Optimizing Financial Risk Assessment with Cloud-Based Monte Carlo Simulations

Venkat Sandeep Imandi: 6829480, vi00085@surrey.ac.uk

https://vi00085comm034lsa.nw.r.appspot.com

Abstract— This project harnesses AWS and Google App Engine (GAE) to implement Monte Carlo simulations aimed at financial trading analysis. It dynamically provisions resources, executes simulations, and securely stores results. By integrating EC2 instances and Lambda functions, the system efficiently handles data processing and analysis, while GAE ensures global accessibility and user interaction. This report delves into the architecture, implementation, and performance of this cloud-based solution.

Keywords—GCP, AWS, lambda, EC2, Three white soldiers, three black crows, S3 bucket

I. Introduction

This report elaborates on the creation of a sophisticated cloud-based API that employs Monte Carlo simulations to assess financial trading strategies. Leveraging Google App Engine and Amazon Web Services, the system processes NVDA stock data to provide insights into trading risks and profitability. The architecture follows NIST SP 800-145 standards, guaranteeing a robust, scalable, and efficient cloud solution for both developers and users.

A. Developer

In accordance with NIST SP 800-145, the developer experience with this cloud platform can be summarized as follows:

In NIST SP800-145	Developer uses and/or experiences
On-demand self-service	Developers can dynamically allocate AWS resources using the API's /warmup endpoint, configuring AWS Lambda for rapid tasks or EC2 instances for more resource-intensive computations, utilizing a pre-configured AMI with essential dependencies. This automation of infrastructure setup allows developers to concentrate on coding and deployment, thereby considerably shortening the time-to-market for new features and applications.
Broad Network Access	The API, hosted on Google App Engine, is globally accessible through standard internet connections, ensuring smooth interaction from any location or device. Designed according to RESTful principles, the API can be easily integrated into diverse applications and services, enhancing both its accessibility and usability.
Resource pooling	AWS provides a shared pool of configurable resources that are dynamically allocated based on demand, maximizing efficiency and minimizing hardware management overhead. This infrastructure optimizes the use of physical resources, resulting in cost savings and enhanced efficiency. Additionally, resource pooling ensures high availability and fault tolerance, as resources can be swiftly reallocated in the event of hardware failures.
Rapid elasticity	The system automatically scales resources up or down based on demand, ensuring high performance and availability during peak times and accommodating sudden usage spikes. This capability guarantees resource availability when needed and prevents over-provisioning during lower demand periods. Auto-scaling policies can be set to trigger based on metrics such as CPU usage, memory consumption, or request rates, enabling the system to handle fluctuations in demand smoothly.
Measured service	AWS services like Lambda and EC2 are closely monitored and metered, optimizing resource allocation and minimizing costs. Detailed cost reports enable users to manage their expenses effectively, providing transparency in usage and spending. This helps developers and businesses plan their budgets more accurately. Additionally, AWS offers tools such as AWS Cost Explorer and AWS Budgets, which allow users to analyze spending patterns and set up alerts for cost overruns, ensuring better financial control.

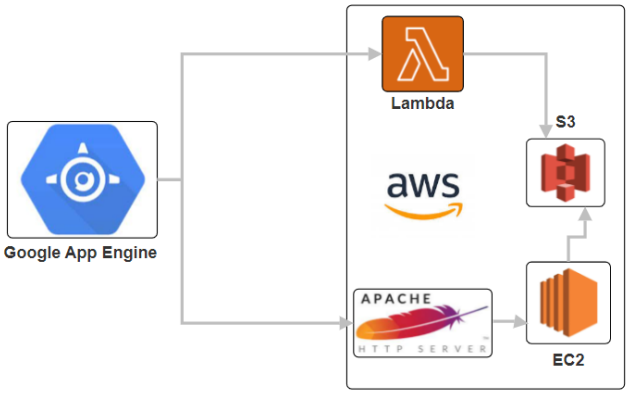
B. User

The user experience, aligned with the SaaS model and NIST SP 800-145, provides a seamless interface for financial risk analysis:

In NIST SP800-145	Developer uses and/or experiences
On-demand self-service	Users can start simulations and handle resources through the platform, choosing between EC2 and Lambda according to their computational needs. The system automates the setup and deployment of resources, offering a self-service model that allows users to carry out complex financial analyses without needing in-depth technical knowledge of the infrastructure. This approach enables users to focus on their primary business functions.
Broad Network Access	The API's global accessibility ensures that users can interact with the platform from any device, anywhere, at any time, facilitating quick decision-making based on market changes. This is particularly beneficial for financial analysts and traders who need to access the system from various locations, including while traveling or working remotely. The system's responsive design ensures compatibility with a wide range of devices, including desktops, laptops, tablets, and smartphones.
Resource pooling	Dynamic allocation of computing resources guarantees that users face no delays or shortages, even during peak demand periods, thereby optimizing cost efficiency. By distributing resources among multiple users and applications, the system achieves higher utilization rates, lowering the cost per user. This approach also enhances load balancing and boosts performance during periods of high usage.
Rapid elasticity	The platform's capacity to scale resources in line with market conditions guarantees optimal performance. This feature is essential for analyzing trading strategies in volatile market periods. The system automatically adjusts resource allocation based on real-time demand, ensuring users have the necessary computational power for their analyses. This elasticity also leads

In NIST SP800-145	Developer uses and/or experiences
	to cost savings, as users pay only for the resources they utilize.
Measured service	Users are billed solely for the resources they consume, with comprehensive usage reports enabling precise budget management. Each simulation result includes a detailed breakdown of costs and financial outcomes. This pay-as-you-go model allows users to effectively manage their expenses, eliminating the upfront costs of traditional infrastructure. Detailed usage reports provide insights into consumption patterns, aiding users in making informed decisions about their resource usage.

II Final Architecture



The API's architecture efficiently manages and executes Monte Carlo simulations for financial trading analysis using NVDA stock data, leveraging both AWS and GAE for scalability and performance:

Resource Initialization and Management:

Warmup and Provisioning: The API dynamically provisions AWS resources based on user input, including EC2 instances with pre-installed software dependencies for intensive data processing or AWS Lambda for less resource-intensive tasks. The /warmup endpoint allows users to specify the number and type of instances, which are then automatically configured and launched. This flexibility ensures that computational resources are tailored to the specific needs of each analysis, optimizing both performance and cost efficiency.

Automated Configuration: The system configures virtual servers based on user specifications, optimizing resource allocation for analysis tasks. This process includes setting up the necessary security groups, networking configurations, and ensuring access to required datasets. By automating these configurations, the system reduces manual intervention, minimizes errors, and accelerates the setup process, enabling rapid deployment of analytical tasks.

Data Processing and Analysis Execution:

Parameter Configuration: Users configure simulation parameters such as historical data period, number of simulations, transaction type, and position size. These parameters tailor each simulation to specific analytical needs, ensuring relevant and accurate results. The ability to customize these parameters allows for flexible and targeted analyses, catering to diverse financial strategies and market conditions.

User-Selected Resource Allocation: The system allows users to choose between AWS Lambda for lightweight tasks and EC2 instances for more computationally intensive analyses. This user-driven approach provides flexibility in balancing cost and performance. Lambda functions can be utilized for quick, parallel processing of smaller tasks, while EC2 instances handle larger, more complex computations based on the user's selection. This ensures that users have the appropriate resources for their specific analytical needs.

Execution and Data Storage: Simulations run to predict market behaviour and trading outcomes, with results stored in AWS S3 for secure access and further analysis. The data is stored in a structured format, facilitating easy retrieval and post-processing. This structured storage not only ensures data integrity and security but also provides a scalable solution for managing large volumes of analytical data.

Result Retrieval and Processing:

Secure Data Archiving: Analysis results are safely stored in AWS S3, offering a dependable archive for assessing risk and profitability. Utilizing S3 ensures data durability and security, capitalizing on AWS's strong storage infrastructure. This secure storage method safeguards sensitive financial data, ensuring adherence to data privacy regulations and industry standards.

Structured Access to Results: Results are retrieved from S3 and displayed via GAE, offering users structured access to detailed reports and visualizations. The API endpoints provide various ways to query and retrieve analysis results, supporting a wide range of user requirements. This structured access allows users to efficiently navigate and interpret the analytical data, facilitating informed decision-making.

Enhanced User Experience:

Interactive Visualizations: The platform includes interactive charts and visualizations to aid in the interpretation of trading strategies. These visual tools provide users with a clear and intuitive understanding of their data, enhancing decision-making capabilities. By presenting data in a visually engaging format, the platform helps users to quickly grasp complex analytical outcomes, making the insights more actionable and valuable.

II. Satisfaction of Requirements

Table I provides an overview of the system's requirements and endpoints, indicating the extent to which each has been satisfied. It categorizes the status of each requirement and endpoint as 'Met', 'Partially Met', or 'Not Met', giving a clear summary of system functionality and highlighting areas that need improvement.

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_warmup_cost /get_endpoints /analyse /get_sig_vars9599 /get_avg_vars9599 /get_sig_profit_loss /get_tot_profit_loss /get_chart_url /reset /terminate /scaled_terminated /get_audit	/get_time_cost	

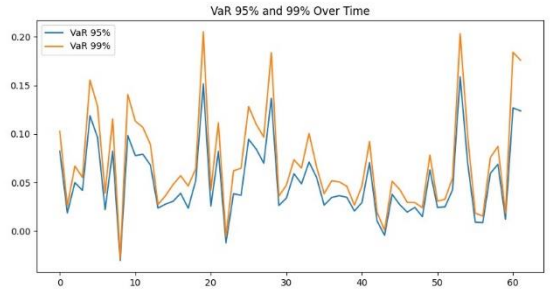
III. RESULTS

TABLE II. RESULTS

S	R	H	D	T	Avg95%	Avg99%	Cost	Time
lambda	3	101	10000	sell	0.05841	0.08131	0.00607	121.49
lambda	3	101	10000	buy	0.05923	0.08266	0.0187	375.92
lambda	1	101	20000	buy	0.05931	0.08265	0.00159	95.552
EC2	3	101	10000	sell	0.05931	0.08234	0.00091	81.929
EC2	3	101	10000	buy	0.05928	0.08271	0.0034	312.97
EC2	1	101	20000	buy	0.05930	0.08274	0.00068	185.33

The data in the table shows that allocating more resources and performing additional iterations lead to increased time and cost expenditures. Figure 2 illustrates a graph from a single system execution.

Fig2 Trend Analysis of Simulated 95% and 99% Value at Risk (VaR) Over Time



The graph titled "VaR 95% and 99% Over Time" shows the Value at Risk at 95% (blue) and 99% (orange) confidence levels over time. It highlights the fluctuations in risk, with the 99% VaR generally higher. This indicates greater risk at higher confidence levels.

IV. COSTS

A. Service Costs

AWS Lambda Costs: AWS Lambda charges $0.0000166667 per GB-second. Handling 10,000 requests with 1 GB of memory each for 1 second costs about $0.1667, offering precise cost control.

EC2 Costs: EC2 t2.micro instances cost $0.0134 per hour. Running an instance for 100 hours costs $1.34, providing predictable pricing for intensive tasks.

B. Operational Costs

AWS S3 Costs: S3 charges $0.023 per GB for storage plus transaction fees. Storing 10 GB with 10,000 GET and 5,000 PUT requests costs approximately $0.30, ensuring affordable data durability.

GAE Costs: Google App Engine charges $0.06 per instance hour and $0.12 per GB of bandwidth. Using 100 instance hours and 50 GB of bandwidth costs around $12.00, supporting scalable applications efficiently.

Based on these assumptions, running the system for 100 users actively engaging in deriving trading strategies would cost approximately $7.08. This estimate demonstrates the scalability and cost-effectiveness of leveraging cloud resources for financial analysis, with costs increasing proportionately with user activity and resource consumption.

REFERENCES

[1] Amazon Web Services, Inc., "AWS Lambda - Serverless Compute," AWS Documentation, 2024.

[2] Odeh, A., Awajan, A., Alzubi, J., "A Survey of Monte Carlo Methods for Financial Risk Management," International Journal of Advanced Computer Science and Applications, vol. 10, no. 7, pp. 261-268, 2019.