

Fair Comparison of Scheduling Algorithms on Heterogeneous Edge Clusters: A Continuous Adaptive Benchmark

Zihang Wang*, Boris Sedlak†, Juan Luis Herrera‡, and Schahram Dustdar*§

* TU Wien, Vienna, Austria

† Universitat Pompeu Fabra, Barcelona, Spain

‡ Universidad de Extremadura, Mérida, Spain

§ ICREA, Barcelona, Spain

Corresponding author: zwang@dsg.tuwien.ac.at

Abstract—Modern Artificial Intelligence (AI) workloads deployed across the heterogeneous tiers of an edge-cloud continuum must satisfy multi-dimensional Service Level Objectives (SLOs) over latency, throughput, and output quality. For each incoming task, the scheduler picks both a target node and a processing mode (e.g., full or reduced inference precision). We call this class of problems *Continuous Multi-Mode Scheduling* (CMMS). Comparing CMMS algorithms fairly is difficult because prior studies typically evaluate each controller in its own stack, under a single workload, and without reporting per-decision overhead. To close these gaps, we present an open source benchmark platform that features (i) a unified controller interface, (ii) a closed-loop workload driver covering multiple workload patterns, and (iii) dual-metric SLO scoring that reports raw SLO (overall compliance) and steady-state SLO (compliance during stable operation) separately. Running six controllers across five cluster configurations and two load regimes (424 episodes), we find that controller rankings are strongly configuration-dependent: a deep reinforcement-learning winner under light workloads loses to a rule-based heuristic by nearly 29 percentage points once load intensifies, at roughly 500× the per-decision operational overhead. We further show that separating raw from steady-state SLOs exposes switching costs that a single aggregate score would otherwise conflate.

I. INTRODUCTION

Modern Artificial Intelligence (AI) workloads are increasingly deployed across the heterogeneous tiers of the computing continuum. The continuum spans edge devices, fog nodes, and accelerators with heterogeneous resources that are often required to sustain multi-dimensional Service Level Objectives (SLOs) over latency, throughput, and output quality [1], [2]. This forces workload schedulers to make continuous runtime decisions: for each incoming task, dispatch it to a suitable node and optimize the execution (e.g., picking an AI model to process data). We call this class of problems *Continuous Multi-Mode Scheduling* (CMMS). Common algorithms and solutions to CMMS problems can be fundamentally different, including rule-based heuristics [3], profile-guided optimization [4], deep reinforcement learning [5], and active inference [6], [7]. Although this holds, a basic question remains hard to answer: *Which class of controller makes the best continuous decisions to maximize SLOs under pressure, and at what cost?*

To this end, we identify three gaps in how CMMS controllers are currently evaluated that make it difficult to fairly compare existing algorithms. First, the algorithm-specific testbeds conflate the algorithm with its test environment. Consequently, published numbers reflect the combination rather than the algorithm alone. For instance, VideoStorm [3], Chameleon [8], and Pensieve [9] each report results on their own custom testbed, which a third-party scheduler cannot reuse. Second, evaluating an algorithm within only a single scenario hides how different scheduling algorithms behave when the workload pattern changes, letting a "single-scenario champion" be mistaken for a global optimum. Specifically, JCAB [10] and Decima [5] evaluate on a single workload pattern, leaving cross-scenario robustness unexamined. Third, the compute cost of the decision itself is rarely accounted for, though on Edge devices this directly consumes resources that otherwise could be dedicated for processing the workloads. Neither Pensieve [9] nor Decima [5] report per-decision latency alongside task-level SLO. Collectively, these gaps are particularly acute in *Distributed Computing Continuum Systems* (DCCS), where heterogeneity of devices and tiers is the rule rather than the exception. In such settings, device heterogeneity and time-varying workloads can make single-testbed, single-scenario, overhead-blind evaluation particularly misleading.

To address these limitations, we present a fairness-first benchmark platform for CMMS in DCCS. The platform enables fair cross-algorithm evaluation through a unified controller interface, diverse workload scenarios that stress the system in different ways, and explicit accounting of per-decision overhead, all on a real heterogeneous continuum testbed under a conjunctive multi-dimensional SLO.

Using this benchmarking platform, we show that scheduler performance is strongly environment-dependent and that higher decision cost does not necessarily lead to better SLO compliance.

Thus, the main contributions of this work include:

- A CMMS benchmark platform with a unified controller interface on a real heterogeneous edge cluster, enabling

fair comparison across algorithms by reducing testbed-specific integration effects.

- A four-scenario evaluation methodology that exercises controllers under different operating conditions, reducing the risk of conclusions that depend on a single workload setting.
- A comparative evaluation of six controllers from three control paradigms, showing that controller behavior varies across scenarios and that per-decision overhead provides important context for interpreting SLO outcomes.

II. BACKGROUND

Although many systems have been proposed for CMMS, fair comparison across them remains difficult. Existing work usually evaluates each scheduler in its own stack, under limited workloads, and often without accounting for decision overhead (§II-A). Benchmarks from related areas address parts of this problem, but not the full CMMS setting (§II-B).

A. Why Fair Comparison Does Not Exist

CMMS-style control has been tackled by a wide spectrum of schedulers, each developed inside its own evaluation environment. Video analytics systems couple profiling, configuration search, and execution into tightly integrated prototypes [3], [8]; adaptive bitrate controllers are trained and evaluated inside their own network simulator [9]; cluster-job schedulers target a single datacenter workload trace [5], [10]. More recently, active inference has been applied to edge AI service routing on a heterogeneous DCCS testbed [6] and to adaptive stream processing under multi-dimensional SLOs [7], broadening the controller space beyond rule-based heuristics and reinforcement learning. In every case, however, the scheduling algorithm is distributed together with its own profiling or execution harness, making results hard to compare across papers. Most studies also consider only narrow workload conditions, leaving robustness under dynamic demand unclear. In addition, scheduler overhead is frequently omitted, even though on edge platforms a single decision can consume a noticeable fraction of a 200 ms SLO budget. As a result, the literature still lacks a clean head-to-head comparison of CMMS algorithms across design families.

B. Why Existing Benchmarks Do Not Help

General-purpose benchmarks address related concerns but not CMMS directly. MLPerf [11], [12] fixes reference models and workloads to compare inference backends, not scheduling algorithms that operate on top of those backends. DeathStar-Bench [13] provides representative microservice workloads, but its focus is the infrastructure and service-mesh substrate rather than per-request scheduling decisions. SPEC elasticity benchmarks [14] target IaaS auto-scaling, i.e., adding or removing replicas in response to load; this differs structurally from continuous multi-mode dispatch, which changes the per-stream processing parameters at each control step. Consequently, none of these benchmarks provides a unified way to evaluate CMMS policies under dynamic workloads while accounting for controller overhead on heterogeneous edge hardware. This gap motivates the benchmark platform presented next.

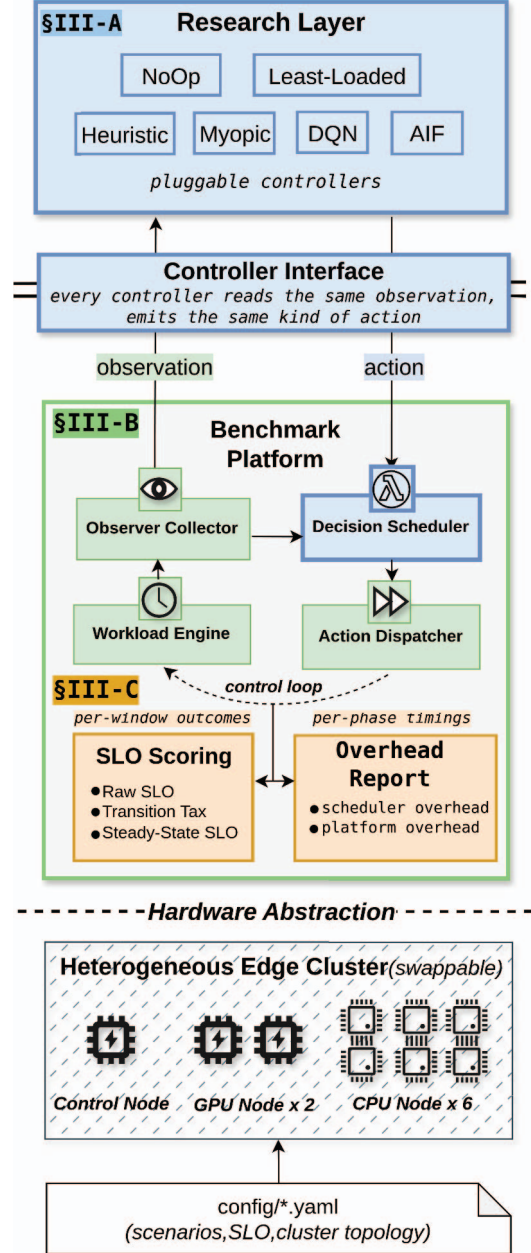


Fig. 1. Three-layer platform architecture. The fixed middle layer comprises the unified interface, closed-loop evaluation harness, and scoring stage.

III. PLATFORM DESIGN

The comparison of CMMS solutions remains constrained by testbed coupling, single-scenario masking, and invisible decision overhead. In response, our platform, as shown in Fig. 1, is organized around three components: a unified controller interface (§III-A), closed-loop evaluation (§III-B), and dual-metric scoring (§III-C). In the following, we describe these three platform components in more detail.

A. Fairness Boundary: The Controller Interface

To ensure a fair comparison across controllers, the platform requires all methods to operate within a unified and restricted decision boundary. At each control step, a controller may only act on the basis of the current system observation, either by keeping the existing allocation unchanged or by making a scheduling decision for a particular stream. To this end, the platform abstracts scheduling behavior into four action modes with stable semantics: FULL, LITE, OFFLOAD, and SKIP, corresponding respectively to full local processing, degraded local processing, remote execution, and skipping the current window. This abstraction captures the principal decision space of edge AI scheduling while avoiding ties to implementation-specific details such as model choice, input resolution, or node configuration, thereby enabling different controllers to be evaluated fairly through online decision-making under a shared task semantics.

The observation is identical for all controllers. It includes per-stream metrics (one-second p95 latency, effective fps, per-window output count, and current processing mode), per-node state (queue depth, rolling inference time, and online status), and cluster-level summaries (GPU/CPU utilization, power draw, and mode distribution). Thus, every controller, regardless of its internal decision logic, operates under the same information boundary.

The platform also fixes two interface-level rules that would otherwise affect comparison. A mode cannot be exited until it has remained active for eight control steps, and over-eager switch requests are dropped. When a controller requests OFFLOAD without specifying a target node, the platform selects the destination by minimum expected completion time (ECT), $ECT = (q + 1) \cdot \bar{t}_{infer}$, as a common default for all controllers.

B. Closed-Loop Control Harness

To evaluate different controllers under comparable conditions, the platform adopts a closed-loop control harness that places workload evolution and control decisions within the same execution loop. Workloads are defined in advance as scenario scripts—ordered lists of timed stream-arrival and -departure events—and replayed event by event by the platform, so that all controllers experience identical pressure trajectories; as a result, observed differences primarily reflect how controllers respond to the same scenario rather than differences in the test environment itself.

At each control step, the platform executes workload events, state observation, decision invocation, and action dispatch in a fixed order, while recording the duration of each phase separately. This design not only preserves consistency in system evolution, but also explicitly separates controller decision time from platform runtime overhead: only the duration of the decision-invocation phase is counted as scheduler overhead, while the remaining phases are reported as platform overhead. After dispatch, the platform uses compensated sleep to maintain a fixed one-second control period, and the initial warmup step is excluded from the episode summary.

C. Dual-Metric SLO Scoring

A single aggregated SLO score can conflate how a controller performs overall with how it performs once its decisions have settled. To make this difference explicit, the platform is built around two primary SLO measures: raw SLO captures overall compliance across all non-SKIP windows, while steady-state SLO excludes transient windows after mode switches and reflects compliance only during stable operation. Their difference is further reported as transition tax, which makes the effect of switching explicit. For example, under heavy load, two controllers separated by 3 pp on Raw SLO collapse to near-parity on Steady-State SLO, with the gap accounted for almost entirely by transition tax (§IV-D).

Concretely, the platform evaluates each stream in each one-second window against three conditions: latency, effective frame rate, and detection quality. SKIP windows are not counted as failures, but are instead constrained by a separate episode-level budget. On this basis, the platform reports Raw SLO, Steady-state SLO, and their difference, rather than collapsing them into a single score that would implicitly impose a subjective weighting across operating phases. In parallel, the platform reports per-step scheduler and platform overheads alongside the SLO metrics, so that decision cost can be examined independently of SLO quality rather than silently absorbed into it.

IV. EVALUATION

This section evaluates whether the three platform components can provide the expected utility—quantified through three research questions (RQ1–RQ3). To address the three RQs, we use our benchmarking platform to compare six scheduling controllers with distinct design philosophies across five cluster configurations and two workload regimes.

TABLE I
EXPERIMENTAL CONFIGURATION.

Parameter	Value
Coordinator	Jetson Orin (5.3 TFLOPS)
GPU workers	up to 2× Jetson Orin Nano (1.7 TFLOPS)
CPU workers	up to 6× Ryzen 7, 3 throttle tiers
Clusters	2-homo, 2-het, 5-node, 8-node (LIGHT); 8-node (HEAVY)
Model	YOLOv8n [15] @ 640×640
Video	720p urban traffic, 30 fps
Control interval	1 Hz
SLO latency	p95 ≤ 200 ms
SLO fps	≥ 10 (OFFLOAD ≥ 5)
SLO DQR	≥ 0.35
Min dwell	8 steps
Scenarios	4 LIGHT (peak 6–8 streams); 4 HEAVY (peak 10–13)
Controllers	NoOp, Least-Loaded, Heuristic, Myopic, DQN [16], AIF [17]; all 6 run on LIGHT and HEAVY
Runs per cell	$n=3$
Total episodes	424

A. Experimental Setup

Table I summarizes the experimental setup: we vary configuration and workload separately to test whether controller performance shifts across settings and under pressure (i.e. HIGH and LOW load). Because all controllers share the same interface, any observed difference in controllers' performance can be directly attributed to the controller, marginalizing the effect of the experimental environment.

DQN is pretrained per (cluster, scenario) pair with a uniform budget, so its HEAVY (high-load, 10–13 streams) results reflect in-distribution rather than transfer performance (training-budget caveat in §IV-E).

1) *Evaluation Scenarios*: Each cluster is evaluated on four stress scenarios that probe distinct temporal request patterns:

- **Ramp-up**: active stream count rises from zero to peak over the first half of the episode, then held.
- **Burst**: steady baseline with short 2–3 stream bursts added and removed every ~ 25 s.
- **Steady-overload**: peak stream count applied from $t=0$ and held throughout.
- **Oscillating**: stream count follows a sin-like pattern with ~ 60 s period.

Each scenario runs at two regimes: **LIGHT** (peak 6–8 streams, within every cluster's sustainable GPU capacity) and **HEAVY** (peak 10–13 streams, deliberately pushing the 8-node cluster beyond local throughput).

2) *Scheduling Controllers*: We compare six controllers spanning distinct decision-making paradigms:

- **NoOp** (*passive baseline*): never changes processing mode.
- **Least-Loaded** (*rule-based*): dispatches OFFLOAD to the node with minimum ECT; no mode switching.
- **Heuristic** (*rule-based*): thresholded state machine; promotes SKIP→LITE→FULL when the per-stream SLO margin is high, demotes when queue backlog is high.
- **Myopic** (*1-step greedy*): selects the (stream, mode, node) triple minimizing a precomputed latency–quality cost.
- **DQN** (*deep RL*) [16]: Double DQN, one seed-0 checkpoint per (cluster, scenario) with a 10-episode budget; HEAVY-workload results therefore reflect in-distribution performance, not transfer.
- **AIF** (*active inference*) [17]: hierarchical generative model with hand-designed (A, B, C, D) matrices and planning horizon 3.

3) *Reproducibility*: To enable third-party verification of our results, we release three classes of artifact:

- **Scenario scripts**: the exact workload traces (stream arrivals and departures) used in every experiment.
- **DQN and AIF checkpoints**: the trained policies reported in Section IV, allowing reviewers to skip the multi-hour training stage.
- **Per-step execution histories**: the raw data from which every figure and table in this paper is regenerated.

On the hardware summarized in Table I, the full 424-episode sweep completes in ≈ 70 hours inside an NVIDIA Container

TABLE II
RAW SLO COMPLIANCE (% , MEAN $\pm 1\sigma$) ACROSS 5 CLUSTER CONFIGURATIONS AND 6 CONTROLLERS. BOLD = PER-CLUSTER WINNER.

Ctrl	2-homo	2-het	5-node	8-L	8-H
NoOp	57 $\pm$ 22	57 $\pm$ 22	56 $\pm$ 22	55 $\pm$ 22	19 $\pm$ 15
Least-Loaded	84 $\pm$ 1	83 $\pm$ 2	84 $\pm$ 1	84 $\pm$ 2	44 $\pm$ 17
Myopic	78 $\pm$ 3	78 $\pm$ 3	79 $\pm$ 4	79 $\pm$ 4	52 $\pm$ 14
Heuristic	84 $\pm$ 2	84 $\pm$ 1	84 $\pm$ 2	84 $\pm$ 2	74$\pm$5
DQN	86$\pm$6	88$\pm$6	91 $\pm$ 4	90 $\pm$ 7	46 $\pm$ 14
AIF	79 $\pm$ 7	80 $\pm$ 6	93$\pm$4	91$\pm$5	71 $\pm$ 6

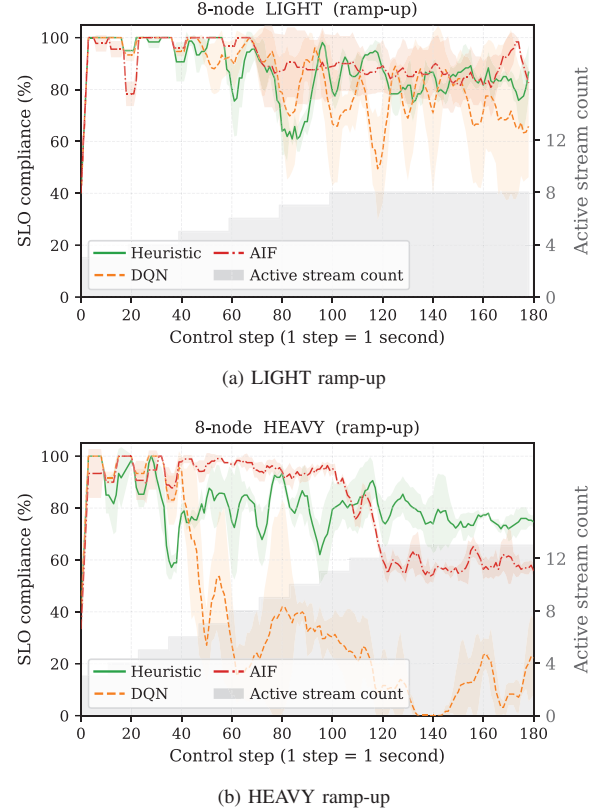


Fig. 2. Per-step SLO compliance on the 8-node cluster under two workload regimes. Gray shading shows active stream count; curves show per-controller SLO ($n=3$, 5-step moving average, $\pm 1\sigma$). Each control step corresponds to one second of wall-clock time.

Toolkit Docker image; all artifacts are available through a public code repository¹.

B. Controller Performance across Load Pattern (RQ1)

To give a fair comparison between different scheduling controllers, we evaluate controllers across increasing cluster sizes and gradually more challenging workload pattern. In particular, we compare per-cluster Raw SLO compliance across all 5 cluster configurations and 6 controllers (Table II; per-cluster winner in bold), and examine per-step SLO traces on the 8-node cluster under LIGHT (Fig. 2a) and HEAVY (Fig. 2b)

¹<https://anonymous.4open.science/r/cmms-benchmark-7D80>

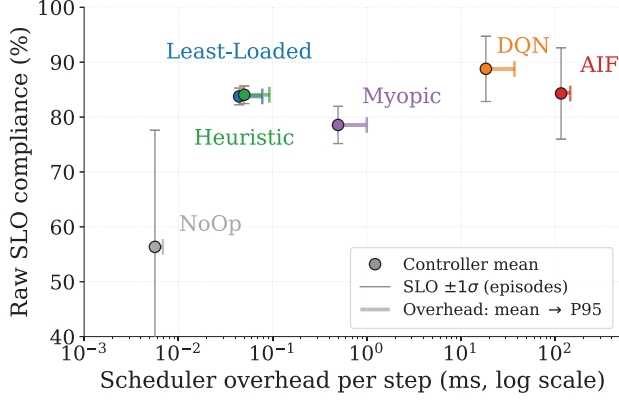


Fig. 3. Raw SLO versus per-step scheduler overhead (log scale) across the four LIGHT cluster configurations. Vertical bars denote SLO standard deviation across episodes; horizontal bars extending to the right of each marker span the mean-to-P95 range of per-step scheduler overhead.

ramp-up to see whether the identity of the leading controller holds across workload regimes. Figures 2a and 2b, together with Table II, show that no controller maintains a stable ranking across configurations. Under LIGHT, Heuristic, DQN, and AIF remain within a similar performance band, and the nominal per-cluster winners (bold) fall inside the run-to-run variance. Under HEAVY, the picture inverts sharply: moving from 8-node LIGHT to HEAVY drops DQN by 44 pp (89.9% $\rightarrow$ 45.8%), AIF by 19 pp, and Heuristic by only 10 pp—the rule-based controller that was not dominant under LIGHT ends up with the highest HEAVY compliance. This rank reversal is what a single-configuration benchmark would miss.

C. Controller Overhead vs. SLO Fulfillment (RQ2)

Next, we compare the computational complexity of scheduling algorithms with their effective SLO fulfillment. Thus, we aim to construct a Pareto front that can answer for resource-constrained environments which algorithm to use. Figure 3 shows that decision cost and control quality are not simply aligned. Under LIGHT, per-step scheduler overhead spans several orders of magnitude across controllers, whereas Raw SLO remains within a much narrower band. In particular, AIF and Heuristic achieve nearly indistinguishable compliance (84.3% vs. 84.1%) despite a roughly $2000\times$ gap in decision cost (115 ms vs. 0.05 ms per step), indicating that higher computational expense does not by itself imply higher control quality.

This becomes even clearer under HEAVY. As load increases, the low-cost Heuristic does not lose competitiveness; instead, it outperforms both AIF and DQN in SLO while remaining substantially cheaper per decision. Once cost and quality are reported together, the identity of the preferred controller is therefore no longer determined by compliance alone, but by the trade-off required to achieve it.

The platform can reveal this trade-off only because algorithm cost is measured separately from platform overhead. Within a

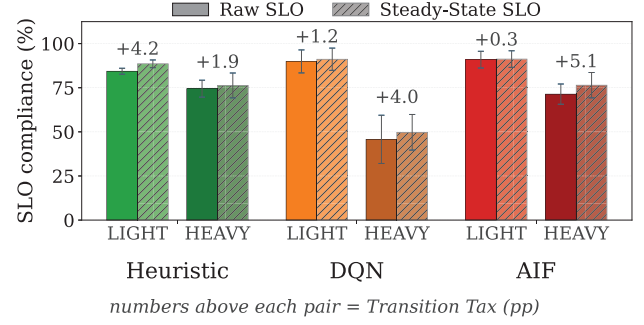


Fig. 4. Raw SLO (solid) and Steady-State SLO (hatched) on 8-node LIGHT (left) and HEAVY (right). Numbers above each pair report Transition Tax in percentage points.

given cluster, the platform-side phases remain broadly stable across controllers, with variation driven mainly by stream count rather than by the controller itself. As a result, `schedule_ns` reflects controller cost as an algorithmic property, rather than an artifact of the harness—an aspect that prior RL-for-scheduling work [9], [5] does not report alongside task-level SLO.

D. Why Dual-Metric Matters (RQ3)

Lastly, we analyze the effect of separating raw from steady-state SLOs. Thus, we aim to reveal the switching costs that emerge from changing configurations, which a single (i.e., raw) score would otherwise conflate. Figure 4 shows why a single raw SLO score is not sufficient to characterize controller behavior. On 8-node HEAVY, Heuristic appears to outperform AIF by 3 pp if comparison is based on raw SLO alone (74.5% vs. 71.4%). Once the same comparison is repeated using Steady-State SLO, however, the gap collapses to 0.14 pp (76.3% vs. 76.5%). The difference between the two controllers therefore lies less in steady-state quality itself than in the cost incurred during switching.

This is precisely why dual-metric scoring matters. Raw SLO reflects compliance over the full execution trace, whereas Steady-State SLO isolates behavior during stable operation by excluding transient windows after mode switches. Their difference, reported as Transition Tax, makes the cost of switching explicit: AIF pays +5.1 pp, nearly three times Heuristic's +1.9 pp, which is where the apparent Raw-SLO gap comes from. Collapsing these two views into a single scalar would therefore conflate steady-state quality with switching cost and obscure a behaviorally important distinction between controllers. From a deployment standpoint, this distinction carries concrete trade-offs: AIF's higher switch rate is a cost under episodic workloads with long stable intervals, but a feature under rapidly shifting traffic.

E. Limitations

These results should be interpreted within four boundaries. First, each (cluster, controller, scenario) cell includes only $n=3$ runs, so 1–2 pp gaps in the LIGHT regime should not be read as stable algorithmic rankings. Second, the HEAVY regime is

evaluated only on the 8-node cluster, and its scaling behavior on smaller clusters is therefore not yet characterized.

Third, DQN uses a fixed 10-episode, single-seed training budget for each (cluster, scenario) pair. A larger budget or multi-seed training could reduce part of its HEAVY deficit, but only at substantially higher training cost. Finally, the current study is limited to one hardware family and one workload type, namely video analytics on a Jetson/Ryzen heterogeneous cluster. Generalization to other CMMS settings, such as ML serving, sensor fusion, or multi-tenant GPU scheduling, remains future work.

V. CONCLUSION

We presented a benchmark platform for Continuous Multi-Mode Scheduling on heterogeneous edge clusters and used it to study whether controller comparisons remain stable across configurations. The central finding is that they do not: controller rankings depend on cluster scale, heterogeneity, and workload regime in ways that a single-setup benchmark would hide. In particular, the apparent winner under light load does not necessarily remain competitive under heavy load, and controllers that look similar under a single aggregate score can separate once decision cost and switching effects are made explicit.

These results reinforce the role of the platform as an evaluation instrument rather than an algorithm-specific testbed. By holding the interaction boundary fixed, separating controller cost from platform overhead, and reporting both Raw and Steady-State SLO, the platform exposes differences that would otherwise be collapsed into a single number or a single scenario. The resulting picture is not that one controller is universally best, but that meaningful comparison requires multiple configurations, explicit cost accounting, and metrics that distinguish steady-state quality from transition behavior.

Our study remains limited to one hardware family and one workload type, namely video analytics on a Jetson/Ryzen heterogeneous cluster. Extending the platform to additional controller families and larger clusters, and validating it in other CMMS settings such as ML serving, sensor fusion, and multi-tenant GPU scheduling, remains future work.

REFERENCES

- [1] B. Sedlak, V. C. Pujol, P. K. Donta, and S. Dustdar, "Equilibrium in the computing continuum through active inference," *Future Generation Computer Systems*, vol. 160, pp. 92–108, 2024.
- [2] A. Lapkovskis, B. Sedlak, S. Magnússon, S. Dustdar, and P. K. Donta, "Benchmarking dynamic slo compliance in distributed computing continuum systems," in *2025 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2025, pp. 93–102.
- [3] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 377–392.
- [4] A. Gujarati, R. Karimi, S. Alzayat, W. Hao, A. Kaufmann, Y. Vigfusson, and J. Mace, "Serving DNNs like clockwork: Performance predictability from the bottom up," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020, pp. 443–462.
- [5] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM SIGCOMM Conference*, 2019, pp. 270–288.
- [6] Z. Wang, B. Sedlak, and S. Dustdar, "Active inference-based adaptive routing for heterogeneous edge AI services," in *Service-Oriented Computing – ICSOC 2025 Workshops (SOC4AI)*. Springer, 2025, to appear.
- [7] B. Sedlak, V. Casamayor Pujol, A. Morichetta, P. K. Donta, and S. Dustdar, "Adaptive stream processing on edge devices through active inference," *Evolving Systems*, 2025.
- [8] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proceedings of the 2018 ACM SIGCOMM Conference*, 2018, pp. 253–266.
- [9] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with Pensieve," in *Proceedings of the ACM SIGCOMM Conference*, 2017, pp. 197–210.
- [10] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proceedings of IEEE INFOCOM*, 2020, pp. 257–266.
- [11] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou *et al.*, "MLPerf inference benchmark," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020, pp. 446–459.
- [12] P. Mattson, C. Cheng, C. Coleman, G. Damos, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf *et al.*, "MLPerf training benchmark," in *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys)*, 2020.
- [13] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson *et al.*, "An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems," in *Proceedings of the 24th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 3–18.
- [14] N. R. Herbst, S. Kounev, A. Weber, and H. Groenda, "BUNGEE: An elasticity benchmark for self-adaptive IaaS cloud environments," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE Press, 2015, pp. 46–56.
- [15] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," <https://github.com/ultralytics/ultralytics>, 2023.
- [16] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.
- [17] T. Parr, G. Pezzulo, and K. J. Friston, *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. MIT Press, 2022.