A Modular Kubernetes Workload Simulator for Evaluating Learning-Based Scheduling Policies

Marios Touloupou[1], Syed Mafooq Ul Hassan[1], Jacopo Castellini[2], Pablo Strasser[2], Alexandros Kalousis[2], Herodotos Herodotou[1]
[1]Cyprus University of Technology, Limassol, Cyprus
marios.touloupou@cut.ac.cy, syedmafooqul.shah@cut.ac.cy, herodotos.herodotou@cut.ac.cy
[2]University of Applied Sciences and Arts Western Switzerland (HES-SO), Carouge, Geneva, Switzerland
jacopo.castellini@hesge.ch, pablo.strasser@hesge.ch, alexandros.kalousis@hesge.ch

***Abstract*—Kubernetes is a popular container orchestration platform in cloud and cloud-edge environments. To properly evaluate new Kubernetes scheduling strategies, especially learning-based ones, a controlled but realistic environment is needed, enabling repeatable experimentation without the overhead of running real clusters. This paper presents a modular Kubernetes workload simulator that supports configurable cluster setups, synthetic workload generation, and pluggable scheduling policies. The simulator enables both rule-based and learning-based schedulers to be evaluated under identical conditions, while providing detailed execution traces and performance metrics. This demo paper showcases its capabilities through some interactive scenarios that highlight its usefulness for the development, testing, and comparative evaluation of different Kubernetes schedulers.**

Keywords-Kubernetes; workload simulation; workload scheduling; experimentation.

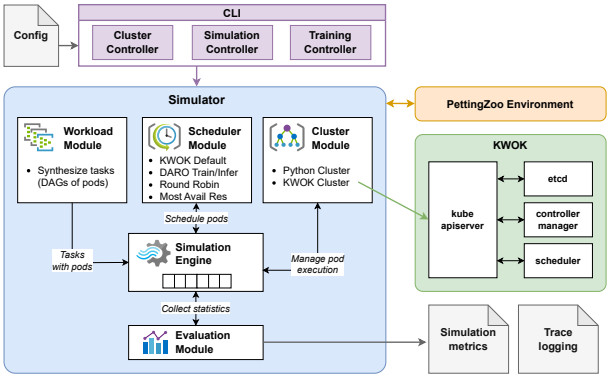

Figure 1. Architecture of the Kubernetes workload simulator.

I. Introduction and Motivation

Evaluating schedulers directly on real cloud or cloud-edge infrastructures is often impractical due to the operational complexity, resource costs, and limited reproducibility of such setups [1]. Moreover, experimentation on real clusters can interfere with production workloads and make it difficult to isolate the impact of scheduling decisions [2].

Recent work highlights the growing interest in learning-based approaches for cloud and cluster resource management, including Kubernetes scheduling [3]. However, existing Kubernetes schedulers and other available simulation environments offer limited support for customized scheduling logic and synthetic workload generation. Many tools either provide fixed scheduling behaviors or operate at a high level of abstraction, which restricts their usefulness for iterative development and fair benchmarking of schedulers [4].

As discussed in recent surveys of cloud, edge, and Internet of Things (IoT) computing, there is an increasing need for lightweight and flexible experimentation environments that bridge the gap between abstract models and real system deployments [5]–[7]. To address these limitations, we designed, implemented, and released a modular open-source Kubernetes workload simulator that provides a controlled, repeatable evaluation framework for different scheduling policies [8]. The simulator was initially motivated by the need to support training and evaluation workflows for learning-based schedulers in a simplified but representative setting. It builds on top of lightweight Kubernetes emulation techniques, such as Kubernetes Without Kubelet (KWOK), to model cluster behavior without the overhead of managing real containers, while extending them with support for dynamic workload synthesis, pluggable scheduling strategies, and detailed execution tracking [9].

The remainder of this paper is structured as follows. Section II presents the architecture and key components of the simulator, Section III demonstrates its use through representative scenarios, and Section IV concludes the paper.

II. Kubernetes Workload Simulator

As depicted in Figure 1, the simulator follows a modular design integrating workload synthesis, scheduling logic, cluster modeling, and evaluation. A command-line interface drives simulation and training workflows, while the simulation engine orchestrates pod execution, scheduling decisions, and statistics collection. The architecture supports interaction with external learning environments and enables both lightweight simulation and Kubernetes-like execution modes.

To balance realism and efficiency, the simulator supports two complementary cluster backends: a Kubernetes-like backend (KWOK [9]) that uses lightweight emulation to model pods and nodes without running actual containers, and a native Python backend with virtual time for fast, controllable experimentation.

The simulator separates cluster modeling, workload generation, scheduling logic, and evaluation into distinct components. The cluster model supports heterogeneous nodes with

```yaml
# Cluster
cluster_type: Python # KWOK or Python
cluster_reset: True  # True: Clear existing cluster, False: Append to existing cluster

# Cluster Nodes
cluster_nodes_cloud: 20
cluster_nodes_edge: 30
cluster_nodes_iot: 50

# Cloud Node Profile
cluster_node_cloud_cpu_dist: {type: normal, mean: 48000, stdev: 12000, min: 16000, max: 96000, round: -3}  # in millicores
cluster_node_cloud_mem_dist: {type: normal, mean: 96000, stdev: 24000, min: 32000, max: 256000, round: -3} # in Mi
cluster_node_cloud_stg_dist: {type: normal, mean: 4000, stdev: 1000, min: 500, max: 10000, round: -2}      # in Gi
cluster_node_cloud_max_pods: 110

# Edge Node Profile
cluster_node_edge_cpu_dist: {type: normal, mean: 8000, stdev: 2000, min: 4000, max: 16000, round: -2}  # in millicores
cluster_node_edge_mem_dist: {type: normal, mean: 16000, stdev: 4000, min: 4000, max: 32000, round: -2} # in Mi
cluster_node_edge_stg_dist: {type: normal, mean: 500, stdev: 50, min: 50, max: 1000, round: -1}        # in Gi
cluster_node_edge_max_pods: 50

# IoT Node Profile
cluster_node_iot_cpu_dist: {type: normal, mean: 2000, stdev: 500, min: 1000, max: 4000, round: -2} # in millicores
cluster_node_iot_mem_dist: {type: normal, mean: 2000, stdev: 500, min: 1000, max: 4000, round: -2} # in Mi
cluster_node_iot_stg_dist: {type: normal, mean: 16, stdev: 4, min: 1, max: 32, round: 0}           # in Gi
cluster_node_iot_max_pods: 30

# Workload
workload_tasks: 500
workload_pods_number_dist: {type: pareto, alpha: 1.2, min: 2, max: 10, round: 0} # Distribution of pods per task
workload_pods_cpu_dist: {type: normal, mean: 1000, stdev: 500, min: 500, max: 4000, round: -2} # in millicores
workload_pods_mem_dist: {type: normal, mean: 2000, stdev: 500, min: 500, max: 8000, round: -2} # in Mi
workload_pods_stg_dist: {type: normal, mean: 1, stdev: 10, min: 0, max: 500, round: 0}         # in Gi
workload_pods_interarrival_dist: {type: poisson, mean: 20, min: 10, max: 60}                  # in seconds
workload_pods_duration_dist: {type: poisson, mean: 60, min: 10, max: 600}                     # in seconds
workload_pods_max_restarts: 5 # The number of times a pending pod can be restarted (Crashloop error of K8s)

# Scheduler
scheduler_type: ROUNDROBIN  # ROUNDROBIN or DEFAULT or MOSTAVAILABLE or RANDOM or DAROINFER or DAROTRAIN
scheduler_reward_type: CompositeReward  # Coop_LB_reward or Cluster_LB_reward or Node_LB_reward or Fragmentation_reward or
scheduler_composite_reward_parts: [Cluster_LB_reward, Node_LB_reward, Fragmentation_reward] # Rewards for CompositeReward
scheduler_composite_reward_weights: [1, 1, 0] # Reward weights to use with CompositeReward

# Simulation
simulation_speedup: 0  # 1=real-time, 0=infinite, other numbers=speedup factor
simulation_seed: 20    # 0 for fully random, any other number for reproduciable runs
```

Figure 2. Example simulator configuration file used in the demonstration.

configurable CPU, memory, and storage capacities, enabling cloud, edge, and IoT-like environments. Workloads are synthetically generated as sets of pods with configurable resource demands, arrival patterns, and execution durations, ensuring reproducibility.

Scheduling is fully pluggable through a unified interface, allowing different strategies to be evaluated within the same setup. During execution, the simulator coordinates pod arrivals, scheduling decisions, and life cycles using a discrete-event engine. Detailed execution traces and performance metrics are collected at the pod, workload, and cluster levels, enabling analysis of scheduling behavior and resource utilization. These metrics include, among others, pod waiting time, completion time, and resource utilization (CPU, memory, and storage) across nodes.

III. Demonstration Scenarios and Observations

The demonstration illustrates how the Kubernetes Workload Simulator enables the evaluation and comparison of scheduling strategies under controlled conditions. A simulated cluster is configured once, and synthetic workloads are submitted while varying only the scheduling policy. This allows direct comparison of pod placement decisions, resource utilization patterns, and overall cluster behavior using execution traces and performance metrics such as waiting times, completion times, and node-level resource usage.

Initial observations indicate that different scheduling policies yield distinct resource utilization patterns across the cluster, underscoring the need for systematic and repeatable evaluation. Although the current demonstration focuses on CPU, memory, and storage resources, the simulator lays the groundwork for more complex scenarios.

The demonstration is executed through a command-line interface that allows users to configure cluster setups, workloads, and scheduling strategies via configuration files. Users can run simulations, switch between schedulers, and observe their impact on pod placement and resource utilization. The simulator also supports evaluating trained learning-based models and comparing them with baseline approaches, including the default Kubernetes scheduler.

Figure 2 shows an excerpt of the configuration file used to define cluster setups (e.g., cloud, edge, and IoT nodes), workload characteristics, and the scheduling policy. A typical scenario fixes the workload while varying the cluster composition or scheduler to enable consistent comparisons.

IV. Conclusion

This paper presents a modular Kubernetes workload simulator for the controlled, repeatable evaluation of scheduling policies. By combining a Kubernetes-like backend with a native Python backend, it enables flexible experimentation across heterogeneous cloud, edge, and IoT environments. The simulator provides execution traces and performance metrics for analyzing scheduling behavior and resource utilization, supporting iterative development and comparative evaluation. Future work will focus on increasing realism through richer system modeling, tighter integration with real Kubernetes clusters, and the addition of a graphical user interface.

Acknowledgments

This work has been supported by the HYPER-AI project, funded by the European Commission under Grant Agreement 101135982 through the Horizon Europe program.

References

[1] M. Menaka and K. S. Kumar, “Workflow Scheduling in Cloud Environment–Challenges, Tools, Limitations & Methodologies: A Review,” *Measurement: Sensors*, vol. 24, p. 100 436, 2022.

[2] Kubernetes, “Scheduling, Preemption and Eviction,” 2023, [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/ (visited on 02/26/2026).

[3] Q. Zhang, M. Chen, and K. Li, “A Survey on Reinforcement Learning for Cloud Resource Management,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–36, 2023. DOI: 10.1145/3566576.

[4] Z. Jian *et al.*, “DRS: A deep reinforcement learning enhanced Kubernetes scheduler for microservice-based system,” *Software: Practice and Experience*, vol. 54, no. 10, pp. 2102–2126, 2024.

[5] S. N. Srirama, “A Decade of Research in Fog Computing: Relevance, Challenges, and Future Directions,” *Software: Practice and Experience*, vol. 54, no. 1, pp. 3–23, 2024.

[6] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D’Andria, “A Survey on IoT-Edge-Cloud Continuum Systems: Status, Challenges, Use Cases, and Open Issues,” *Future Internet*, vol. 15, no. 12, p. 383, 2023.

[7] H. Kuchuk and E. Malokhvii, “Integration of IoT with Cloud, Fog, and Edge Computing: A Review,” *Advanced Information Systems*, vol. 8, no. 2, pp. 65–78, 2024.

[8] Eclipse Research Labs, *Kubernetes Workload Simulator*, 2025. [Online]. Available: https://gitlab.eclipse.org/eclipse-research-labs/hyper-ai-project/k8s-workload-simulator (visited on 02/26/2026).

[9] Kubernetes SIGs, “KWOK: Kubernetes Without Kubelet,” 2023, [Online]. Available: https://kwok.sigs.k8s.io/ (visited on 02/26/2026).