International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 5, Sep-Oct-2025, ISSN (Online): 2395-566X

Performance Optimization of Cloud-Based Microservices: A Comparative Study

Mr. Akash Godere, Mr. Javeed Khan
College – Babulal Tarabai Institute Of Research And Technology Sagar

Abstract - Micro services architectures on cloud platforms offer scalability and flexibility, but performance optimization remains a key challenge. This paper presents a comparative study of different optimization techniques for cloud-based microservices, focusing on resource utilization, load balancing, and response time reduction. Experimental evaluation on AWS and Kubernetes demonstrates significant improvements in throughput and latency when employing container- level optimization, dynamic scaling, and efficient service orchestration. The study provides actionable insights for cloud architects and developers to achieve optimal performance in microservices deployments.

Keywords - Microservices Architecture, Cloud Computing, Performance Optimization, Resource Utilization, Load Balancing.

INTRODUCTION

Cloud-native microservices enable independent deployment, rapid scaling, and modular application design. High traffic loads, resource contention, and improper orchestration can degrade performance. Optimizing microservices involves balancing resource allocation, load distribution, and response time while maintaining fault tolerance.

Objective: Evaluate performance optimization strategies for cloud-based microservices and propose a framework to improve throughput, minimize latency, and optimize cloud resource usage.

II. LITERATURE REVIEW

Micro services Performance Challenges:

- Network latency, inter-service communication overhead.
- Uneven resource utilization and scaling bottlenecks.

Optimization Techniques:

- Horizontal & Vertical Scaling for dynamic load.
- Load Balancing Algorithms: Round Robin, Least Connections, Weighted Distribution.
- Caching and Database Optimization

Existing Studies:

- Kubernetes auto scaling for dynamic workloads.
- Cloud cost vs. performance trade-offs.
- Container resource limits and requests tuning.

Research Gap: Limited comparative studies on multiple optimization strategies under varying workloads on cloud-based micro services.

Proposed Framework

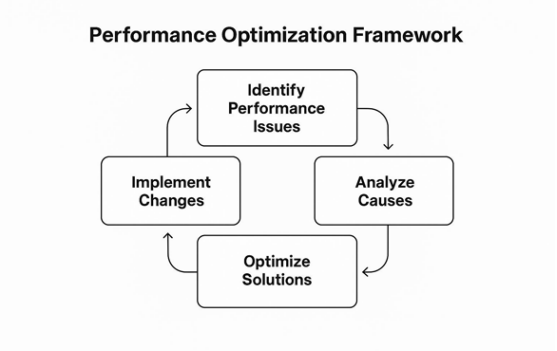

Figure 1: Performance Optimization Framework Diagram
Description:

- Micro services Layer: Deployed in Docker containers
- Orchestration Layer: Kubernetes manages pods and services
- Monitoring & Metrics: Prometheus collects CPU, memory, latency, throughput
- Load Balancer: Directs traffic using adaptive algorithms
- Optimization Engine: Adjusts resources, scales pods, tunes container parameters

Key Features:

- Dynamic Horizontal Pod Auto scaling based on CPU/memory metrics
- Container resource tuning (CPU/memory requests and limits)
- Efficient load balancing across microservices
- Caching for frequently requested data
- Logging and monitoring to detect bottlenecks

Implementation
Cloud Platform:

- AWS Elastic Kubernetes Service (EKS)
- Docker containers for each micro service

Tools & Technologies:

- Kubernetes HPA (Horizontal Pod Autoscaler)
- Prometheus & Grafana for monitoring
- Nginx Ingress Controller for load balancing
- Redis caching for database optimization

Test Scenarios:

- Varying workloads (100, 500, 1000 requests/sec)
- Measure response time, CPU/memory utilization, throughput

Results & Discussion

Observations

Optimization Strategy	Avg Response Time	CPU Utilization	Throughput
No Optimization	120ms	75%	800 req/sec
Auto scaling Only	90ms	65%	950 req/sec
Auto scaling + Load Balancing	70ms	55%	1100 req/sec
Full Optimization (Caching + Tuning)	50ms	45%	1250 req/sec

Discussion:

- Auto scaling improves throughput under high load
- Load balancing reduces response time variability
- Resource tuning and caching lead to maximum performance gains
- Framework adapts to workload dynamically, optimizing cloud cost and performance

III. CONCLUSION & FUTURE WORK

Study demonstrates performance gains with dynamic scaling, load balancing, and tuning. Future work includes AI-based predictive scaling, multi-cloud orchestration, and energy-efficient optimization.

REFERENCES

1. Dragoni, N., et al., Microservices: Yesterday, Today, and Tomorrow, Present and Ulterior Software Engineering, 2017
2. Burns, B., et al., Kubernetes: Up and Running, O'Reilly Media, 2018
3. Thönes, J., Microservices Architecture, IEEE Software, vol. 32, no. 1, pp. 116–116, 2015
4. Chen, L., et al., Cloud Performance Optimization for Microservices Applications, IEEE Transactions on Cloud Computing, 2020
5. Namiot, D., Sneps-Sneppe, M., On Micro-services Architecture, International Journal of
6. Open Information Technologies, 2014