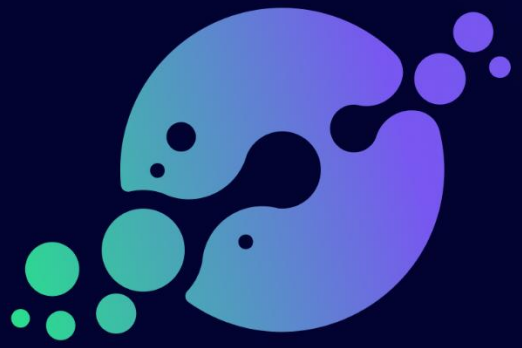
Cyprus University of Technology

Hes·so Haute Ecole Spécialisée de Suisse occidentale

Funded by the European Union



HYPER-AI

Revolutionising big data applications with autonomous cloud-to-edge resources

IARIA

CLOUD COMPUTING 2026

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

• Marios Touloupou • Syed Mafooq Ul Hassan

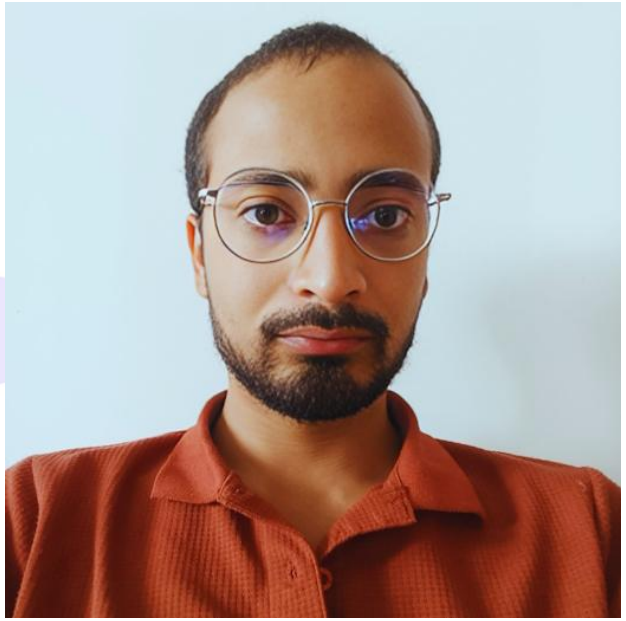
• Jacopo Castellini • Pablo Strasser

• Alexandros Kalousis • Herodotos Herodotou

Presenter: Syed Mafooq Ul Hassan, Cyprus University of Technology (syedmafooqul.shah@cut.ac.cy)

About Me

- Research associate at Data Intensive Computing Research Lab at Cyprus University of Technology
- PhD candidate in the Department of Electrical Engineering, Computer Science and Engineering at Cyprus University of Technology
- MSc In Intelligent Critical Infrastructure Systems from University of Cyprus



Cyprus University of Technology

dicl Data Intensive Computing Research Lab

https://dicl.cut.ac.cy/people

Presentation Roadmap

- **Motivation**: Why scheduler evaluation is hard on real clusters
- **Simulator Design**: Core capabilities, architecture, and backends
- **Demonstration**: Configuration-driven experiments and observations
- **Takeaways**: Conclusions and future work

Motivation: Why this Simulator is Needed

Core problem: Testing schedulers directly on real infrastructures is costly, operationally complex, and difficult to reproduce.

1. Real-cluster experimentation: Requires substantial infrastructure effort, time, and cost. Reproducing exact same conditions across runs is hard.

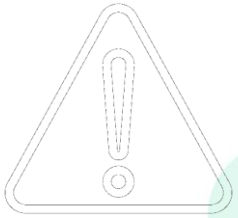
2. Production risk: Running experiments on live systems can interfere with real workloads and hide the true causal effect of the policy.

3. Learning-based schedulers: Need many controlled and repeatable runs for both training and fair evaluation.

Shortcomings of the Existing State of the Art

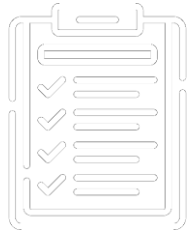
Limitations of current Simulators:

- Limited support for customized scheduling logic
- Limited support for synthetic workload generation
- Many tools use fixed behaviors or work at a very high level of abstraction
- This reduces their usefulness for iterative development and fair benchmarking



What we needed:

- A lightweight but flexible experimentation environment
- Support for cloud, edge, and IoT-style scenarios
- Repeatable execution under controlled conditions
- A bridge between abstract models and real Kubernetes deployments



Kubernetes Workload Simulator: Design

Configurable cluster setups

- Controlled cloud, edge, and IoT-like environments

Synthetic workload generation

- Statistically generated pods and task structures

Pluggable schedulers

- Rule-based and learning-based policies in one framework

Detailed traces and metrics

- Execution logs for analysis and benchmarking

Central idea

- Evaluate different scheduling policies under identical conditions rather than changing the whole experimental setup each time.

Kubernetes Workload Simulator: Architecture

CLI and controllers

- Configuration flows through cluster, simulation, and training controllers

Three core modules

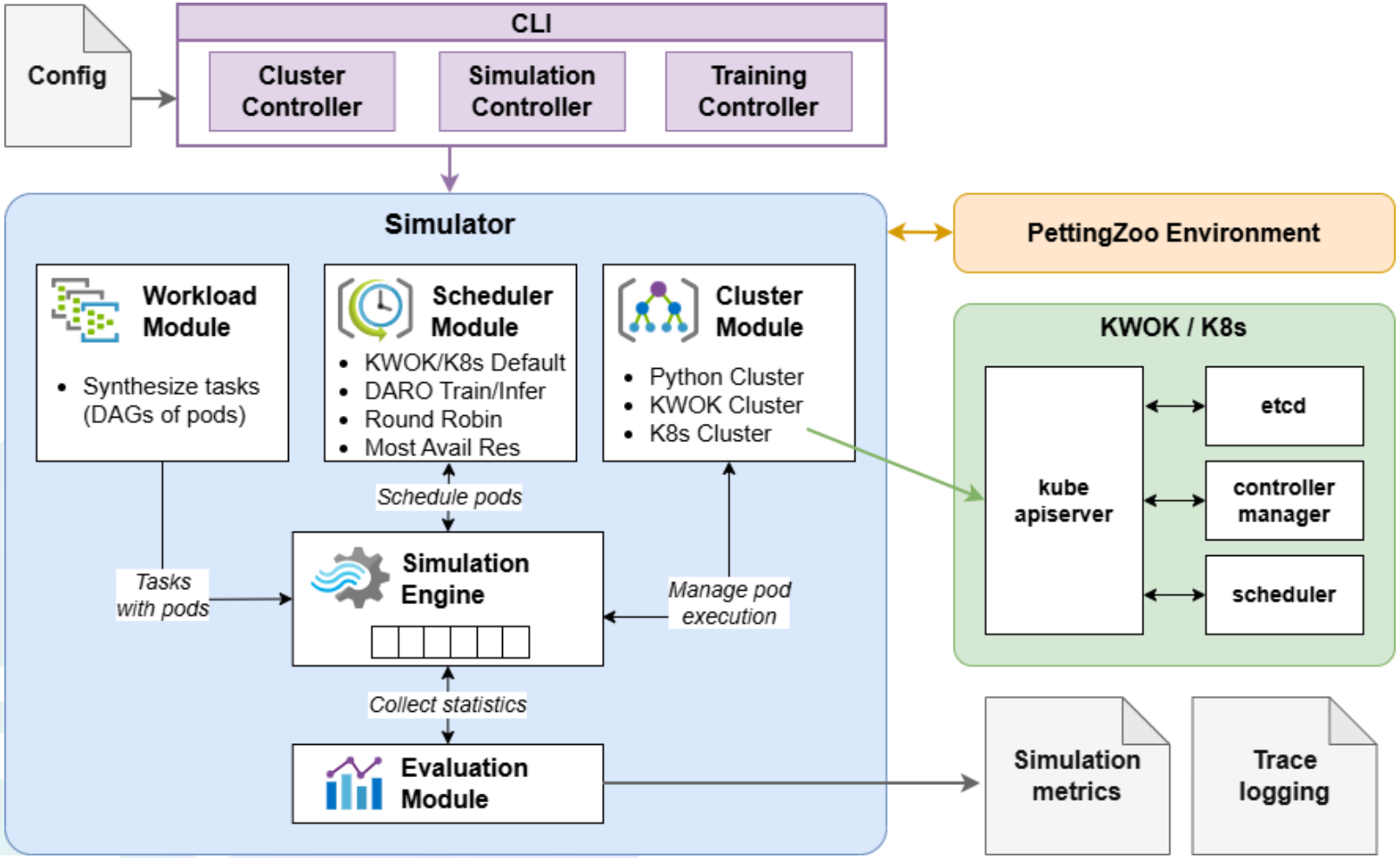
- Workload, scheduler, and cluster modules remain separated

Simulation engine

- Coordinates pod execution, scheduling decisions, and lifecycle management

Evaluation and integration

- Outputs metrics and trace logs
- Supports external learning environments



Kubernetes Workload Simulator: Cluster backends

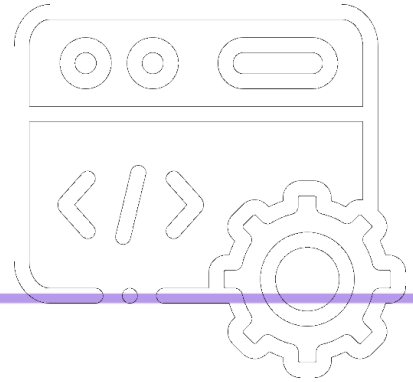
Two execution modes: realism vs. speed

Kubernetes-like backend (KWOK)

- Lightweight emulation of nodes/pods without running containers
- Preserves Kubernetes control-plane style components
- Useful when you want “Kubernetes-like” behavior and APIs

Native Python backend (virtual time)

- Discrete-event simulation with controllable timing
- Fast iteration for training/ hyperparameter sweeps
- Ideal for repeatability and large-scale experiments

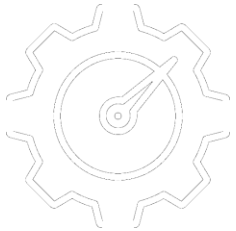


Workloads and Schedulers

Generate workloads, swap schedulers, and compare fairly

Workload synthesis

- Synthetic tasks are built as DAGs of pods
- Configurable: arrivals, resource demands, durations
- Reproducibility by using fixed seeds/config files



Pluggable scheduling

- Unified interface: swap policies without changing the workload
- Baseline examples: Kubernetes default, Round Robin, Most Available Resources
- Supports learning-based schedulers (training/inference workflows)

End-to-End Simulation Process

Reproducibility comes from controlled workload generation and a unified scheduler interface

Cluster Modeling

- Heterogeneous nodes with configurable CPU, memory, and storage
- Can emulate cloud, edge, and IoT-like environments
- Backends remain separate from workload and scheduler logic

Pluggable Scheduling

- Unified scheduler interface
- Examples in the architecture: default Kubernetes logic, round robin, most available resources, DARO train/infer
- Swap policies without changing the workload

Workload Generation

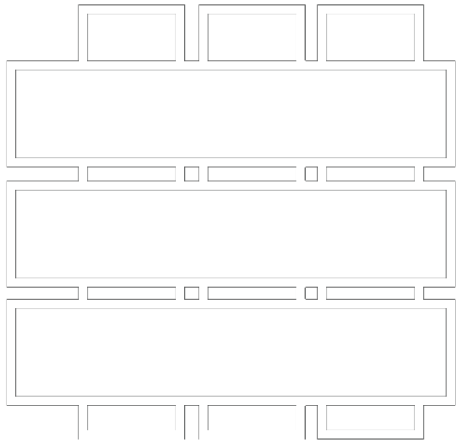
- Synthetic pods and tasks
- Configurable resource demands, arrivals, and durations
- Reproducible scenarios through configuration control

Evaluation Outputs and Metrics

The simulator supports both benchmarking and deeper diagnostic analysis.

Detailed traces & performance metrics

- Pod-level: waiting time, completion time, lifecycle events
- Cluster/node-level: CPU, memory, and storage utilization over time
- Workload-level summaries for direct scheduler comparison
- Execution trace logging to analyze placement decisions



Why these outputs matter

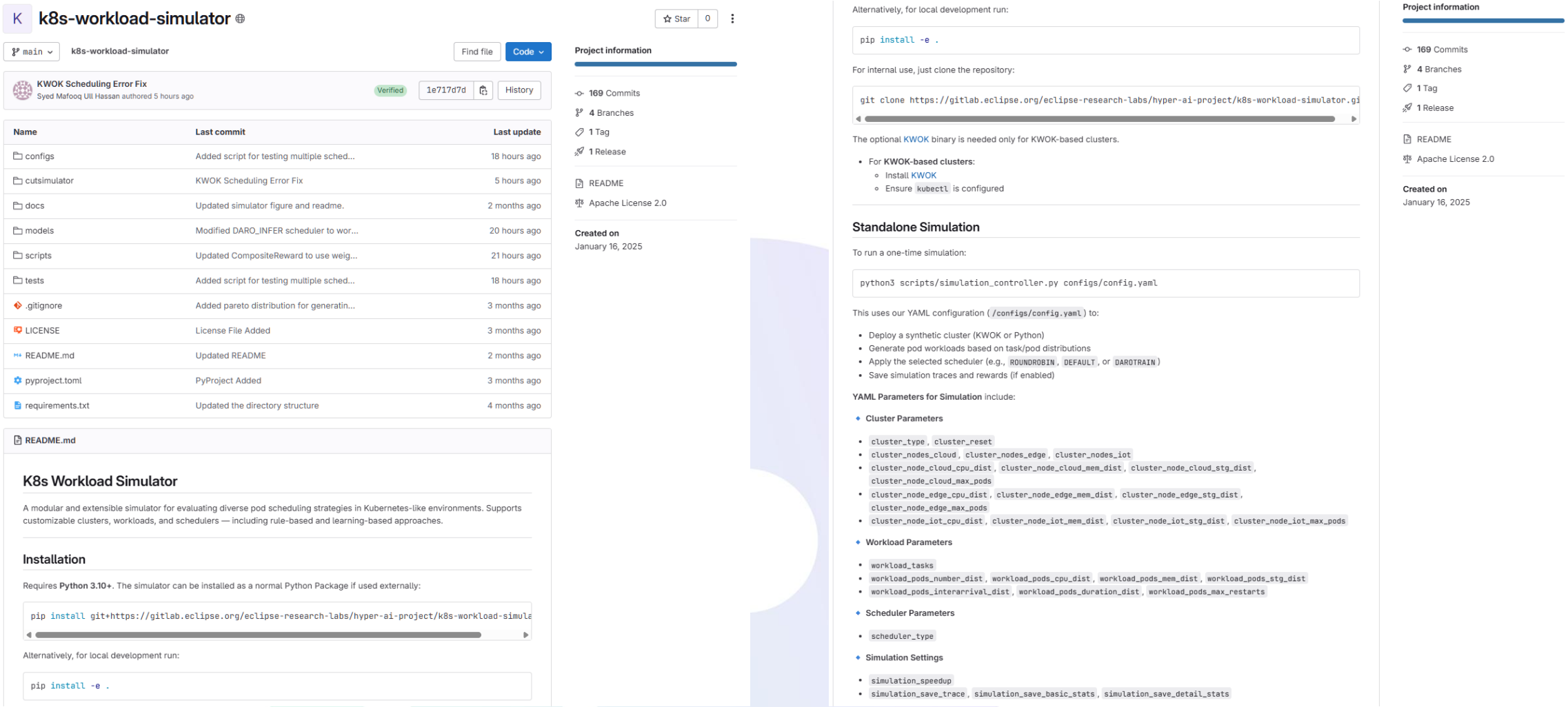
- They make it possible to compare policies fairly, but also to understand why a policy behaves the way it does over time.

Demonstration

config.yaml

```yaml
# Logging
logging_output: 3    # 1 = Console, 2 = File, 3 = Both
logging_level: INFO  # CRITICAL, ERROR, WARNING, INFO, DEBUG

# Cluster
cluster_type: Python # KWOK or Python or K8s
cluster_reset: True  # True: Clear existing cluster, False: Append to existing cluster

# Cluster Node Counts
cluster_nodes_cloud: 4
cluster_nodes_edge: 3
cluster_nodes_iot: 3

# Cloud Node Profile
cluster_node_cloud_cpu_dist: {type: normal, mean: 8000, stdev: 1000, min: 6000, max: 10000, round: -2}   # in millicores
cluster_node_cloud_mem_dist: {type: normal, mean: 16000, stdev: 2000, min: 12000, max: 20000, round: -2} # in Mi
cluster_node_cloud_stg_dist: {type: normal, mean: 2000, stdev: 1000, min: 500, max: 10000, round: -2}    # in Gi
cluster_node_cloud_bdw_dist: {type: normal, mean: 10000, stdev: 1000, min: 1000, max: 100000, round: -2} # in Mbps
cluster_node_cloud_max_pods: 110

# Edge Node Profile
cluster_node_edge_cpu_dist: {type: normal, mean: 4000, stdev: 800, min: 3000, max: 6000, round: -2}   # in millicores
cluster_node_edge_mem_dist: {type: normal, mean: 8000, stdev: 1000, min: 6000, max: 12000, 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_bdw_dist: {type: normal, mean: 1000, stdev: 200, min: 200, max: 10000, round: -1}   # in Mbps
cluster_node_edge_max_pods: 50

# IoT Node Profile
cluster_node_iot_cpu_dist: {type: normal, mean: 1000, stdev: 300, min: 500, max: 2000, 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: 8, stdev: 1, min: 1, max: 32, round: 0}             # in Gi
cluster_node_iot_bdw_dist: {type: normal, mean: 500, stdev: 200, min: 100, max: 1000, round: -1}    # in Mbps
cluster_node_iot_max_pods: 30

# Workload
workload_tasks: 20
workload_pods_number_dist: {type: pareto, alpha: 2, min: 2, max: 8, round: 0}  # Distribution of number of pods in each task
workload_pods_cpu_dist: {type: normal, mean: 1000, stdev: 1000, min: 500, max: 4000, round: -2}   # in millicores
workload_pods_mem_dist: {type: normal, mean: 2000, stdev: 2000, 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_bdw_dist: {type: normal, mean: 1000, stdev: 2000, min: 100, max: 10000, round: -1}  # in Mbps
workload_pods_require_bdw_prob: 0.3 # The probability that a pod has a bandwidth requirement
workload_pods_interarrival_dist: {type: poisson, mean: 5, min: 3, max: 6} # in seconds
workload_pods_duration_dist: {type: poisson, mean: 6, min: 2, max: 9} # in seconds
```

Kubernetes Workload Simulator: Code Repository



k8s-workload-simulator

Star 0

main · k8s-workload-simulator · Find file · Code

KWOK Scheduling Error Fix
Syed Mafooq Ull Hassan authored 5 hours ago · Verified · 1e717d7d · History

Name	Last commit	Last update
configs	Added script for testing multiple sched...	18 hours ago
cutsimulator	KWOK Scheduling Error Fix	5 hours ago
docs	Updated simulator figure and readme.	2 months ago
models	Modified DARO_INFER scheduler to wor...	20 hours ago
scripts	Updated CompositeReward to use weig...	21 hours ago
tests	Added script for testing multiple sched...	18 hours ago
.gitignore	Added pareto distribution for generatin...	3 months ago
LICENSE	License File Added	3 months ago
README.md	Updated README	2 months ago
pyproject.toml	PyProject Added	3 months ago
requirements.txt	Updated the directory structure	4 months ago

README.md

K8s Workload Simulator

A modular and extensible simulator for evaluating diverse pod scheduling strategies in Kubernetes-like environments. Supports customizable clusters, workloads, and schedulers — including rule-based and learning-based approaches.

Installation

Requires **Python 3.10+**. The simulator can be installed as a normal Python Package if used externally:

```
pip install git+https://gitlab.eclipse.org/eclipse-research-labs/hyper-ai-project/k8s-workload-simula
```

Alternatively, for local development run:

```
pip install -e .
```

Project information

169 Commits
4 Branches
1 Tag
1 Release

README
Apache License 2.0

Created on
January 16, 2025

Alternatively, for local development run:

```
pip install -e .
```

For internal use, just clone the repository:

```
git clone https://gitlab.eclipse.org/eclipse-research-labs/hyper-ai-project/k8s-workload-simulator.gi
```

The optional KWOK binary is needed only for KWOK-based clusters.

- For **KWOK-based clusters**:
  - Install KWOK
  - Ensure `kubectl` is configured

Standalone Simulation

To run a one-time simulation:

```
python3 scripts/simulation_controller.py configs/config.yaml
```

This uses our YAML configuration (`/configs/config.yaml`) to:

- Deploy a synthetic cluster (KWOK or Python)
- Generate pod workloads based on task/pod distributions
- Apply the selected scheduler (e.g., `ROUNDROBIN`, `DEFAULT`, or `DAROTRAIN`)
- Save simulation traces and rewards (if enabled)

YAML Parameters for Simulation include:

🔹 **Cluster Parameters**

- `cluster_type`, `cluster_reset`
- `cluster_nodes_cloud`, `cluster_nodes_edge`, `cluster_nodes_iot`
- `cluster_node_cloud_cpu_dist`, `cluster_node_cloud_mem_dist`, `cluster_node_cloud_stg_dist`, `cluster_node_cloud_max_pods`
- `cluster_node_edge_cpu_dist`, `cluster_node_edge_mem_dist`, `cluster_node_edge_stg_dist`, `cluster_node_edge_max_pods`
- `cluster_node_iot_cpu_dist`, `cluster_node_iot_mem_dist`, `cluster_node_iot_stg_dist`, `cluster_node_iot_max_pods`

🔹 **Workload Parameters**

- `workload_tasks`
- `workload_pods_number_dist`, `workload_pods_cpu_dist`, `workload_pods_mem_dist`, `workload_pods_stg_dist`
- `workload_pods_interarrival_dist`, `workload_pods_duration_dist`, `workload_pods_max_restarts`

🔹 **Scheduler Parameters**

- `scheduler_type`

🔹 **Simulation Settings**

- `simulation_speedup`
- `simulation_save_trace`, `simulation_save_basic_stats`, `simulation_save_detail_stats`

Project information

169 Commits
4 Branches
1 Tag
1 Release

README
Apache License 2.0

Created on
January 16, 2025

https://gitlab.eclipse.org/eclipse-research-labs/hyper-ai-project/k8s-workload-simulator

Conclusions and Future Work

A controlled framework for developing and comparing Kubernetes scheduling policies

- A modular simulator for controlled, repeatable evaluation of Kubernetes scheduling policies
- Supports heterogeneous cloud/edge/IoT-style clusters and synthetic workloads
- Enables fair comparisons via shared configurations, execution traces, and metrics

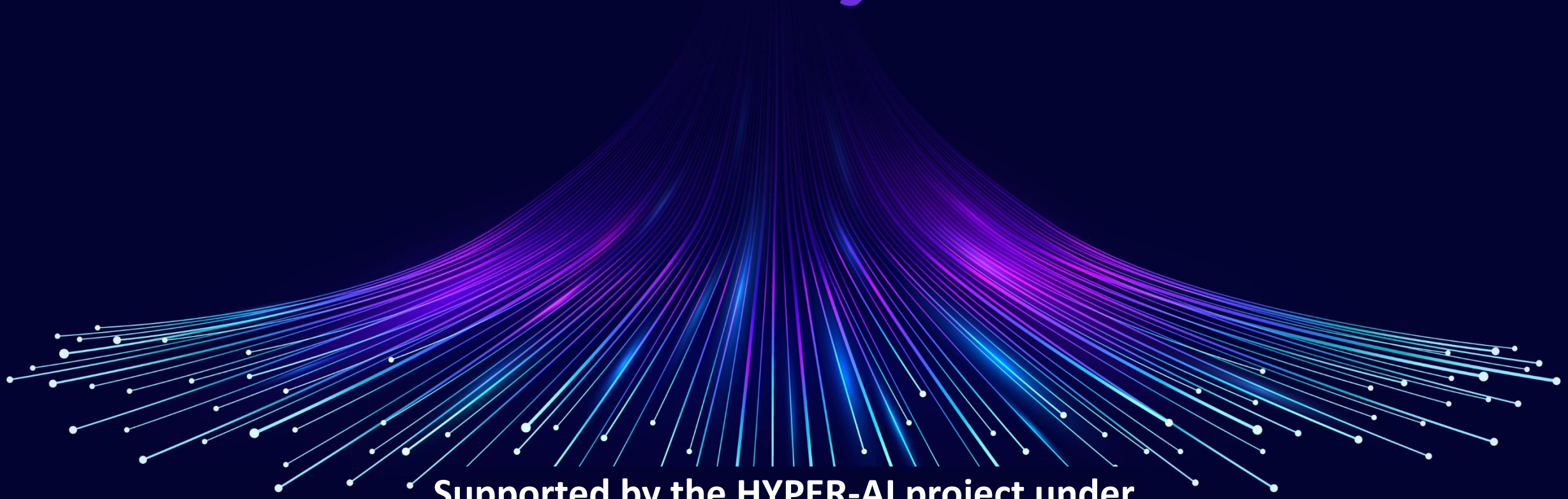
Future Work

- Richer system modeling
- Tighter integration with real Kubernetes clusters
- Graphical user interface (GUI)

HYPER-AI

Revolutionising big data applications with autonomous cloud-to-edge resources

Thank you



Supported by the HYPER-AI project under Horizon Europe Grant Agreement 101135982