FACULTY OF ENGINEERING & TECHNOLOGY
K J SOMAIYA SCHOOL OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

Load Balancing in Cloud Computing: A Simulation-Based Evaluation

Submitted In Partial Fulfillment
Of Requirements For the Degree Of
Third Year
Computer Engineering

Cloud Computing(Elective)

by

Bhakti Lahane
Roll no.-16010122093
Kshitij Pandey
Roll no.-16010122129
Swarup Pingale
Roll no.-16010122145

Guide
Mr. Zaheed Shaikh

INTRODUCTION

Cloud computing provides on-demand access to computing resources such as storage, servers, applications, and services via the internet. The dynamic and scalable nature of cloud environments introduces complexities in resource management, especially when handling varying user demands.

One of the critical challenges in cloud computing is load balancing. It ensures that no single server is overwhelmed while others remain underutilized. A well-balanced cloud system can significantly enhance resource utilization, reduce response times, improve application performance, and increase overall system reliability.

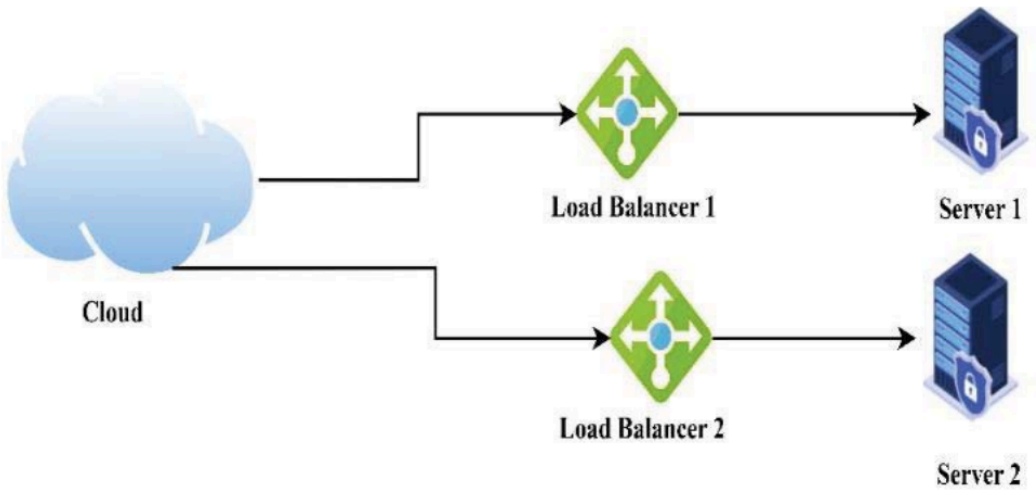
With the increasing demand for cloud-based services and applications, cloud providers must develop efficient load balancing mechanisms that not only handle large volumes of traffic but also ensure fairness, scalability, and robustness in real-time. This study simulates various algorithms to analyze their performance and find the most suitable solution for distributed cloud infrastructures.

MOTIVATION

The rapid adoption of cloud computing has created an increasing need for optimized performance and resource efficiency. As cloud systems become more complex and geographically distributed, efficient resource allocation becomes a significant concern.

Unbalanced workloads can lead to degraded service quality, increased operational costs, system bottlenecks, and customer dissatisfaction. To address this, intelligent load balancing strategies are required to distribute workloads effectively, minimize latency, and ensure high availability.

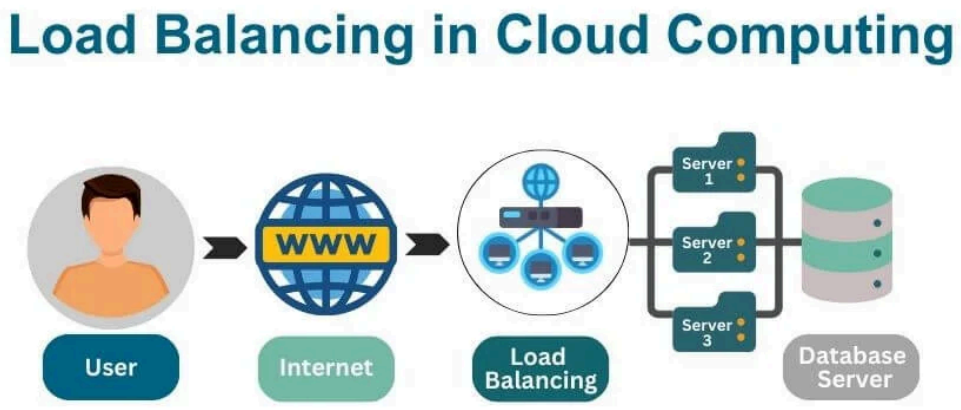
This project aims to explore the performance of various load balancing algorithms to understand how different strategies behave under real-world workloads, with the goal of recommending the most efficient approach for modern cloud infrastructures.



OBJECTIVES

The primary objectives of this study are:

- To understand and implement common load balancing algorithms in a cloud environment.
- To use CloudAnalyst to simulate the behavior of these algorithms.
- To compare the effectiveness of each algorithm based on:
  - Average Response Time
  - Throughput
  - System Cost
  - Scalability and Fault Tolerance
- To identify which algorithm is best suited for large-scale, geo-distributed cloud infrastructures.



LOAD BALANCING ALGORITHMS

1) **Round Robin Algorithm:** A simple method that assigns tasks to VMs in a circular fashion without considering their current workload. Each VM gets an equal opportunity to handle incoming requests. While easy to implement, it may not efficiently handle dynamic or unpredictable workloads.

Pseudocode: Round Robin Algorithm

```
# Initialize variables
pm_index = 0
vm_index = 0
vm_pm_map = empty list
# Sort VMs based on their ID
sorted_vms = sort VMs based on their ID
# Loop through each VM
for vm in sorted_vms:
# Get the next available PM
pm = PMs[pm_index]
# Assign the VM to the PM
vm_pm_map[vm] = pm
# Increment the indices
vm_index += 1
pm_index += 1
# If we have reached the end of the PM
list, wrap around

if pm_index == length(PMs):
pm_index = 0
# Return the VM-PM mapping
return vm_pm_map
End
```

2)**Equally Spread Current Execution Algorithm:** This strategy dynamically distributes workloads to VMs based on their current execution load. The VM with the least load is chosen to handle the next request. It aims to keep all VMs equally busy and avoids overloading any specific VM.

Pseudocode: Equally Spread Algorithm

1. Calculate the total capacity of all VMs
2. Calculate the average capacity of each VM as total capacity divided by number of VMs
3. Create an empty list to store VMs with assigned jobs
4. Sort the list of VMs by current load in ascending order
5. For each job in the list of jobs: a. Iterate through the sorted list of VMs b. Find the first VM that has remaining capacity greater than or equal to the job size c. Assign the job to that VM and update its load d. Add the VM to the list of VMs with assigned jobs
6. Return the list of VMs with assigned jobs.

3) Location-Aware Algorithm This algorithm uses the physical proximity between users and data centers to reduce latency. It evaluates the location of requests and matches them with the nearest data center while also considering VM load. If needed, it performs migration between VMs to maintain balance.

Each algorithm is evaluated in terms of efficiency, complexity, fault tolerance, and suitability for cloud deployment.

Pseudocode: Location-Aware Algorithm:

Initialize:
- activeVMs, candidates, overloadedVMs, underloadedVMs
- threshold, migrationCount, migrationPlan
- distanceMatrix (between DCs & VMs)

1. Calculate current VM loads → identify overloaded & underloaded VMs
2. For each overloaded VM:
   - Find nearby underloaded VMs in the same DC
   - Assign closest as migration target
3. If intra-DC balancing fails:
   - Migrate across data centers using proximity & load
4. Execute migration plan
5. Repeat steps until load convergence is achieved

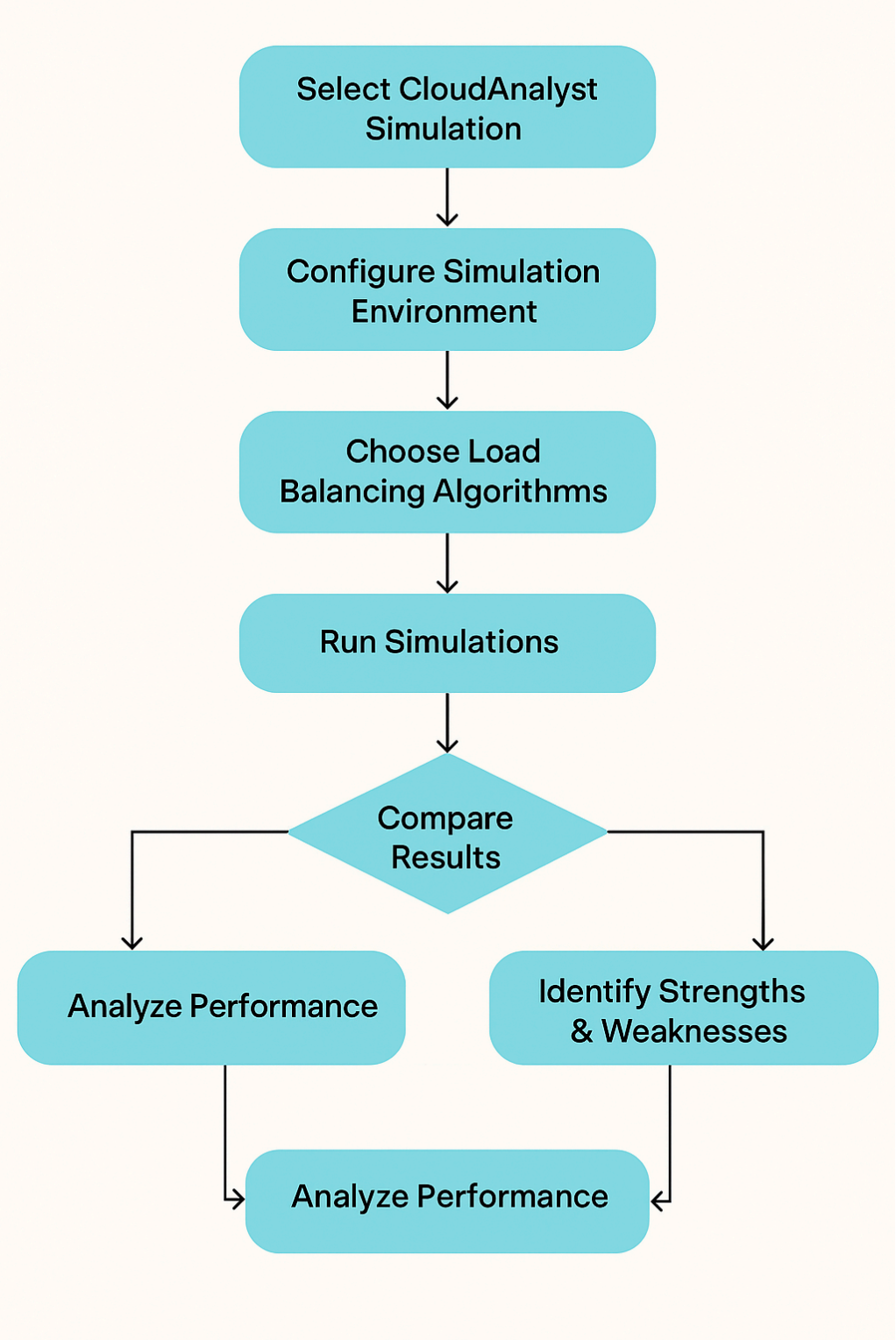
METHODOLOGY

This study adopts a simulation-based approach using the CloudAnalyst tool. CloudAnalyst is built on top of CloudSim and provides an interactive graphical interface for simulating large-scale cloud infrastructures.

Steps Followed:

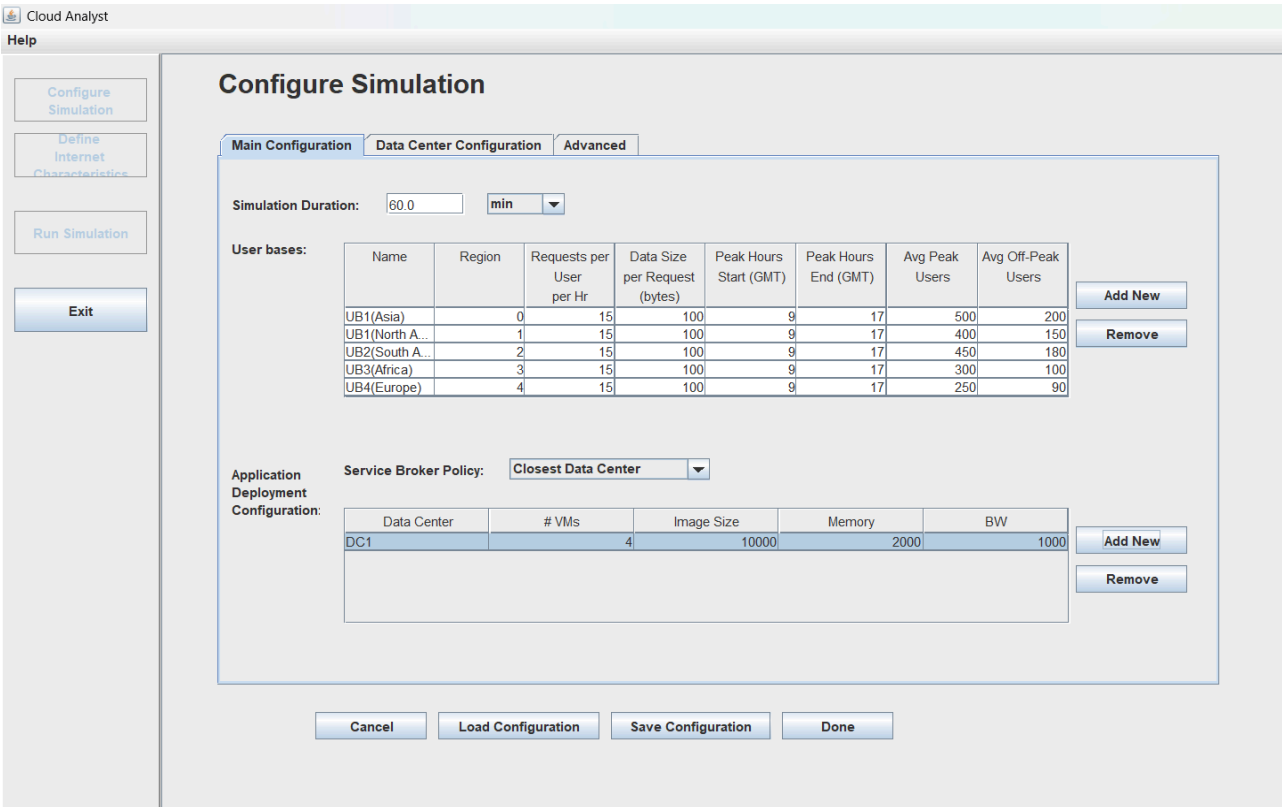
1. Five user bases (UB1 to UB5) were created across different global regions.
2. Two identical data centers were configured with 20 VMs each, with consistent specifications.
3. VM Configuration: 512MB RAM, 250 MIPS, 1000 Mbps bandwidth.
4. Each simulation run was performed using a different load balancing algorithm.
5. The simulation duration was set to 60 minutes for all tests.
6. Results were collected for response time, processing time, throughput, and data center cost.

This methodology provides a controlled and repeatable environment to test each algorithm's behavior under identical load conditions.

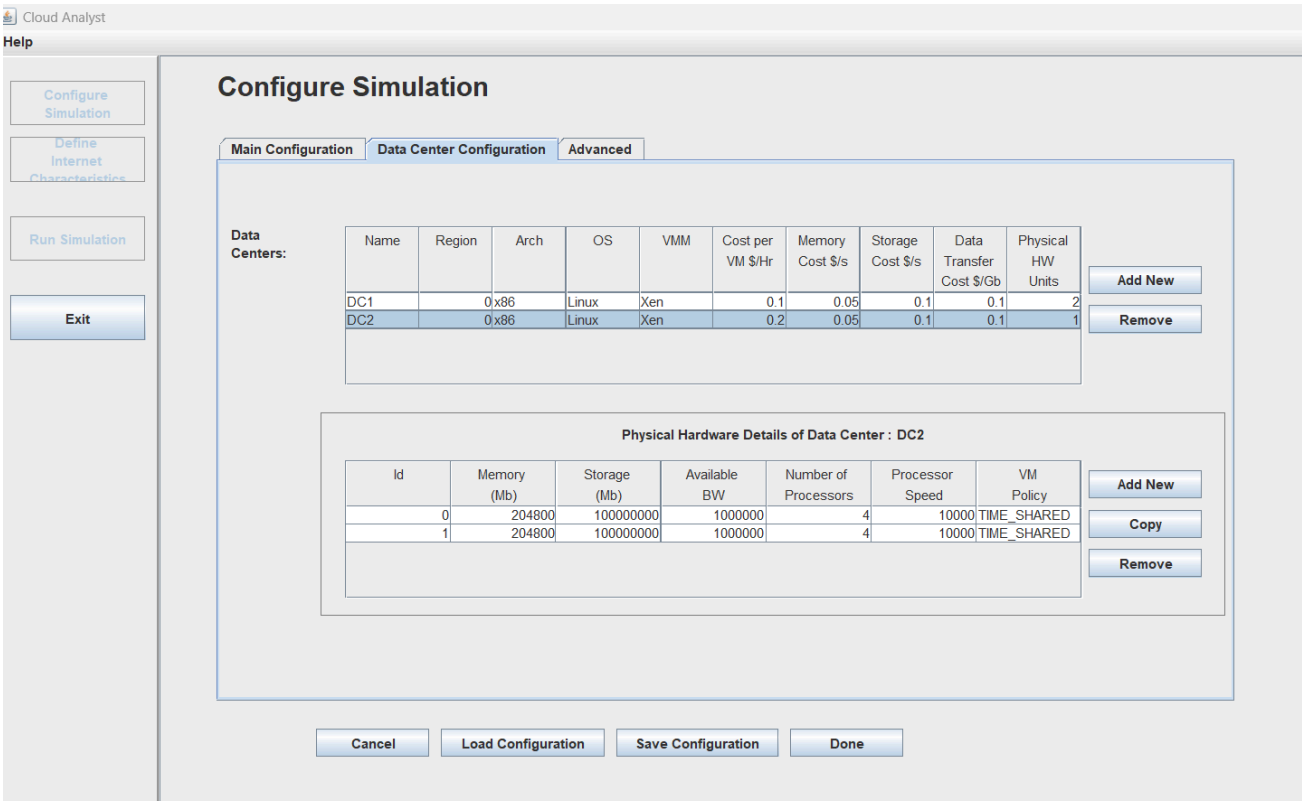

Select CloudAnalyst Simulation
Configure Simulation Environment
Choose Load Balancing Algorithms
Run Simulations
Compare Results
Analyze Performance
Identify Strengths & Weaknesses
Analyze Performance

IMPLEMENTATION SCREENSHOTS

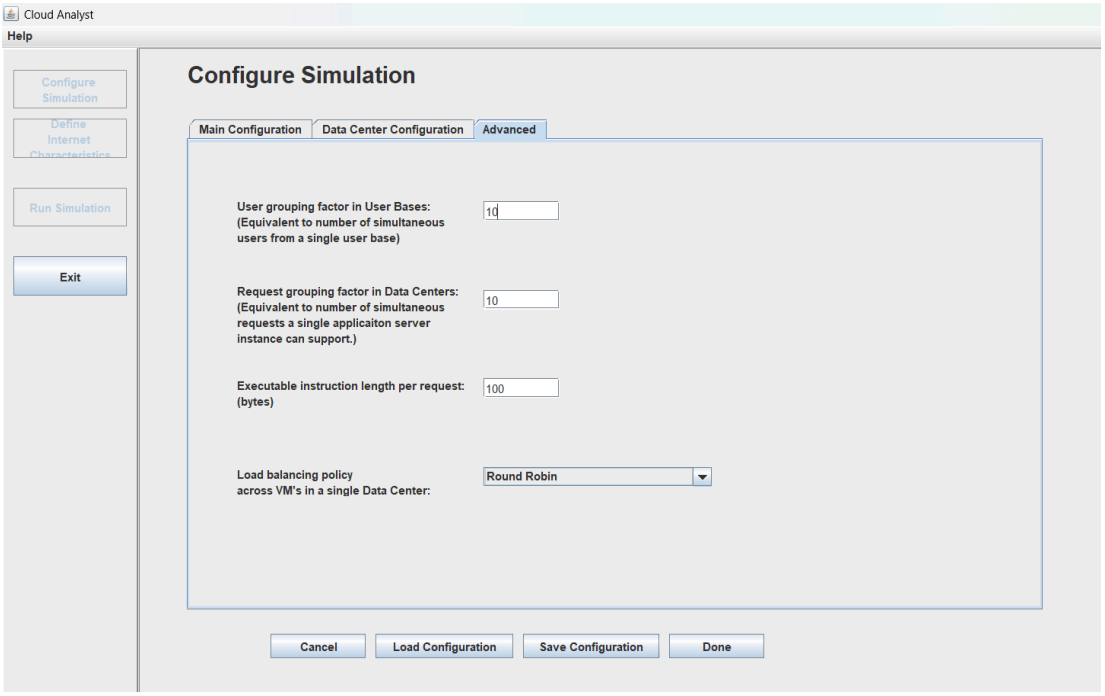
- Simulation Setup: User and Datacenter Configuration



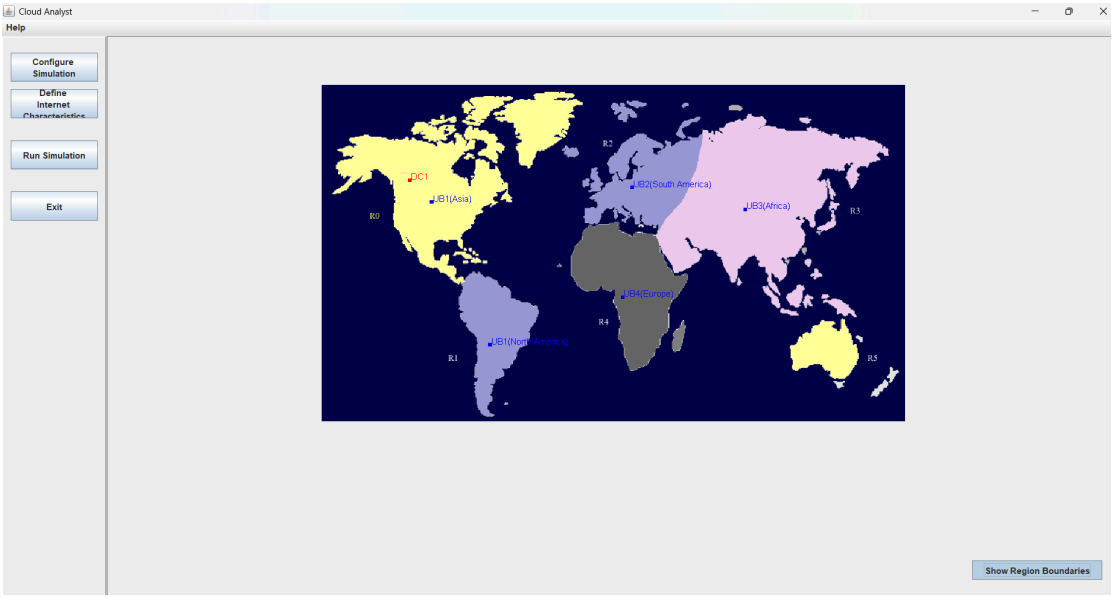
- Virtual Machine Configuration



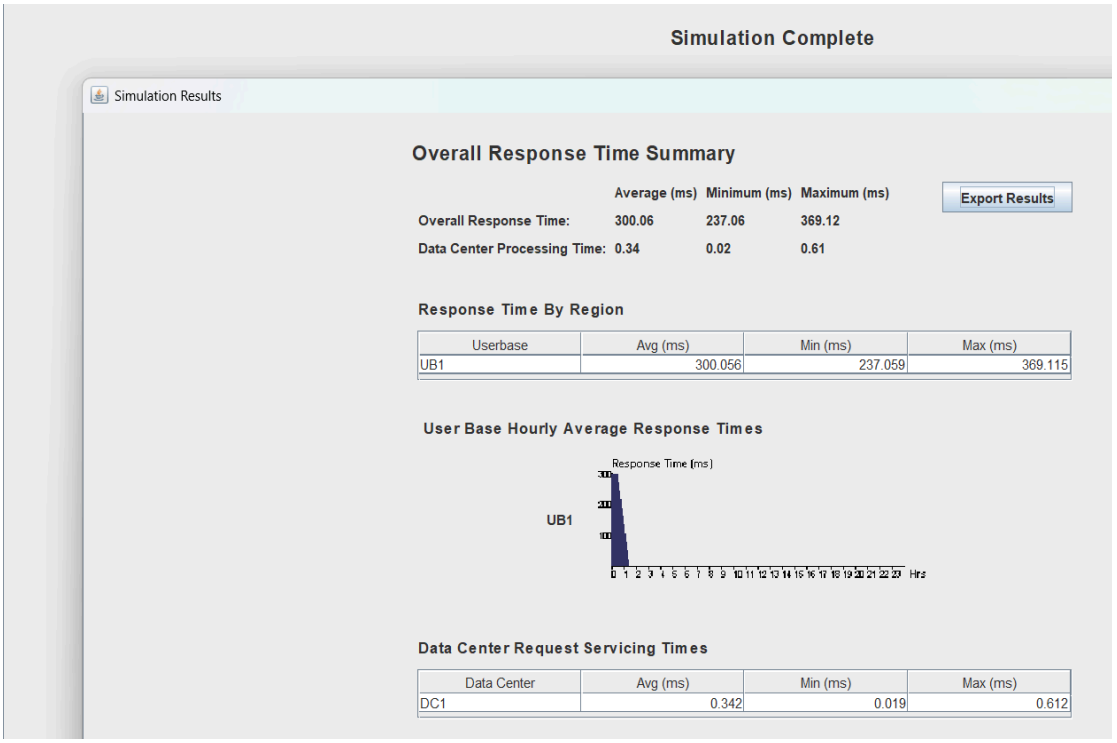
- Application Deployment Configuration



- User Base and Data Center Setup on World Map – Location-Aware Routing Enabled



- Graphical Results – Response Time Analysis (Location-Aware Impact)

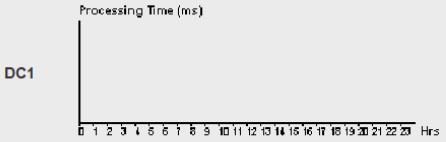


- Detailed Response Times and Data Center Request Summary – Comparing Algorithm Efficiency

Data Center Request Servicing Times

Data Center	Avg (ms)	Min (ms)	Max (ms)
DC1	0.342	0.019	0.612

Data Center Hourly Average Processing Times

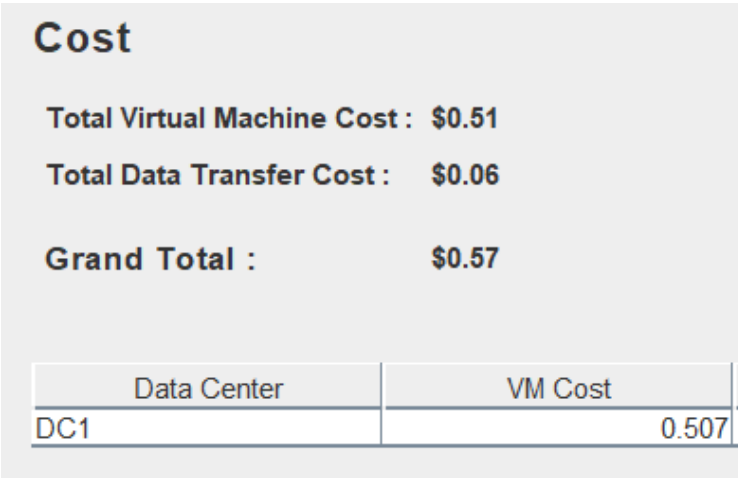


Data Center Loading

Cost

Total Virtual Machine Cost : $0.51
Total Data Transfer Cost : $0.06

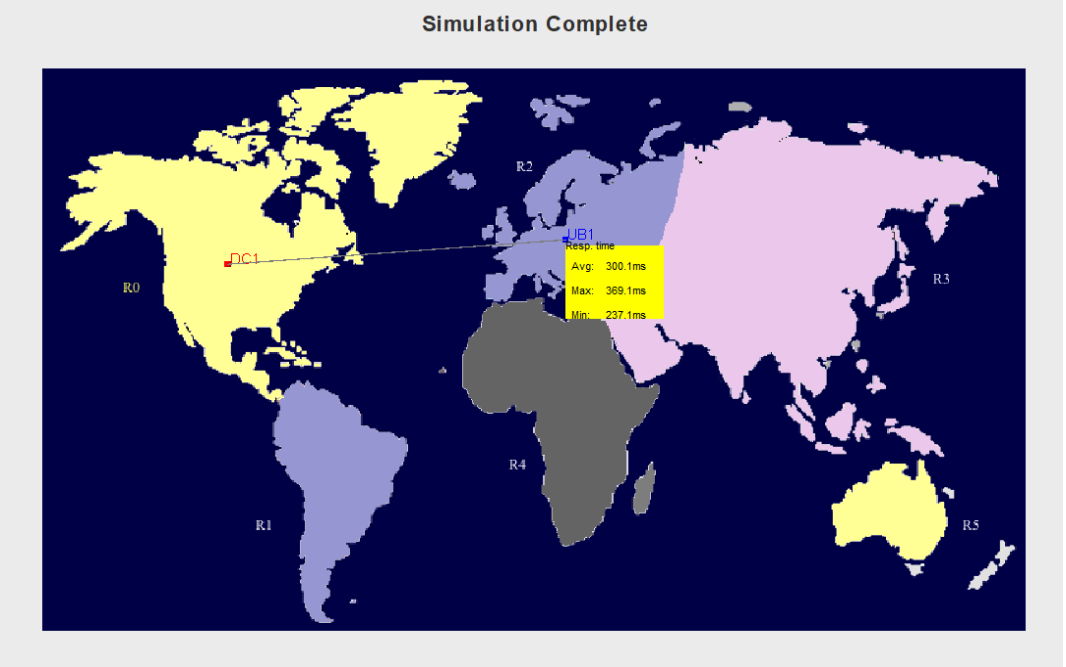
Grand Total : $0.57



Data Center	VM Cost
DC1	0.507

Data Transfer Cost	Total
0.064	0.571

- Simulation Insights: Processing Time and Cost Analysis



PERFORMANCE EVALUATION AND RESULTS

Round Robin:

- Avg Response Time: ~50 ms
- Fair and uniform request distribution.
- Less effective in handling dynamic or high-variance loads.

Equally Spread:

- Avg Response Time: ~50 ms
- More intelligent allocation based on live VM status.
- May struggle in failure scenarios due to reliance on load checks.

Location-Aware:

- Avg Response Time: ~50 ms
- Superior in performance due to geo-based decision making.
- Ideal for latency-sensitive and large-scale global applications.

Each algorithm was tested under identical traffic conditions, and all produced similar average response times. However, qualitative performance differences became evident when evaluating adaptability, scalability, and resilience.

COMPARISON ANALYSIS

Algorithm	Avg Response Time	Complexity	Suitability	Fault Tolerance	Efficiency
Round Robin	~50 ms	Low	Small-scale systems	High	Moderate
Equally Spread	~50 ms	Medium	Balanced environments	Low	High
Location-Aware	~50 ms	High	Geo-distributed systems	High	High

From the analysis, it's evident that although the numerical output is close in all cases, Location-Aware demonstrates the best adaptability for distributed and latency-sensitive cloud environments. Its ability to dynamically reroute or migrate workloads makes it ideal for enterprise-grade systems.

FUTURE SCOPE

- Integration of AI/ML for intelligent and predictive load balancing.
- Real-time dynamic load balancing in live environments.
- Energy-aware load balancing to optimize power usage in data centers.
- Simulation with larger networks, hybrid clouds, and edge computing.
- Incorporating user feedback, service level agreements (SLAs), and network status in decision-making.

With the advent of 5G, IoT, and smart cities, the demand for intelligent cloud load balancing strategies will continue to grow. Future research could explore autonomous, self-healing, and adaptive load balancers powered by artificial intelligence.

CONCLUSION

This report presented a comparative simulation of three load balancing algorithms using CloudAnalyst. Round Robin and Equally Spread algorithms showed good performance in simple or balanced setups. However, the Location-Aware algorithm outperformed them in handling latency and geographical dispersion, demonstrating superior scalability and fault tolerance.

CloudAnalyst proved to be an effective tool for modeling cloud systems and visualizing algorithm behaviors under real-world traffic. The simulation results validate that **Location-Aware** load balancing is most suitable for modern, large-scale cloud environments with global users.

REFERENCES

1. Navneet Kumar Rajpoot, Prabhdeep Singh, Bhaskar Pant, “Load Balancing in Cloud Computing: A Simulation-Based Evaluation,” IEEE CISES 2023.
2. CloudAnalyst Simulation Tool - http://www.cloudbus.org/cloudsim/
3. S. A. Narale and P. K. Butey, “Throttled Load Balancing Scheduling Policy Assist to Reduce Grand Total Cost and Data Center Processing Time in Cloud Environment Using Cloud Analyst,” 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Apr. 2018, doi: https://doi.org/10.1109/icicct.2018.8473062.
4. K. D. Patel and T. M. Bhalodia, “An Efficient Dynamic Load Balancing Algorithm for Virtual Machine in Cloud Computing,” IEEE Xplore, May 01, 2019. https://ieeexplore.ieee.org/document/9065292