Risk Assessment and Prediction for Financial Time Series Using GAE and AWS Scalable Cloud Service

Seyed ali, Shahebrahimi: 6838826, ss05110@surrey.ac.uk

https://adept-bridge-413513.nw.r.appspot.com/

***Abstract*—**

This report demonstrates the implementation and design of the cloud-native API system for analyzing assessing risk and predicting profitability to financial trading signals with time series data. In this system, multi-cloud services are employed. It is a mixture of GAE (Google App Engine) and AWS (lambda, Ec2), and it makes an API for users to interact with through curl HTTP for executing Monto Carlo simulations. In addition, AWS Lambda and AWS EC2 are employed for computationally intensive tasks. This application is designed and established with five critical characteristics to comply with NIST SP 800-145 for scalability, resource optimization, and cost-effectiveness. At the same time, users and developers safely interact with each other under the NIST standards. Cost, time, and performance will be evaluated to evaluate the system's efficiency.

Keywords— Cloud Computing, NIST SP 800-145, AWS Lambda, EC2, GAE, Monte Carlo Simulation, Financial Risk Assessment

I. INTRODUCTION

This report detail the design Relate to the system comprehensively and very clearly in compliance with NIST SP 800-145 utilizing Google App Engine (GAE) for API hosting and Amazon Web Services (AWS) Lambda and EC2 for carryon computational intensive task in respect to (A) what the developer uses/experiences and (B) what a user of such a system would use/experience:

A. Developer

In NIST SP800-145	Developer uses and/or experiences
On-demand self-service	The developer who can manage to set up the system, allocate the resources automatically, and assign the resources without human interaction to provide a capable system. Using the AWS Lambda functions and EC2 instances, it can also be managed, scaled, and terminated via HTTP and have seamless power to all needed infrastructure, even the complex ones.
Broad network access	*The developer can access the whole system with the GAE link, making it available on the Internet with most electronic devices that have standard browsers on them. The developer can also manage and interact with these devices. This flexibility is valuable in a distributed environment since all the developer teams utilise different devices.Also,this broad network access facilitates distributed development and evluating environments.*
Resource pooling	*Cloud services, such as AWS EC2 and Lambda, Allow the developer to allocate resources and computational power among different customers based on demand. The customer does not know the location of the services, allowing the developer to concentrate on the logic of the application rather than any other trifling thing, such as the hardware for simplification and optimization resources.*
Rapid elasticity	*The system is designed to scale rapidly if any demanding computation is needed. For example, when the number of users increases during peak times, Additional Lamda and Ec2 can be allocated to meet this demand. The developer has the elastic potential to manage the resources and scale out the system automatically; it also needs to be scaled up and down based on its capabilities. This ensures that the system is still responsive to the demand without manual intervention, increasing efficiency. Thus, this dynamic computational demand in peak hours can be allocated to ensuring reposnibility and efficiency.*
Measured service	*The developer utilizes the monitoring and controlling tools that AWS providers provide to optimize and control the whole system in terms of efficiency and cost-effectiveness. Tools such as AWS CloudWatch give the devloper this ability to to fine-tune their applications for cost efficiency.this is essential for controling the cosf specefily ina cloud in which utlizing directly have great impacts for billing.*

B. User

In NIST SP800-145	User uses and/or experiences
On-Demand Self-Service:	Users can enjoy the smooth and uninterrupted application operation without human interaction, just interacting with the service provider. This ensures that the user experience is as quick and reliable as possible.
Broad Network Access:	*The user can access the system through a wide range of devices worldwide, offering them more convenience, flexibility, and usability in different contexts.*
Resource pooling	*Users do not need to worry about the Sharing and computing resources that have been allocated because the provider uses the pooling system to allocate resources to allocate enough resources.*
Rapid elasticity	*The users feel they have indefinite capabilities because the system can handle all types of load smoothly, especially when intensive analyses are needed to ensure that performance is not reduced under heavily loaded conditions.*
Measured service	*Users can also utilize the metering capabilities to monitor their spending costs through the provided features. This ensures they are aware of their resource cost and consumption, helping them to make informed decisions about their usage.*

II. FINAL ARCHITECTURE

This architecture is designed to handle the whole system, and the computationally demanded task while considering the system's scalability and effectiveness by observing the cloud computing principles. For designing this system first, we need to consider some crucial major criteria, we consider

computing the past(h) days and generating many simulations according to the trends the user requested. Because the system involves calculation, many simulations must be done in the bases of the dataset and parameters.

Google app engine (GAE) is employed for hosting the website in which all API is hosted there. Either EC2 or Lambda is being used as a computation base for user selection.

HTTP Gateway handles the creation, analysis, and termination of Lambda and EC2. All instance IDs are stored to be used later in a file in the S3 bucket. All of them are instances of IDs accessed by the lambda controller to terminate and run aging.

AWS CloudWatch service is used to monitor the usage.

Statistics of the AWS services.

In addition, Matplotlib is employed to generate data visualization.

The API is responsible for creating and terminating EC2 instances.

The system is using the warmup to reduce the wait time for user or users and ask how much computation is needed. Also, to perform all the system to the working condition. In the /warmup API in made to while the user choice (s) for service type (lambda or Ec2), and r is used for (scale out), which is used for number of the Ec2 instances or Lambda user might needed. In addition, in warmup is trying to send number of the parallel requests for waken up the AWS or EC2 instances according to the user choices and make it ready to avoid cold start issue.

Analysis:

In the **analysis,** the endpoint should be used after the warmup endpoint to avoid the system's cold start. First, the system strives to retrieves the data from Yahoo Finance using the yfinance library. My system uses the AMZN dataset. All the input data, such as history, stand for h (minimum days needed for calculation of the mean and standard deviation, d in stands for the number of shots, type stands in (sell or buy), and p number of days which price difference will be compared to predict if the signal was profitable or not. To achieve better performance and decrease the time, the Thread service is used to parallelize the executions of the simulations.

In the case of Lambda, I aim to split the data because this splitting approach in terms of the data in my architecture is because when I strive to measure time, I understand it is more cost-effective and time-effective than using the various Lambda. Using Lambda for different parts is more economical for the user, as it reduces the time and cost of using different lambda systems.

The analysis is done by in 2 logical system the first part in the GAE, and the second part is simulated in the one of the scalable services in the Lambda or Ec2 spending to user preferences. This distributed approach not only optimizes resource but also ensures that the system remains flexible and scalable, capable of handling vast verity of the loads without inconvenient on performance. It is worth to mention that because the GAE the lightweight front-end for data management.

Scaled ready is checking if the newly Lambda or EC2 instances status.

It is a request to the HTTP API Gateway of the AWS that is linked to the lambda function named simulate with simulation logic in it. Parameters are mean, std and shots. In the case of EC2, we send one request per one EC2 instance, in total r requests per signal. The requests for the EC2 are sent to the public DNS names of the instances at port 8080 and are handled by another Flask Api which can be launched listening to the port.

III. Satisfaction of Requirements

Satisfaction of the Requirements is provided in Table I. the requirements that were MET, PARTIALY MET, or NOT MET, and list the fully working endpoints.

TABLE I. Satisfaction of Requirements/Endpoints and Code use/Creation

	MET	***PARTIALLY MET***	***NOT MET***
Requirements	i.	iv.	ii. iii
Endpoints	/warmup resources_ready /get_endpoints /analyse /get_sig_vars9599 /get_avg_vars9599 /reset /terminate /resources_terminated /get_sig_profit_loss /get_tot_profit_loss /get_chart_url /get_time_cost /get_audit	warmup /get_warmup_cost	

IV. Results

For evaluation of the system performance. Six experiments were carried out table II with various setups to identify relation between the r and d and the corresponding time. The parameter of the h, p, t and s in unchanged throughout the whole experiments to identify the relation between scale(r) and d, and the time that is totally related with cost of the

system, which is obvious that while the utilizing of more lambda the cost will be increased starting with the 10,000 to 40,000 shots for simulation. In the table II, in the three first experiment the (r) is constant 2 and the time increased from 8.02 to 13.55,26.57, respectively. In the three second experiment as the r increase to three the time also decreased as you can see in the table II. It is also wroth to mention that as the shots is double the time increase approximately make doubles as well.

In conclusion, it is readily apparent that with if the number of the scale the analysis time is reduced. To get the bill of the AWS Lambda we can employ Cloud watch. Thus, it is completely review in the section **V**.

TABLE II. RESULTS (TABLE INCLUDED FOR CONVENIENCE – USE NUMBERS OF ROWS/COLUMNS AS APPROPRIATE FOR PRESENTING RESULTS)

r	*h*	*d*	*p*	*Time(sec)*
2	101	10000	7	8.02
2	101	20000	7	13.58
2	101	40000	7	26.57
3	101	10000	7	6.34
3	101	20000	7	11.09
3	101	40000	7	21.10

V. COSTS

To evaluating the costs of AWS are calculated according to the resources used by the user, or the resources allocated by the provider for every user. It is worth noting that the costs need to be appropriately managed; otherwise, the price can be staggering. We need to mention two curial price criteria: full-scale use and the price of each recourse.

In the first part, let me make some speculation about the day and the number of the user. This system is run any delayed in the GAE, and considering it is run for two weeks. Also, we deem that half of the users use Lamba and half of them utilize the Ec2 and r=4 and shots =100,000 per r to have less calculation time. In term of the AWS, a Lambda function with 128Mb of memory in US East (N. Virginia) costs $0.0000021 per second [2][3]. For the t2. micro instance of EC2, the cost is $0.0116 per hour [2]. S3 storage price $0.023 per GB of data storage for the first 50TB [5].

Cost Calculation for GAE

Cost of a GAE B2 instance in Us-east4 region is about 0.11$ per hour

1,000 users×30 minutes each per day×365 days=10,950,000 minutes,

10,950,000 minutes/60 minutes=182,500 hours of runtime on GAE

182,500 * 0.11=20075$ per year

b) Simulations:

2 buy/sell signals×4 simulations per signal=8 per run

1,000 users×10 runs per day×365 days=3,650,000 runs

3,650,000 runs×8 simulations per run=29200000

Simulation Time*:*

29200000 simulations×20 seconds per simulation=584 000 000 seconds

584000000 seconds/60 seconds=9733333 of simulation Because the load on the split between the Lambda and EC2

9733333 minutes/2=4866666 minutes per service (Lambda/EC2)

d) Cost breakdown

- **GAE:182,500 * 0.11=20075$ per year**
- **Lambda**: 4866666 minutes×$0.0000021 per second×60=
- $10.21
- **EC2**: 4866666 minutes×$0.0116 per hour/60=$940
- **S3Storage**: 2,805 GB×$0.023 per GB=$64.52

e) Total_Yearly_Cost:21089 $

REFERENCES

1. [1] Peter Mell, and Timothy Grance, "The NIST Definition of Cloud Computing," September 2011
2. [2] Amazon Web Services, "AWS Lambda Pricing", AWS Lambda, AWS
3. [3] "Amazon EC2 On-Demand Pricing", AWS cost pricing
4. [4] L.Gilliam , "Amazon Web Services (AWS), Getting Started", Week2 Lab,COMM034 2022-2023, University Surrey
5. [5] L.Gilliam , "Coursework description LSA, 2022-23", Coursework COMM034 2022-2023, University of Surrey
6. [6] Seyed ali shahebrahimi 2022-2023, University of Surry