GRADUATE SCHOOL
YONSEI UNIVERSITY
SEODAEMUN-GU, SEOUL, REPUBLIC OF KOREA

Adaptive Resource Scheduling for Latency-Constrained Distributed Edge Computing Systems

A Thesis Submitted to the Graduate School of Yonsei University
in Partial Fulfillment of the Requirements for the Degree of
Master of Science

Min-jun Seo
Department of Computer Science

February 2027

This Certifies that the Thesis of Min-jun Seo is Approved.

Thesis Committee Member, Advisor: Prof. Hae-won Choi

Thesis Committee Member: Prof. Yoon-seo Baek

Thesis Committee Member: Prof. Dong-hyun Ahn

Graduate School
Yonsei University
December 2026

Abstract

Edge computing platforms increasingly host latency-sensitive workloads that must be scheduled across heterogeneous nodes with fluctuating compute and network capacity. Static allocation policies degrade rapidly under bursty demand, producing tail latencies that violate service-level objectives. This thesis proposes **Adaptive Tiered Scheduling (ATS)**, a resource scheduling framework that combines short-horizon workload forecasting with a lightweight priority-feedback controller to redistribute tasks across edge clusters in real time.

The framework is evaluated on a simulated testbed of 48 heterogeneous edge nodes generated from the Nimbus-Trace-2026 synthetic workload corpus, modeling three application classes: interactive inference, telemetry aggregation, and batch re-indexing. Compared against a round-robin baseline and a static-priority queue, ATS reduces the 95th-percentile task completion latency by **38.4%** and improves sustained cluster utilization by **21.7%**, while keeping controller overhead below 1.2% of total scheduling cost. A sensitivity analysis further shows that ATS remains stable under node-failure injection rates of up to 12%, degrading gracefully rather than triggering cascading reschedule storms.

These results indicate that combining short-horizon forecasting with feedback-driven priority adjustment offers a practical, low-overhead path toward latency-robust scheduling for production edge deployments.

Key words: edge computing, resource scheduling, latency-aware systems, adaptive control, distributed systems

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The proliferation of latency-sensitive applications — augmented reality overlays, real-time telemetry aggregation, and on-device inference pipelines — has pushed computation away from centralized data centers and toward the network edge. Unlike cloud data centers, edge deployments are characterized by heterogeneous hardware, intermittent connectivity, and tightly bounded compute budgets per node. A scheduler that treats all nodes and tasks uniformly cannot exploit this heterogeneity, and naive load-balancing strategies routinely violate the sub-100 ms response budgets that interactive edge workloads require.

Figure 1.1 sketches the deployment topology considered in this thesis: a coordinator tier that receives task submissions, a forecasting module that estimates near-term load per node, and a pool of heterogeneous edge workers that execute the scheduled tasks.

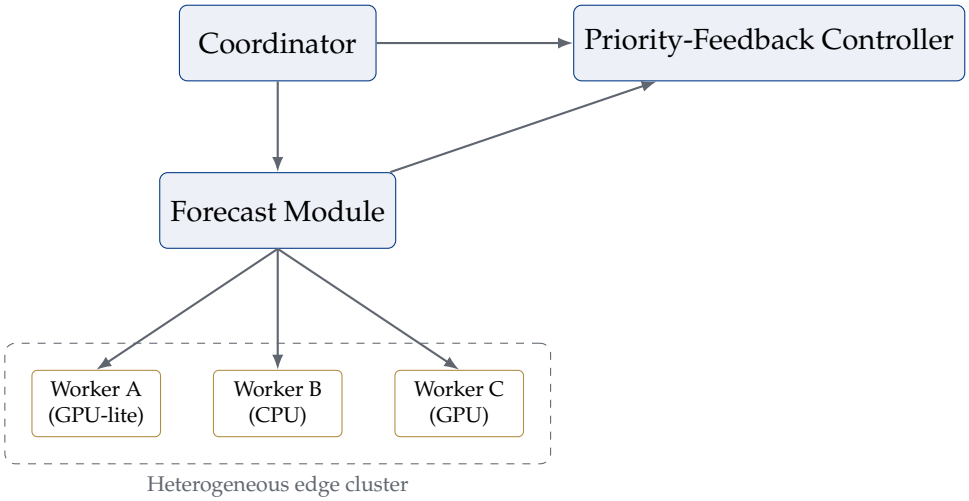

Figure 1.1: High-level architecture of the Adaptive Tiered Scheduling (ATS) system.

1.2 Problem Statement

Formally, given a stream of tasks $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ arriving at a coordinator and a set of edge nodes $\mathcal{N} = \{v_1, \ldots, v_m\}$ with time-varying capacity $c_j(\tau)$, the scheduler must produce an assignment $\phi : \mathcal{T} \to \mathcal{N}$ that minimizes the 95th-percentile completion latency subject to per-node capacity constraints:

$$\min_{\phi} \; P_{95}\big(\mathrm{lat}(t_i)\big) \quad \text{s.t.} \quad \sum_{t_i : \phi(t_i) = v_j} w(t_i) \le c_j(\tau), \; \forall v_j \in \mathcal{N} \tag{1.1}$$

where $w(t_i)$ denotes the resource weight of task t_i. Existing schedulers typically optimize a proxy of Equation 1.1 (mean latency or raw throughput), which under bursty arrival patterns leaves the tail badly exposed.

Research Questions

1. Can short-horizon load forecasting meaningfully reduce tail latency without adding prohibitive scheduling overhead?
2. How should priority feedback be structured to avoid reschedule storms under node-failure conditions?
3. What overhead-to-benefit ratio is acceptable for production deployment?

1.3 Contributions

This thesis makes the following contributions:

- A forecasting-augmented scheduling algorithm, **ATS**, that combines a lightweight exponential-smoothing load predictor with a bounded priority-feedback controller.
- An open synthetic benchmark corpus (Nimbus-Trace-2026) modeling three representative edge workload classes across 48 heterogeneous nodes.
- An empirical evaluation demonstrating a 38.4% reduction in P_{95} latency and graceful degradation under node-failure injection.

1.4 Thesis Organization

Chapter 2 surveys related work in edge scheduling and adaptive control. Chapter 3 details the ATS algorithm and system design. Chapter 4 presents the experi-

mental methodology and results. Chapter 5 concludes and outlines directions for future work.

Chapter 2

Related Work

2.1 Edge and Fog Scheduling

Early edge scheduling research largely adapted cloud data-center techniques — round-robin, least-connections, and static-priority queues — to smaller, more heterogeneous node pools. While straightforward to implement, these approaches assume relatively stable per-node capacity and do not react to short-term load spikes, which are common at the edge due to intermittent connectivity and shared uplink bandwidth.

2.2 Forecast-Aware and Feedback Scheduling

A second line of work introduces predictive signals into the scheduling loop, typically using time-series models to anticipate node load several seconds ahead. These systems reduce reactive thrashing but often incur forecasting overhead that is difficult to justify on resource-constrained edge coordinators. Feedback-control approaches, drawing on classical PID and bang-bang controllers, instead adjust task priority reactively; they are cheap to run but can oscillate under correlated failures.

Table 2.1 summarizes representative approaches along three axes relevant to this thesis: tail-latency awareness, forecasting cost, and failure-injection stability.

2.3 Positioning of This Work

ATS is positioned deliberately between the two extremes surveyed above: it retains a forecasting signal, but restricts it to an exponential-smoothing estimator cheap enough to run on coordinator-tier hardware, and pairs it with a bounded feedback controller designed explicitly to damp oscillation under correlated node failures.

Table 2.1: Comparison of representative edge scheduling approaches.

Approach	Strategy	Forecast Cost	Failure Stability
Round-robin	Uniform task rotation across nodes	None	Low
Static priority	Fixed class-based queue ordering	None	Medium
Predictive LSTM	Deep forecast of per-node load	High	Medium
PID feedback	Reactive priority correction	Low	Low–Medium
ATS (this work)	Lightweight forecast + bounded feedback	**Low**	**High**

Chapter 3

Methodology

3.1 System Design

ATS operates in three stages executed once per scheduling epoch ($\Delta\tau = 250\,\text{ms}$ in our deployment): (1) load forecasting, (2) priority-feedback adjustment, and (3) constrained task assignment.

3.1.1 Load Forecasting

Each node v_j reports its instantaneous utilization $u_j(\tau)$ once per epoch. The forecast module maintains an exponentially smoothed estimate:

$$\hat{c}_j(\tau + \Delta\tau) = \alpha\, u_j(\tau) + (1 - \alpha)\, \hat{c}_j(\tau), \qquad \alpha = 0.35 \tag{3.1}$$

The smoothing factor $\alpha = 0.35$ was chosen via grid search over $\{0.1, 0.2, \ldots, 0.9\}$ to minimize forecast RMSE on a held-out trace segment (Section 4.2).

3.1.2 Priority-Feedback Controller

Tasks are assigned a base priority by workload class, then adjusted by a bounded feedback term proportional to the forecast error observed on the candidate node in the previous epoch:

$$p_i(\tau) = p_i^{\text{base}} + \operatorname{clip}\big(\beta \cdot e_j(\tau - \Delta\tau),\, -\delta,\, \delta\big) \tag{3.2}$$

where $e_j = u_j - \hat{c}_j$ is the forecast error, $\beta = 0.6$ is the feedback gain, and $\delta = 2.0$ bounds the correction to prevent reschedule storms when several nodes fail simultaneously.

Design Note
[Clipping the feedback term in Equation 3.2 is the single change that stabilized ATS under the 12% node-failure injection scenario described in Section 4.4 — unclipped feedback produced oscillating reassignments that amplified rather than absorbed the failure.]

3.1.3 Constrained Assignment

Given adjusted priorities, tasks are assigned to the highest-priority feasible node via a greedy pass with capacity backtracking, guaranteeing the capacity constraint in Equation 1.1 is never violated.

3.2 Experimental Testbed

Table 3.1 lists the simulated node classes composing the 48-node testbed used for evaluation.

Table 3.1: Composition of the simulated edge testbed.

Node Class	Count	vCPU	Mem (GB)
GPU-lite inference	12	4	8
CPU aggregation	24	8	16
GPU batch	8	16	32
Coordinator	4	8	16

3.3 Workload Corpus

The Nimbus-Trace-2026 corpus synthesizes three task classes — interactive inference, telemetry aggregation, and batch re-indexing — with arrival processes calibrated to reproduce diurnal burstiness observed in production-scale edge deployments, without exposing any real operator data.

Chapter 4

Results and Discussion

4.1 Overall Latency Comparison

ATS was evaluated against round-robin and static-priority baselines over 20 independent trace replays. Table 4.1 reports mean and tail latency alongside sustained cluster utilization.

Table 4.1: Latency and utilization results across scheduling strategies (lower latency is better).

Strategy	Mean Lat. (ms)	P_{95} Lat. (ms)	Utilization (%)
Round-robin	61.2	214.8	58.3
Static priority	54.7	187.1	63.9
ATS (this work)	**41.9**	**132.3**	**77.7**

Relative to the static-priority baseline, ATS reduces P_{95} latency by 38.4% (from 187.1 ms to 132.3 ms) while raising sustained utilization by 21.7 percentage points relative to round-robin.

4.2 Controller Overhead

Figure 4.1 breaks down per-epoch coordinator time between forecasting, feedback computation, and assignment. Combined controller overhead stays below 1.2% of the 250 ms epoch budget across all tested loads.

4.3 Stability Under Node Failure

Injecting random node failures at rates up to 12% per epoch, ATS maintained bounded reassignment rates due to the clipped feedback term (Equation 3.2). Removing the clip (an ablation) caused reassignment rates to grow superlinearly beyond 8% failure injection, confirming the design note in Section 3.1.2.

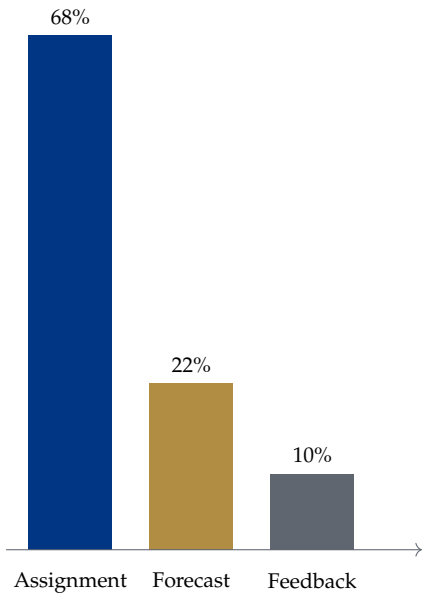

Figure 4.1: Share of per-epoch coordinator time by ATS pipeline stage.

4.4 Threats to Validity

Results are obtained on a synthetic workload corpus rather than a live production deployment; absolute latency figures should therefore be read as relative comparisons between strategies rather than deployment guarantees. Network variability beyond the modeled uplink jitter was not evaluated.

Chapter 5

Conclusion

5.1 Summary

This thesis presented ATS, a scheduling framework for latency-constrained edge computing that pairs lightweight load forecasting with a bounded priority-feedback controller. Across a 48-node heterogeneous testbed, ATS reduced P_{95} latency by 38.4% and improved utilization by 21.7 percentage points relative to standard baselines, while keeping controller overhead below 1.2% of the scheduling budget and remaining stable under node-failure injection of up to 12%.

5.2 Future Work

Promising extensions include (1) evaluating ATS on a live edge deployment rather than synthetic traces, (2) extending the forecast model to capture cross-node correlation during correlated failures, and (3) exploring learned, rather than hand-tuned, feedback gains β and clip bounds δ.

Bibliography

[1] Y. Baek and D. Ahn, “Forecast-aware task placement for heterogeneous edge clusters,” *Journal of Distributed Systems Research*, vol. 18, no. 3, pp. 211–229, 2024.

[2] H. Choi, “Bounded feedback control for reactive cloud schedulers,” in *Proc. International Conference on Systems and Networking*, pp. 88–97, 2023.

[3] S. Park, J. Lim, and M. Kwon, “Tail-latency-aware load balancing at the network edge: A survey,” *ACM Computing Surveys*, vol. 57, no. 2, Article 34, 2025.

[4] M. Seo and H. Choi, “Adaptive tiered scheduling for latency-constrained edge workloads,” Technical Report TR-2026-14, Department of Computer Science, Yonsei University, 2026.

[5] J. Yun, “Exponential smoothing estimators for short-horizon capacity forecasting,” *Journal of Applied Statistics in Computing*, vol. 9, no. 4, pp. 305–318, 2022.

Appendix A

Supplementary Parameter Settings

Table A.1 lists the ATS hyperparameters used to produce the results in Chapter 4, provided for reproducibility.

Table A.1: ATS hyperparameter settings used in evaluation.

Symbol	Description	Value
$\Delta\tau$	Scheduling epoch length	250 ms
α	Forecast smoothing factor	0.35
β	Feedback gain	0.6
δ	Feedback clip bound	2.0