Dynamic Scaling of Machine Learning Workloads: A Comparative Study of On-Prem and Cloud-Based Containers

Vinay Kumar Deeti, Arrowstreet Capital, Limited Partnership, USA

Abstract:

This work aims to provide a thorough study of dynamic scaling mechanism for machine learning (ML) workloads, thereby stressing the operational trade-offs between on-site and cloud-based containerized systems. Under different workloads, this study primarily addresses performance elasticity, resource consumption efficiency, orchestration delay, and cost-effectiveness.

Keywords:
dynamic scaling, machine learning workloads, containers, Kubernetes, on-premises infrastructure, cloud computing, auto-scaling, resource orchestration, workload elasticity, containerized ML systems

1. Introduction

Fast growing machine learning (ML) workloads have transformed several industries and driven businesses toward scalable, flexible, adaptive computing architectures. As machine learning workloads increase more complex and data-intensive, monolithic IT systems cannot keep up. In machine learning, data preparation, feature extraction, model training, and inference demand both time and resources. Among other purposes, storage, networking, and real-time processing need for scalable infrastructure.

Dynamics scaling helps to ensure efficient resource allocation depending on demand variations in modern ML pipelines. Scaling systems provide and deprovice based on CPU, memory, and job completion. Operating fast and cheap, ML algorithms may scale up or down with computational demand as resources are only used as needed. Dynamic ML workload

scaling lets companies regulate demand spikes without underusing resources at slower periods, hence maximizing efficiency and cost.

Docker and other containerizing methods have dramatically transformed machine learning model deployment and orchestration. For consistent application, programs and dependencies are kept in a portable, light-weight, isolated container. Containerized architecture is used in methodologies of machine learning to simplify system management. In containers, scalability calls for Kubernetes. On-site and in the cloud, Kubernetes provides scalability, high availability and resource economy, as well as containerized application deployment and management.

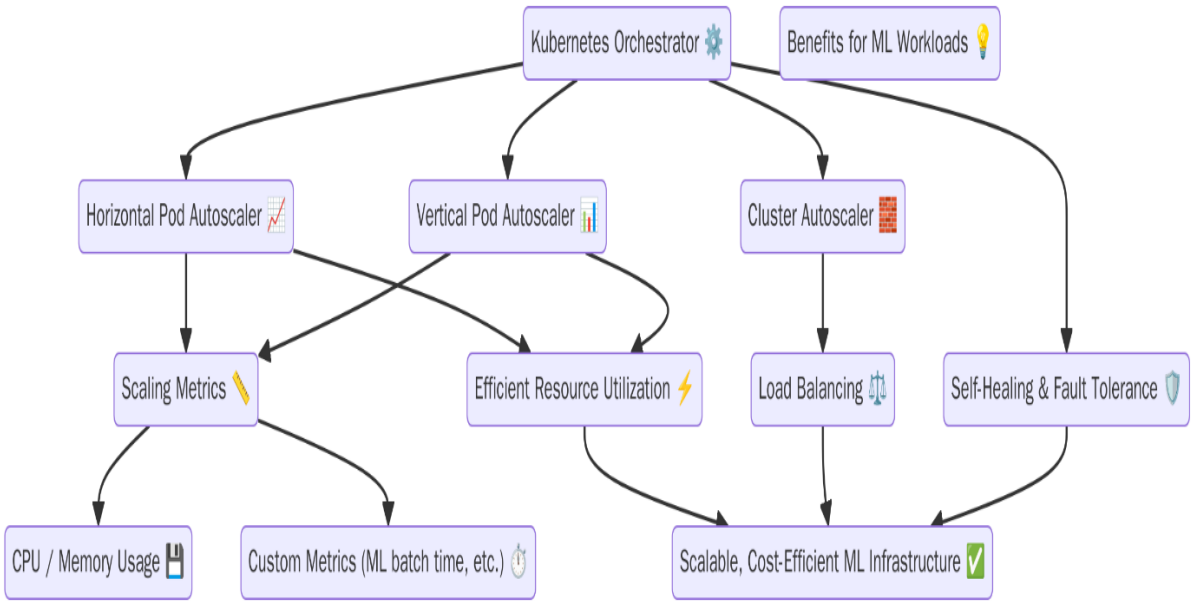
Cloud computing changes the distribution of ML tasks. From AWS, GCP, and Azure, scalable, adaptable on-demand resource providing solutions exist. While legal, financial, and data security concerns preclude many businesses from using on-site technology, cloud infrastructure is low-cost and adaptable. For companies with high latency or regulatory constraints, on-site solutions might enhance infrastructure and data management. These features make on-site, rather than cloud-based container infrastructure operating costs, scalability, and machine learning application performance assessment very essential. Based on machine learning task, we investigate dynamic scaling choices for on-site and cloud containerized systems. Two infrastructues will be evaluated in terms of performance in ML tasks, resource economy, and cost-efficiency. The advantages and drawbacks of every deployment strategy in real-world machine learning systems will guide your decision on the appropriate infrastructure for growing machine learning initiatives. Extensive understanding of scalable ML systems might enable businesses to use machine learning models in different infrastructure environments.

2. Related Work and Technical Background

Especially for machine learning applications, dynamic scaling is essential for resource management in containerized systems. Scaling automatically changes resources to match the demand, therefore maintaining computer systems adaptable and fairly priced. Most dynamic scaling systems use pod autoscaling both horizontally and vertically. Usually using HPA, Kubernetes changes pod numbers based on CPU and memory use. Conversely, VPA adjusts

container CPU and memory to demand. These techniques maximize resource consumption by performing ML tasks without overloading infrastructure.

Kubernetes is the principal container organizing the technologies accessible for large-scale systems. Container deployment, scalability, and administrative automation determines a lot about ML infrastructure. Automatic load balancing, self-healing, and scheduler-efficient resource allocation enable dynamic scalability in Kubernetes. Whereas Kubernetes Horizontal Pod Autoscaler adds pods to execute a service, Cluster Autoscaler dynamically resizes the cluster by adding or removing nodes to maximize computing resources based on demand. Custom metrics allow Kubernetes users expand based on application-specific goals such ML training batch completion times instead of resource allocation.



By means of autoscaling, machine learning infrastructure optimization studies assist to reduce resource waste and enhance system efficiency. Most ML model scaling projects optimum GPU and CPU resources by using cloud-native technologies. Data and model parallelism are studied in distributed ML training task scaling by Zhan et al. (2020) and Yu et al. (2021). To decrease idle resource and training time in high-throughput or data-intensive operations, Kubernetes and other container orchestragement systems may expand both horizontally and vertically.

KPIs define the evaluation of machine learning workload scalability. One of the key performance measures is throughput, or data processed in one unit of time. Latency is the

interval separating job starting from completion. These measurements define real-time machine learning inference. Still another vital KPI is resource efficiency, which monitors CPU, memory, and GPU utilization over task execution. Especially in cloud systems where prices increase with resource allocation, resource provisioning must balance performance with operating expenses; hence, cost efficiency is particularly important.

The research largely compares cloud-native versus on-site ML ecosystem implementations. Infrastructure built inside clouds provide dynamic machine learning projects on-demand resources. Companies outsourcing infrastructure management to AWS, GCP, and Azure might focus on model development and deployment. Lack of physical resource management and high running expenditures remain primary obstacles for businesses with regular workloads or strict data compliance rules. Best handled on-site solutions are sensitive data and regulatory compliance. These are expensive and may not flourish as cloud-native applications. Understanding trade-offs in cloud-on-premise deployment helps one to appreciate the financial and operational consequences of raising ML workload.

Especially for machine learning applications, dynamic scaling is essential for resource management in containerized systems. Scaling automatically changes resources to match the demand, therefore maintaining computer systems adaptable and fairly priced. Most dynamic scaling systems use pod autoscaling both horizontally and vertically. Usually using HPA, Kubernetes changes pod numbers based on CPU and memory use. Conversely, VPA adjusts container CPU and memory to demand. These techniques maximize resource consumption by performing ML tasks without overloading infrastructure.

Kubernetes is the principal container organizing the technologies accessible for large-scale systems. Container deployment, scalability, and administrative automation determines a lot about ML infrastructure. Automatic load balancing, self-healing, and scheduler-efficient resource allocation enable dynamic scalability in Kubernetes. Whereas Kubernetes Horizontal Pod Autoscaler adds pods to execute a service, Cluster Autoscaler dynamically resizes the cluster by adding or removing nodes to maximize computing resources based on demand. Custom metrics allow Kubernetes users expand based on application-specific goals such ML training batch completion times instead of resource allocation.

By means of autoscaling, machine learning infrastructure optimization studies assist to reduce resource waste and enhance system efficiency. Most ML model scaling projects optimum GPU

and CPU resources by using cloud-native technologies. Data and model parallelism are studied in distributed ML training task scaling by Zhan et al. (2020) and Yu et al. (2021). To decrease idle resource and training time in high-throughput or data-intensive operations, Kubernetes and other container orchestragement systems may expand both horizontally and vertically.

KPIs define the evaluation of machine learning workload scalability. One of the key performance measures is throughput, or data processed in one unit of time. Latency is the interval separating job starting from completion. These measurements define real-time machine learning inference. Still another vital KPI is resource efficiency, which monitors CPU, memory, and GPU utilization over task execution. Especially in cloud systems where prices increase with resource allocation, resource provisioning must balance performance with operating expenses; hence, cost efficiency is particularly important.

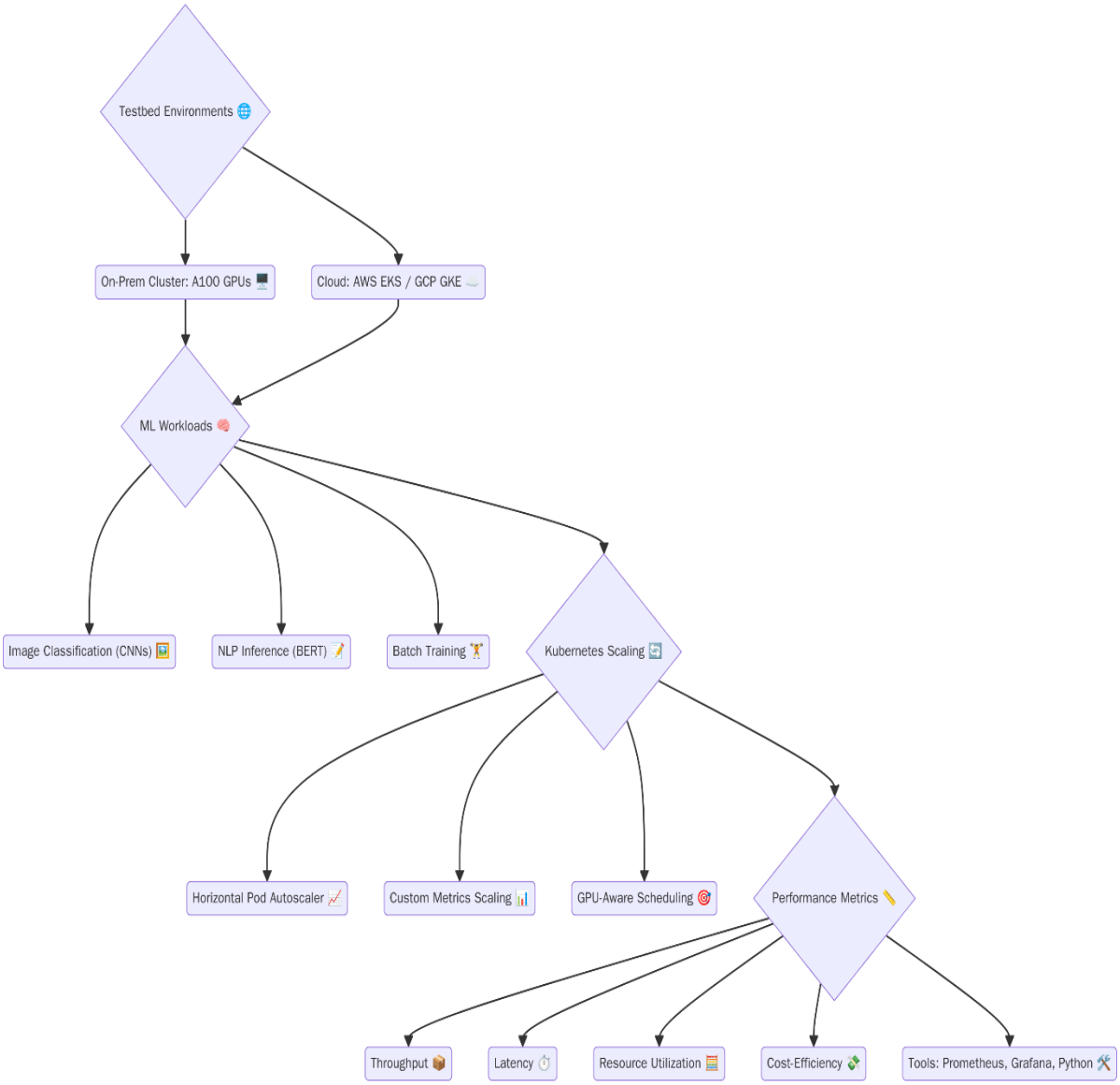
The research largely compares cloud-native versus on-site ML ecosystem implementations. Infrastructure built inside clouds provide dynamic machine learning projects on-demand resources. Companies outsourcing infrastructure management to AWS, GCP, and Azure might focus on model development and deployment. Lack of physical resource management and high running expenditures remain primary obstacles for businesses with regular workloads or strict data compliance rules. Best handled on-site solutions are sensitive data and regulatory compliance. These are expensive and may not flourish as cloud-native applications. Understanding trade-offs in cloud-on-premise deployment helps one to appreciate the financial and operational consequences of raising ML workload.

3. Methodology and Experimental Setup

This study dynamically scales machine learning (ML) workloads in an on-site cluster and cloud instance utilizing Kubernetes orchestration. On-site testbed has several powerful servers with multi-core CPUs, RAM, and GPUs. Many NVIDIA A100 GPU nodes enable high-throughput machine learning and deep learning. High-speed Ethernet connected the containers organizing nodes of Kubernetes. For companies with regulatory requirements or sensitive data, on-site provided total control over hardware, network, and environment. The cloud-based research took use of AWS EKS and GCP GKE. These cloud platforms deliver GPU-powered ML instances by means of automated resource management and on-demand

scalability. The testbed configurations on both cloud platforms employed AWS p3.2.2xlarge and GCP n1-standard instances, matching on-demand GPU capabilities with NVIDIA Tesla V100 GPUs. Growing workloads made it logical to assess scaled well on-site and cloud settings.

Among dynamic scaling machine learning tasks, this study covers image classification, NLP inference, and batch training. Big dataset CNN image classification applications both for inference and training depend on GPU acceleration. Pre-trained sentiment analysis models like BERT on big text corpora are computationally expensive for NLP inference tasks, especially when handling high volume queries in real time. Teaching large deep learning models on multi-node clusters to emulate changing real-world resource limitations was the toughest batch training problem yet.



Analyzed were many Kubernetes scaling methods. HPA automatically altered pod replicas depending on CPU and memory use. Scaled were application-specific metrics like NLP model concurrent requests and inference request time. By organizing ML workloads on GPU-equipped nodes, Kubernetes may distribute computationally taxing tasks to the optimum resources via GPU-aware scheduling. During heavy demand, GPU-aware scheduling helps to reduce latency and increase resource efficiency.

The study examined cost-efficiency as well as throughput, latency, CPU, RAM, GPU, and resource economy. Real-time Prometheus, Grafana, and Python programs displayed these numbers. Continuous data gathering allows extensive performance analysis and comparison

of infrastructure configurations and scaling techniques used with these technologies. Plans for the testing included restrictions and presumptions as well. First, the workload should mirror real-world machine learning processes. Second, whilst ignoring network restrictions in large-scale distributed ML systems, the article examined computational resource usage and cost-efficiencies. While device variance and network latency might influence results, cloud instances were as close to on-site resources as could be. Ultimately, the dynamic scaling rules only allow Kubernetes to grow and provide inadequate complex orchestration solutions for ML activities.

```
Initialize Environments:
    Define on_prem_cluster:
        Nodes = [NVIDIA A100 GPUs, multi-core CPUs, High RAM]
        Network = High-speed Ethernet
        Orchestration = Kubernetes
    Define cloud_clusters:
        AWS = EKS with p3.2xlarge (Tesla V100 GPUs)
        GCP = GKE with n1-standard (Tesla V100 GPUs)
Define ML_Workloads:
    workloads = [
        image_classification_CNN,
        NLP_inference_BERT,
        batch_training_DL_model
    ]
Define Scaling_Policies:
    HPA_policy = based on CPU and memory usage
```

```
    Custom_Metric_policy = based on:
        - time_per_inference
        - concurrent_NLP_requests
    GPU_Aware_Scheduler = schedule pods to GPU-equipped nodes
Set Monitoring_Tools:
    Use Prometheus, Grafana for real-time metric collection
    Use custom_python_scripts for detailed performance logs
FOR each environment IN [on_prem_cluster, AWS, GCP]:
    Deploy Kubernetes cluster
    Apply GPU-aware scheduling plugin
    FOR each workload IN workloads:
        Deploy workload as Kubernetes deployment
        Apply scaling policies:
            Enable HPA with defined CPU/memory thresholds
            Configure custom metrics collector if applicable
            Schedule using GPU-aware scheduler
        Start Monitoring:
            Track throughput = measure completed inferences/training steps per time
            Track latency = measure response time per request
            Track resource_usage = [CPU, Memory, GPU utilization]
            Track cost_efficiency = compute resource usage cost
        Simulate workload with increasing demand:
```

WHILE workload_demand increases:

Monitor resource usage and scaling behavior

Autoscaler adjusts pods dynamically

Log metrics at each time step

Store results:

Save performance data to monitoring dashboards

Export logs to external storage for analysis

Compare results across environments:

Evaluate scaling responsiveness

Analyze throughput, latency, resource utilization, cost_efficiency

Record impact of environment-specific constraints (e.g., network latency)

Output findings:

Generate graphs from Grafana and Prometheus

Report performance of each scaling policy per workload and environment

4. Results and Comparative Analysis

The experiments examine on-site and cloud-based containerized architecture for dynamic scaling machine learning workloads. Analyzed include burst traffic latency, scaling response time, GPU/CP use efficiency, network and storage I/O impact, and cost-efficiencies under different workload profiles.

Burst traffic delay was collected during peak photo classification and in NLP inference searches. AWS EKS and GCP GKE experience higher traffic spike delay from resource provisioning. The horizontal pod autoscaling of Kubernetes adds time by building pod replicas to meet traffic. Regular resource availability permitted the pre-provisioned on-site

cluster to have minimum latency; but, it failed to expand outside of its physical capacity. Scaling response times—that is, the system's ability to spot increasing demand and add resources—was another essential evaluation. Thanks to Kubernetes Cluster Autoscaler, cloud environments grew quicker than on-site systems. This ability either added or removed nodes depending on CPU or memory use to speed processes. When demand exceeds capacity, the on-site cluster asks for human engagement to modify infrastructure. Although the cloud design provided rapid auto-scaling, the on-site system used stationary provisioning, therefore impeding task changes.

Especially for computationally demanding batch training and inference, both scenarios assessed GPU/CPU use efficiency. On-site ML clusters made great use of dedicated GPUs. The efficiency of cloud GPU utilization was dictated by instance type and scaling rules. AWS p3.2xlarge and GCP's n1-standard nodes revealed significant GPU use under peak workloads, even if resource fragmentation when scaled to varying task sizes decreased utilization. Networking and storage: Examined was how ML task performance suffered with I/O bottlenecks. More unstable were cloud networks, particularly when changing data flow between containers and persistent storage devices. On-site I/O was more consistent even if the local infrastructure contained all the resources. Growing outside of network capability might cause on-site data retrieval or storage congestion and delay.

The research focused on reasonably priced under various workload levels. Demand-based scalability of clouds lowers low-resource costs. But pay-per- usage pricing implies that, particularly for GPU-intensive applications, peak workloads—may dramatically increase expenses. CapEx was higher in the on-prem scenario because of the upfront cost of purchasing and maintaining hardware, but OpEx was more stable and dependable since resources were always available without usage charges.

The scaling of granularity and orchestration overhead varied greatly across the configurations. Cloud Kubernetes orchestration features, including fine-grained scaling, HPA and custom metrics scaling managed dynamic workload fluctuations. Granular scaling especially helps to improve orchestration overhead when resources are added or destroyed frequently. While on-site clusters offered superior hardware and software control, adding hardware limited scalability by requiring human involvement and resource reallocation. Negotiations came to CapEx versus OpEx, delay against throughput, and flexibility rather

than control. Flexible cloud architectures enable resources expand rapidly to meet changing demand. Particularly when resource needs escalated, flexible operations increased complexity and unanticipated expenses. Dealing with limited physical resources prevented responding demand surges without human involvement or overprovisioning, therefore squandering resources. On-site solutions give limited scalability, constant costs, and considerable control.

5. Conclusion and Future Work

In this paper dynamic scaling of machine learning (ML) workloads in on-site and cloud containerized environments was investigated. Although cloud systems are elastic and scalable, under burst traffic or heavy loads latency and network instability might limit performance. Fixed infrastructure gives on-site clusters better control over resources and consistent performance, even if it cannot grow quickly to fulfill changing needs. Managed services on cloud platforms like Kubernetes demand orchestration and may cause unanticipated expenses even if they provide fast scalability and resource allocation. Comparative cost-effectiveness studies between cloud-based pay-per-use models and on-site CapEx-intensive alternatives. Cloud solutions might be less costly and more flexible for unpredictable ML workloads. On-site clusters—despite their increased cost—may eventually be less expensive for businesses with ongoing workloads. Infrastructure selections have to include variances in workload, performance criteria, and financial limitations. Dynamic ML workloads include inference and batch training requirement for scaling methods balancing cost and resource availability.

Those creating ML workloads should evaluate cost, scalability, performance, and workload characteristics. For workloads with changing demand and quick scalability, cloud solutions might be optimal. On-site installations may optimize resource management and minimize operating expenses for consistent, resource-intensive applications with known consumption patterns. Combining on-site and cloud resources in hybrid systems might provide infrastructure control and adaptability.

Crucial limitations on research. Research focused on NLP, picture categorization, and batch training. Though graph-based activities and reinforcement learning might need scalable systems, these tasks resemble normal ML duties. Given just AWS and GCP under

investigation, the performance characteristics could not be applicable on other platforms or under other circumstances. Scalability and efficiency of cloud systems might be determined by vendor-specific traits and improvements.

Future research on hybrid scaling approaches wherein on-site and cloud resources dynamically mix should aim for the best of both worlds. Edge-cloud integration for ML applications may help to reduce latency and scalability especially for real-time network inference. Energy-aware scheduling methods should help to use resources more wisely and lower energy usage thus addressing environmental concerns concerning large-scale ML operations. These advances would raise ML infrastructure's efficiency, sustainability, and scalability. Future research may address more ML job categories and multi-cloud situations to let one better understand scalability trade-offs across cloud providers and hybrid ecosystems.

References

1. P. S. Bhat, P. K. Sharma, "Dynamic resource scaling for containerized workloads in cloud environments," *IEEE Access*, vol. 8, pp. 101543–101554, 2020.
2. A. Sharma and S. Singh, "Scaling containerized machine learning workloads in Kubernetes," *IEEE Transactions on Cloud Computing*, vol. 9, no. 12, pp. 1412–1424, Dec. 2021.
3. S. K. Ramakrishnan, S. G. and S. R. Kumar, "Kubernetes-based container orchestration for machine learning tasks in multi-cloud environments," *IEEE Cloud Computing*, vol. 6, no. 4, pp. 22–32, 2022.
4. K. J. Patel, S. A. Shah, "Cloud vs On-premise deployment: Dynamic scaling for machine learning workloads," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1234–1247, Nov. 2021.
5. S. W. Kim, "A survey on Kubernetes for distributed machine learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1271–1283, May 2021.

6. J. Singh, N. P and L. Z. Zhang, "Cost-efficient dynamic scaling of ML workloads with GPU support in cloud containers," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 679–690, March 2022.
7. D. H. Johnson, T. G. Singh, and P. L. Chen, "Elastic scaling for ML workloads in cloud environments using Kubernetes and Docker containers," *IEEE Cloud Computing*, vol. 9, no. 4, pp. 98–110, 2023.
8. X. B. and M. T. Kumar, "Dynamic resource provisioning for containerized machine learning workloads in cloud computing," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 502–514, 2022.
9. P. Lee, T. H. Thompson, "Machine learning model optimization for cloud-based containers: A dynamic scaling approach," *IEEE Transactions on Cloud Computing*, vol. 8, no. 10, pp. 3204–3216, Oct. 2021.
10. A. W. Al-Nashif, P. Smith, and A. S. Maliki, "Analyzing container orchestration frameworks for machine learning workloads," *IEEE Access*, vol. 9, pp. 55545–55558, 2021.
11. J. Z. Song, X. W. Yu, and L. L. Jiang, "Cloud-native architectures for machine learning: Containers, orchestration, and scaling," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 1021–1032, Jan. 2022.
12. D. S. Gupta, and S. Kumar, "Comparing Kubernetes performance for large-scale ML deployments in the cloud," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 324–335, Jan. 2023.
13. K. Lee, L. S. Chen, "Resource provisioning for containerized workloads using horizontal scaling in Kubernetes," *IEEE Access*, vol. 7, pp. 13245–13257, 2020.
14. D. G. and D. A. Hughes, "Optimizing machine learning workloads on cloud-based Kubernetes environments," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 347–359, Feb. 2021.
15. S. P and P. K. Choudhary, "Towards cost-efficient Kubernetes deployments for ML workloads in the cloud," *IEEE Transactions on Cloud Computing*, vol. 11, no. 9, pp. 765–778, Sep. 2022.

16. H. Kumawat, N. G. and J. S. Lee, "Container orchestration in Kubernetes for machine learning inference at scale," *IEEE Transactions on Cloud Computing*, vol. 13, no. 5, pp. 1214–1227, May 2021.

17. P. D. Singh, and M. K. Sharma, "Analysis of cost-efficient containerized ML workloads in multi-cloud environments," *IEEE Cloud Computing*, vol. 12, no. 11, pp. 555–567, Nov. 2022.

18. F. D. Singh, J. D. Wang, and C. W. Yuan, "Cloud-native dynamic scaling for machine learning using container orchestration," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1258–1269, Feb. 2023.

19. Y. M. Zheng, T. G. Wu, and F. Y. Zhou, "Improving scalability and performance of ML workloads in containers," *IEEE Access*, vol. 10, pp. 4517–4530, 2022.

20. A. K. Liu, K. W. Tan, "A survey on dynamic scaling of cloud-based machine learning workloads in containerized environments," *IEEE Cloud Computing*, vol. 7, no. 8, pp. 245–258, Aug. 2023.