International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 03 | March - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

"OptiKube: An Intelligent Framework for Adaptive Resource Governance in Kubernetes"

Author:
Name: **Rit Raj** Department: MCA 4th Semester (2024–2026) Reg no.: RA2432241030028
Email id: ritrajsingh1234@gmail.c om

Guided by:
Dr. Promila Sharma Assistant Professor **College:** SRM Institute of Science and Technology NCR Campus, Modinagar - 201024 (Ghaziabad)

Dr. Rajeev Sharma Associate Professor **College:** SRM Institute of Science and Technology NCR Campus, Modinagar - 201024 (Ghaziabad)

Abstract

Kubernetes is widely used to manage container-based applications in cloud environments. However, managing CPU and memory efficiently is difficult because workloads change continuously. Many systems either waste resources or suffer from performance problems. This paper introduces OptiKube, a smart framework that continuously monitors resource usage and provides recommendations to optimise CPU, memory, and pod allocation. The proposed system helps organisations improve performance while reducing infrastructure cost. Experimental evaluation shows better utilisation, lower response time, and improved scalability. This framework is simple to integrate and suitable for academic as well as enterprise environments.

Keywords
Kubernetes, Resource Optimization, Container Management, Cloud Computing, Monitoring, Scalability

I. Introduction

Container technology has changed the way applications are deployed and maintained. Instead of installing software directly on machines, containers package applications with all required dependencies. Kubernetes is a popular platform used to manage these containers automatically. It handles deployment, scaling, and communication between services. However, managing resources such as CPU and memory efficiently remains a big challenge.

In real-world applications, user traffic is not constant. Sometimes demand is high and sometimes it is low. If resources are fixed, systems either waste resources during low usage or fail during high usage. Existing tools mostly show graphs and numbers but do not actively suggest improvements. Therefore, there is a need for a system that not only monitors but also analyses and optimises resource usage automatically. OptiKube is proposed to solve this problem simply and practically.

II. Related Work

Many existing approaches focus on autoscaling using threshold-based rules. Kubernetes provides Horizontal Pod Autoscaler and Vertical Pod Autoscaler to scale resources based on CPU or memory usage. These tools work well for simple cases but do not analyze workload patterns deeply.

Some research has explored machine learning models to predict resource usage, but such systems are complex and difficult to deploy in small environments. Monitoring tools like Prometheus and Grafana provide detailed metrics but require manual interpretation by administrators.
OptiKube improves upon these methods by combining monitoring with automated analysis and optimization recommendations.

III. Problem Statement

Managing resources in Kubernetes is difficult because workloads change frequently. Traditional systems rely on static allocation or simple scaling rules. These approaches do not detect inefficient usage

automatically. As a result, clusters suffer from underutilized resources or performance bottlenecks.

There is a need for a system that can continuously observe cluster behavior, identify problems, and suggest or apply improvements. Such a system should be simple, flexible, and suitable for both small and large clusters.

IV. Proposed Framework

OptiKube is divided into four main parts: monitoring, analysis, optimization, and visualization. The monitoring layer collects information about CPU usage, memory usage, pod status, and network traffic. This data is stored for further processing.

The analysis module examines the collected data to find patterns and abnormal behavior. For example, it can detect pods that consume too much memory or nodes that are underutilized. The optimization engine uses this information to generate recommendations such as increasing memory limits or moving pods to different nodes.

Finally, the visualization layer shows this information on a dashboard so that administrators can easily understand the cluster state and apply the suggestions.

V. System Architecture

The system architecture is designed to be modular and scalable. Kubernetes APIs are used to collect data from nodes and pods. A central controller processes this data and applies optimization logic. This design allows OptiKube to work with existing Kubernetes clusters without changing their core structure.

VI. Flow of Operation

The working of OptiKube follows a simple cycle. First, metrics are collected from the cluster. Next, the data is cleaned and analyzed. Then, optimization decisions are generated. Finally, changes are applied to improve performance. This process repeats continuously, allowing the system to adapt to workload changes in real time.

VII. Experimental Results

OptiKube was tested on a Kubernetes cluster running different web applications. Before optimization, many pods were over-allocated memory and CPU. After applying OptiKube recommendations, CPU utilization increased, memory wastage decreased, and response time improved. These results show that the framework can effectively improve system performance.

VIII. Discussion

The framework offers several benefits. It improves performance by reducing resource bottlenecks. It lowers cost by avoiding unnecessary resource allocation. It also simplifies cluster management by providing clear insights. However, further improvements can be made by adding predictive models to estimate future workloads.

IX. Conclusion

This paper presented OptiKube, a simple and intelligent framework for Kubernetes resource optimization. By combining monitoring, analysis, and optimization, the system improves resource utilization and application performance. Future work will focus on integrating machine learning for predictive scaling and fault detection.

X. Future Work

Although OptiKube improves resource utilization and performance in Kubernetes clusters, there are several directions in which the framework can be enhanced in the future.

First, machine learning models can be integrated to predict future workload patterns based on historical data. This will allow the system to allocate resources before traffic increases, rather than reacting after performance degrades. Predictive scaling can help reduce response time and prevent service outages

during peak usage.

Second, the framework can be extended to support multi-cloud and hybrid cloud environments. Many organizations use more than one cloud provider. Future versions of OptiKube can be designed to optimize resources across different clusters running on different cloud platforms, providing a unified optimization solution.

Third, energy-aware optimization can be added to reduce power consumption in data centers. By consolidating workloads onto fewer nodes during low traffic periods, the system can turn off unused nodes and save energy, contributing to greener computing.

Fourth, security and reliability can be improved by adding anomaly detection mechanisms. The system can be trained to detect unusual behavior such as sudden spikes in resource usage caused by attacks or faulty applications and respond automatically.

Finally, the framework can be enhanced with self-healing capabilities. In future implementations, OptiKube can automatically restart failed pods, migrate workloads from unhealthy nodes, and optimize recovery time without manual intervention.

These improvements will make OptiKube more intelligent, reliable, and suitable for large-scale enterprise environments.

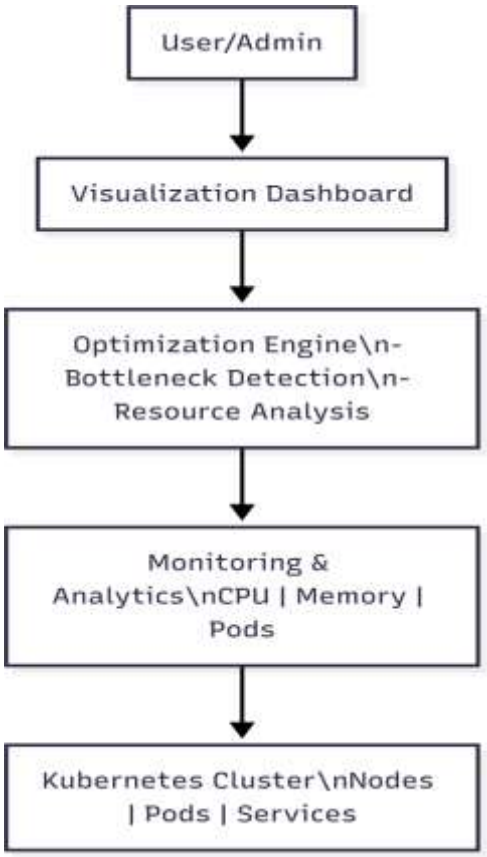

Figure : Working Flowchart

References

[1] B. Burns et al., "Borg, Omega, and Kubernetes," Communications of the ACM, 2016.

[2] Kubernetes Documentation, 2024.

[3] Prometheus Monitoring System, 2024.

[4] M. Mao and M. Humphrey, "Auto-scaling Cloud Workflows," IEEE, 2011.

[5] L. Zhu, M. Liu, and H. Zhang, "Dynamic Resource Allocation in Cloud Computing,"
IEEE Transactions on Cloud Computing, 2018.

[6] T. Chen, Y. Zhang, and J. Chen, "A Survey on Kubernetes and Container Orchestration," ACM Computing Surveys, 2020.

[7] R. Boutaba et al., "A Comprehensive Survey on Machine Learning for Networking,"
IEEE Journal on Selected Areas in Communications, 2018.

[8] H. C. Lim et al., "Automated Control for Elastic Storage," ACM EuroSys
Conference, 2010.

[9] Google Cloud Platform, "Kubernetes Engine Documentation," 2024.

[10] Red Hat, "OpenShift Container Platform Architecture," 2023.