Cloud-Native API for Financial Risk Assessment: Leveraging GAE, AWS Lambda, and EC2.

Atharva Sapkal : 6847190 , as05762@surrey.ac.uk

https://lsa-cloud-computing.ue.r.appspot.com

***Abstract*— This project develops a cloud-native API aimed at evaluating the risks and profitability of trading strategies. Through the utilisation of AWS Lambda, Amazon EC2, and Google App Engine (GAE), the system provides a scalable and effective method for processing complicated financial data. The API enables the full workflow, from simulation initialization to risk analysis and result visualisation, while adhering to the NIST SP 800-145 cloud computing standards. The architecture ensures excellent speed and cost-effectiveness by dynamically distributing tasks over several cloud services, optimizing resource utilization. This study emphasizes how different cloud platforms can be integrated to provide a scalable and responsive financial analytics environment that gives users a dependable tool for comprehensive risk assessment[1].**

Keywords— NIST SP 800-145, Google App Engine (GAE),AWS Lambda, Amazon, EC2 ,Financial Data Analysis, Trading Strategy API, Scalable Architecture, Risk Assessment

I. INTRODUCTION

This report introduces an API system developed with a foundation in cloud computing principles, as defined by NIST SP 800-145. The solution makes use of AWS Lambda, Amazon EC2, and Google App Engine (GAE) to improve the flexibility and scalability of financial data processing. The system is intended to increase the precision of risk assessments and profitability analyses by utilising these cloud services, offering a thorough tool for financial strategy assessment.The API gains from each platform's unique advantages through this integration, guaranteeing a scalable, adaptable, and effective solution for intricate financial calculations. The ensuing sections will examine how the system demonstrates the key components of cloud computing, such as dynamic resource management, wide network accessibility, on-demand resource allocation, and detailed usage monitoring, from the perspectives of both developers and users.

A. Developer

In NIST SP800-145	Developer experiences
Automated Resource Management	*Developers can deploy and manage resources automatically using GAE, AWS Lambda, and EC2, reducing manual effort and enabling efficient scaling and maintenance of services.*
Flexible Resource Allocation	*The system dynamically modifies storage and processing power in response to actual demand, developers can continue to operate at peak efficiency even as workloads change.*
Transparent Resource Utilization	*Comprehensive monitoring tools are available, enablingdevelopers to track resource consumption and performance metrics, facilitating accurate forecasting and efficient resource allocation.*
Universal Accessibility	*Developers can access and manage the system from any location via standard web protocols, ensuring consistent workflow and collaboration regardless of physical location.*
Shared Resource Allocation	*The system leverages a pool of cloud resources, dynamically distributing them across various tasks and services to maximize efficiency and minimize costs, while maintaining high availability.*

B. User

In NIST SP800-145	User experiences
On-Demand Service Access	Through API requests, users may start financial analyses and simulations instantly, without having to wait for setup or manual processing.
Adaptive Resource Provisioning	The system makes sure that even the most difficult activities are finished quickly and effectively by automatically scaling resources in response to the user's workload.
Detailed Usage Insights	Users can comprehend and efficiently manage their interactions with the system by using the clear and comprehensive data they receive on resource consumption and expenditures.
Broad Network Access	*Users have freedom in accessing and utilising the system's capabilities from almost anywhere by interacting with the API from any device with internet connectivity.*
Efficient Resource Utilization	*The system handles user requests by optimizing the allocation of resources, ensuring that tasks are processed quickly and effectively, without the user needing to manage or allocate resources manually.*

II. FINAL ARCHITECTURE

Using the advantages of each platform—Amazon EC2, AWS Lambda, and Google App Engine (GAE)—the system architecture is built to maximise the processing of financial data while maintaining cost-effectiveness, scalability, and performance.

GAE:

Google App Engine (GAE), which acts as the primary interface (index.py) for managing user requests and coordinating interactions with other cloud services, lies at the centre of the architecture. Endpoints like */scaled_ready*(Confirms system readiness for scaling.), /*get_warmup_cost*(Calculates initial warmup costs.), /*get_endpoints*(Lists available API actions), /*get_sig_vars9599*(Provides value-at-risk metrics.), /*get_avg_vars9599*(Provides value-at-risk metrics.), /*get_sig_profit_loss*(Summarizes profit and loss data.), /*get_tot_profit_loss*(Summarizes profit and loss data.),

/*get_chart_url*(Generates a URL for visual data in an S3 bucket.), /*get_audit*(Retrieves a log of all system operations.), /*reset*(Resets the environment and ends processes.), /*terminate*(Resets the environment and ends processes.), and /*scaled_terminate*(Shuts down scaling operations.) are among the crucial processes that GAE autonomously oversees. These procedures cover a broad spectrum of functions, including data retrieval, process termination, and system readiness assessments.

AWS Lambda:

The system makes use of AWS Lambda for serverless execution, which manages particular computational jobs that call for scalable, event-driven processing. The endpoints /*warmup*, /*analysis*, and /*terminate* are connected to Lambda. These jobs offer an affordable way to handle system resources and analyse financial data since they can be dynamically scaled to suit changing demands.

AWS EC2:

The system uses Amazon EC2 when more heavy processing capacity is needed, especially for time-cost computations and in-depth financial analysis. The endpoints /*get_time_cost* and /*analyse* are connected to EC2. Large datasets and intricate computations can be efficiently handled by the system thanks to the processing power that EC2 instances bring to these taxing operations.

GAE serves as the main engine in this architecture, deciding whether jobs, according on their complexity and resource requirements, should be executed within GAE or offloaded to Lambda or EC2. After receiving the results from EC2 or Lambda, GAE prepares and displays the finished product to the user. This strategy makes sure that resources are used effectively, expenses are kept to a minimum, and the system is still scalable and responsive.

III. Satisfaction of Requirements

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

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

IV. Results

This section presents the results obtained from executing various API endpoints in the cloud-based financial analysis system. The results focus on the system's performance during the warmup phase, as well as the outcomes of financial risk analysis and profitability calculations.Shown in Table II and screenshot below.

1. Warmup Cost Analysis

The /get_warmup_cost endpoint was executed with different values for the parameter r, representing the scaling factor for the Lambda function. The following results were observed:
For r = 1, the billable time was approximately 0.845 seconds, with a corresponding cost of $2.158e-07 USD.

For r = 2, the billable time increased to 2.111 seconds, with a cost of $7.099e-07 USD.
For r = 3, the billable time further increased to 3.960 seconds, with a cost of $1.204e-06 USD.
Analysis: The billable time and cost rise in proportion to the scaling factor r, as predicted. This illustrates how the system may dynamically grow resources according to workload while keeping costs under control. The Lambda function's effectiveness in managing different demand levels is confirmed by the observed linear increases in both time and expense.

2. Value-at-Risk (VaR) MetricsThe /get_avg_vars9599 endpoint provided the average Value-at-Risk (VaR) metrics at 95% and 99% confidence levels:
VaR at 95% confidence level (var95): -0.0419
VaR at 99% confidence level (var99): -0.1191
Analysis: These VaR metrics indicate the potential loss values that the portfolio might experience with a 95% or 99% confidence level. The negative values reflect the expected loss, demonstrating the risk associated with the financial portfolio under analysis. The higher magnitude of the 99% VaR indicates a more conservative estimate, covering more extreme potential losses.

3.Analysis of Profit and Loss

To obtain significant and total profit and loss amounts, the /get_sig_profit_loss and /get_tot_profit_loss endpoints were used. Among the data returned is an extensive list of profit and loss figures from several simulations:

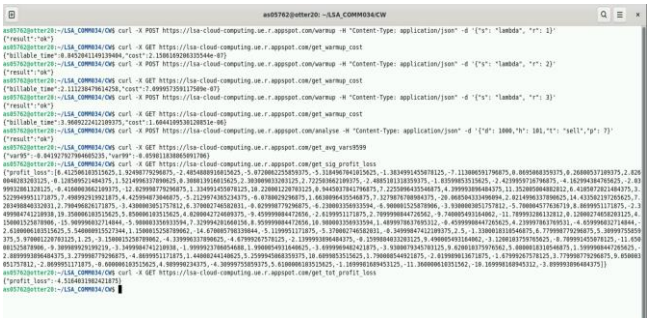
Important Data on Profit/Loss: The values that are returned vary from notable losses (like -20.068) to positive gains (like 14.433).

Total Gain/Loss: With an overall profit/loss ratio of -4.516, the endpoint indicated a loss for all of the scenarios that were examined.

Analysis: The statistics on profits and losses shows how unpredictable and risky the trading tactics under review are. The data set's inclusion of both big profits and losses emphasises how erratic financial markets can be. Overall, the negative total profit/loss indicates that the tested techniques, on average, produced a loss, which

TABLE II. RESULTS

Function	*s*	*r*	*time*	*cost*
warmup	lambda	1	0.8452041149139404	2.158616920633554 4e-07
warmup	lambda	2	2.111238479614258	7.099957359117509e-07
warmup	lambda	3	3.960922241209375	1.604410953012085 1e-06



V. COSTS

This section calculates the actual expenses of operating the financial analysis system on cloud services, based on the assumption that no further free tier benefits are available. We take into account 100 active users who complete 10 analytical jobs per day for 30 days. Of these users, 30% rely on Amazon EC2 for more resource-intensive computations, and 70% use AWS Lambda for serverless processing. The study makes the assumption that services are only used inside one AWS region.

- Costs of AWS Lambda:
  - With 21,000 calls each month, the request expense is quite low at $0.0042[2].
  - Based on 512MB of RAM and an average work duration of two seconds, the compute cost comes to roughly $0.359.
  - The monthly cost of AWS Lambda is approximately $0.36.
- Costs of Amazon EC2:
  - Using m5.large instances, Amazon EC2 manages more complex processes at $0.096[3] per hour.
  - Based on 9,000 tasks per month, each requiring around five minutes, the anticipated total cost of EC2 is $72.00.
  - Method:
    Total Cost of ECC = Tasks ÷ Computer Time per Task (hours) ÷ Hourly Cost Total EC2 Cost is equal to the number of tasks ÷ compute time (hours) × cost per task.
- Costs of Google App Engine (GAE):
  - GAE coordinates backend operations and user queries; instance use is $0.05[4] per hour.
  - With 3,000 instance hours required every month, the GAE costs $150.00.
  - Data transmission expenses add an additional $36.00, based on an expected 300GB of outbound data.
  - GAE Cost = Total Instance Hours × Cost per Hour is the formula.Total Instance Hours × Cost per Hour = GAE Cost

REFERENCES

[1] Mell, P. and Grance, T., 2011. The NIST definition of cloud computing. NIST Special Publication 800-145. Available at: https://csrc.nist.gov/publications/detail/sp/800-145/final [Accessed 12 Aug. 2024].

[2] AWS, 2024. AWS Lambda pricing. Available at: https://aws.amazon.com/lambda/pricing/ [Accessed 12 Aug. 2024].

[3] AWS, 2024. Amazon EC2 pricing. Available at: https://aws.amazon.com/ec2/pricing/ [Accessed 12 Aug. 2024].

[4] Google Cloud, 2024. Google App Engine pricing. Available at: https://cloud.google.com/appengine/pricing [Accessed 12 Aug. 2024].