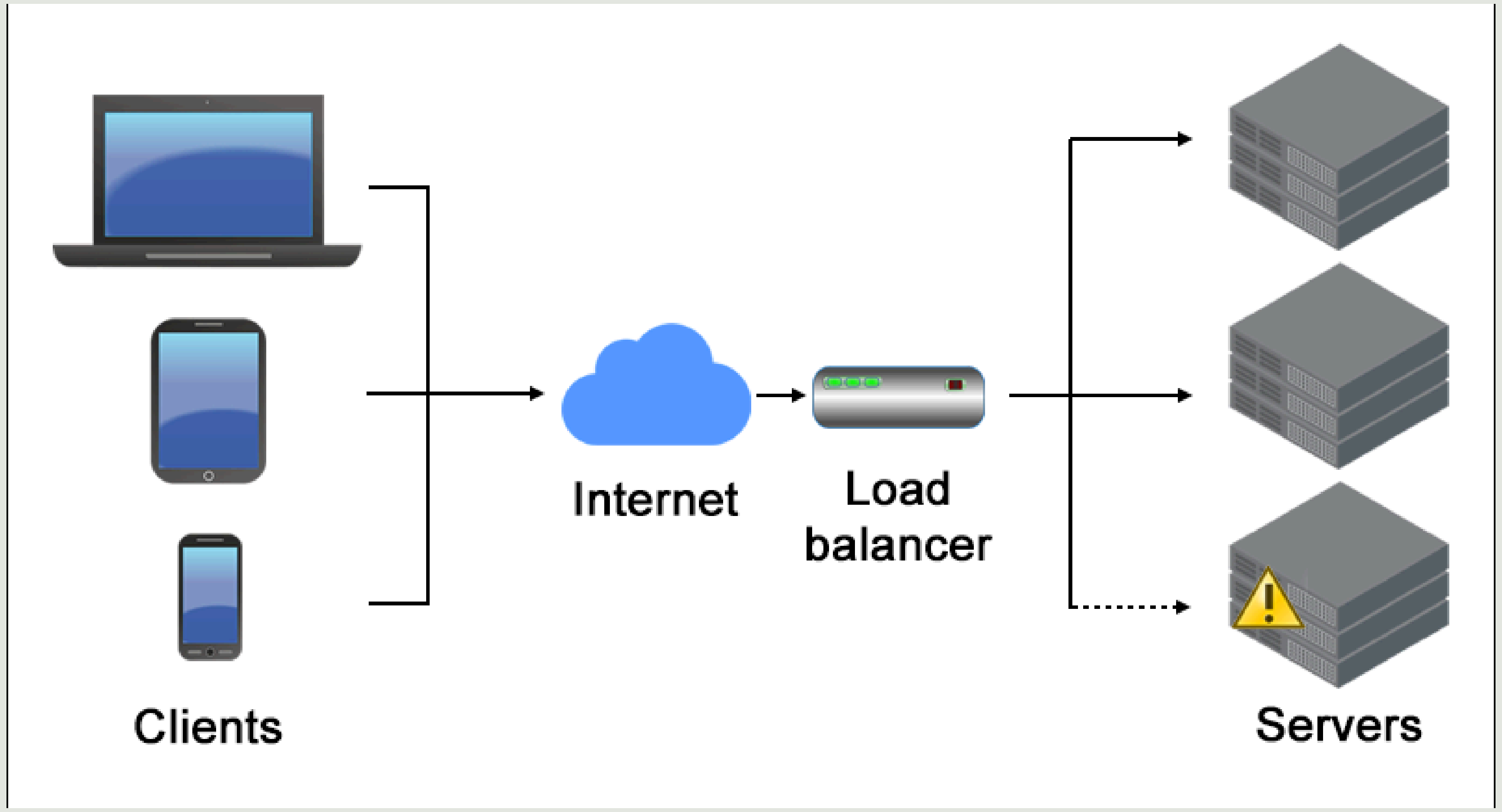
A Hybrid Load Balancing Method to Achieve Quality of Service (QoS) in Cloud-based Environments

TEAM MEMBERS :

SAKETH PABBU - S2023001019
NERELLA VENKATA SRIRAM - S20230010164
SOMESWARKUMAR BALAM - S20230010230



Introduction :

- Purpose: Develops an advanced Hybrid Load Balancing System to optimize resource allocation in cloud environments, ensuring Quality of Service (QoS) guarantees.
- Core Mechanism: Utilizes a Dynamic Reward Based (Q-Learning) policy to adaptively switch between:
    - A simple, low-overhead algorithm for efficiency during low-stress scenarios.
    - A complex, highly optimizing algorithm for enhanced performance during high-demand scenarios.
- Objective: Balances resource utilization and performance by intelligently adapting to varying cloud workloads.

Motivation :

- **Cost-Efficiency :** Ensures high server utilization ($\bar{U}$) to justify operational costs. Minimizes queue times to maintain efficient resource use.

- **Performance Degradation:** Addresses limitations of static load balancing strategies like Round Robin , which fail under heterogeneous workloads .Prevents bottlenecks and reduces Response Time Metric ($\bar{T}$_R).

- **The Overhead Dilemma:** Dynamic algorithms (e.g., ACO) provide superior QoS but incur high computational overhead.The hybrid approach mitigates overhead by selecting the appropriate algorithm based on system stress, optimizing CPU cycle usage

.

Gaps Addressed

- Static Methods (Round-Robin): Simple, low-overhead routing but suffers from Load Blindness, failing to address imbalance (σ) from variable tasks.
- Dynamic Methods (ACO): Intelligent, QoS-driven routing based on real-time state but incurs high computational overhead, inefficient when idle.

Hybrid Model Options

- Simple Threshold (50%): Switches at fixed utilization; rigid, unstable.
  - Formula: **ACO if $\bar{U} \geq L$; else RR.**
- Probabilistic Weighted: Tunable selection; manual adjustments needed, lacks adaptability.
  - Formula: **$P_{ACO} = 0$ if $\bar{U} < 30\%$; $(\bar{U} - 30\%)/40\%$ if $30\% \leq \bar{U} \leq 70\%$; 1 if $\bar{U} > 70\%$.**
- Q-Learning (Chosen): Adapts and predicts using Average Utilization (μ) and Load Imbalance (σ).
  - Formula: **$\text{Action} = \text{argmax}_{A \in \{RR,ACO\}} Q(S(\mu,\sigma),A)$.**

Objectives :

- **Multi-Algorithm Implementation:** Develop and integrate three distinct load balancing approaches - Static Round-Robin, Dynamic Ant Colony Optimization, and Adaptive Q-Learning Hybrid.
- **Intelligent Algorithm Selection:** Demonstrate that the Q-Learning component automatically switches between RR and ACO to maintain optimal performance under varying load conditions.
- **AWS Cloud Deployment:** Execute the complete simulation on AWS infrastructure using EC2 instances, with real-time logging to DynamoDB and performance monitoring through CloudWatch.
- **Performance Superiority:** Quantitatively prove that the Hybrid approach achieves significantly better response times and more balanced server utilization compared to using individual algorithms alone.
- **Scalability Verification:** Systematically test and validate that all algorithms maintain consistent performance and reliability when scaling from 20 to 50 servers handling 100-2000 tasks.

Tools , Packages and Cloud Resources :

Category	Technology	Purpose
Platform	AWS Free Tier	Cloud deployment and simulation environment
Language	Python 3.8+	Core programming language with AWS

Service	Usage
EC2 (t2.micro)	Runs simulation scripts
DynamoDB	Stores QoS metrics & results
CloudWatch	Monitors performance metrics
IAM	Manages security permissions

Package	Usage
boto3	AWS SDK for DynamoDB & CloudWatch
pandas	Data analysis & metrics calculation
matplotlib	Performance charts & graphs
numpy	Mathematical calculations & Q-Table
SimPy	Discrete-event simulation

Implementation

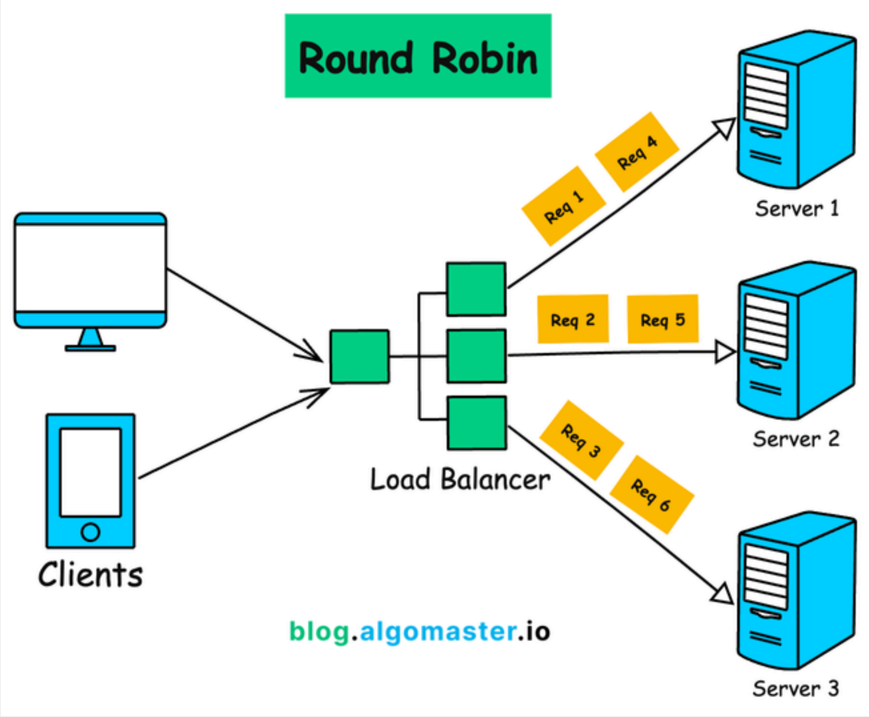
- **Launch EC2 Instance:** Start t2.micro with Amazon Linux 2, configure SSH access for remote simulation execution
- **Setup IAM Role:** Create EC2 role with DynamoDB write access and CloudWatch metrics permissions for secure service integration
- **Install Python Environment:** Install Python 3.8+, pip, and required packages (boto3, pandas, numpy, matplotlib) on EC2
- **Create DynamoDB Table:** Provision 'QoS_Metrics' table with LogID partition key to store response times and utilization metrics
- **Deploy Simulation Code:** Upload Python scripts implementing RR, ACO, and Hybrid algorithms to EC2 instance
- **Execute Load Testing:** Run simulation for 1000+ tasks across 20-50 virtual servers, logging metrics to DynamoDB via boto3
- **Monitor in CloudWatch:** Push custom metrics (response time, utilization) to CloudWatch for real-time performance monitoring
- **Generate Visualizations:** Use matplotlib to create comparison charts and save results for analysis

Load Balancing Algorithms

Static Round-Robin (RR) Algorithm :

RR is a low-overhead baseline for load distribution. It operates statically, assigning tasks sequentially without considering the servers' current load. Its purpose is to prioritize simplicity and efficiency during low-stress scenarios. This algorithm is essential for the Hybrid model as it conserves valuable CPU cycles when complex optimization isn't needed.

Core Formula: **Next Index = (Current Index + 1) (mod Total Servers)**

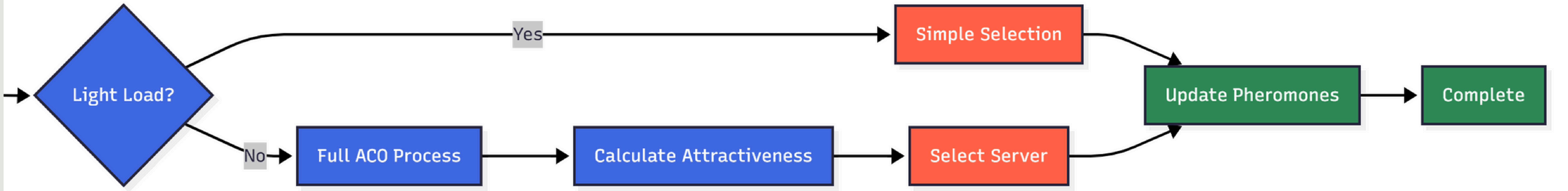


blog.algomaster.io

Dynamic Ant Colony Optimization (ACO) :

ACO provides a high-intelligence, QoS-driven dynamic solution. Inspired by ants, it selects the optimal server by weighing two factors: the server's historical success (τ / Pheromone) and its current availability (η / Heuristic). ACO is invoked by the Q-Agent when Load Imbalance (σ) is high, as it guarantees the task is routed to the single best resource to minimize latency. It maintains its intelligence by updating pheromone levels after every task.

Core Formula: **Attractiveness(Si) ∝ (Pheromonei)^α × (Heuristici)^β**



Q-Learning Decision Rule Strategy :

The Q-Learning strategy implemented is an intelligent, self-optimizing approach for task routing. The Q-Agent selects between Round Robin (RR) and Ant Colony Optimization (ACO) strategies based on the current system's load metrics (μ,σ). In the code, the agent consults a pre-defined Q-Table to find the action with the highest expected reward (Q). This policy ensures the system intelligently minimizes the usage of the high-cost ACO algorithm while maximizing overall QoS and efficiency.

A. Reward (R)

The reward reflects action quality via Quality of Service (QoS) metrics:

R = w1 × (1 / Response Time) + w2 × (Utilization)

- The w1 and w2 weights are used in the reward function to balance QoS priorities. They are calculated from the measured output data, ensuring the agent prioritizes fixing the metric that is currently performing the worst .
- Metrics sourced from AWS CloudWatch.

B. Bellman Equation

The Q-Table updates for state (S) and action (A) using:

Q(S, A) ← (1 - α) Q(S, A) + α [R + γ max_A' Q(S', A')]

- α: Learning rate (0 to 1).
- R: Immediate reward.
- γ: Discount factor (0 to 1).
- S': Next state.
- max_A' Q(S', A'): Maximum Q-value for next state’s actions.

C. Decision Logic

The Q-Agent selects the action with the highest Q-value:

If **Q(S, RR) > Q(S, ACO)**, tasks are routed using static Round Robin.

If **Q(S, ACO) > Q(S, RR),** tasks are routed using dynamic ACO.

Action selection follows:

Action = argmax_A ∈ {RR, ACO} Q(S(μ, σ), A)

Where S(μ, σ)is defined by:

μ: Average Utilization (μ) across servers.

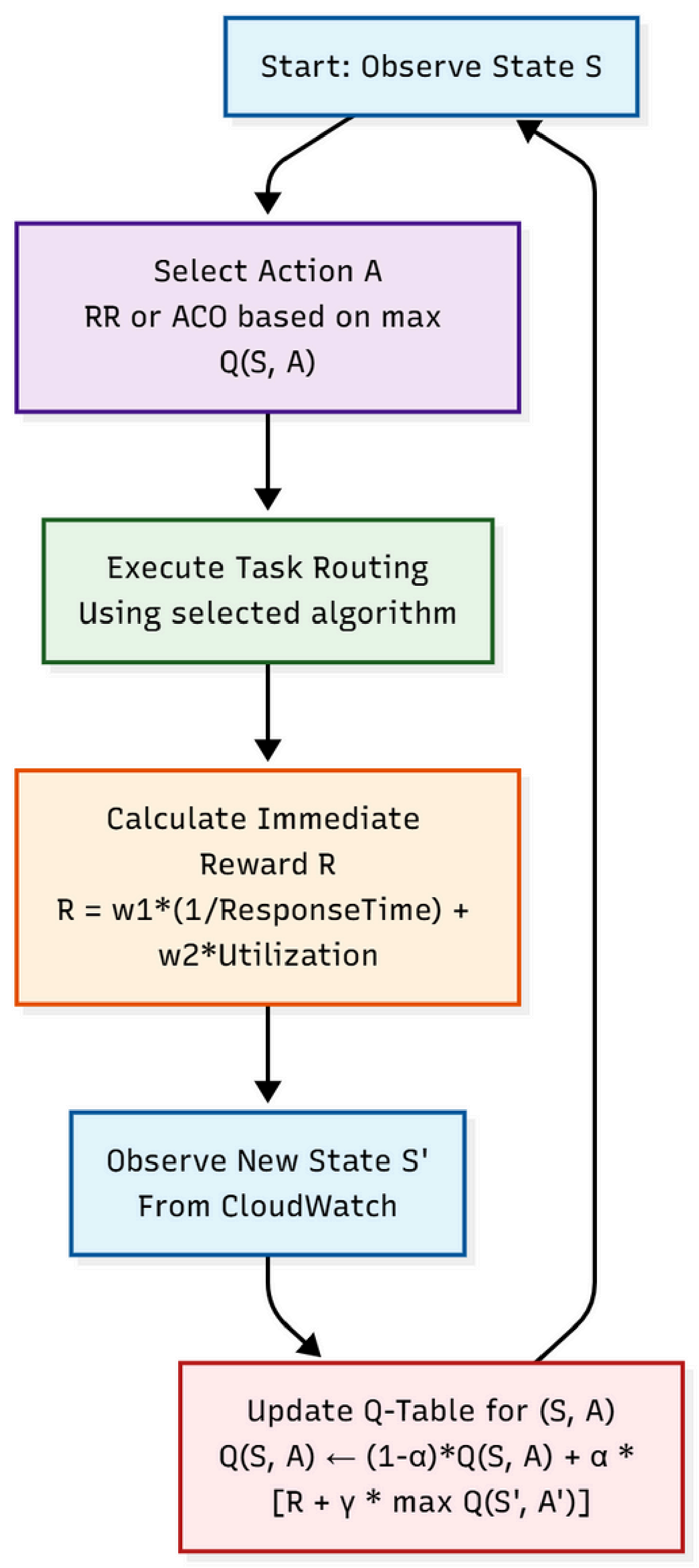
σ: Standard Deviation (σ) of utilization across servers.

D. CloudWatch Integration

AWS CloudWatch provides real-time metrics (response time, utilization) to update state S and compute reward R, enhancing decision-making and enabling performance visualization.

The Q-Learning strategy adapts dynamically:

In upscale scenarios, the Q-learning strategy adapts based on the number of servers and tasks: for low loads (e.g., 25 tasks), it initially favors RR for balanced distribution and low overhead. As load increases (e.g., to 5000 tasks), it switches to ACO to optimize routing dynamically, reducing response times and improving utilization through learned pheromone-based decisions.



Start: Observe State S

Select Action A
RR or ACO based on max
Q(S, A)

Execute Task Routing
Using selected algorithm

Calculate Immediate
Reward R
R = w1*(1/ResponseTime) +
w2*Utilization

Observe New State S'
From CloudWatch

Update Q-Table for (S, A)
Q(S, A) ← (1-α)*Q(S, A) + α *
[R + γ * max Q(S', A')]

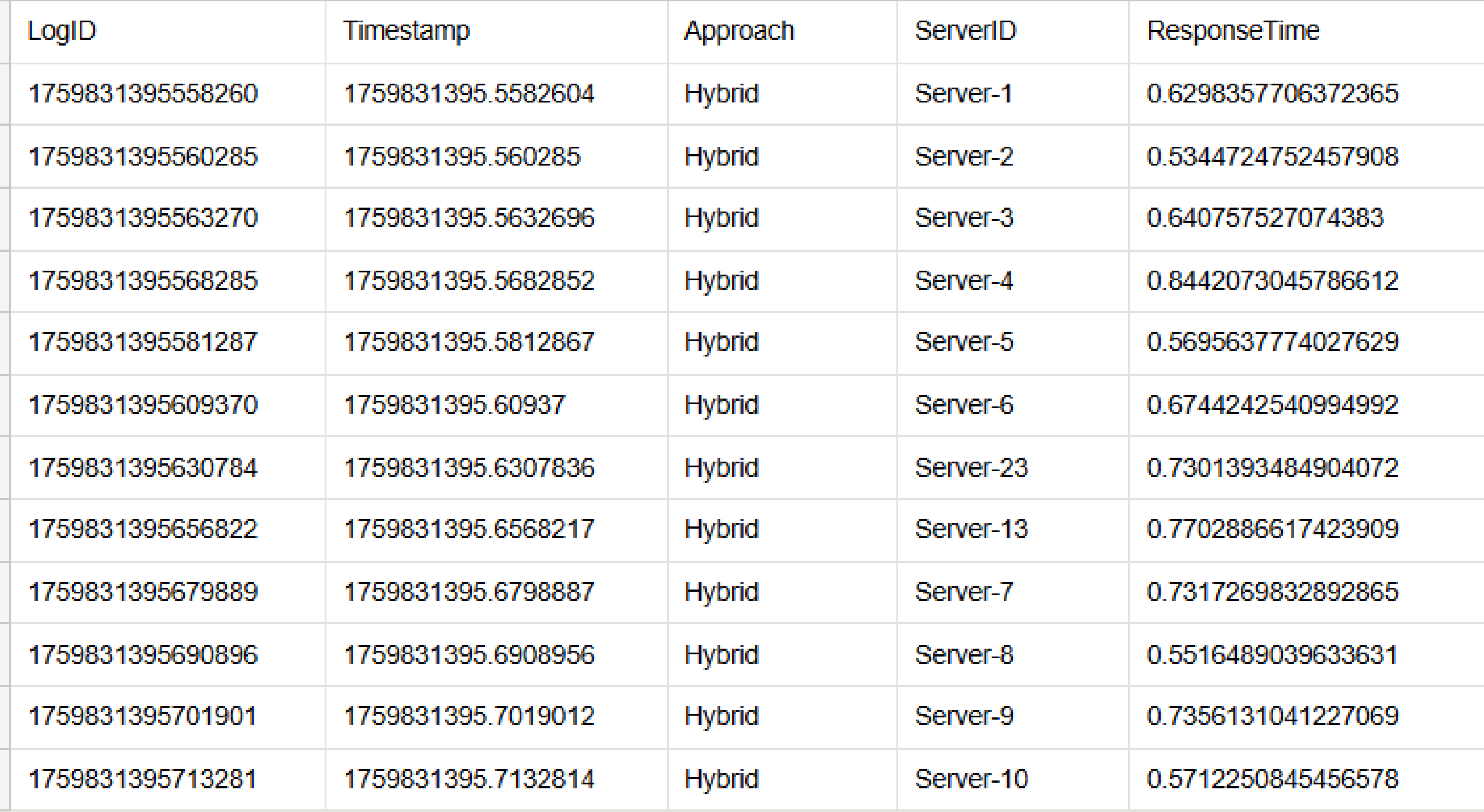
QoS Improvement and Visualization

QoS Improvement Mechanisms

1. **Algorithm Efficiency -** Maintains lower response times and higher utilization than ACO , RR by optimizing task routing with Q-Learning, balancing static Round Robin and ACO.
2. **Efficient Algorithm Switching** - Q-Learning picks Round-Robin or ACO for best response times via a reward function. If Q(S,RR) > Q(S,ACO), it uses Static Round-Robin; if Q(S,ACO) > Q(S,RR), it uses Dynamic ACO.
3. **Ensuring QoS with Load Distribution** - Prevents QoS degradation from varying task loads across 20-50 servers processing 100-2000 tasks by dynamically distributing load using Q-Learning, maintaining balanced performance.

Performance Scatter Plot

- X-Axis: Response Time (seconds) - Lower is better
- Y-Axis: Server Utilization (%) - 60-80% optimal
- Point Colors:
  - 🔵 Blue: RR
  - 🔴 Red: ACO
  - 🟢 Green: Hybrid
- Point Size: Number of tasks processed
- Purpose: Shows Hybrid's optimal balance of low response time and utilization.



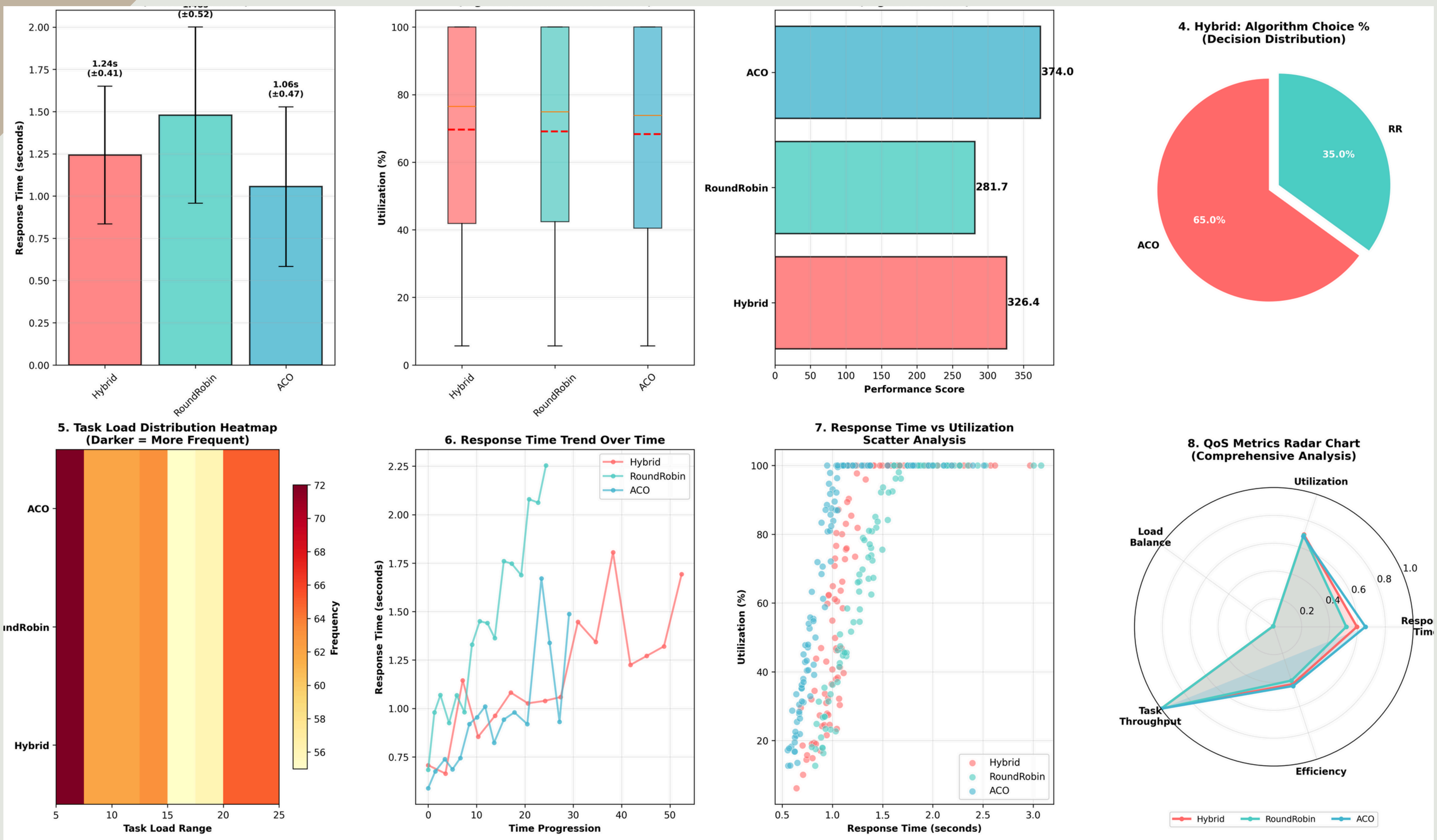
LogID	Timestamp	Approach	ServerID	ResponseTime
1759831395558260	1759831395.5582604	Hybrid	Server-1	0.6298357706372365
1759831395560285	1759831395.560285	Hybrid	Server-2	0.5344724752457908
1759831395563270	1759831395.5632696	Hybrid	Server-3	0.640757527074383
1759831395568285	1759831395.5682852	Hybrid	Server-4	0.8442073045786612
1759831395581287	1759831395.5812867	Hybrid	Server-5	0.5695637774027629
1759831395609370	1759831395.60937	Hybrid	Server-6	0.6744242540994992
1759831395630784	1759831395.6307836	Hybrid	Server-23	0.7301393484904072
1759831395656822	1759831395.6568217	Hybrid	Server-13	0.7702886617423909
1759831395679889	1759831395.6798887	Hybrid	Server-7	0.7317269832892865
1759831395690896	1759831395.6908956	Hybrid	Server-8	0.5516489039633631
1759831395701901	1759831395.7019012	Hybrid	Server-9	0.7356131041227069
1759831395713281	1759831395.7132814	Hybrid	Server-10	0.5712250845456578

Real-time Metrics

- Live Graphs: Real-time response time and utilization plots.
- CloudWatch Dashboards: AWS monitoring integration.
- Performance Scores: Real-time QoS effectiveness.

Performance Charts

Performance Score , Load Heatmap , Time Trend , Scatter Analysis , Radar Chart:



Response Time (seconds)
1.24s (±0.41)
(±0.52)
1.06s (±0.47)
Hybrid
RoundRobin
ACO
Utilization (%)
Performance Score
374.0
281.7
326.4
4. Hybrid: Algorithm Choice %
(Decision Distribution)
RR
35.0%
65.0%
ACO
5. Task Load Distribution Heatmap
(Darker = More Frequent)
undRobin
Task Load Range
Frequency
6. Response Time Trend Over Time
Response Time (seconds)
Time Progression
7. Response Time vs Utilization
Scatter Analysis
Utilization (%)
Response Time (seconds)
8. QoS Metrics Radar Chart
(Comprehensive Analysis)
Utilization
Load Balance
Respo Tim
Task Throughput
Efficiency

Individual Contributions :

Member	Combined Responsibilities
Saketh Pabbu	Round Robin & ACO algorithm implementation . AWS CloudWatch monitoring system . UI development & user interaction .
Nerella Venkata Sriram	Complete Q-Learning algorithm & state management . AWS EC2 infrastructure setup . Q-value update rules & reward system
Someshwar Kumar Balam	SimulatedServer class & workload generator . AWS DynamoDB database management . Data visualization by fetching from DynamoDB using pandas/matplotlib .

Conculsion :

Validation of Adaptive Intelligence (QoS Optimization): The Q-Learning policy provides a superior mechanism for load management. By integrating the learned policy (Q-Table), the system intelligently minimizes overhead (RR is selected at low load) while maximizing performance (ACO is selected when σ or μ is high), thus achieving an optimal balance between $\bar{T}R$ and $\bar{U}$.

References :

Shrimal, M., Sharma, A. K., & Patel, M. (2024). A Novel Hybrid Load Balancing Method to Achieve Quality of Service(QoS) in Cloud-based Environments. 2024 3rd International Conference on Sentiment Analysis and Deep Learning (ICSADL)

THANK YOU ....