Value at Risk estimation using parallel lambda, EC2, S3 with Google App Engine

Given Name: Muaad Siala

https://var-s-350014.ew.r.appspot.com

***Abstract*— This paper discusses a cloud system that allows users to determine five critical parameters in order to assess the value at risk for Amazon prices. The input for the system is scalable service, number of resources, the length of the price history, number of shots, and trading signal (Buy or Sell). The system is composed of six components, the first component is Google App Engine, the second component is Amazon API Gateway, the third component is Lambda function, the fourth component is Elastic Compute Cloud (Amazon EC2), the fifth is Amazon Simple Storage (S3), the sixth Google Image Charts. The components of the system and their interactions are discussed. This paper also discusses the experimental results of the system. Finally, this paper also discusses the satisfaction of the requirements, results and services costs has been offered.**

Keywords—Cloud Computing, Lambda, Elastic Compute Cloud, Google App Engine, Simple Storge Service, Amazon API Gateway, Value at Risk.

I. Introduction

The cloud system provides the user with a way to specify five parameters which are scalable service type (S), a number of resources (R), Length of Price History (H), Number of shots (D), and Trading Signals (T). The main goal of the system is to produce the average of the 95% value at risk and 99% value at risk for Amazon prices. The system will initialize the number of resources for the analysis using Lambda, EC2, and S3. In addition, the system gives the user a way to save the values of risk analysis to estimate the overall running costs. Per requirements to solve the problem, the author proposes a cloud system that includes three characteristics of the essential characteristics according to NIST SP 800-145 [1]. First, the system involves an on-demand self-service which is implemented in order to calculate the risk by utilizing the lambda function provided by AWS. Second, the system involves broad network access by deploying the front-end application on the Google App Engine, which is made public allowing any user from any device to access the network. Third, the system required a rapid elasticity in order to give the user the ability to specify the number of resources to warm up to compute the VaR analysis. We can look at the system from two main points of view. The point of view is the developer, and the second point is the user.

A. Developer

The system is composed of two parts, each with its own service model. The first part was built using The Function as a Service (FaaS) model in which the developer reaps the benefit of the AWS Lambda function and S3 [2]. The second part was built using The Platform as a Service (PaaS), which allows the developer to utilize and host the application's front end on the Google App Engine [1]. In terms of deployment models, the application has been deployed under the public cloud [1].

B. User

As mentioned above, the system provides the user with a simple front end to enter the five parameters (S, R, H, D, T) via the web browser. By inserting S and R, the system allows the user to use the on-demand self-service determined by the number of EC2 resources/ and a number of Lambda function invocations. Consequently, the user can increase the required infrastructure to a large extent without disturbing the Google App Engine host operations [3]. In addition, the system provides the user with the flexibility to save the result of each selection over time. In relation to NIST SP 800-145, the users are eligible to use the on-demand self-service from any device (Mobile, Tablet, Laptop) at any time which leads to the "public cloud" model.

II. Final Architecture

A. Major System Components

The system is the combination of 6 components, each component has a specific configuration and different roles within the system.

1) Google App Engine:

Google App Engine (GAE) is a platform as a service that allows the consumer to deploy and host their website. The hosting service is free up to a particular quantity of resources used, and it is only available in the standard environment, not the flexible environment. The fees were only charged if the application required additional storage, bandwidth, or instance hours. In this system, Google App Engine provides a standard environment: sandbox restricts, runs on small instances which made the front-end application hosted on it. The GAE cannot write to disk but can deploy files. The GAE cannot use non-whitelisted binary libraries. The GAE limits CPU and memory options available. The GAE can retrieve URLs, but not open direct network connections [4]. So, it's much needed for the Amazon API gateway to communicate with the rest of the system components. Google App Engine has six features that made it the best for hosting applications. The following are the features: All-Time Availability, User-friendly platform, Enhanced scalability, Cost savings, assure a faster time to market for more descriptions see [5].

2) AWS Lambda Function:

AWS Lambda is a serverless compute service that handles the core compute resources for the user while the codes run in response to events. The AWS Lambda allows the developer to enlarge other AWS services (e.g., initialize EC2, write/read from S3). AWS Lambda is a service that runs code in response to a variety of events such as HTTP

requests through Amazon API Gateway and modifications to items in Amazon Simple Storage Service (S3) buckets [7]. AWS provides 1 million requests free by using Lambda [8]. Also, instead of allocating infrastructure for peak capacity upfront, pay just for the computation time you use- per millisecond. In this system, lambda has been used in parallel to calculate risk values using threads in python.

3) Amazon API Gateway:

The Amazon API Gateway service is a part of AWS serverless infrastructure, it enables users to build, distribute, maintain, monitor, and secure REST, and HTTP at any scale. API developers can create APIs that connect to AWS, and other web services [6]. Also, it provides the system with the support needed for stateless [HTTP and REST] APIs and extensible authentication mechanisms like Lambda authorizer functions and AWS IAM roles [6].

4) Elastic Compute Cloud (Amazon EC2):

In the Amazon Web Services (AWS) Cloud, Amazon Elastic Compute Cloud (Amazon EC2) provides computational power that is scalable. Using Amazon EC2 decreases the upfront hardware expenditure required, allowing you to create and deploy applications much faster [9]. The main concepts are Instance, Amazon Machine Image (AMI), Elastic IP address, Region, Availability zone, Security group, Block storage volume, and Snapshots [10]. Because of its speedy development process, and to implement the functionality, including the software package running in the instance, Amazon EC2 was chosen over other AWS services. As a cluster of instances, however, Amazon ECS or Amazon EMR might be more dependable options [10].

5) Amazon Simple Storage Service (S3):

Amazon S3 is a service that allows the system to store and access a limitless amount of data at any time from any location [11]. It permits the system to save all the data needed for the functions to run efficiently. The Amazon prices (yfinance) data, value at risk 95% and 99% generated by lambda, the details of the EC2 resources, and its temporary results are all of them retained for the system usage. Because of its simplicity of accessing and storing data, lack of infrastructure management requirements, cheaper cost based on system utilization, and lack of advanced data storage functionalities, Amazon S3 was chosen over other amazon storage services.

6) Google image charts:

Google Charts is a great way to present statistics on the website. From simple line charts to complex hierarchical treemaps, the chart gallery contains a wide range of ready-to-use chart formats. Google Charts tools are robust, easy to use, and completely free. It provides the system a way for showing a line each for the 95% and 99% risk values for each signal and two lines relating the averages over each such that higher and lower risk signals can be seen readily [12].

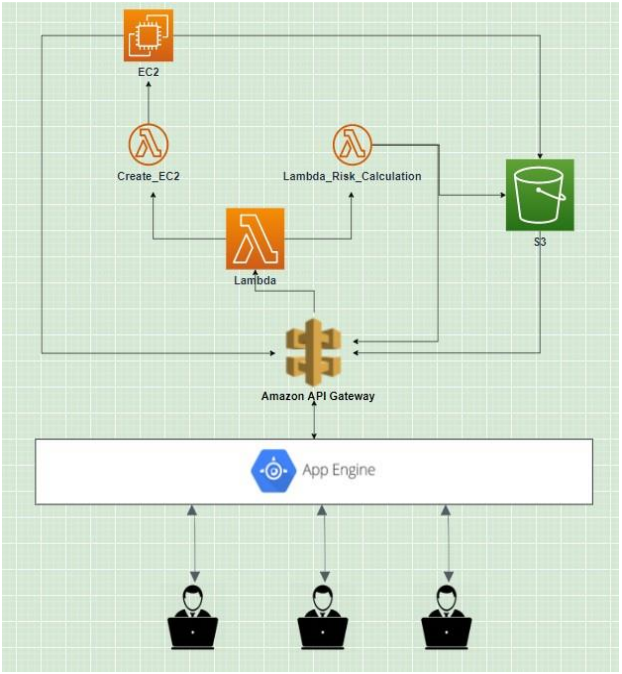

Figure 1. Cloud System Overview

B. System Component Interactions

As mentioned in the section above (Major System Components), the six components have been in use for the system. In this section, I will explain the interaction between those components. The section is divided into two main scenarios. The first scenario is when calculating the value at risk using the Lambda function. The second scenario is when the user asks for the calculation of the value at risk using the other scalable service (EC2).

1) Calculating Risk using Lambda Function:

The user enters the URL (https://var-s-350014.ew.r.appspot.com) the google app engine will render the home page that contains a form. The home page asks the user to enter the five-parameter required to generate the risk analysis. After submitting the HTML form, the parameters are collected by Google App Engine which calculates the mean (mu) and the standard deviation (std) for the prices with a specific length of history price (e.g., 200 days). And then send them using the HTTP POST method in the Amazon API Gateway.

Afterwards, the google app engine will establish a direct connection to lambda and send a JSON containing five keys: mean (mu), standard deviation (std), length of history price (H), number of shots (D), and length of prices (L) that generated in google app engine.

The lambda function will get the five keys and calculate the risk values at 95% and at 99%. In parallel lambda, GAE will do multiple calls using threads in python. The parallel model has made more copies of the same functionality in order to respond to the user demand. The Lambda function (Calculate_Risk_Averages_Funtion) is executed with the parameters (mu, std, H, D, L), L is the length of dataset. The returned results will send back to the app engine via API gateway to render on the result page for the user. Meanwhile, the GAE will take the returned result from (Calculate_Risk_Averages_Funtion) as well as the user selection of (S, R, H, D, T) and then send them to store in S3 using lambda invocation (Write_in_S3_Funtion).

Amazon API Gateway is responsible for the communicationsbetween the lambda, S3 and GAE. In case the user requests anaudit page the GAE engine will establish a connection and invocate Lambda to read from S3 (Read_From_S3_Function). Then the lambda returns the history of the user's previous experiments. On the other hand, EC2 has been created using the lambda function too. EC2 container calculate the risk values and send them to S3 as well as send them back to google app engine.

III. Results

The result of the system has been produced to carried out to see how different parameters affect the system outcome. The system utilizes two services to calculate the risk values. The two services (Lambda and EC2) are affected by the same parameters which are **number of resources (R)**, length of price history **(H)** and number of shots **(D)** (Monte Carlo size). The system returns the value at risk for 80,000 shots after 4.4 seconds in average. And it takes up to 15 seconds in average for 1 million shots of Monte Carlo size. However, Due to bugs in EC2 implementation the system unable to return runtime costs for using EC2 service.

Ex	*S*	*R*	*H*	*D*	*Time*
1	Lambda	4	200	80,000	4.4
2	Lambda	4	200	160,000	5.6
3	Lambda	4	200	320,000	7.8
4	Lambda	4	200	500,000	10.8
5	Lambda	4	200	1,000,000	17.2

Table 1. Shows the table from Audit page.

IV. Costs

The cost of cloud computing is important aspect for businesses whose running their application on AWS, GAE, Azure, and others. Consequently, minimizing the costs of cloud services for their operations is essential to reach best cloud infrastructure. As I mentioned above the system used six components. The costs of each component are different. The cost calculation of lambda (512 MB) function is totally relied on memory specification, duration of execution, and how many calls for the function. The cost of EC2 (t2.micro) for us-east-1 region is determined by runtime and volume of data send and receive from the instance. Table 2. Shows the approximately of costs of each component of the system.

Services	Cost for	Cost in $
Google App Engine	Europe west 1	$ 0.05 per hour per instance [14]
Google Image chart	No region	0
Amazon API gateway	Number of Requests	$ 1.00 per million [17]
Lambda functions	Execution duration + number of calls	$ 0.0000000083 per 1 Ms [16]
Elastic Compute Cloud	Data in and out	$ 0.0116 per hour [15]
Simple Storage Service	For first 50 TB / month	$ 0.023 per GB [18]

Table 2. shows the approximate costs for the system components.

References

[1] P. Mell, T. Grance (2011), "The NIST Definitaion of Cloud Computing" (U.S. Department of Commerce, Washington, D.C), NIST Special Publication 800-145 (SP 800-145).

[2] L. Gillam, "FaaS (PaaS) overview", Lecture Week 4, COMM034-Cloud Computing, 2021-2022, University of Surrey.

[3] D. Rountree, L. Castrillo, "On-Demand Self-Service" in the Basic of Cloud Computing, 2014. Science Direct.

[4] L. Gillam, "Google App Engine", Lecture Week 2, COMM034-Cloud Computing, 2021-2022, University of Surrey.

[5] A. Lomas, "What is Google App Eninge, its Advantages and How it can benefit your Business" Net Solutions, 2021.

[6] Amazon Web Services, Inc., "What is Amazon API Gateway", Developer Guide, Amazon API Gateway, AWS Documentation, 2022.

[7] Amazon Web Services, Inc., "Lambda Features", Developer Guide, AWS Lambda, AWS documentations, 2022.

[8] Amazon Web Services, Inc., "Lambda Overview", Developer Guide, AWS Lambda, AWS documentations, 2022.

[9] Amazon Web Services, Inc., "What is Amazon EC2?", Developer Guide, Amazon EC2, AWS Documentation, 2022.

[10] G. Reese, "Cloud Application Architectures", Chapter 2 – EC2, 2009.

[11] G. Reese, "Cloud Application Architectures", Chapter 2, - S3, 2009.

[12] Google Developer Guide, Google Charts, 2022.

[13] Amazon Web Services, Inc., "Amazon Simple Storage Service", Developer Guide, AWS Documentation 2022.

[14] Google Cloud, "App Engine Pricing", App Engine Standard Enviroemnt Pricing, 2022.

[15] Amazon Web Service, Inc., "Amazon EC2 On-Demand Pricing", AWS Documentation 2022.

[16] Amazon Web Service, Inc., "Amazon Lambda Pricing", AWS Documentation 2022.

[17] Amazon Web Service, Inc., "Amazon API Gateway Pricing", AWS Documentation 2022.

[18] Amazon Web Service, Inc., "Amazon S3 Pricing", AWS Documentation 2022.

[19] AWS CLI Command Refrence, "Stop instance" AWS documentation. 2022.

[20] Victor Bonilla Pardo, "Serverless web application for risk identification and risk assessment in a financial asset" 2020, University of Surrey.

[21] SARAV AK, Devops Junction "Copy files from EC2 to S3 Bucket" 2022.

[22] Utkarsha Bakshi, "How to Upload File to S3 using Python AWS Lambda", 2022.