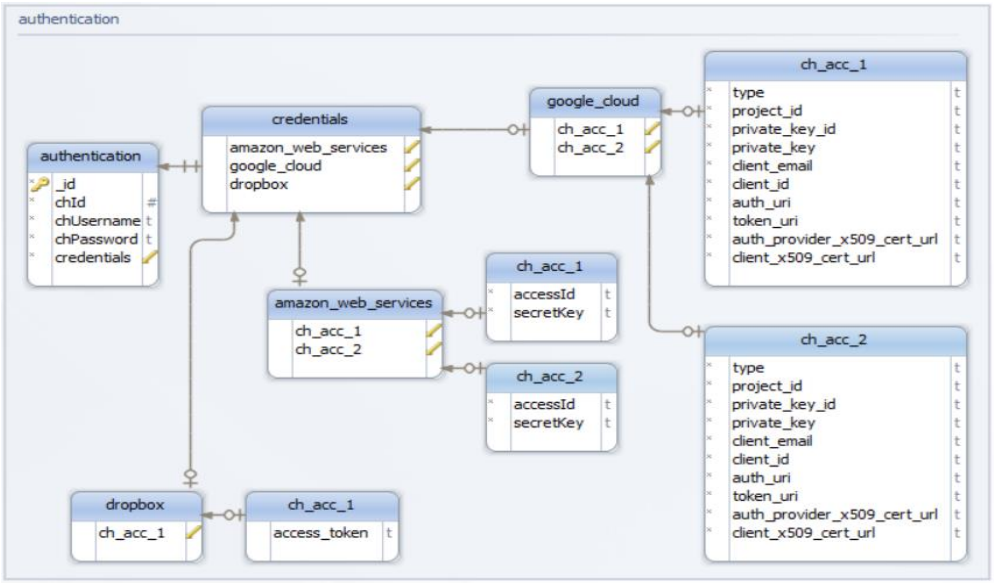

Figure 4.9 authentication schema

This collection is used to store the access keys and secret keys in a secure manner. Fig. 4.9 shows the schema of authentication collection. ‘chId’ and ‘chUsername’ fields of each document are used to store the login credentials of the Cloud Hopper application. ‘credentials’ field is an embedded sub-document that further contains sub-documents in the form of arrays. They store the access keys and secret keys of each account and follows ‘One-to-Many’ relationship model. Before storing these credentials, they are being encrypted by AES 256 bit and 32-bit block algorithm. Further, when the credentials are retrieved from the database, they are decrypted and then used.

4.6.2 scheduled_tasks Collection

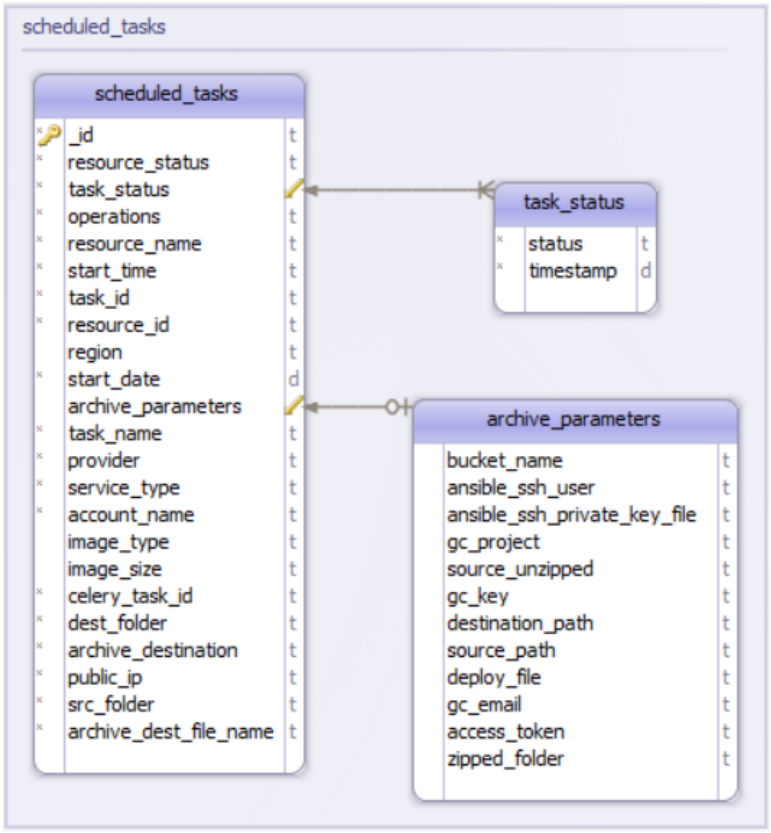

Figure 4.10 scheduled_tasks schema

This collection is being used for storing each task's details. Most of the fields shown in the Fig. 4.10 of scheduled_tasks schema shows the fields that are related to a resource. All the details of each resource that has been scheduled along with the timestamp for which the task has been scheduled to run and their statuses are saved. Snapshots of the details of the task are stored each time the status of the task changes.

The ‘task_status’ and ‘archive_parameters’ are the sub-fields. The ‘task_status’ field uses ‘One-to-Many’ relationship with the ‘status’ documents of the task. ‘archive_parameters’ uses ‘One-to-One’ relationship model which stores the details of the archiving task.

4.6.3 Workflow Collection

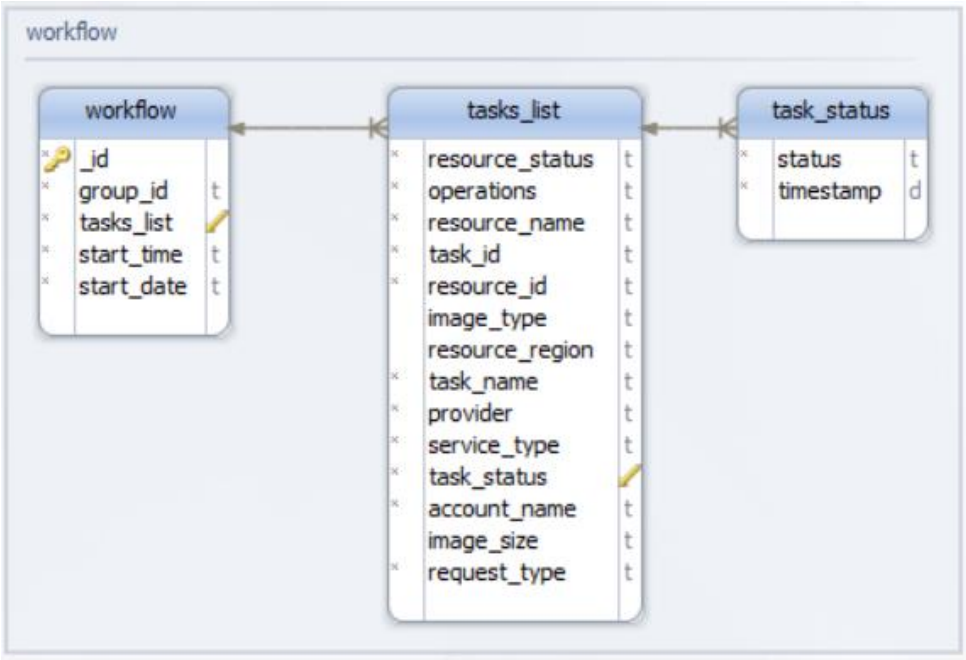

Figure 4.11 workflow collection schema

This collection has been used to store the details of the scheduled workflows. All the tasks under a workflow are grouped under a ‘group_id’ field. The ‘task_list’ field of

each group follows the ‘One-to-Many’ relationship model and contains the array of dictionaries. Each dictionary consists of all the details of each task along with their statuses and their timestamps at which they change as can be seen in the Fig. 4.11.

CHAPTER 5

IMPLEMENTATION

In this chapter, we will discuss the implementation of the architecture presented in the previous chapter. The application can be either hosted on local environment or on the public cloud. For our experiments, application is hosted on the local environment. A Mongo DB server was installed on the AWS EC2 instance. Below section gives an overview of few main screens of the application.

5.1 Screenshots

5.1.1 Dashboard

Figure 5.1 Dashboard Screen

Fig. 5.1 depicts the dashboard of the application that is designed to show the number of resources in use and their usage in the last seven days. Also, it shows the number of tasks scheduled and completed.

5.1.2 Key Store

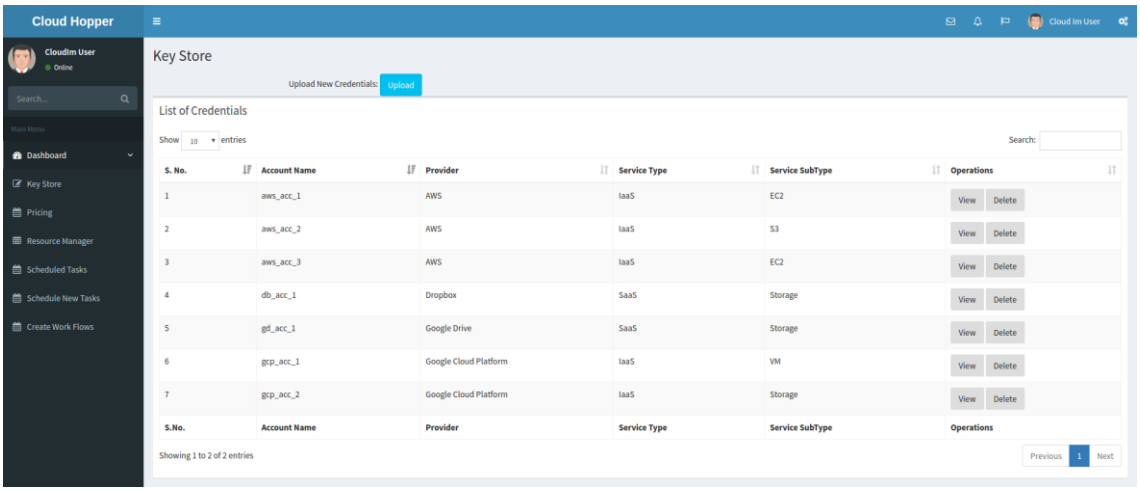

Figure 5.2 Key Store Screen

Fig. 5.2 illustrates the functionality of key store where one can upload, edit, and delete the key details.

5.1.3 Resource Manager

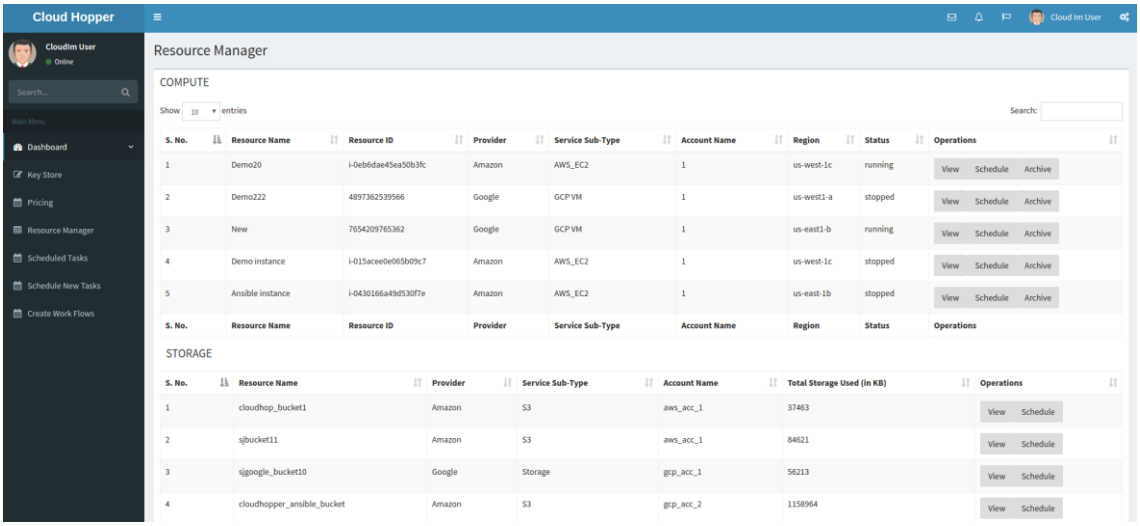

Figure 5.3 Resource Manager

As can be seen in the Fig. 5.3, resource manager screen shows the details of the resources in use along with the scheduling and archiving provision.

5.1.4 Scheduled Tasks

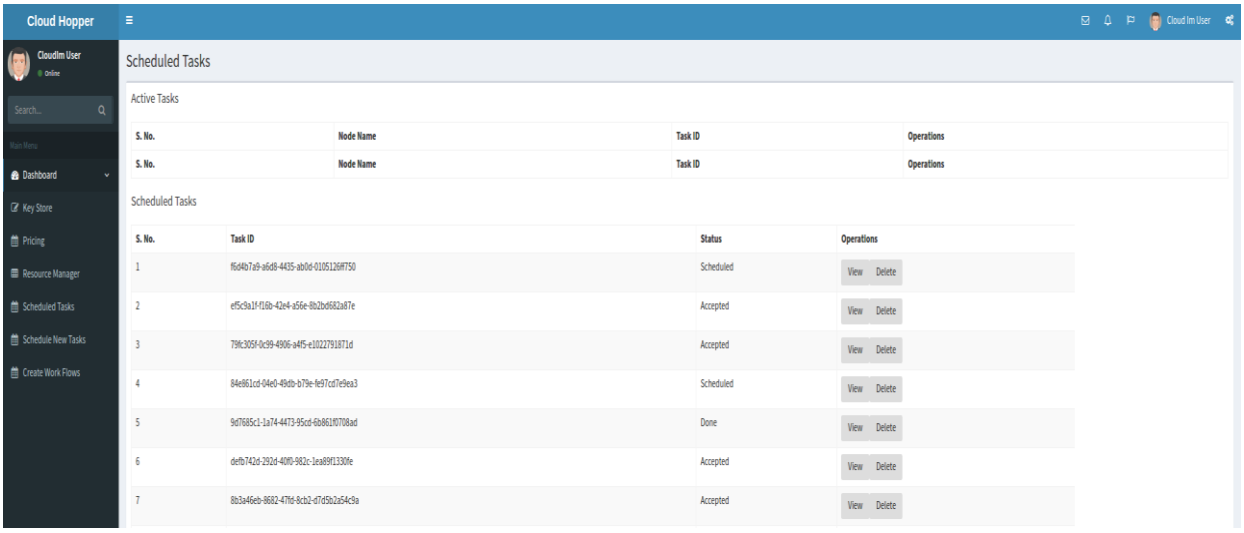

Figure 5.4 Scheduled Tasks Screen

Scheduled Tasks screen presents user the details about the tasks that are being scheduled and the tasks that are being triggered and are in active state. It also gives the ability to the user to delete the task at any point of time. Fig. 5.4 gives an overview of the screen.

5.1.5 Schedule New Tasks

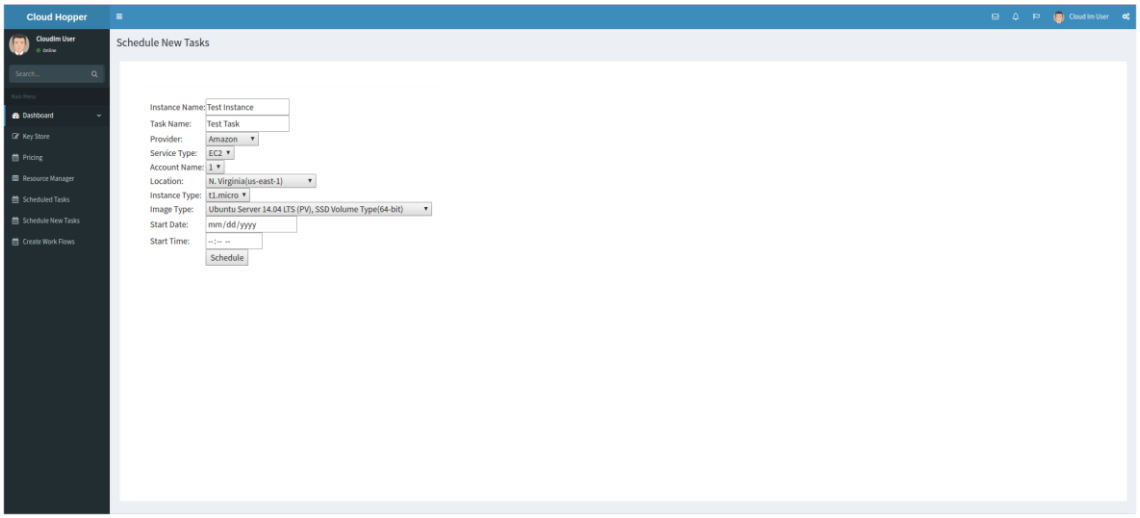

Figure 5.5 Schedule New Tasks Screen

When a new task is to be created, and scheduled, it can be done from the Schedule New Tasks screen as shown in the Fig. 5.5.

5.1.6 Workflows

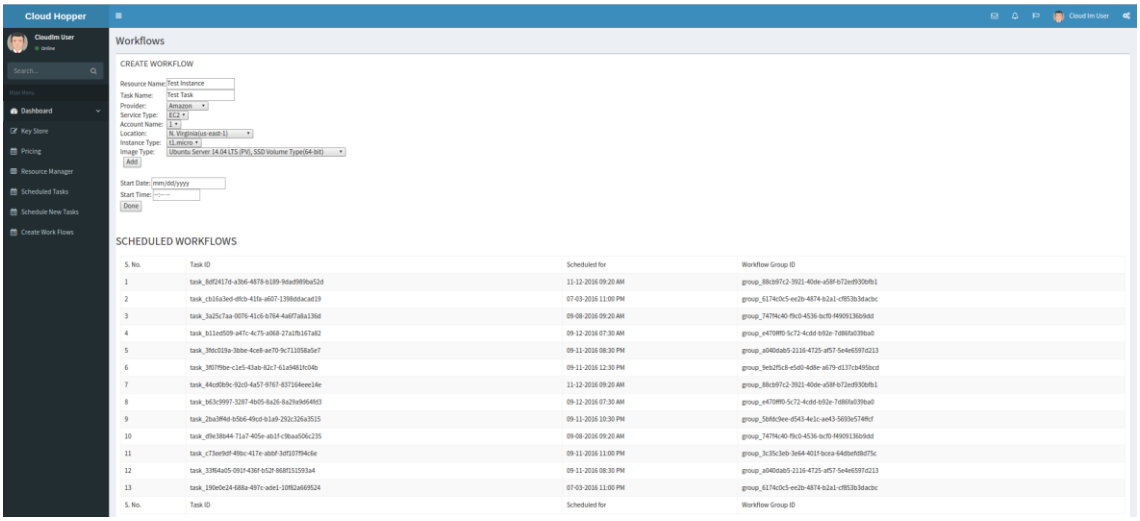

Figure 5.6 Workflows Screen

As depicted in the Fig. 5.6, this screen can be used to create a workflow which is a series of tasks to be run in an order. These workflows can also be scheduled and the details of each workflow are presented in a tabular form

5.1.7 Pricing

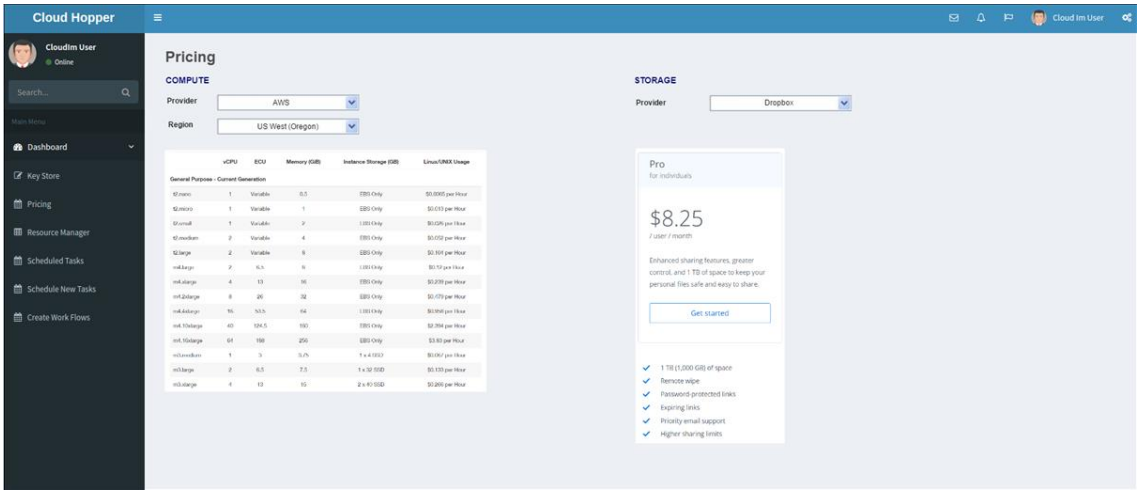

Figure 5.7 Pricing Screen

The pricing screen as shown in the Fig. 5.7 gives user an idea of the prices of the different services offered by the cloud providers.

5.1.8 Slack Bot

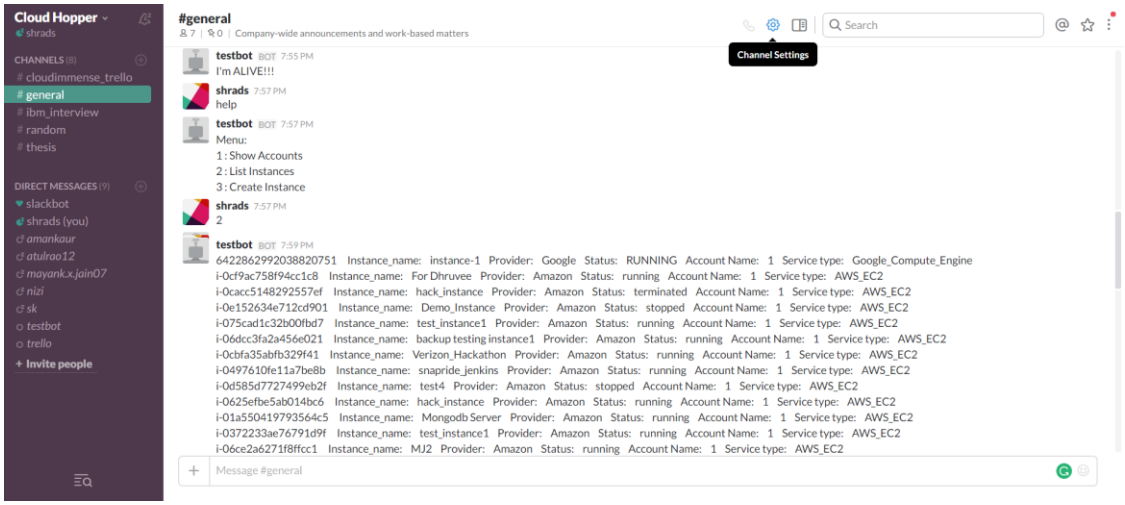

Figure 5.8 Slack Bot Screen

Fig. 5.8 illustrates the Slack Bot interface.

5.1.9 Alerts

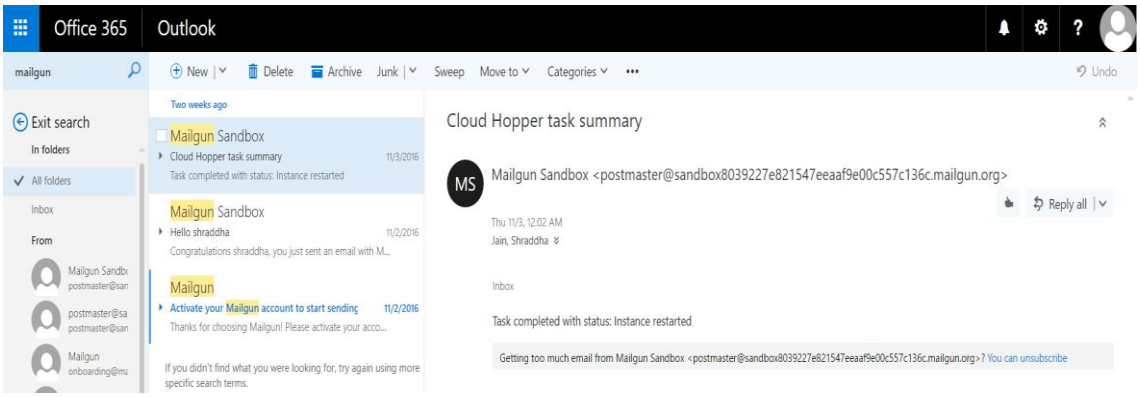

Figure 5.9 Alerts Screen

Fig. 5.9 shows an example of the email alert triggered by Mailgun through Cloud Hopper regarding a task completion to the user.

CHAPTER 6

PRICE ANALYSIS

6.1 Compute prices comparison and analysis

These days, companies who are using cloud services do not want to dump all the workload of multiple services to a single cloud provider, rather, they try to distribute the work on multiple clouds as per the providers who are offering most efficient services and for the best prices. Keeping that in mind, in this thesis, research on the prices of various services provided by some IaaS and SaaS cloud players is done.

Instance Type	AWS Instance Type	Hourly $	Monthly $	Google Cloud Platform Instance Type	Hourly $	Monthly $
1 core, 3.75 GB memory	m3.medium	0.067	48.91	n1-standard-1	0.036	25.95
2 core, 7.5 GB memory	m3.large	0.133	97.09	n1-standard-2	0.071	51.5
4 core, 15 GB memory	m3.xlarge	0.266	194.18	n1-standard-4	0.141	102.6
8 core, 30 GB memory	m3.2xlarge	0.532	388.36	n1-standard-8	0.281	204.8

Table 6.1: AWS vs. GCP compute pricing

The comparison of the monthly cost of various instance types of AWS and GCP is shown in the Table. 6.1. The AWS pricing data is collected from Amazon EC2 Pricing webpage [41]. And for Google Cloud Platform, it is obtained from Google Cloud Platform developer's console after logging into its console and choosing the compute services [42]. All the comparisons are made for the image type as 'Linux', instance types of the region 'US West', specifically, for AWS, it is 'us-west-1 (Oregon)', and for GCP, it is 'us-west1-a'. The monthly cost is calculated for a total of 730 hours.

The cost comparison Table. 6.1 gives the glimpse of monthly costs incurred for one virtual machine of some instance types for AWS and GCP. Now, if we see it as a big picture and consider a scenario where in a company, out of 500 virtual machines of 'Linux' image type and region as 'us-west-1' are in running state. Out of them, 80 AWS machines of instance type 'm3.large' are of no use and are still running without being monitored. Those machines if left running for a month will cost total:

Total unused VMs x Cost of each instance type = Total cost of the unused VMs

80 x 97.09 (in USD) = 7767.2 (in USD)

The total cost of 500 VMs would have been:

Total VMs x Cost of each instance type = Total cost of the VMs

500 x 97.09 (in USD) = 48545 (in USD)

Above calculations show that if the 80 needlessly provisioned or over-provisioned machines are terminated, it reduces the cost by up to 16% per month.

Let us take another scenario, where there are total 500 AWS virtual machines created by a company. Out of them, 300 are of instance type ‘m3.medium’ and rest 200 are of instance type ‘m3.2xlarge’. From these 500 VMs, let us consider 100 VMs of ‘m3.medium’ type and 70 VMs of ‘m3.2xlarge’ are in running state and are no longer being used. The total monthly cost of these VMs will be:

For ‘m3.medium’ instance type:

Total unused VMs x Cost of each instance type = Total cost of the unused VMs

100 x 48.91 (in USD) = 4891 (in USD)

For ‘m3.2xlarge’ instance type:

Total unused VMs x Cost of each instance type = Total cost of the unused VMs

70 x 388.36 (in USD) = 27185.2 (in USD)

Total cost = 4891 + 27185.2 = 32076.2 (in USD)

The above results show the total amount which a company is losing on the unused VMs. In this scenario, the total cost of all the VMs for a month would have been:

Total VMs x Cost of each instance type = Total cost of the VMs

300 x 48.91 (in USD) = 14673 (in USD)

200 x 388.36 (in USD) = 77672 (in USD)

Total = 79135 (in USD)

If the 170 (100 +70) unused machines are managed properly, then the company would save up to 41.29 %.

If we consider another scenario where a company has 300 VMs on AWS, each of ‘Linux’ operating system image and the region as ‘us-west-1’. Out of them, 200 are of ‘m3.medium’ type and 100 are of ‘m3.2xlarge’ type. Also, it has 200 VMs on GCP, each of ‘Linux’ image, region as ‘us-west1-a’ and ‘n1-standard-8’ instance type. Out of the 300 AWS VMs, 100 of ‘m3.medium’ type and 50 ‘m3.2xlarge’ are no more in use. Out of 200 GCP VMs, 100 are of no use and are in running state. In this scenario, the total monthly cost for these VMs will be:

AWS VMs:

Total unused VMs x Cost of each instance type = Total cost of the unused VMs

100 x 48.91 (in USD) = 4891 (in USD)

50 x 388.36 (in USD) = 19418 (in USD)

GCP VMs:

Total unused VMs x Cost of each instance type = Total cost of the unused VMs

100 x 204.8 (in USD) = 20480 (in USD)

Total cost = 4891 + 19418 + 20480 = 44789 (in USD)

The above scenarios show the extra cost which companies are paying per month. If the third scenario is considered for six months, then the total amount the company is losing will be:

Total cost = 6 x 44789 = 268734 (in USD)

The total cost of 500 VMs, some on AWS and some on GCP would have been:

AWS VMs:

Total VMs x Cost of each instance type = Total cost of the VMs

200 x 48.91 (in USD) = 9782 (in USD)

100 x 388.36 (in USD) = 38836 (in USD)

GCP VMs:

Total VMs x Cost of each instance type = Total cost of the VMs

200 x 204.8 (in USD) = 40960 (in USD)

Total = 89578 (in USD)

The total cost which could be saved on these unmanaged and over-provisioned VMs would be up to 50% a month which is a huge saving for any company. Thus, if these resources are managed properly, they can help in maintaining the budgets of a company.

Moreover, the Table. 6.1 can also be used to compare the VM prices on AWS and GCP. It will help in deciding on the economic options for the company or a user which ultimately will assist in keeping the budgets stable for them.

6.2 IaaS Storage prices comparison and analysis

	Standard Storage	Standard - Infrequent Access Storage	Glacier Storage
First 1 TB / month	$0.0300 per GB	$0.0125 per GB	$0.007 per GB
Next 49 TB / month	$0.0295 per GB	$0.0125 per GB	$0.007 per GB
Next 450 TB / month	$0.0290 per GB	$0.0125 per GB	$0.007 per GB
Next 500 TB / month	$0.0285 per GB	$0.0125 per GB	$0.007 per GB
Next 4000 TB / month	$0.0280 per GB	$0.0125 per GB	$0.007 per GB
Over 5000 TB / month	$0.0275 per GB	$0.0125 per GB	$0.007 per GB

Table 6.2: AWS S3 pricing

Table. 6.2 illustrates the pricing of AWS S3 storage for 'US West (Oregon)' region (as per 11/5/2016). Monthly bill can be estimated by using it.

Multi-Regional Storage (per GB per Month)	Regional Storage (per GB per Month)	Nearline Storage (per GB per Month)	Coldline Storage (per GB per Month)
$0.026	$0.02	$0.01	$0.007

Table 6.3: GCP storage pricing

Table. 6.3 gives the pricing of GCP storage (as per 11/5/2016). Both tables are being used for price comparison of the storage services offered by the different cloud service providers which will be helpful in deciding on the economical options.

6.3 SaaS Storage prices comparison and analysis

Till now, we analyzed and compared the prices of IaaS (Infrastructure as a Service) such as compute and storage provided by AWS and GCP. Let us analyze the pricing of some SaaS (Software as a Service) providers such as Dropbox and Google Drive.

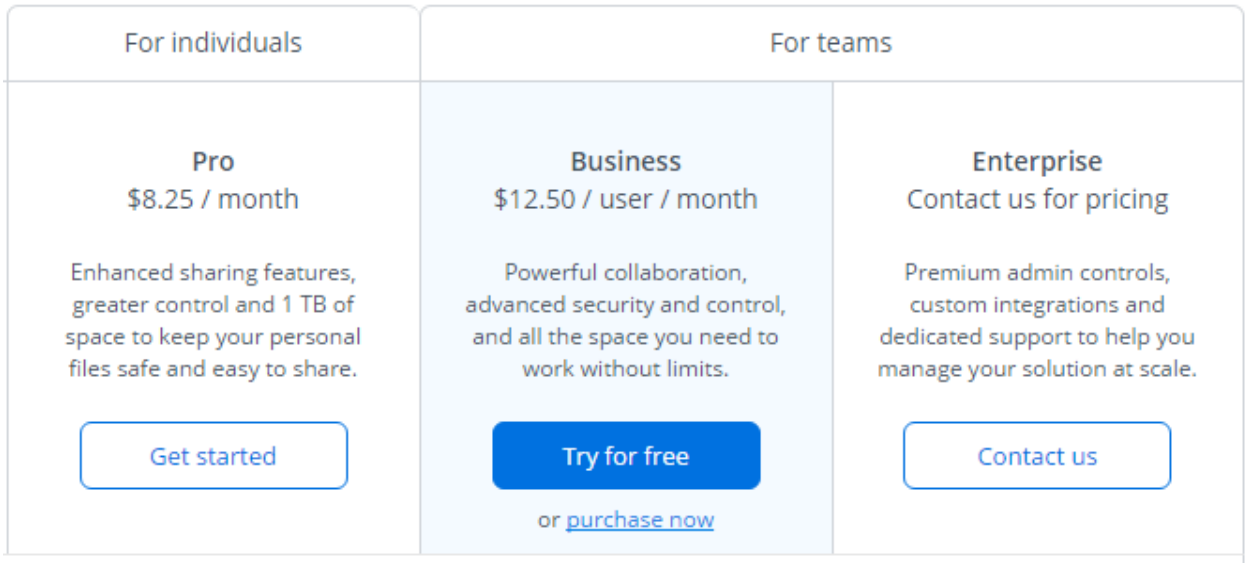

Figure 6.1 Dropbox pricing

Fig. 6.1 gives the overview of pricing for personal or a business account for a month (as per 11/6/2016). And, Fig. 6.2 presents the pricing on a Google Drive account for a month.

Total storage	Monthly price
15 GB	Free
100 GB	$1.99
1 TB	$9.99
10 TB	$99.99
20 TB	$199.99
30 TB	$299.99

Figure 6.2 Google Drive Pricing

Let us consider a scenario where a company has some over-provisioned VMs which are of no use and are costing unnecessary. The company wants to terminate them. But those machines have some of the important files and folders which need to be archived before the machines are put down. Cloud Hopper provides a provision to archive the directories and documents either to the IaaS services such as AWS S3, Google Cloud Storage or PaaS services such as Dropbox and Google Drive and even on the local machine on few clicks of buttons through a simple user interface. Let us calculate and compare the pricing of 1TB storage for a month on the below-mentioned service providers. First, let us take IaaS services:

AWS S3: (for standard storage and location US West (Oregon))

Cost for 1TB a month:

0.0300 (in USD) x 1024 GB = 30.72 (in USD)

GCP Storage: (for regional storage)

Cost for 1TB a month:

0.02 (in USD) x 1024 GB = 20.48 (in USD)

GCP storage comes out to be a cheaper option to choose among IaaS based storage services. AWS also charges for Request Pricing and Data Transfer Pricing [43]. So, the AWS S3 final price goes even higher.

Now, let us do the similar cost comparison for SaaS-based storage services:

Dropbox:

If we choose the business account for a user, Dropbox charges $12.50 for unlimited storage for a month.

Cost for a month:

12.50 (in USD)

Google Drive:

Cost for 1TB a month:

9.99 (in USD)

Thus, since we have more affordable and cheaper options for storage, a company sticking to only IaaS services can even move on with the above described SaaS services and can cut down on the sprawling costs on these storage services.

As per the VMware whitepaper on ‘controlling virtual machine sprawling’ [44], below three scenarios are shown on the virtual machines, which gives some statistics that shows how companies can save a lot of amount on over-provisioned, under-used or temporary virtual machines.

Scenario 1: A typical company with 1,000 virtual machines—with 5 percent of machines created without proper business justification and another 5 percent over-provisioned—could easily save over USD $100,000 to USD $150,000 in capital expenditures through better control of the front-end provisioning process. By delivering the right sized machine at the right service level, companies can eliminate waste, improve resource utilization, and lower costs [44].

Scenario 2: A typical company with 1,000 virtual machines that have five percent of its machines used for temporary applications can expect to save approximately USD $60,000 to USD $70,000 by automating the reuse of leased machines. Environments such as development and testing, which have a larger need for temporary machines, can expect even greater savings [44].

Scenario 3: A typical company with 1,000 virtual machines and 10 percent of its resources consumed by inactive and abandoned virtual machines can expect to save between USD $80,000 and USD $100,000 on capital expenditures annually [44].

CHAPTER 7

EXPERIMENTS AND RESULTS

This chapter discusses the experiments and comparisons done that are used to analyze the performance of various services offered by multiple cloud service providers. The details of the experiments are discussed in each of the below sections.

7.1 Experimental Setup

The experiments were performed using the below configurations:

- Programming language: Python 2.7
- Framework: Django, Django REST, Celery 4.0
- Database: MongoDB 3.4, Redis 2.10.5
- Cloud Platforms: Amazon Web Services, Google Cloud Platform
- DevOps: Ansible 2.2
- Network upload speed: 70 Mbps
- Host location: US Central

7.2 AWS Compute Experiments

This section shows the performance analysis of creation, deletion, and listing of various instances on AWS in different US regions. The experiments 1, 2 and 3 were conducted on the ‘Ubuntu 14.0’ virtual machine of ‘t2.micro’ type, on ‘AWS’ in all the US

regions (except for US-East-2 (Ohio)) that include ‘US-West-1 (N. California)’, ‘US-West-2 (Oregon)’ and ‘US-East-1 (N. Virginia)’.

Experiment 1: EC2 instance creation time comparison across the US regions

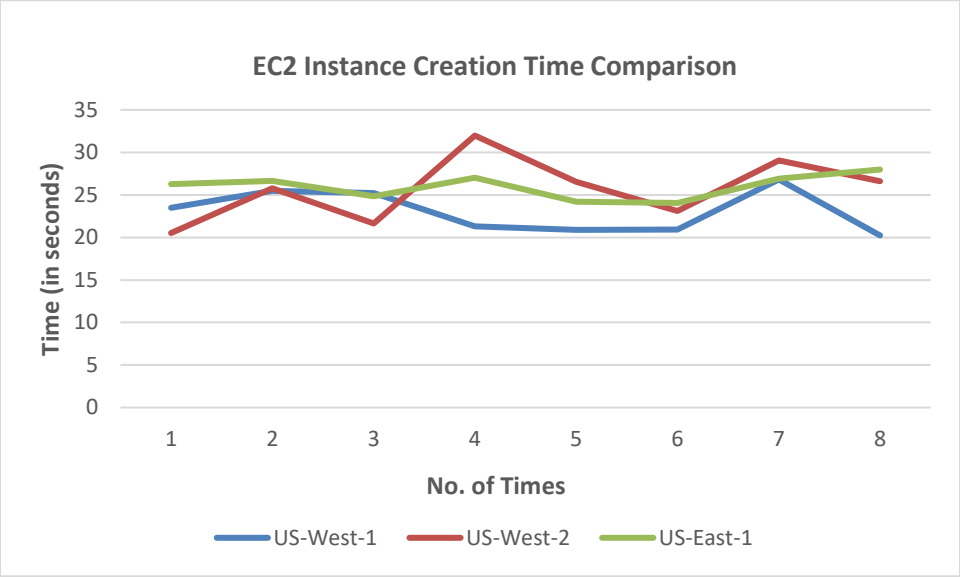

Figure 7.1 EC2 Instance creation time comparison

In this experiment, through Cloud Hopper, a machine with the configurations mentioned in the beginning were created eight times in each US region and the time of creation of each machine was recorded. Fig. 7.1 shows the time taken to create eight EC2 instances.

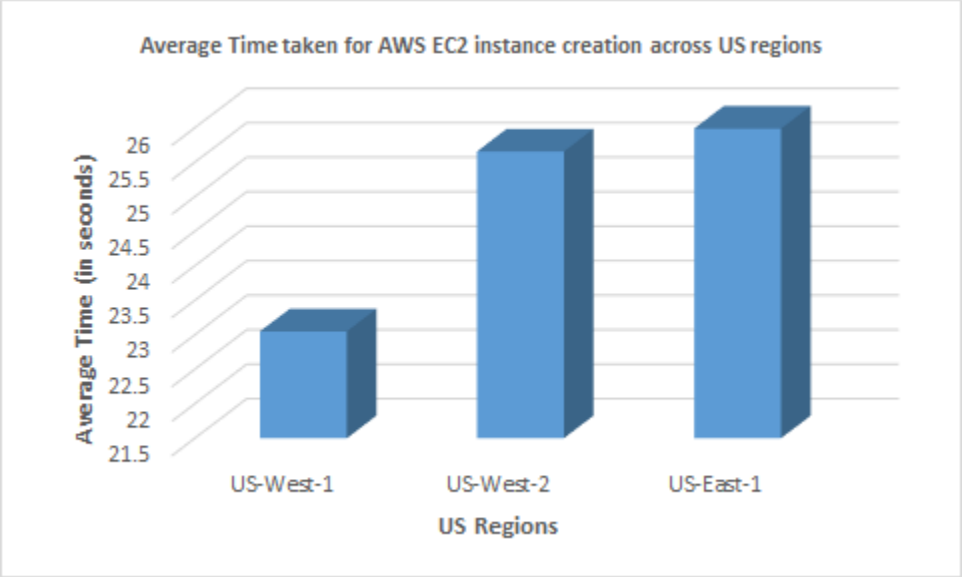

Figure 7.2 Average time for AWS EC2 instance creation

Fig. 7.2 shows the average time taken to create an EC2 instance across the US regions. It can easily be deduced that time taken to create an instance is least in 'US-West-1' region followed by 'US-West-2' and the region 'US-East-1' took the maximum time. Thus, it shows that it is faster to create an instance in 'US-West-1' region

Experiment 2: EC2 instance deletion time comparison across US regions

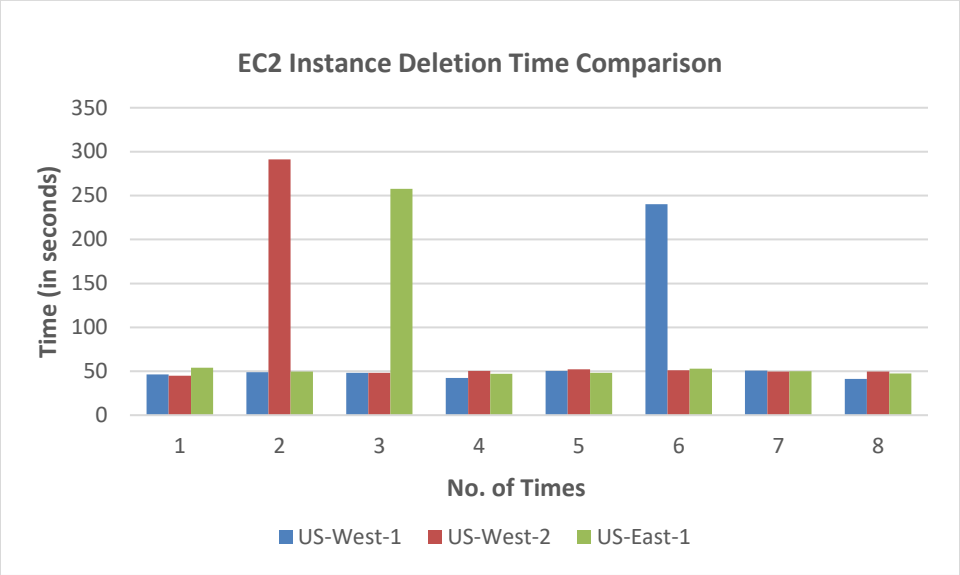

Figure 7.3 EC2 Instance deletion time comparison

Fig. 7.3 shows the time taken to delete eight EC2 instances across US regions and which are in ‘active’ or ‘stopped’ state. We can see that deletion time was almost the same for each instance in ‘US-West-1’ region.

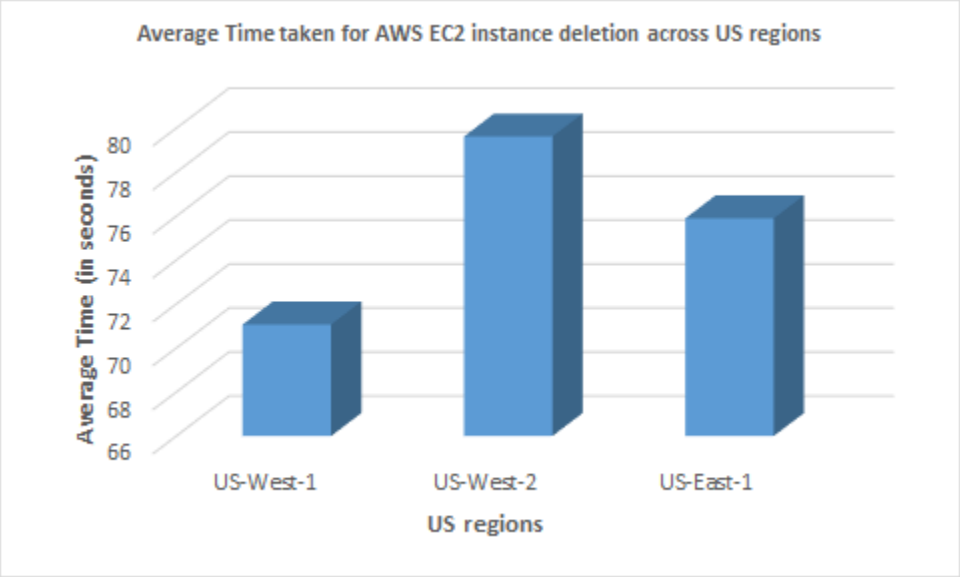

Figure 7.4 Average time for AWS EC2 instance deletion

Fig. 7.4 depicts the average time taken to delete an instance across the US regions. It can easily be deduced that time taken to delete an instance is least in 'US-West-1' region followed by 'US-East-1' and the 'US-West-2' region took the maximum time. So, it shows that it is faster to delete the instance in 'US-West-1' region.

Experiment 3: EC2 instances listing time comparison across US regions

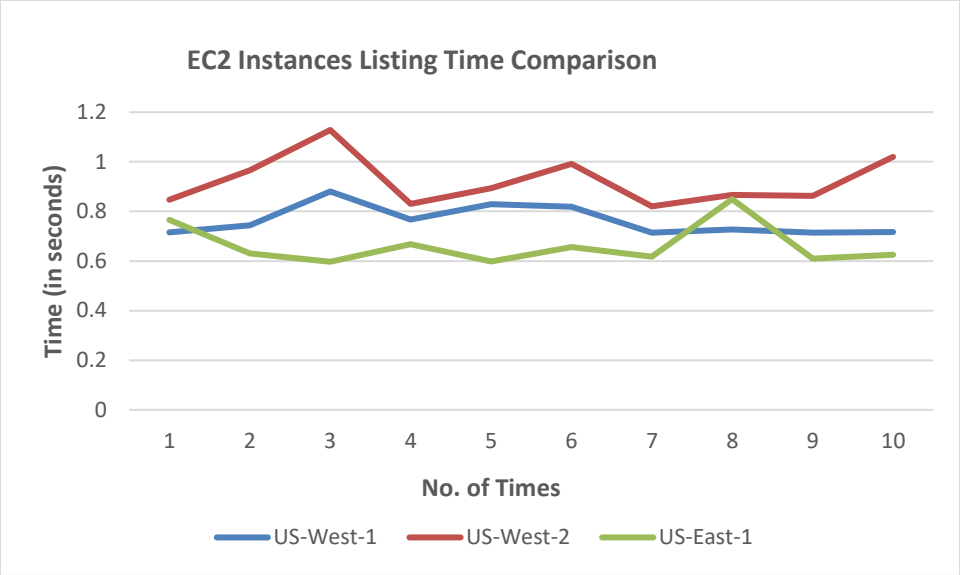

Figure 7.5 EC2 instances listing time comparison

In this experiment, listing times of 10 instances in each US region were recorded. The listing was done ten times for each region. Fig. 7.5 shows the time taken to list ten instances ten times and in whichever state they are. We can deduce that listing time was highest for ‘US-West-2’ region.

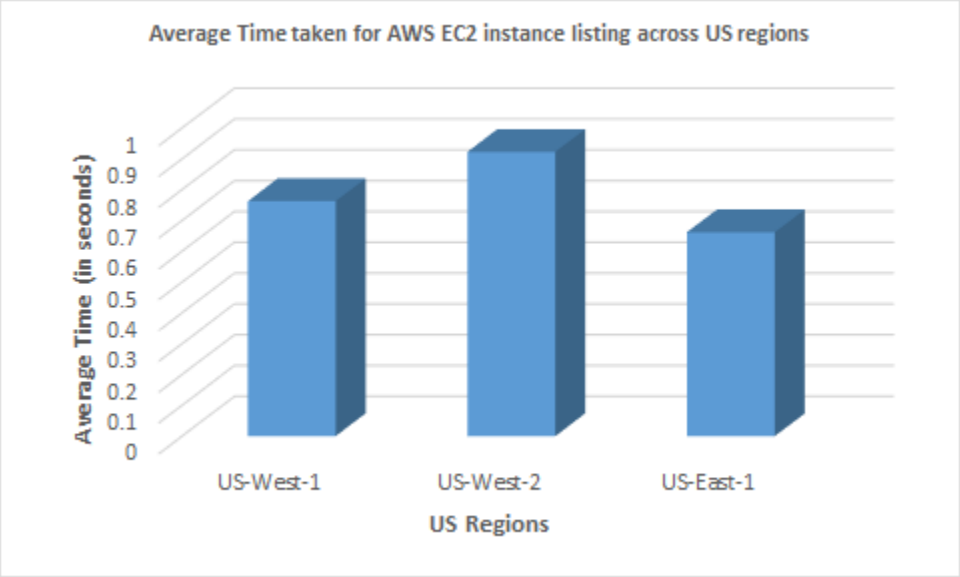

Figure 7.6 Average time for AWS EC2 instance listing

Fig. 7.6 illustrates the average time taken to list ten instances in the US regions. 'US-East-1' took the least time to list the instances.

From experiments 1, 2 and 3, we can deduce that creation and deletion of an instance is faster if it is done in 'US-West-1' region and listing of instance is faster in 'US-East-1' region on AWS when the Cloud Hopper is hosted in US central location. So, using 'US-West-1' to maintain an instance is a better deal when the application is hosted in US central location.

7.3 Google Cloud Compute Experiments

This section shows the performance analysis of creation, deletion, and listing of various instances on GCP in different US regions. The experiments 4, 5 and 6 were

conducted on the ‘Ubuntu 14.0’ virtual machine of ‘n1-standard-1’ type, on ‘Google Cloud Platform’ in all the US regions that include ‘us-west1-a’, ‘us-west1-b’, ‘us-central1-a’, ‘us-central1-b’, ‘us-central1-c’, ‘us-central1-f’, ‘us-east1-b’, ‘us-east1-c’ and ‘us-east1-d’.

Experiment 4: Instance creation time comparison across the regions of United States

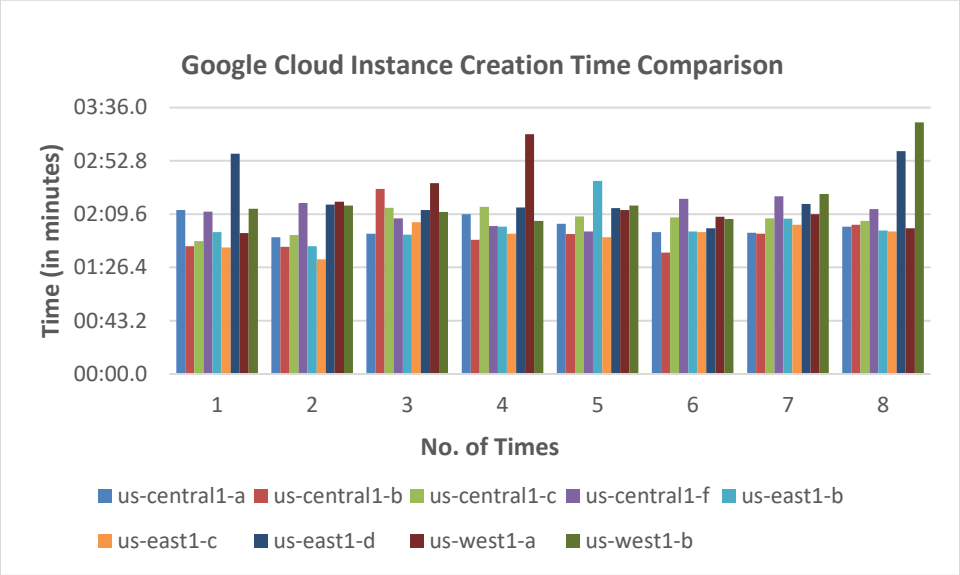

Figure 7.7 Google Cloud instance creation time comparison

In this experiment, through Cloud Hopper, a machine with the configurations mentioned in the beginning were created eight times in each US region and the time of creation of each machine was recorded. Fig. 7.7 shows the time taken to create 8 EC2 instances in all the US regions.

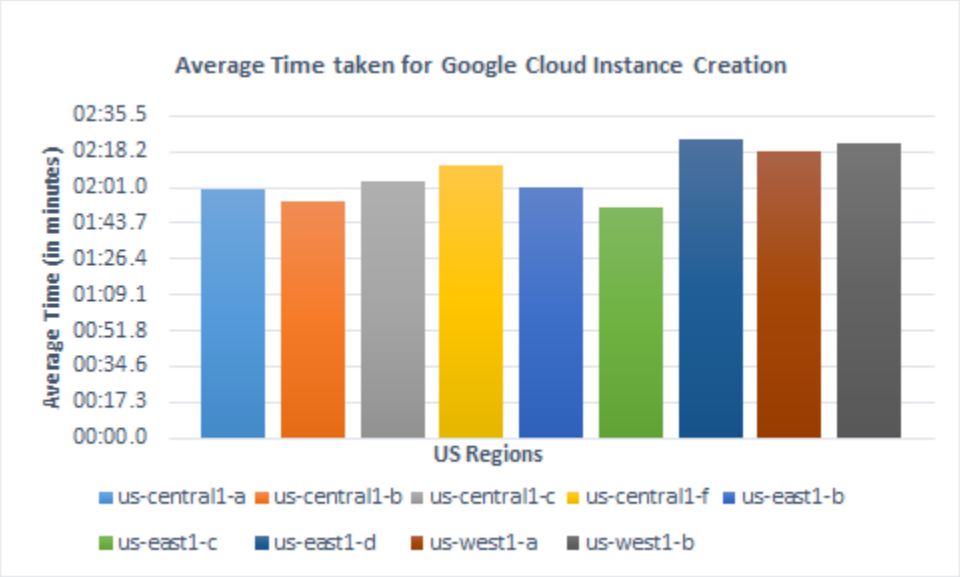

Figure 7.8 Average time for Google Cloud instance creation

Fig. 7.8 shows the average time taken to create an instance across the US regions. It indicates that time taken to create an instance is least in ‘us-east1-c’ while region ‘us-east1-d’ took the maximum time. Thus, it can be deduced that it is faster to create an instance in ‘us-east1-c’ region when the Cloud Hopper application is hosted in US Central region.

Experiment 5: Google Cloud instance deletion time comparison across US regions

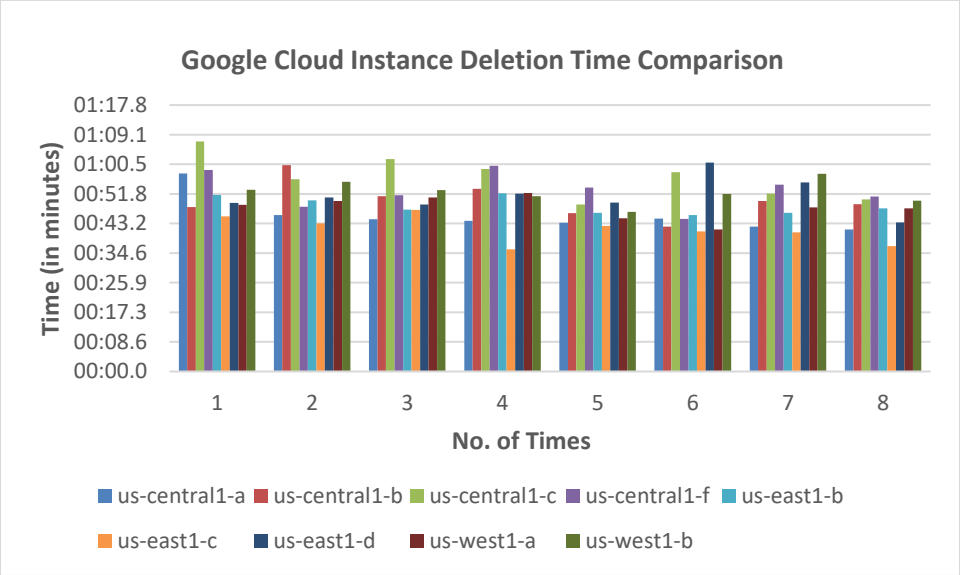

Figure 7.9 Google Cloud instance deletion time comparison

Fig. 7.9 shows the time taken to delete eight instances across US regions and which are in ‘active’ or ‘stopped’ state.

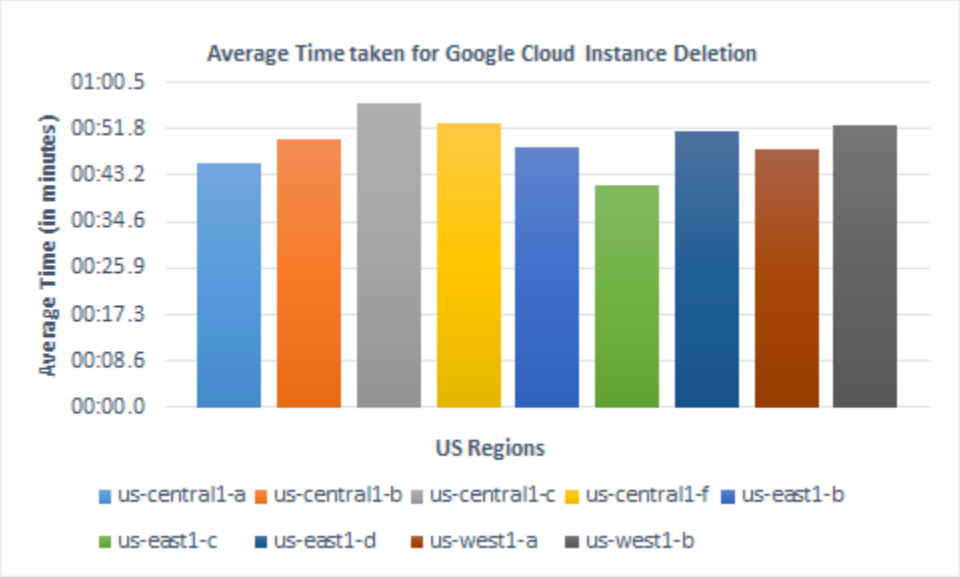

Figure 7.10 Average time for Google Cloud instance deletion

Fig. 7.10 depicts the average time taken to delete an instance across the US regions. It can easily be deduced that time taken to delete an instance is least in 'us-east1-c' region and the 'us-central1-c' region took the maximum time. So, it shows that it is faster to delete the instance in 'us-east1-c' region.

Experiment 6: Google Cloud instances listing time comparison across US regions

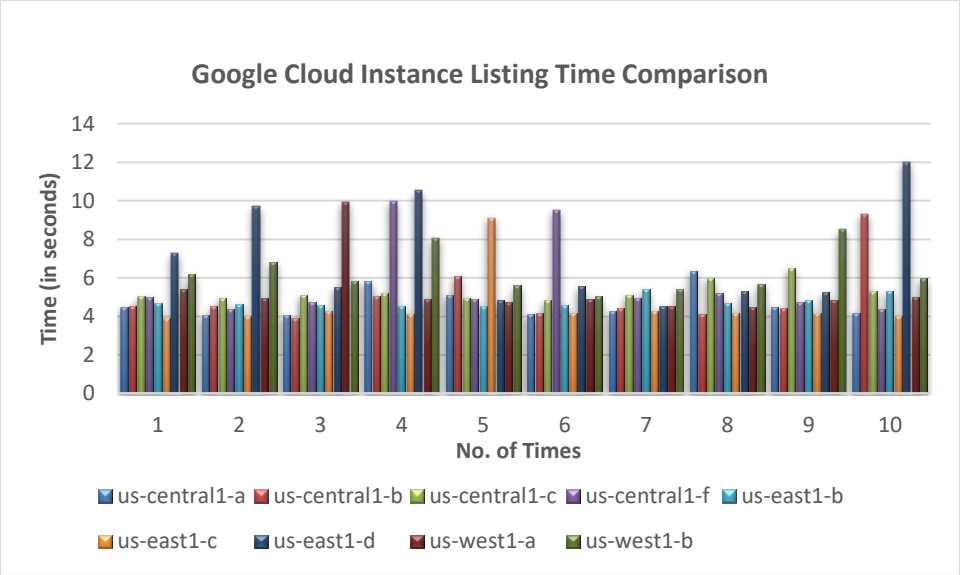

Figure 7.11 Google Cloud instances listing time comparison

In this experiment, listing times of 10 instances in each US region were recorded. The listing was done ten times for each region. Fig. 7.11 shows the time taken to list ten instances ten times and in each of the US region.

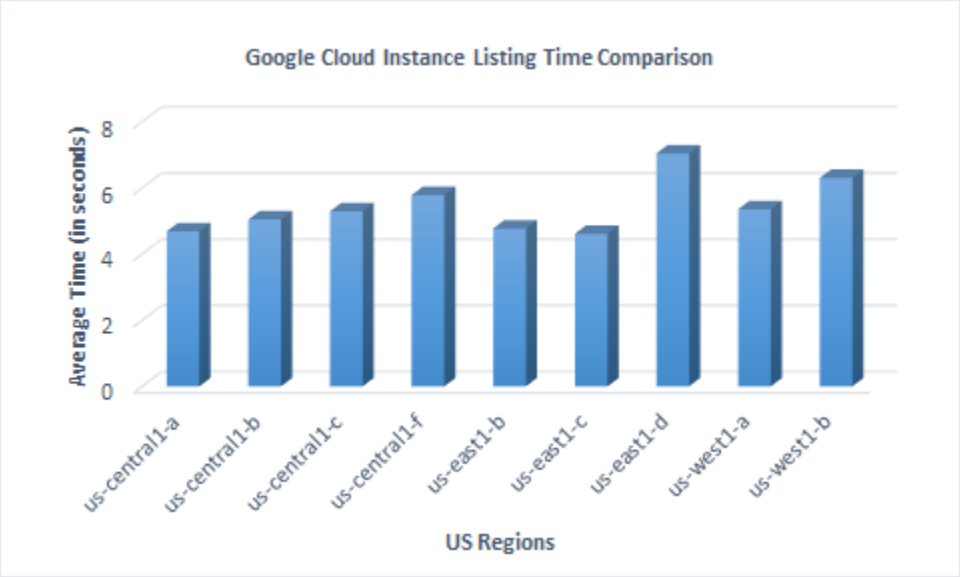

Figure 7.12 Average time for Google Cloud instance listing

Fig. 7.12 shows the average time taken to list ten instances in the US regions. ‘us-east1-c’ took the least time to list the instances.

From experiments 4, 5 and 6, we can deduce that creation, deletion, and listing of an instance is faster if it is done in ‘us-east1-c’ region on Google Cloud Platform when the Cloud Hopper is hosted in US central location. So, using ‘us-east1-c’ to maintain an instance is a better deal when the application is hosted US central location.

7.4 GCP vs. AWS Compute Experiments

In this section, a time comparison of creation, deletion, and listing of various instances on AWS and GCP in one US regions is done. The experiments 7, 8 and 9 shows this comparison in the coming sections. To conduct the experiments, for AWS,

the ‘Ubuntu 14.0’ virtual machines of ‘t2.micro’ type, in the ‘US-West-1’ region were taken and for GCP, ‘Ubuntu 14.0’ virtual machines of ‘n1-standard-1’ type in ‘US-West1-a’ were taken.

Experiment 7: AWS EC2 and GCP instance creation time comparison

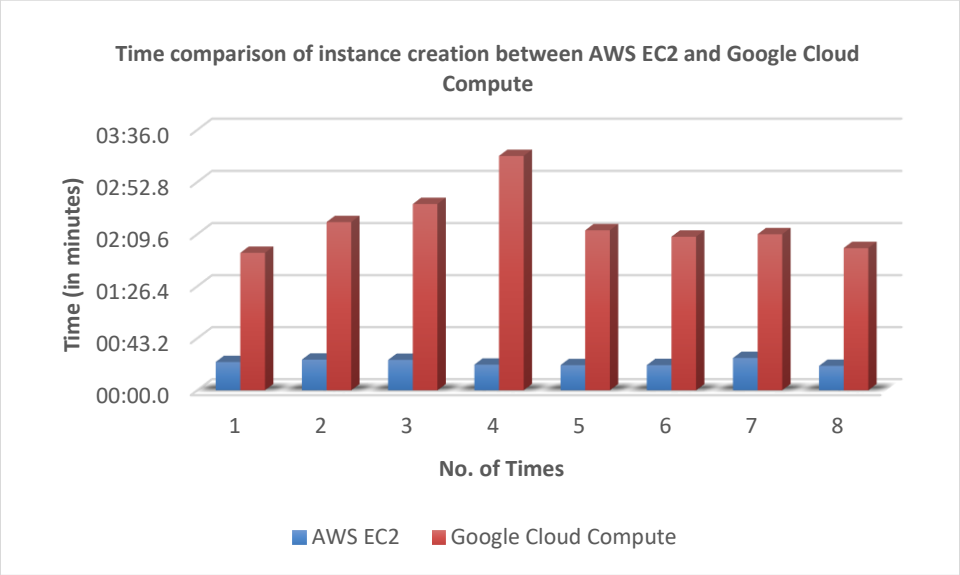

Figure 7.13 AWS EC2 and GCP instance creation time comparison

This experiment was conducted to record the creation times of 8 instances on the AWS EC2 and GCP. Fig. 7.13 clearly shows that creating an instance on AWS EC2 was always faster as compared to on GCP Compute.

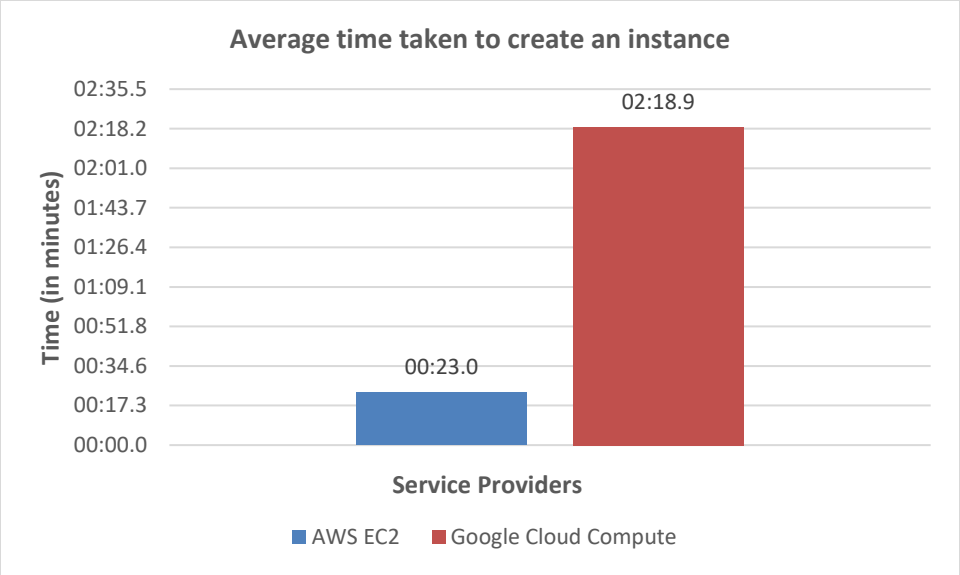

Figure 7.14 Average time for instance creation on AWS EC2 and GCP

Fig. 7.14 illustrates the average time taken to create an instance on AWS EC2 and GCP Compute. While it took average 2 minutes and 18 seconds to create an instance in 'US-West1-a' on Google Cloud Compute, it took only 23 seconds on AWS in 'US-West-1' region, when the application is hosted in US Central region. Thus, we can deduce that creating an instance on AWS is way faster than creating on GCP.

Experiment 8: AWS EC2 and GCP instance deletion time comparison

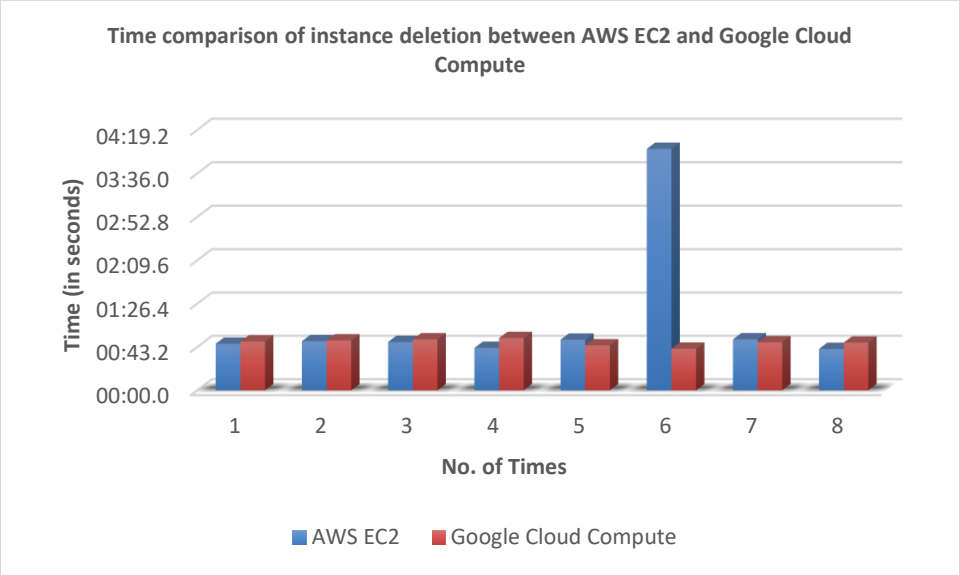

Figure 7.15 AWS EC2 and GCP instance deletion time comparison

This experiment was conducted to record the deletion times of 8 instances on the AWS EC2 and GCP. From Fig. 7.15, we can see that in all the cases apart from one, the time taken to delete an instance is almost same on both AWS and GCP.

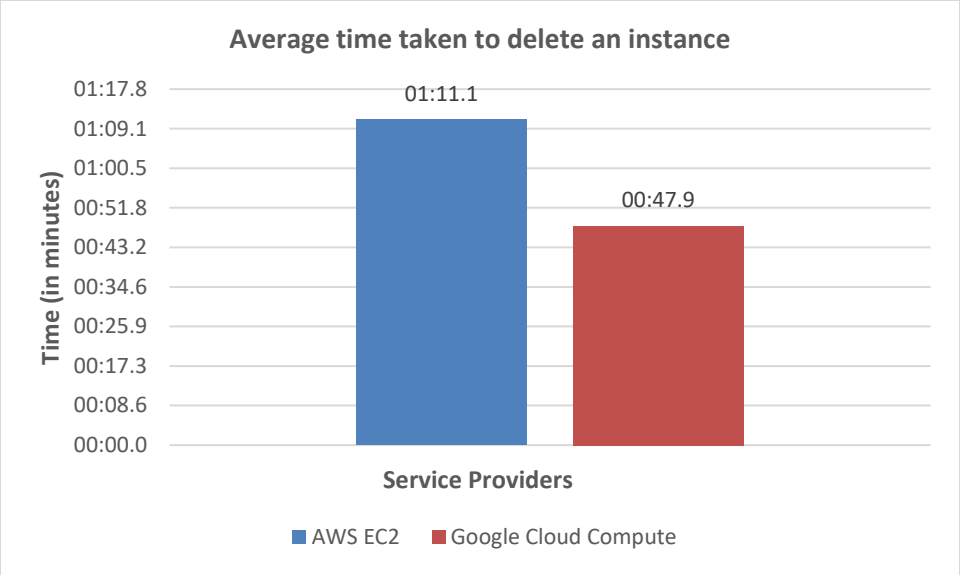

Figure 7.16 Average time for instance deletion on AWS EC2 and GCP

When the average deletion times for an instance on AWS and GCP were calculated, though deletion time was faster on GCP, there was not much difference between both AWS and GCP. Fig. 7.16 depicts the comparison.

Experiment 9: AWS EC2 and GCP instance listing time comparison

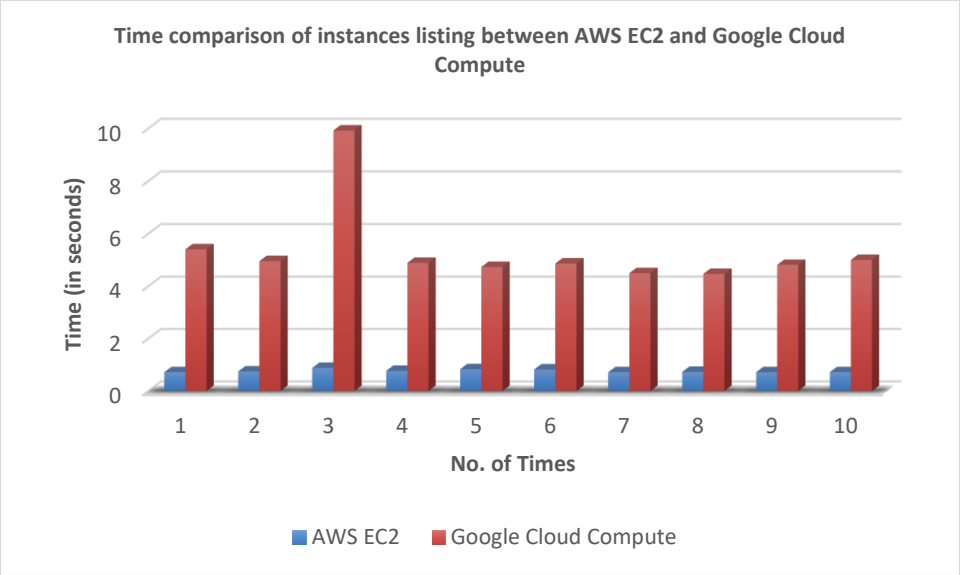

Figure 7.17 AWS EC2 and GCP instance listing time comparison

This experiment was conducted to record the listing times of 10 instances on the AWS EC2 and GCP. This experiment was run ten times. From Fig. 7.17, we can see that in all the cases, the time taken to list instances was very less on AWS as compared to GCP.

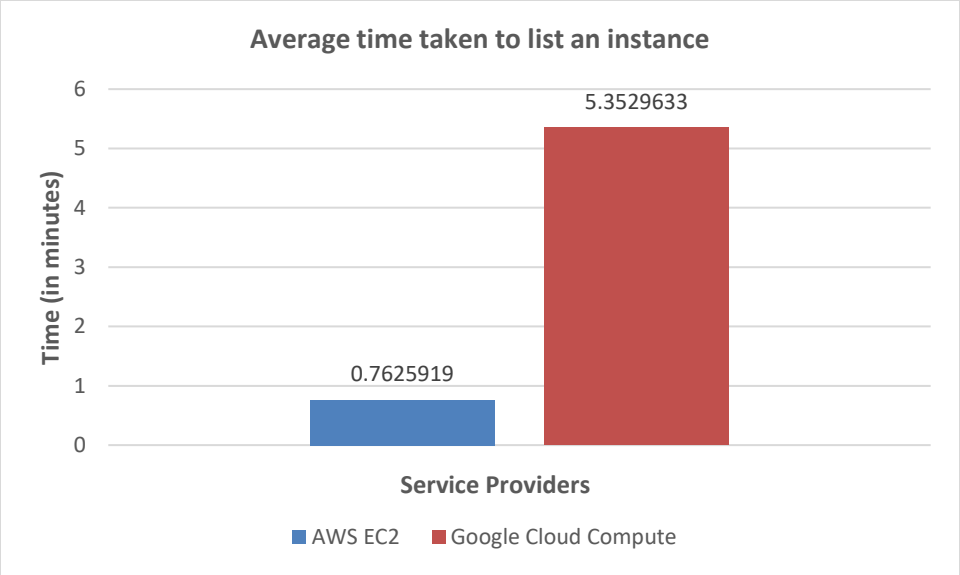

Figure 7.18 Average time for instance listing on AWS EC2 and GCP Compute

When the average listing times for an instance on AWS and GCP were calculated, listing on AWS was very fast as compared to instance on GCP. Fig. 7.18 depicts the comparison.

So, from the experiments 7, 8 and 9, we can deduce that creation and listing times for instance was faster on AWS and the deletion time was comparatively same on both AWS and GCP when the application is hosted in US Central region, a user can opt for AWS platform to save execution time.

7.5 Storage Experiments

This section shows the performance analysis of creation, deletion, and listing of buckets on AWS S3 and GCP storage. The experiments 10, 11 and 12 give an overview of each experiment.

Experiment 10: Time comparison to create buckets on AWS S3 and Google Cloud Storage

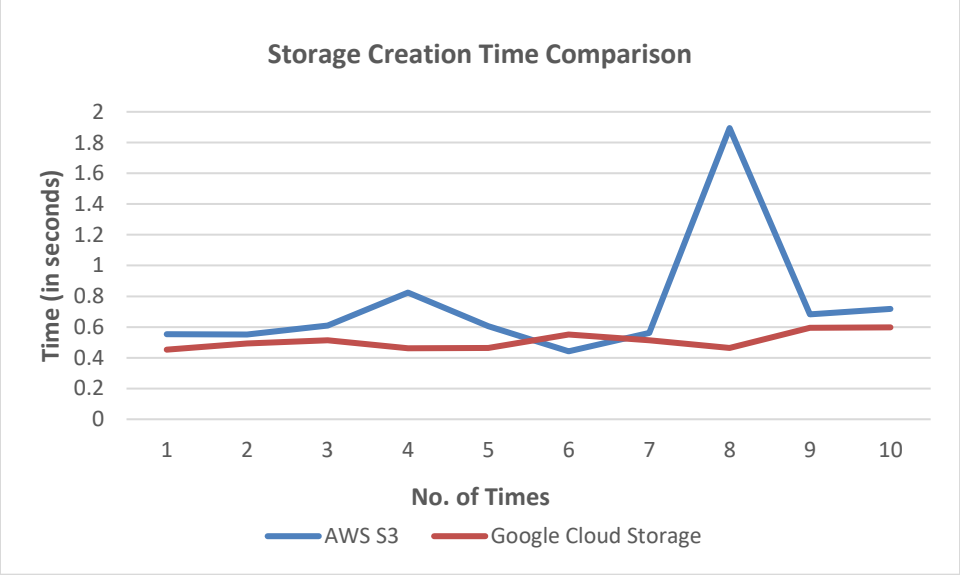

Figure 7.19 Time taken to create buckets on AWS S3 and GCP Storage

This experiment was conducted by creating ten buckets on each of AWS and GCP storages. The creation time of each bucket was recorded and plotted on a graph to see the difference in the bucket creation times on both AWS and GCP. Fig. 7.19 shows

the comparison and portrays that the creation time of bucket was almost constant for GCP and less than AWS.

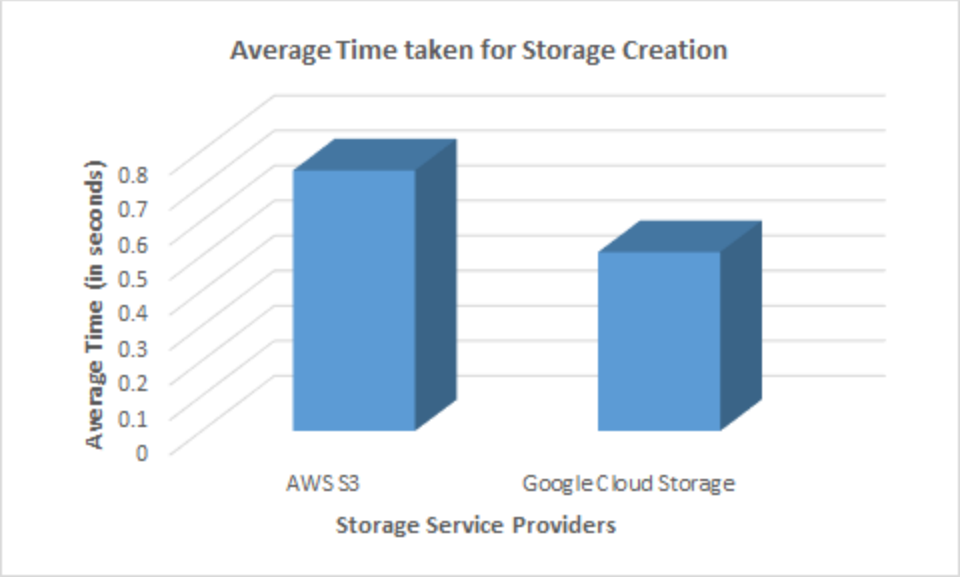

Figure 7.20 Average time for creating a bucket on AWS S3 and GCP Storage

Fig. 7.20 shows the average time taken to create a bucket on AWS S3 and GCP Storage. Thus, it shows that it is faster to create a bucket on GCP as compared to AWS.

Experiment 11: Time comparison to delete buckets on AWS S3 and Google Cloud Storage

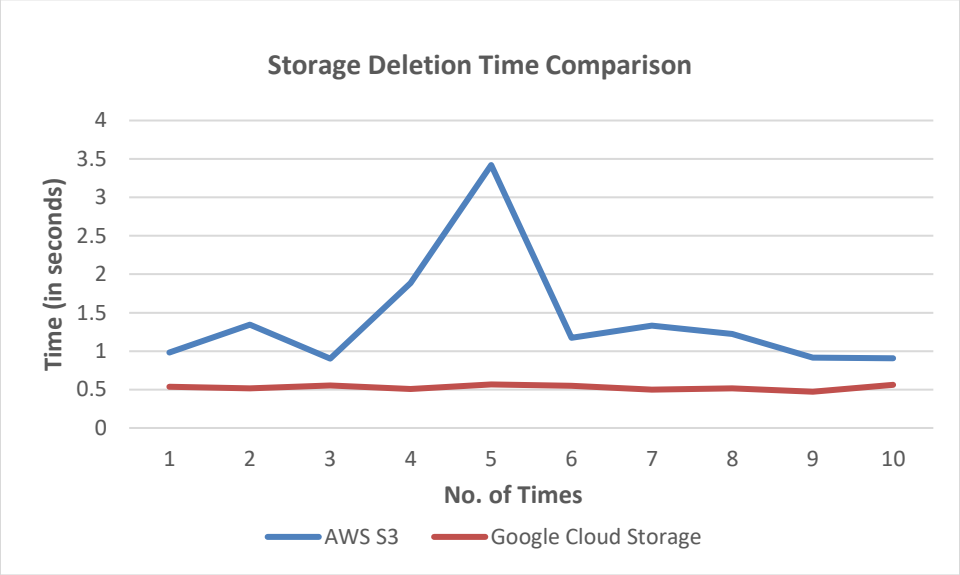

Figure 7.21 Time taken to delete buckets on AWS S3 and GCP Storage

In this experiment, a comparison of time taken to delete ten buckets on AWS S3 and GCP storage is done whose results are depicted in the Fig. 7.21. From the figure, it can easily be deduced that deletion of bucket took almost constant time every time on GCP storage. Also, the deletion was faster on GCP as compared to GCP every time.

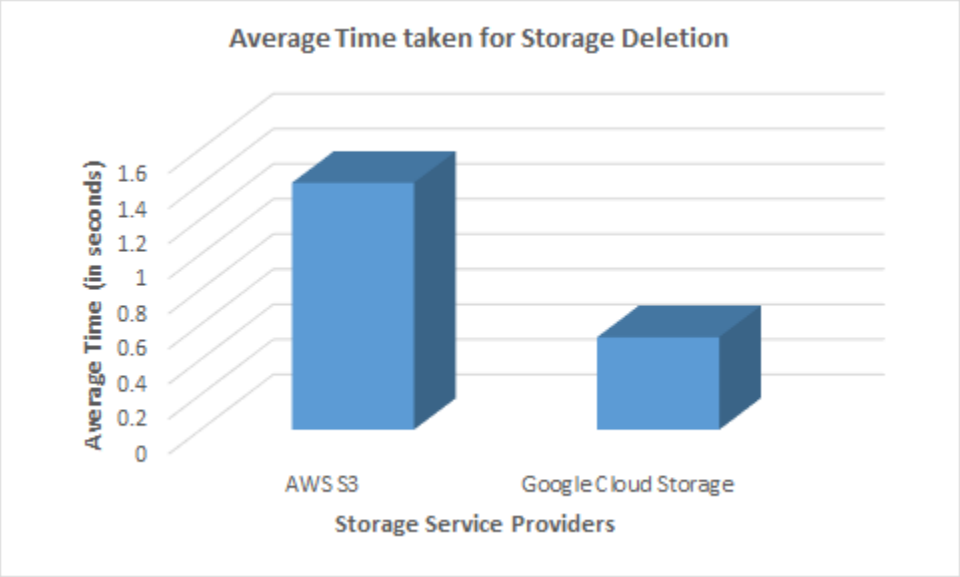

Figure 7.22 Average time for deleting a bucket on AWS S3 and GCP Storage

Fig. 7.22 shows the average time taken to delete a bucket on AWS S3 and GCP Storage. It demonstrates that it is faster to delete a bucket on GCP as compared to AWS.

Experiment 12: Comparison of time to list buckets on AWS S3 and Google Cloud Storage

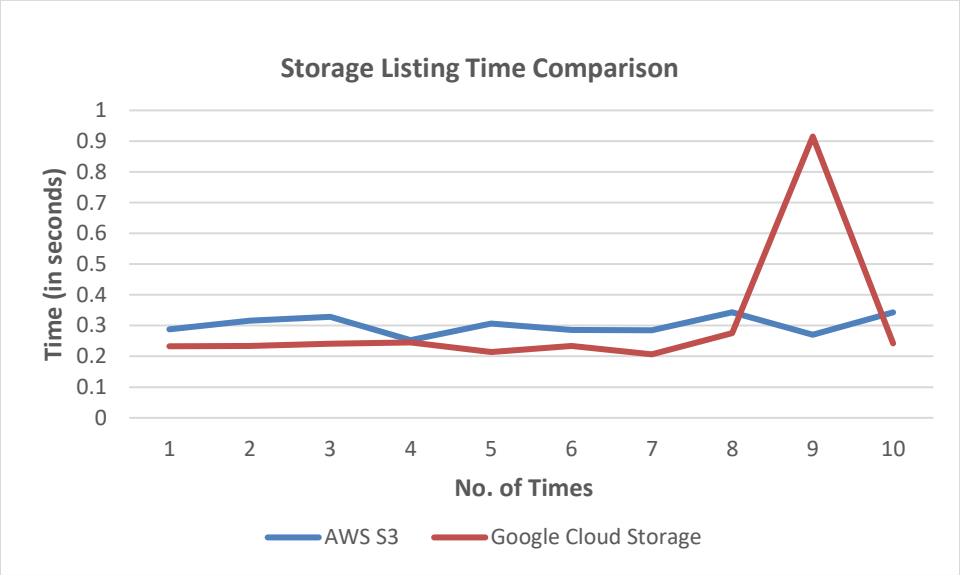

Figure 7.23 Time taken to list buckets on AWS S3 and GCP Storage

In this experiment, a comparison of time taken to list ten buckets ten times on AWS S3 and GCP storage is done whose results are depicted in the Fig. 7.23. From the figure, we can observe that listing of buckets on AWS S3 was constant and for GCP storage also it was almost constant except at one instance. Also, the time taken on both the service providers was very similar.

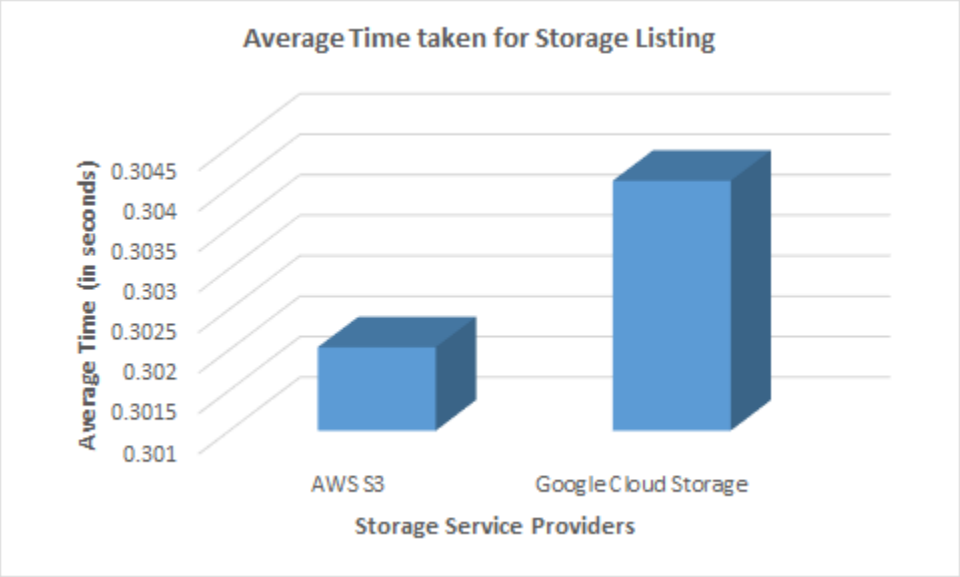

Figure 7.24 Average time for listing buckets on AWS S3 and GCP Storage

Fig. 7.24 shows the average time taken to list ten buckets on AWS S3 and GCP Storage. Both the service providers took time in the range of 3-4 seconds.

From the experiments 10, 11 and 12 we can see that performance of GCP storage was better than AWS S3 on the basic create, delete and list operations on buckets. If given a choice between AWS S3 and GCP storage, it would be better to select GCP storage while taking the time taken for each operation and the application being hosted in US Central region.

7.6 Archiving Experiments

This section shows the performance analysis of archiving of documents and folders from EC2 to storage services such as Dropbox, AWS S3, and Localhost. The

experiments 13 and 14 give an overview of each experiment. All the experiments are conducted using a 100 MB file. The file was transferred from an AWS EC2 instance and a GCP compute instance to Dropbox, AWS S3 and to the local machine.

Experiment 13: Comparison of time taken for archiving of documents and folders from AWS EC2 to Dropbox, AWS S3, and Localhost.

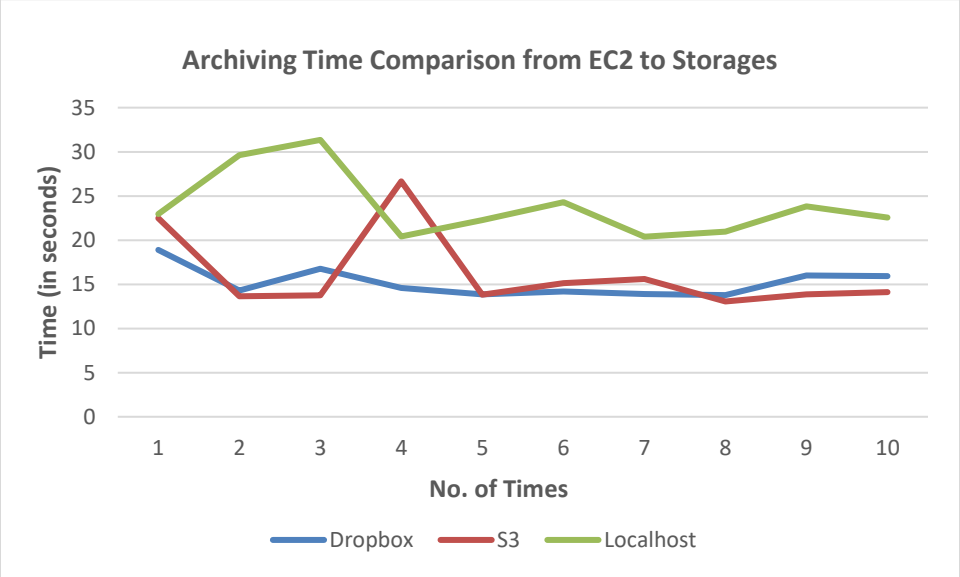

Figure 7.25 Time comparison of archiving from AWS EC2 to Dropbox, AWS S3 and on localhost

This experiment was conducted to find out the time taken to transfer a 100 MB file from an AWS EC2 instance to Dropbox, AWS S3 and the local machine. This experiment was conducted ten times. Fig. 7.25 shows the time taken each time to

transfer the file to various storage service providers. Moving to local machine took a longer time than on the cloud service providers such as AWS and Dropbox.

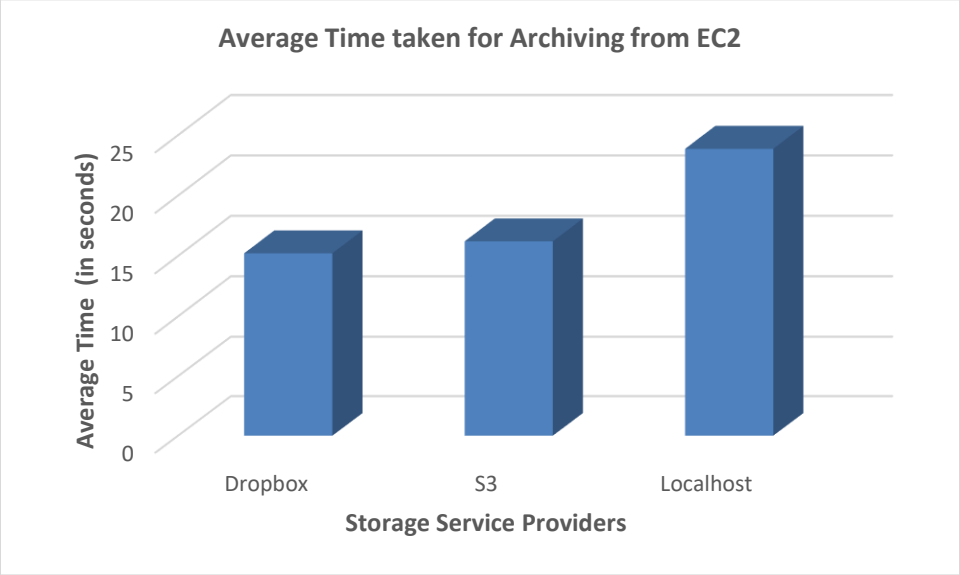

Figure 7.26 Average time comparison for archiving from AWS EC2 to Dropbox, AWS S3, and localhost

When the average time taken to archive from AWS EC2 to Dropbox, AWS S3 and localhost was plotted on the graph as shown in the Fig. 7.26, it was observed that archiving to Dropbox and AWS S3 took almost same time whereas, while archiving to local machine took longer than others. This is mainly due to the reason that the experiment to transfer from AWS EC2 to localhost was done on the home network with the upload speed of 70 Mbps and download speed of 15 Mbps. Whereas, AWS maintains

its own high-speed dedicated lease lines, so the transfer on AWS S3 and Dropbox from AWS EC2 were done using the super-fast AWS network.

Experiment 14: Comparison of time taken for archiving of documents and folders from Google Cloud compute instance to Dropbox, AWS S3, and Localhost.

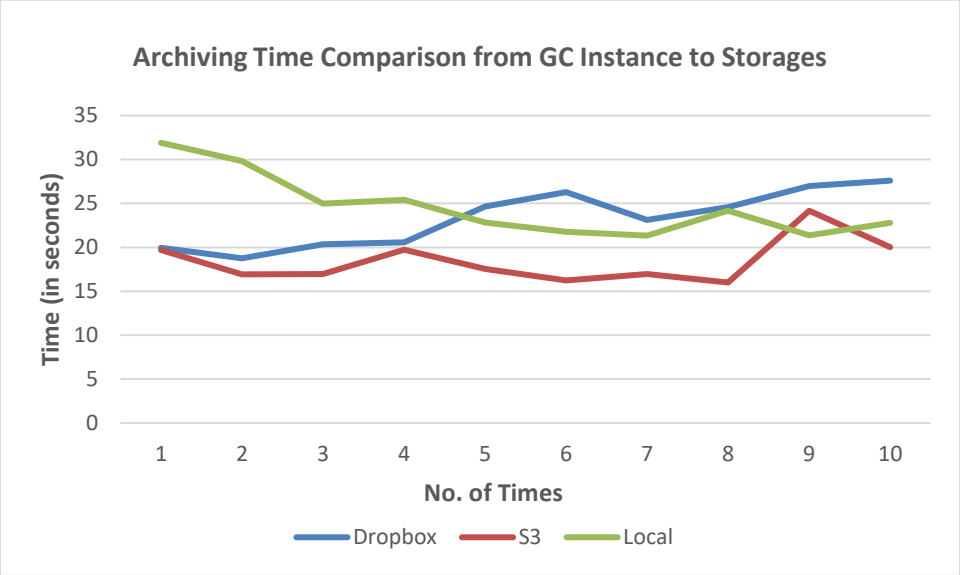

Figure 7.27 Time comparison of archiving from GCP instance to Dropbox, AWS S3 and on local machine

This experiment was conducted to find out the time taken to transfer a 100 MB file from a Google Cloud compute instance to Dropbox, AWS S3 and the local machine. This experiment was conducted ten times. Fig. 7.27 shows the time taken each time to

transfer the file to various storage service providers. Moving to AWS S3 took the least time as compared to other storages.

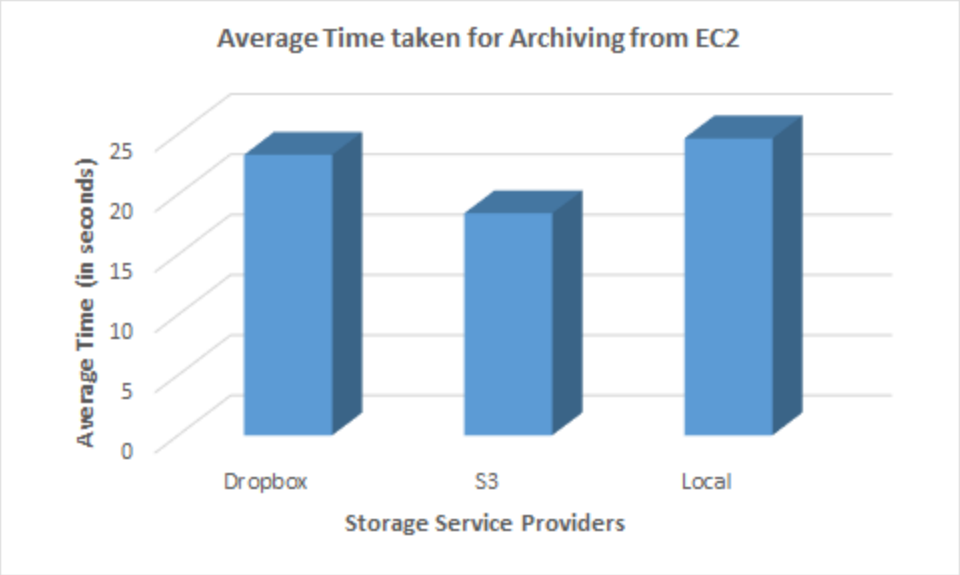

Figure 7.28 Average time comparison for archiving from GCP compute instance to Dropbox, AWS S3 and localhost

When the average time taken to archive from Google Cloud Compute instance to Dropbox, AWS S3, and localhost and plotted on the graph as shown in the Fig. 7.28, it was observed that archiving to AWS S3 took the least time whereas, while archiving to local machine took most time than others. As explained in the previous section where the AWS was maintaining its high-speed lease lines, Google Cloud Platform also maintains their own dedicated lease lines. The experiment to transfer from Google Cloud Compute instance to localhost was done on the home network with the upload speed of

70 Mbps and download speed of 15 Mbps. And the transfer on AWS S3 and Dropbox from Google Cloud Compute instance was done using the Google Cloud Platform network. Thus, it was faster on AWS S3 and Dropbox.

CHAPTER 8

SUMMARY AND CONCLUSION

This thesis started with a goal to provide a unified cloud solution to manage heterogeneous clouds from a single platform which would help enterprises who are using the portfolio of multiple clouds to save the cost of resources. The burgeoning future of multi-cloud framework strengthens the need for a reliable, cost-efficient, faster and a one-stop shop for the resources provided by multiple cloud service providers. A strong framework was presented in this thesis which can handle heterogeneous clouds. The packages were designed in such a manner that they are modular and any new module or a package can be easily integrated with the framework. This framework also provides an ability to set lifetime to the resources.

An application was developed based on the above-described framework which provided a user-friendly resources management console along with the features to schedule and monitor the individual resources as well as the workflows. An additional feature to archive the documents and folders to the various storages services was also offered. A common key store to store all the credentials in a secure manner was designed. A dashboard of the application was designed to show the data of the number of resources in use and a trend of usage of those resources in the last 7 seven days. Also, the data related to the tasks scheduling was presented there. Another screen of the application was designed to show the prices of the compute and storage services. A usage alert feature was also presented to send the timely alerts to the user. Another

interesting feature, named Slack Bot which performed interactive bot based operations was also presented.

After the implementation of application section, a detailed analysis of prices of compute and storage services was done showing the price comparison of similar services offered by different cloud providers. Alternate options for storage were suggested based on the prices. It has been demonstrated that if the resources are chosen correctly across multiple cloud providers, companies can save a lot of money.

In the later part of the thesis, experiments and analysis were conducted to compare the performance of various services offered by multiple cloud providers. The comparison helps in deciding on the services provided by the cloud providers which can be used to design the infrastructure of an application in an optimal manner.

Heterogeneous cloud computing will remain a major area of research as most of the enterprises are moving towards multi-cloud ecosystem and Cloud Hopper can be an efficient multi-cloud management tool for them which in future can be enhanced with the upcoming technologies.

CHAPTER 9

FUTURE WORK

So far in this thesis, integration of IaaS and SaaS services has been done with the framework of Cloud Hopper. The framework is designed in such a way that in future, PaaS services can be easily integrated. A new PaaS package can be added to the 'cloud_resource_lib' library to leverage the services provided by the PaaS providers such as Heroku and AWS Elastic Beanstalk.

The prototype is built on the services offered by the public cloud service providers such as Amazon Web Services and Google Cloud Platform. The services provided by the private cloud built on the platform such as OpenStack can also be integrated by simply writing the modules which can be plugged into the Cloud Hopper framework.

In this thesis, the prototype was built by using the compute services offered by AWS and GCP, storage services provided by AWS and GCP and the SaaS providers such as Dropbox and Google Drive. There are a lot of services offered by the cloud providers which can be studied and can be integrated with the application framework. The implementation can be enhanced to services such as Databases, and Containers. This can also be achieved by simply creating modules for each service type and plugging them in the correct packages of Cloud Hopper.

For archiving of documents from the compute instances on AWS and GCP to storage services, only the storage services offered by IaaS service providers such as AWS S3 and SaaS service providers such as Dropbox and Google Drive was

implemented in the prototype. Exploration of other storage service providers and integrating them into the framework will give more options to the users to choose.

Another area to consider is the real-time analysis and comparison of prices offered for different services by various cloud providers. This will help the users in comparing and choosing the best options available across multiple clouds in their budget.

REFERENCES

[1] Behrend, T. S., Wiebe, E. N., London, J. E., & Johnson, E. C. “Cloud computing adoption and usage in community colleges.” Behaviour & Information Technology, 2011, 30(2), 231-240.

[2] I. Cloud, "What is cloud computing?," 2016. [Online]. Available: https://www.ibm.com/cloud-computing/learn-more/what-is-cloud-computing. Accessed: Dec. 5, 2016.

[3] Microsoft, "Office 365 for business," 2016. [Online]. Available: https://products.office.com/en-us/business/get-latest-office-365-for-your-business-with-2016-apps?wt.srch=1&wt.mc_id=AID522514_SEM_5SOqeNbs. Accessed: Dec. 5, 2016.

[4] I. Bojanova and A. Samba, "Analysis of Cloud Computing Delivery Architecture Models," 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, Biopolis, 2011, pp. 453-458.
doi: 10.1109/WAINA.2011.74
keywords: {Internet;cloud computing;software architecture;Internet;cloud computing delivery architecture models;cloud computing infrastructure metrics;cloud service model architectures;Cloud computing;Computational modeling;Computer architecture;Google;Hardware;Servers;Software;Amazon EC2;Cloud Computing;Google Apps;IaaS;Microsoft Azure;Oracle Fusion;PaaS;SaaS},
URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5763543&isnumber=5763390

[5] Peter Mell and Timothy Grance (September 2011). The NIST Definition of Cloud

Computing (Technical report). National Institute of Standards and Technology: U.S. Department of Commerce. doi:10.6028/NIST.SP.800-145. Special publication 800-145.

[6] Microsoft (2016). Microsoft azure: Cloud computing platform & services. . Retrieved from https://azure.microsoft.com/en-us/?b=16.48

[7] Software » OpenStack open source cloud computing software. Retrieved December 5, 2016, from https://www.openstack.org/software/

[8] 19— (2016). Cloud computing. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Cloud_computing

[9] Summary of the Amazon EC2, Amazon EBS, and Amazon RDS service event in the EU west region. (2016). Retrieved December 5, 2016, from https://aws.amazon.com/message/2329B7/

[10] Laing, B. (2012, September 3). Summary of windows azure service disruption on Feb 29th, 2012. Retrieved from https://azure.microsoft.com/en-us/blog/summary-of-windows-azure-service-disruption-on-feb-29th-2012/

[11] Cloud industry insights. (2006). Retrieved December 5, 2016, from http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-surveyFebruary18, 2015, Posted by Kim Weins

[12] Bishop, T. (2014, November 5). InterConnections – the Equinix Blog. Retrieved December 5, 2016, from https://blog.equinix.com/blog/2014/11/05/and-the-winner-is-the-multi-cloud/#!prettyPhoto

[13] Chen, Y., & Huang, X. (2013). Cloud-based resource scheduling management and its application - with agricultural resource scheduling management for example. Advances in Information Sciences and Service Sciences, 5(4), 191. Retrieved from https://login.ezproxy.uta.edu/login?url=http://search.proquest.com.ezproxy.uta.edu/

docview/1620059831?accountid=7117

[14] Chen, X., Zhang, Y., Zhang, X., Zheng, X., Guo, W., & Chen, G. (2013). Architecture-based integrated management of cloud resources. IEEE Conferences, 474-481. doi:http://dx.doi.org.ezproxy.uta.edu/10.1109/CLOUDCOM-ASIA.2013.45

[15] Geng, Y., & Kou, J. (2012). Construction of heterogeneous data integration model on cloud computing. Journal of Jinan University / Science and Technology, 26(4), 384-389. Retrieved from https://login.ezproxy.uta.edu/login?url=http://search.proquest.com.ezproxy.uta.edu/docview/1136337071?accountid=7117

[16] Do, C. T., Tran, N. H., Tran, D. H., Pham, C., Alam, M. G. R., & Hong, C. S. (2015). Toward service selection game in a heterogeneous market cloud computing. Paper presented at the 44-52. Retrieved from https://login.ezproxy.uta.edu/login?url=http://search.proquest.com.ezproxy.uta.edu/docview/1695350083?accountid=7117

[17] Blog, D. (2015, October 15). Mist.Io: Cloud management in your pocket: Retrieved December 5, 2016, from https://mist.io/

[18] AWS cost management tools. Retrieved December 5, 2016, from https://www.gorillastack.com/

[19] MultCloud. (2016). Manage, move, copy, and migrate files between cloud storage services with MultCloud. Retrieved December 5, 2016, from https://www.multcloud.com/

[20] ACCELLION. (2014, April 2). Operational and security risks of public cloud subscriptions. Retrieved December 5, 2016, from http://www.accellion.com/blog/cloud-sprawl-operational-and-security-risks-public-cloud-subscriptions

[21] Cohen, R. (2013, April 16). The cloud hits the mainstream: More than Half of U.S. Businesses now use cloud computing. Forbes. Retrieved from http://www.forbes.com/sites/reuvencohen/2013/04/16/the-cloud-hits-the-mainstream-more-than-half-of-u-s-businesses-now-use-cloud-computing/#7987560e67c2

[22] Wilson, D. (2014, March 18). Cloud 'sprawl' causing business inefficiencies. . Retrieved from http://www.techradar.com/news/internet/cloud-services/cloud-sprawl-causing-business-inefficiencies-1234913

[23] Welcome to python.org. (2016, December 4). Retrieved December 5, 2016, from https://www.python.org/

[24] Welcome to python.org. (2016, December 4). Retrieved December 5, 2016, from https://www.python.org/

[25] 2, to A. E. (2016). Elastic compute cloud (EC2) cloud server & hosting – AWS. Retrieved December 6, 2016, from https://aws.amazon.com/ec2/

[26] Introduction to Amazon S3 (2016). Retrieved from https://aws.amazon.com/s3/

[27] Cloud storage - online data storage | Google cloud platform. Retrieved December 6, 2016, from https://cloud.google.com/storage/

[28] Dropbox. Retrieved December 6, 2016, from https://www.dropbox.com/

[29] Google drive - cloud storage & file backup for photos, Docs & more. Retrieved December 6, 2016, from https://www.google.com/drive/

[30] Slack. Slack: Be less busy. Retrieved December 6, 2016, from https://slack.com/

[31] Rackspace. (2016). Transactional Email API service for developers by Rackspace. Retrieved December 6, 2016, from https://www.mailgun.com/

[32] Canvas: Designing work-flows — celery 4.0.0 documentation. (2009). Retrieved December 6, 2016, from

http://docs.celeryproject.org/en/latest/userguide/canvas.html#chains

[33] Palach, J. (2016). Understanding celery's architecture [Book]. Retrieved from https://www.safaribooksonline.com/library/view/parallel-programming-with/9781783288397/ch07s02.html

[34] Hat, R. (2016). About modules — Ansible documentation. Retrieved December 6, 2016, from http://docs.ansible.com/ansible/modules.html

[35] Redis. Retrieved December 6, 2016, from https://redis.io/

[36] MongoDB. (2016). MongoDB for GIANT ideas. Retrieved December 6, 2016, from https://www.mongodb.com/

[37] Hat, R. (2016). Playbooks — Ansible documentation. Retrieved December 6, 2016, from http://docs.ansible.com/ansible/playbooks.html

[38] Simple mail transfer protocol (2016). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[39] Rackspace. (2016). Transactional Email API service for developers by Rackspace. Retrieved December 6, 2016, from https://www.mailgun.com/

[40] MongoDB. Introduction to MongoDB — MongoDB manual 3.2. Retrieved December 6, 2016, from https://docs.mongodb.com/v3.2/introduction/

[41] EC2 instance pricing – Amazon web services (AWS). (2016). Retrieved December 6, 2016, from https://aws.amazon.com/ec2/pricing/on-demand/

[42] Google cloud platform. Retrieved December 6, 2016, from https://console.cloud.google.com/

[43] Cloud storage pricing – Amazon simple storage service (S3) – AWS. (2016). Retrieved December 6, 2016, from https://aws.amazon.com/s3/pricing/

[44] Retrieved December 6, 2016, from https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/v

mware-controlling-virtual-machine-sprawl-white-paper.pdf

[45] Christie, T. Django REST framework. Retrieved December 7, 2016, from http://www.django-rest-framework.org/

BIOGRAPHICAL INFORMATION

Shraddha Jain joined the University of Texas at Arlington in spring 2015. She received her Bachelor's Degree in Computer Science from Uttar Pradesh Technical University, Lucknow in 2010. She worked as a senior systems engineer with Infosys, Ltd., Bangalore in India until December 2014, after which she decided to pursue her Master's Degree in Computer Science at University of Texas at Arlington. She interned at Cloud Elements as a software developer at Dallas, Texas.

Her current research interests are in the areas of cloud computing, data mining, big data technologies, DevOps and software development.