COST OPTIMIZATION IN AWS

by Aishwarya D

Submission date: 13-Dec-2021 01:30PM (UTC+0530)
Submission ID: 1728959759
File name: IEEE_Paper.pdf (1.07M)
Word count: 4685
Character count: 23799

COST OPTIMIZATION IN AWS

Celin Mary
Computer Science
PES University
Bangalore, India
marycelinjoby@gmail.com

D Aishwarya
Computer Science
PES University
Bangalore, India
aishdharni26@gmail.com

Swarup Banik
Computer Science
PES University
Bangalore, India
swarupbanik28@gmail.com

Varshini G
Computer science
PES University
Bangalore, India
varshinigunaranjan@gmail.com

Prof.Sushma.E
Department of Computer Science
PES University
Bangalore, India
sushmae@pes.edu

***Abstract*—The study involves different cost strategies and methods involved to benefit the organizations, companies which tend to use the cloud services predominantly due to its flexibility and scalability. Yet it has its own performance and risks involved which has to be viewed as a whole architecture of cloud. Cloud services may look simple and open to all but the costs behind them benefit the organizations only when it's measured and optimised based on their needs regularly as the pricing system of these keeps fluctuating and getting updated monthly and yearly. Bringing up techniques to reduce the cost or optimising the models based on cost structure is the goal of the project work addressed here. The study not only deals with some techniques to optimise cost with time but also has a detailed view on others' work and approaches used along with their drawbacks to fit our solution effectively.**

***Index Terms*—Optimization, AWS, Cloud, Pay-as you go, Instances, Scalability**

I. Introduction

Cloud Technology is the new buzz in the market which is gaining popularity among all kinds of businesses, enabling them to access application software over high internet connection without the need for investing in software or hardware. The growth of cloud over last few years has been exponential and enterprises are looking forward to ho their data on the cloud instead of private servers. It works on the principle of web-based email clients, allowing users to access all features of the system without having to store data on their own systems.

Cloud services are **pay-as-you-go**. Everything on cloud has easy access to company's data which saves time and money in project start s. If you are not taking advantage of what cloud is offering, then at least you won't be wasting money over it. Same is applied when it comes to data storage, you will be charged for the space you need. Pooling of resources in cloud computing offers great benefits to business organisations. It helps in achieving efficiency as cloud facilitates access of applications and data from any part of the world just with an internet connection. Since there is resource pooling and many number of users are sharing these resources, it helps in cost savings. It is also flexible, helps in scalability for the companies with fluctuating workloads. While working in cloud it is also important to optimize AWS bills while running your workloads. While adopting the cost optimization methods it is also important to keep the performance and capacity you require. There should not be any loss in productivity, reliability or efficiency.

It is not unusual to read headlines claiming businesses are overspending in the cloud, that a double-figure percentage of money is being wasted on unused services, or that millions of business provision resources with more capacity than they need. AWS offers solutions to these problems, but it's important to understand and implement services with cost optimization in mind for the betterment of business.

The main goal of our research is to address the question of how to take care of the cost-effective compute services and pricing options that gives the ability to effectively manage costs while still maintaining high performance and complete business requirements using AWS.

II. PROBLEM DEFINITION

As we come across day to day cloud based issues it has come to notice that cost and the reports regarding usage, being a bigger impact for all the common users and companies. There are several challenges due to this and there's always a need to overcome it, since everything today is over the cloud as it is simple, feasible and it helps to save a significant amount of money and it does not require any investment on hardware or software to work with, it's much more convenient. Cloud services are like open service providers which help you to store 10 folds of the data on an infrastructure which is generated enormously by various organisations and companies. But when the challenges are discussed, it's observed that there are

drawbacks to all these services which are granted by the cloud and need to be at least taken care of or minimised by certain methods and strategies without affecting the performance and efficiency. Previously a finance system was used to set an overall fixed budget and later the resources were given to use, no purchase was made until everyone had the opportunity to meet across the company on whether or not it was needed. But now the cloud has made the work much easier to handle and make payments. Our research focuses on the services provided by AWS and mainly on S3, EC2, RDS etc and the pricing models of these individually which gives a rough idea about the cost structure so that it is analysed and taken care of eventually. The key drivers of cost would include storage, bandwidth used, computation , transfer of data and also the pricing model you wish to choose. AWS helps you with spend and bills with their cost optimisation tools which tracks all the usage. To prioritize the cost we need to look after two things in hand : One is savings and other is about approximate work to be done to recommend a ***cost optimization technique***. Based on the technique referred, the user can have a rough idea about savings on each of the services. This directly cannot lead to an accurate amount of savings but it depends on the work chosen to be spread over cloud platforms. To work on this, every organisation must know their basic requirements initially, so that they are allocated with the best of the resources and the right service on the go and they can pay as they consume it. It's always necessary to maintain business requirements but not to spend too much on the expenditure. This is the problem we are hoping to solve in our research.

A. Overview of the research

The entire goal of the project is to understand how cloud is being used, it's components, it's services, how the bills are being charged, how reports are analysed, how instances are handled etc, and how to optimise cost on AWS, how industries, companies and organisations can try overcome these challenges slowly. All this works on one policy pay-as you go over cloud. Cloud benefits are which give users a clear overview of their data, how it's being utilised in the environment which ensures more visibility as each of the services is tracked and independently measured as you go. It becomes hard to figure out when you have tons of data flowing and analysing how they are bought, who is purchasing those services and the whole ownership cost for the cloud, and this issue is when suddenly users or an organisation switch from normal premises to cloud infrastructure.

B. A few basic approaches

The simplest way to get away with all cost driving bill elements are to make sure idle instances are turned off and are no longer in use making sure that the data is also not lost while shutting them instantly, for this we might need to store the data until you completely delete it or remove, or take a snapshot of it beforehand until you require it again. Another way is to have auto start and stop for VM's which saves a lot of expenses in the long run as the VM's are to be used only during the development phase and do not require to be running throughout the day. Next comes right sizing the instances according to your needs which is discussed in most of the cost optimization techniques, as it is the most common method to resolve cost and make customization to CPU Ram usages. Our basic initiative to this is to create a web page and maintain all the scripts that optimise cost and ***maintain feasibility***, which should try to solve and make sure all the cloud cost is optimised to an extent.

III. Related Work

A. Fundamentals of AWS pricing

Steven Noble & Oliver Berger [1] have drawn an overview on the ability of cloud services to optimize cost to match your needs, even with the change in needs over time. AWS services helps you to build sophisticated applications by paying for the individual services you consume. Hence it is important to understand how AWS pricing works across various services.

The work describes how individual services are charged, different services use different pricing models. When you choose to a run virtual server in cloud using Amazon EC2, you have three pricing models to choose, on-demand, reserved or spot. Most people usually start with on-demand pricing simply because it's very simple to use and is charged based on an hourly basis depending on the instance type you choose. There is no long term commitments or upfront payments required in this pricing model. This model is useful when you can't actually predict the workloads of your business, you can either scale up or down the compute capacity depending on your workload needs which makes this model flexible. Reserved instances are charged based on compute capacity within a time frame. It's like a commitment to run an instance with specific compute capacity for a certain period, usually one to three years. Since there is a commitment, amazon will give a discount price.

When it comes to spot instances, Amazon's data centers will have extra processing powers ,they are not all at 100% capacity at 100% of the time, so this extra processing power is going idle. So, these unused resources can be requested at sleep discounts, you can actually decide your hourly price for the instance. But once any other customer requests for the same compute capacity for higher price than yours, your instance will automatically cut out. Therefore this pricing model should be used in case of non-critical data.

B. An optimisation of simulation execution on Amazon EC2 spot market

Another work which represents cost optimization on EC2 service. This paper mainly focuses on spot pricing mechanisms apart from the other two methods of pricing - such as on demand and reserved instances. Bogumił Kamiński *, Przemysław Szufel [2] have proposed an algorithm for cost and time optimization for computing on cloud clusters, which can be used practically in real time, it's implemented using python which is now openly up for optimizing Amazon EC2 instance pricing model. The paper deals particularly with end

user perspective only eliminating all the drawbacks from other approaches and attempting for a practical use for bidding on EC2. They have used a framework which compares different strategies of bidding from other referred work which was discussed in this paper and they have limited their research work by using virtual machines which are rented out over the cloud.

An EC2 cloud simulator was implemented as a tool to analyze different strategies which uses the boto library of amazon for connecting with ec2 infrastructure of bidding and they use two types: static fixed bid method and adaptive bidding. The drawback of this work is that due to constant changes in amazon pricing policies the approaches which are inferred from this paper may or may not be applicable although it's a real life approach and has been tested statistically.

C. SuMo Analysis and Optimization of Amazon EC2 Instances

This research work [3] developed a tool known as SuMo which helps in increasing the performance and utilization of resources making sure that cost is minimized which still offers the same load of work. The tool provides resource utilization solutions by analyzing and bringing together all the data from AWS. They have gone through the different methodologies between private and public cloud to figure out the differences between the parameters in use, for public cloud the main focus is on resource utilization and its cost as it's directly proportional to the work effectively, where as in private cloud the access is completely given to the administrator. The paper deals with some algorithms which includes profiling of resources and spike detection and resizing of instances or resources. Further the paper takes through amazon cloud watch which discusses some common metrics such as CPU utilization, Disk read and write operations , Disk read and write bytes and Network in/out. Each of these individual metrics are addressed at a time by the cloudwatch. Since the paper mainly talks about public cloud analysis they have discussed management decisions such as spike detection, Profiling, Workload consolidation, Resource resizing and Planning. The results of using CUO in their work have few drawbacks in giving the 100% output, such as a number of upgrades that chooses higher capacity instances which leads to lower utilization of resources or a particular resource.

D. Statistical analysis of Amazon EC2 cloud pricing models

In this paper [4] they have performed statistical analysis on two methods of pricing that are on demand and on spot pricing of instances. On demand can only be stopped by the user with high availability and spot prices are on spot driven and they are stopped when it goes above the initial user defined price which has low availability. Their research led to multiple linear regression equations that brings an idea of costs available on ec2 instances . For on spot they have used time smoothed moving averages to analyse the prices statistically. According to other researchers from the paper's point of view it's found that EC2 pricing models mostly take care of on demand and spot pricing only which try to explain the gaussian (mog) distribution [5] although there is no significant approach or any such distribution which fits completely to determine the pricing model as a whole. They have compared 2014 and 2016 data based on the prices from the history and estimated for almost six different rented vm's or instances, the on demand instances look like they are mostly depending on RAM and capacity, this was discovered by using multiple regression The work also performed linear and multiple regression for on demand pricing of instance and have ended up restricting to only compute-optimised and memory-optimised types of instance then the statistical analysis of this is summarized in a table format for comparing with on spot instances later.

E. Cost optimization on Amazon S3 Pricing model

Mayur Palankar & Adriana Iamnitchi [6] have mainly focused on Amazon Simple Storage Service pricing structure. There is an increase in the need of storage space which lets the organisations switch from their storage space to s3 buckets. With the network technologies getting advanced and the sudden increase in the need of storage space has encouraged the majority of the companies to outsource their storage needs. The cloud storage providers provide a way that intends to conceptualize the complexities. There may be some literature work where they have raised concerns about security i.e about privacy, about availability. The Amazon Simple Storage Service [7] uses four usage patterns which are as follows: i)The usage pattern used here for the Amazon Simple Storage Service is that it can be used to store the data from the website itself and as we know that each object in the amazon simple storage service has a unique hypertext transfer protocol-uniform resource locator, therefore the stored-content will be delivered directly from the simple storage service. ii) To host an inert website completely we can use the simple storage service, the main reason behind this is that the amazon simple storage service is of low-cost(as compared to other services) and it is also vastly available iii) Apart from the storing of the web documents or hosting an inert website the amazon simple storage service can also be used as the data storing for analysing and computing large scale data which often is financing or converting a file from one format to the other, the reason behind this is that a user can access the data from many connecting devices and doesn't need to depend on a single connecting device. iv)The Amazon simple storage service offers services which usually have an extremely durable and provides a safe backup for perilous data.

IV. PROPOSED METHODOLOGY

A. Basic Requirements

Since the project involves more of a research work , the study deals with only a minimum amount of requirement to work with and build a script for effective cost optimisation

- To control multiple AWS services from the command line and automate them through scripts we have made use of AWS command line interface (CLI)
- The languages used are python and R for automation.

- Amazon Cloudwatch [8] helps to set alarm or threshold which is set to measure the various metrics captured in AWS.
- The Service dashboard uses Shiny Package
- To Solve Linear problems the LpSolveAPI library has been used.
- For Automation we have made use of Boto3 library

B. Implementation of various services

Writing scripts to reduce the costs and which will help in managing the following services :

1) Idle EC2 instances : To Identify and stop idle EC2 instances based on CPU utilization and bandwidth used. EC2 instances are considered idle when the average CPU utilisation is less than 2% and average network I/O is less than 5MB for the last 7 days.
2) Idle RDS : To identify RDS instances which have no connection in the last 7 days. The criteria to detect idle rds is when the average number of database connections has been less than 1 for the last 7 days
3) Underutilised EBS volumes :To identify idle EBS volume (unattached or had less than 1 IOPS per day for last 1 week) The main criteria for detecting underutilized ebs volumes is as follows:
   - Based on Usage

   Severely Underutilized: Avg Used % < 35% Moderately Underutilized: Avg Used % >= 35% and < 50%
   - Based on Read Throughput

   Severely Underutilized: Avg Read Ops % < 20% Moderately Underutilized: Avg Read Ops % >= 20% and < 50% Write Throughput Severely Underutilized: Avg Write Ops % < 20% Moderately Underutilized: Avg Write Ops % >= 20% and < 50%
4) Legacy Instances: To identify if Legacy instances are in use and new generation instances should be used over previous generation instances.
5) Unassociated Elastic IP : To identify unassociated Elastic IP which is not associated with running EC2 instances. The reserved public IP's are assigned to EC2 instances and given a particular region, the user can choose to release it explicitly by detecting any unassociated IP. The IP address remains unaffected even if the instances go through changes like stopping, starting and restarting
6) RI recommendations (Reserved instances) : It is a simulation service for deriving the cost-optimized configuration of AWS EC2 instances using Reserved Instances. Currently, AWS provides five kinds of reserved instances pricing policies, which is cost efficient at most 77% compared with on-demand instances. If you input your expected usage of AWS EC2 instances, you can get the cost optimized RI configuration. The project is written in R . Shiny package is used for the service dashboard. LpSolveAPI library is used for solving linear problems.

The output of the scripts are provided in a csv format with all the details of the services and region and will indicate extra wastage of AWS services. Ex : Instances , EBS block, snapshots Elastic IP, RDS instances etc. Providing us the automatic (such as direct closing of instances) and semi automatic approaches (such as decision on budgets and managing costs based on the service)

C. Working of each service

1.To stop idle EC2 instances :

i) Check all regions and loop through all the regions in AWS and check for running EC2 instances

ii) Connect to cloudwatch We need to connect to cloudwatch to fetch the metrics of CPU utilization and network IO of the individual EC2 instances.

$$c = boto3.client('cloudwatch', region\_name == region)$$

iii) After getting the statistics from the metrics, compare the CPU utilization and memory IO of each individual ec2 instances with the minimum CPU utilization and network IO *if average < minimum and sum1+sum2 < 52428800:*

$$print("About \quad to \quad stop \quad instance", i['InstanceId'])$$

iv) Stopping the underutilized instances:

After comparing with the minimum values, if the instances is found to be underutilized, stop the ec2 instance.

$$inst = ec2res.Instance(i['InstanceId'])$$

$$inst.stop()$$

2. To detect idle RDS instances with no connection in the last 7 days:

i) Create a dataframe :

Region Name, RDS Name, Connections in last 7 Days, Status

ii) Connect to cloudwatch

$$mon = boto.ec2.cloudwatch.connect\_to\_region(str(r.name))$$

$$con = boto.rds.connect\_to\_region(str(r.name))$$

iii) Listing all the database instances

$$dbins = con.get\_all\_dbinstances()$$

If not in dbins :

Data = No database instance, 0, idle

iv) Writing to csv file

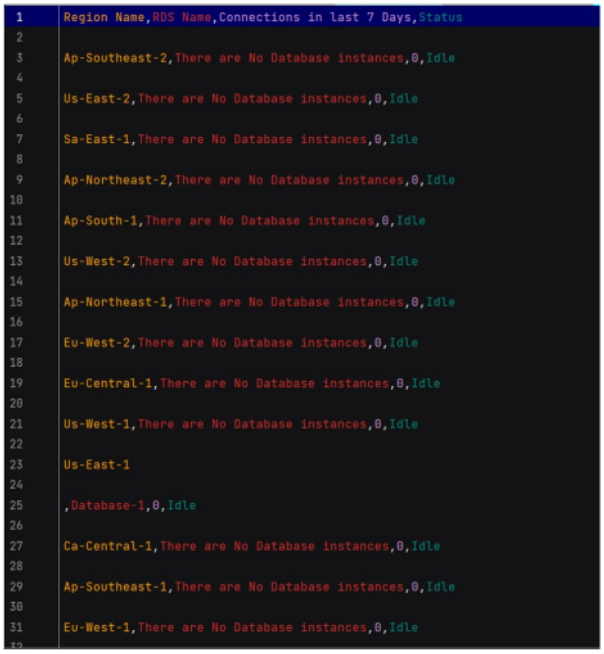
$$csvwriter.writerow(data)$$

v) Getting the statistics for DB Connections over the past 1 week using cloudwatch

get metric statistics (600, datetime.datetime,now()-datetime.timedelta(Seconds=604800), datetime.datetime.now(), "DatabaseConnections", 'AWS/RDS', 'Sum', dimensions='DBInstanceIdentifier': [str(db.id)])

vi) Stop the instances if no connection

$rds = boto3.client('rds', region\_name = r.name)$ $response = rds.stop\_db\_instance(DBInstanceIdentifier = name)$ // it should be rds instance name



```
Region Name,RDS Name,Connections in last 7 Days,Status

Ap-Southeast-2,There are No Database instances,0,Idle

Us-East-2,There are No Database instances,0,Idle

Sa-East-1,There are No Database instances,0,Idle

Ap-Northeast-2,There are No Database instances,0,Idle

Ap-South-1,There are No Database instances,0,Idle

Us-West-2,There are No Database instances,0,Idle

Ap-Northeast-1,There are No Database instances,0,Idle

Eu-West-2,There are No Database instances,0,Idle

Eu-Central-1,There are No Database instances,0,Idle

Us-West-1,There are No Database instances,0,Idle

Us-East-1

,Database-1,0,Idle

Ca-Central-1,There are No Database instances,0,Idle

Ap-Southeast-1,There are No Database instances,0,Idle

Eu-West-1,There are No Database instances,0,Idle
```

Fig. 1. Idle RDS

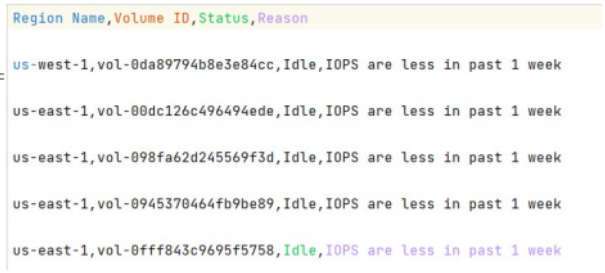
Figure 1 shows the RDS instances which has no connection in the past 7 days and are being utilized unnecessarily, this helps the user to have an overview of all regions aws well. 3. To detect underutilized ebs volumes :
i) Creating a data frame - Region Name, Volume ID, Status, Reason Reason describes if IOPS are less in past one week or IOPS are more in past one week based on threshold set in the script. For testing the threshold set is 10000000 and if it exceeds then IOPS are more in past one week, else it is less for the past one week.
ii) Checking if the volume is in use or attached to instance :

if not a.status: data = [str(r.name), str(a.id), "In-Use", "Volume Not attached to any Instance"] csvwriter.writerow(data) print("The volume is not attached to any instance in region:", str(r.name)) else: print(str(r.name.title()))

Figure 2 shows the underutilized EBS volumes that are idle in the region.

4. To identify if Legacy instances are in use and recommend for new generation instances:

i) Creating a data frame - Region Name, Instance id, Legacy Instance, Type, Remarks
ii) Getting the type of the instance $c = e = c.instance\_type$ c = str(c) c = c[0:2]
iii) Fill the columns with appropriate data data =

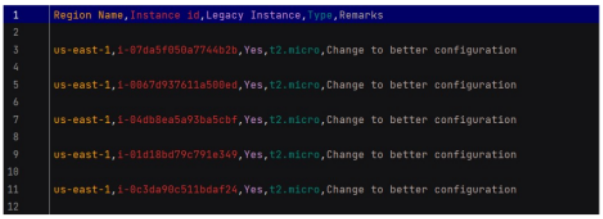


```
Region Name,Volume ID,Status,Reason

us-west-1,vol-0da89794b8e3e84cc,Idle,IOPS are less in past 1 week

us-east-1,vol-00dc126c496494ede,Idle,IOPS are less in past 1 week

us-east-1,vol-098fa62d245569f3d,Idle,IOPS are less in past 1 week

us-east-1,vol-09453704464fb9be89,Idle,IOPS are less in past 1 week

us-east-1,vol-0fff843c9695f5758,Idle,IOPS are less in past 1 week
```

Fig. 2. Underutilized ebs volumes

[str(reg.name), str(d.id), "Yes", str(e), "Change to better configuration"]

iv) Checking the instance type and write data into csv if c == t2 or m2 or c2 or hi2 or m2 or cr1 or hs1 csvwriter.writerow(data)



```
Region Name,Instance id,Legacy Instance,Type,Remarks

us-east-1,i-07da5f050a7744b2b,Yes,t2.micro,Change to better configuration

us-east-1,i-0867d937611a500ed,Yes,t2.micro,Change to better configuration

us-east-1,i-04db8ea5a93ba5cbf,Yes,t2.micro,Change to better configuration

us-east-1,i-01d18bd79c791e349,Yes,t2.micro,Change to better configuration

us-east-1,i-0c3da90c511bdaf24,Yes,t2.micro,Change to better configuration
```

Fig. 3. Legacy instances

Figure 3 shows the recommendation to the user to move to a better configuration in case an user uses an old generation instances or smaller versions below the requirement.

5. To identify unassociated Elastic IP which is not associated with running EC2 instance
i) Create a data frame Regions, Unassociated Ip Address
ii) Connect to aws using boto3 and obtain all addresses $connection = boto.ec2.connect\_to\_region(r\_name)$ $addresses = connection.get\_all\_addresses()$
iii) Checking if the IP has an address or not and write data into CSV $if(ins\_id == None)$: data = ["", address] csvwriter.writerow(data)

Figure 4 shows the Elastic IP address which is not associated with any ec2 instance and is being used unneccarily can be detected using the script across all regions on AWS.

6. Reserved Instances Recommendation : Figure 5 shows the implementation of the user interface of RI and how the prices are calculated, then optimizing and getting the usage of instances every month, this optimization is done by calculating the mean here. The results are plotted graphically.

- Features:

1. Expected Usage of Instances - Provide the usage of on demand instances over a time period to get cost optimized RI .
2. Previous Upfronts - Provide previous upfront detail if available to enable type of upfront preferred.
3. Platform - Provide the platform type being used

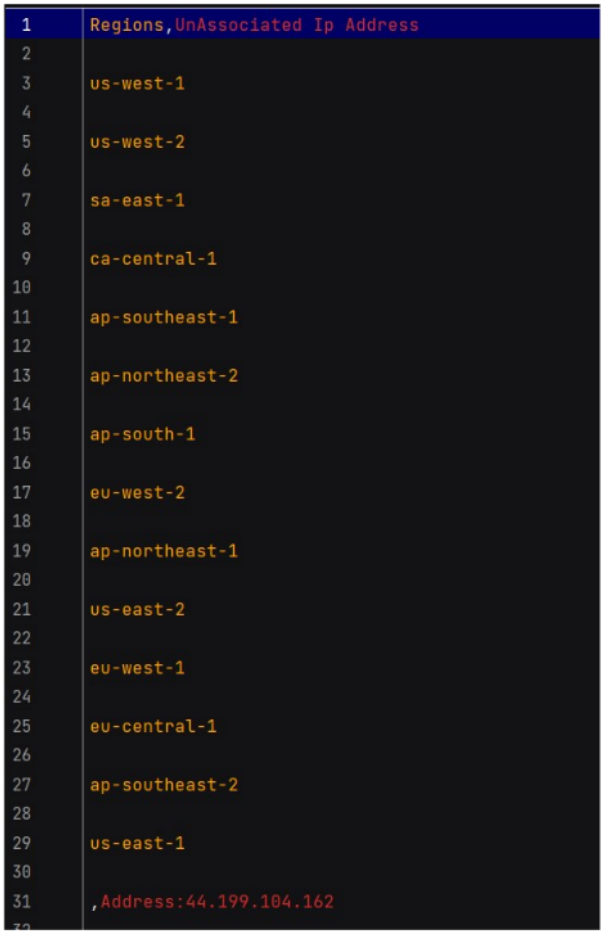


Fig. 4. Unassociated Elastic IP

because each platform has their own hourly rate by AWS.
4. Region -Provide the region of deployment because each platform has their own hourly rate by AWS.
5. Period- Provide the Time period for RI Recommendation
6. RI Pricing- Five types of Pricing provided: i) No upfront ii) Partial Upfront(1 year) iii) Partial Upfront(3 year) iv) All upfront(1 year) v)All upfront(3 year)

Figure 6 shows after clicking on JOB SUBMIT , it displays the usage instances , no.of months and no. of server types(instances types) Figure 7 displays the pricing information of AWS according to the plan selected.

- Cost Analysis and Graphical Representation of Optimized Cost and also provides Cost Gain details

Figure 8 shows the analysis of all the instances
1. Larger the upfront,Greater the Discount.
2.Cost Optimized way is taken out by the mean rate of usage. Using RI , the hourly rate is lowered significantly.

```
#UI of the RI
library(shiny)
shinyUI(fluidPage(
        titlePanel("SE01 RI RECOMMENDATION"),
        sideBarLayout , numericInput ,sliderInput ,tabPanel))


#Price Calculation
rawPrice <- function(priceFile) {
 avgDays <- 365 / 12  # Average days per month
 rawData <- read.csv(priceFile, header = T, sep = ",")
 rawData <- cbind(rawData,
             "On.Demand_Month" =
              as.numeric(
                substring(rawData[, "On.Demand_Hour"], 2, 6)
              ) * 24 * avgDays)

#Optimization
library(lpSolveAPI)

# Get instance usage per month
optResult <- getInstanceUsage(optResult, rawUps, valueX, thisMonth)

 # Get cost
 optResult <- getCost(optResult, prices)

#Optimization done by mean usage calculation
incProgress(1 / (ncol(optResult[["instsTotal"]]) - 1), detail = inst)
sum(instsNM1.opt[, -1]) / sum(instsTotal[, -1]) * 100

#Output Plot
library(ggplot2)
cost.plot <- data.frame(
   Month = as.character(instsTotal[, 1]),
   Base = rowSums(cost.base) - instsTotal[, 1],
   Opt = rowSums(cost.opt) - instsTotal[, 1],
   Base.cum = rowSums(cumsum(cost.base)) - cumsum(instsTotal[, 1]),
   Opt.cum = rowSums(cumsum(cost.opt)) - cumsum(instsTotal[, 1])

    myTheme(0) + xlab("Month") + ylab("Cost ($)") + ggtitle("Monthly Cost")
```

Fig. 5. RI implementation

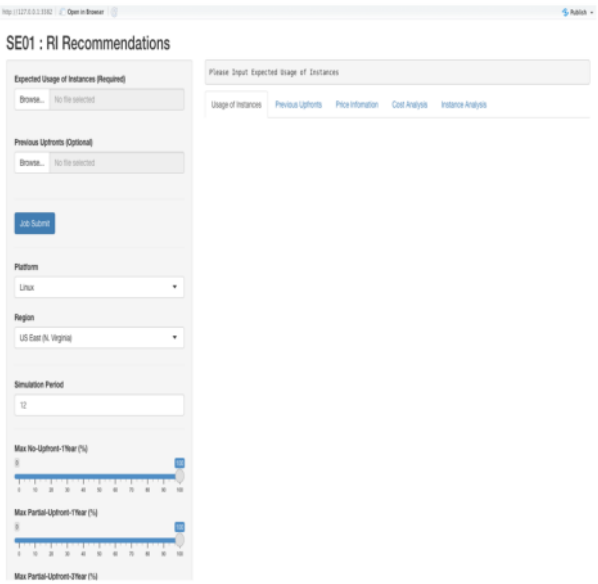


Fig. 6. User interface

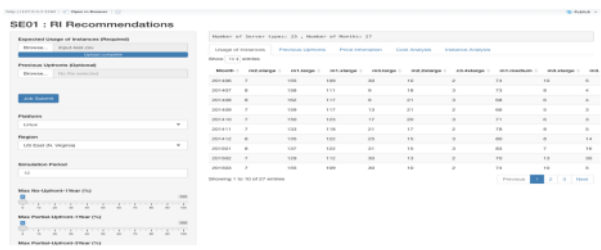


Fig. 7. Meta data of instances

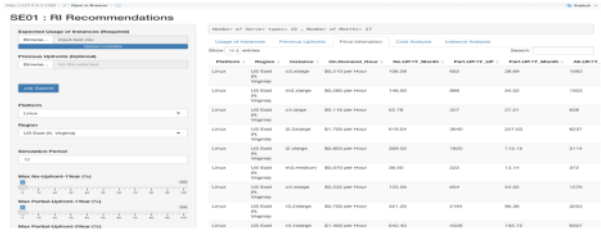


Fig. 8. Pricing info of the current plan

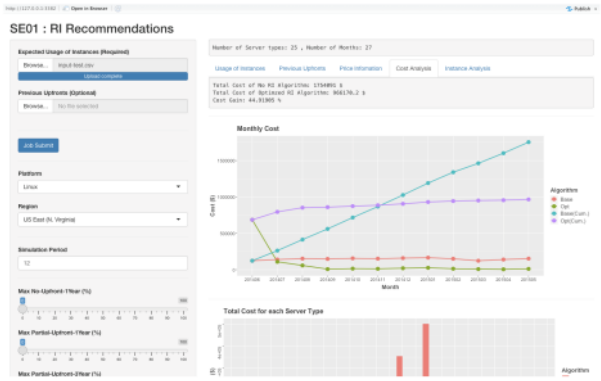


Fig. 9. Graphical representation of cost gains

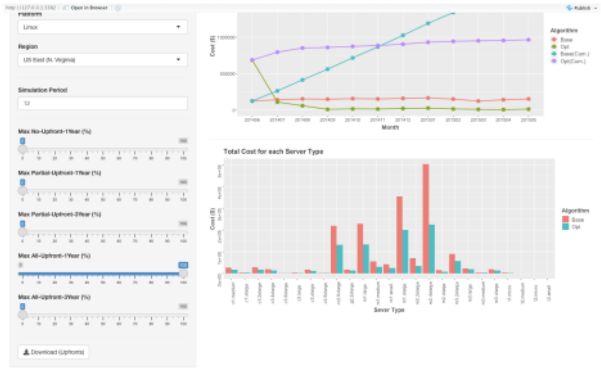


Fig. 10. Graphical Representation of optimized cost

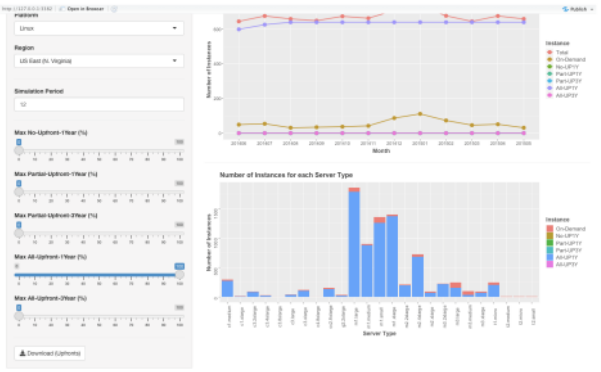


Fig. 11. Instance analysis

V. DISCUSSION AND CONCLUSION

We have conducted an extensive literature and product survey to understand the concepts of cost optimization. We have incorporated features in our product that will help in the optimization of the cost of AWS by taking care of its resources and it's performance. The results obtained at the end of our research were based on drawbacks of the previous work and taking away some valuable inputs to complete our research work. We have put out a few automation scripts which help in optimising the cost and usage of various services such as EC2, RDS, EBS, Elastic IP and Recommendation systems [9] We managed to configure and get access to the resources to modify with the help of AWS CLI and SDK named boto3 to implement the process and the scripts are implemented using python and R. For RI recommendations, a user interface has been developed using various packages and libraries which is a stimulation for obtaining the optimised ec2 instances that use reserved instances and provides the linear graphical representation. The work ensures to entail the implementation of our approach in multiple ways and fine tuning different aspects to achieve optimal performance.

The automation scripts provides the monitoring and a quick check of the usage of certain aws resources & recommendation, that can be done based on this usage data and reports gathered for the optimization of the cost. We will also be using SDK to connect to the aws resources so that we can automate the whole process for the ease of the users. The script contains all the constraints required to manage cost, which automatically can help the user or an organisation who wishes to build applications, optimise the cost for the cloud usage.

REFERENCES

[1] Steven Noble & Oliver Berger, "11 surefire ways to reduce your AWS EC2 costs"(14 Jan 2021)

[2] B. Kaminski, P. Szufel, On optimization of simulation execution on Amazon EC2 spot market, Simulat.Modell. Pract. Theory (2015)

[3] P . Kokkinos · T. A. Varvarigou · A. Kretsis · P. Soumplis · E. A. Varvarigos, "SuMo: Analysis and Optimization of Amazon EC2 Instances" , 2014

[4] Gustavo Portella1 Genaina N. Rodrigues1 Eduardo Nakano2 Alba C.M.A.Melo1 "Statistical analysis of Amazon EC2 cloud pricing models", 2018
[5] Javadi B, Thulasiram RK, Buyya R. Characterizing spot price dynamics in public cloud environments. Future Gener Comput Syst. 2013;29(4):988-999.
[6] Mayur Palankar# , Adriana Iamnitchi # , Matei Ripeanu * , Simson Garfinkel , "Amazon S3"
[7] Simson L. Garfinkel TR-08-07 (Citation:2017) , "An Evaluation of Amazon's Grid Computing Services:EC2, S3, SQS".
[8] Amazon Cloudwatch : https://docs.aws.amazon.com/whitepapers/latest/how-aws-pricing-works/how-aws-pricing-works.pdf
[9] AWS : https://aws.amazon.com/blogs/compute/10-things-you-can-do-today-to-reduce-aws-costs/

COST OPTIMIZATION IN AWS

ORIGINALITY REPORT

5%	3%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.tothenew.com Internet Source	2%
2	Submitted to PES University Student Paper	1%
3	Gustavo Portella, Genaina N. Rodrigues, Eduardo Nakano, Alba C.M.A. Melo. "Statistical analysis of Amazon EC2 cloud pricing models", Concurrency and Computation: Practice and Experience, 2018 Publication	1%
4	Woo-Chan Kim, Ohyun Jo. "Cost-optimized configuration of computing instances for large sized cloud systems", ICT Express, 2017 Publication	<1%
5	www.salesforce.com Internet Source	<1%
6	Submitted to Newham College of Further Education, London Student Paper	<1%
7	docs.aws.amazon.com Internet Source	

		<1%
8	Submitted to University of Limerick Student Paper	<1%
9	docplayer.net Internet Source	<1%
10	awsdocs.s3.amazonaws.com Internet Source	<1%
11	www.i-scholar.in Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		