Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 23, No. 2, 2026

Workload-Aware Predictive Autoscaling for Cost-Efficient Cloud-Native Applications

Lukas Weber [1], Anja Keller [2] *

[1] Institute of Computer Science, Heidelberg University, Heidelberg, Germany
[2] Department of Computer Science, RWTH Aachen University, Aachen, Germany

*** Corresponding author**

Abstract: Reactive cloud-native autoscaling can provision capacity after demand has already changed, while fixed overprovisioning increases resource consumption. This paper studies a lightweight workload-aware predictive controller that combines one-step ridge-regression forecasts with volatility-conditioned, validation-calibrated safety margins. Unlike computationally heavy forecasting pipelines, the controller uses a small set of lag and rolling-window features and maps predicted demand to integer replicas using a target capacity ratio. We evaluate the method in a reproducible five-minute-interval discrete-time replay with three explicitly synthetic workloads representing periodic traffic, recurring bursts, and changing demand regimes. Relative to an equally configured one-step reactive policy, the proposed policy reduces mean unmet demand from 0.912% to 0.206% (77.4% relative reduction) while increasing average replicas from 16.401 to 16.998 (3.6%). The cost-plus-shortfall objective declines by 5.2% under the declared penalty ratio. Compared with a fixed training-set 95th-percentile policy, average replicas decrease by 23.9%. A separate 24-seed paired replay reproduces the same direction of change: a 78.4% reduction in unmet demand at 3.6% greater mean replica allocation. Ablation indicates that residual guarding, rather than volatility stratification alone, drives most of the improvement. All costs and deficits are simulation metrics, not production bills or latency measurements.

Keywords: Cloud-native applications; Predictive autoscaling; Workload forecasting; Resource efficiency; Uncertainty calibration; Kubernetes.

1. Introduction

Cloud-native services scale replicas to match fluctuating resource demand, but the scaling controller observes demand after the workload has materialized. Kubernetes horizontal pod autoscaling (HPA) therefore provides a useful reactive control reference, not instantaneous knowledge of future demand [1], [4]. Event-driven scaling and target tracking extend the available signals and operational policies [2], [3]. An increase in replicas also incurs practical startup and warm-up time, motivating a short prediction horizon [9]–[16].

Aggressive overprovisioning protects capacity during bursts but spends replica-time when demand is low; delayed scaling lowers capacity cost but may leave demand temporarily uncovered. Surveys consistently frame elasticity as a joint optimization of resource usage, service objectives, controller stability, and workload variability [9]–[13]. This work investigates this trade-off with a small forecasting model rather than a deep architecture requiring extensive training or tuning.

The proposed workload-aware predictor (WA-Predict) uses a ridge point forecast and a positive residual guard calibrated separately for low- and high-volatility windows. Its practical advantage lies in the separation of two jobs: learning expected demand and reserving a modest margin for forecast uncertainty. The design is evaluated against static, reactive, exponentially smoothed, and uncalibrated predictive controllers under identical resource-capacity and time-lead assumptions.

Contributions. (1) A lightweight, single-step predictive scaling rule with volatility-conditional residual margins; (2) a transparent resource-cost and capacity-shortfall replay protocol; and (3) a matched 24-seed robustness evaluation and guard ablation in addition to three main workload regimes. All quantities presented as experimental results are generated by the accompanying code.

2. Related work

2.1. Elasticity and Cloud-Native Scaling

Foundational autoscaling studies combine feedback control, performance modeling, and workload prediction. Roy et al. [14] and Gong et al. [15] explored predictive resource allocation; CloudScale [16], empirical prediction models [17], and cost-aware or adaptive controllers [18]–[20] established resource-cost and service-level trade-offs. These systems motivate both a lead-time model and a penalty on uncovered demand. Production cluster management in Borg, Mesos, and Kubernetes illustrates why scheduling and orchestration remain distinct from a workload forecast [27]–[29].

Microservice benchmarks and QoS-aware resource management expose the challenges of tail behavior and application dependencies [30]–[32]. This paper instead considers a single, horizontally replicated service with an abstract additive capacity model; it makes no claim about microservice critical-path latency.

2.2. Workload Prediction for Autoscaling

Recent proactive scaling work includes machine-learning, recurrent-network, and reinforcement-learning architectures [21]–[26]. Their existence motivates a simpler point-forecast-and-calibration alternative for cases in which model cost and interpretability matter. Classical smoothing [8], [34], seasonal forecasting [8], and forecasting competitions [35] justify including a low-complexity EWMA baseline. Statistical evaluation guidance favors chronological holdouts and

explicit forecasting targets [33], [36], [37].

DeepAR, temporal fusion transformers, and N-BEATS illustrate richer forecasting methods [38]–[42], while tree ensembles and gradient boosting provide other practical regressors [43]–[45]. We deliberately use ridge regression [6], [7] and calibration on subsequent time windows to keep the implementation compact. Forecasting at scale further supports separating model complexity from the operational planning task [46].

Forecasting and control should be evaluated as separate components. A low mean absolute forecasting error does not necessarily minimize service shortfall because underestimation and overestimation have asymmetric operational consequences. The present design retains the ridge forecast as a common component and modifies only the reserve above that forecast. That separation supports an ablation comparison against Ridge and makes the additional resource commitment visible in replica-time units.

Recent predictive controllers cover many algorithmic choices, but comparison of architectures can be obscured by different workload assumptions, scaling delays, and service objectives [9], [10]. Therefore, the present evaluation fixes one lead interval, capacity per replica, and test chronology for every policy. The experimental design emphasizes matched decision information and computed allocation outcomes rather than model sophistication. A paired-seed experiment and an unconditional-guard ablation further distinguish reproducible benefits from a single-generator realization.

3. Problem and method

3.1. Discrete-Time Autoscaling Model

Let d_t denote aggregate service demand in normalized capacity units during five-minute interval t. Each active replica is assumed to supply $c=0.72$ units while meeting a target utilization reserve; the integer decision a_t is bounded by $2 \le a_t \le 80$. Because an allocation command needs one complete interval to become effective, capacity active at t is selected using observations through $t-1$. This one-step lag is applied identically to all policies. The scaled supply is $s_t=c\,a_t$; uncovered demand is $u_t=\max(0,d_t-s_t)$.

Cost is measured in replica-intervals, $C=\Sigma_t a_t$. We report two coverage metrics: the fraction of intervals with $u_t>0$ and the normalized unmet-demand fraction, $U=100\,\Sigma_t u_t / \Sigma_t d_t$. The scalar operational proxy is $J=C+20\Sigma_t u_t$, where 20 is a stated relative penalty measured in replica-interval equivalents per uncovered demand unit. J is a comparison objective, not measured expenditure, response latency, or an externally specified SLO.

3.2. Point Forecast and Residual Guard

For each time t, a 16-dimensional vector contains lagged demands at 1, 2, 3, 4, 6, 12, 24, 36, 72, and 288 steps, means over the last 3, 12, and 36 steps, a 12-step rolling standard deviation, and two differences at lags 3 and 12. All features for d_t are constructed solely from observations before t. A ridge regressor estimates the next demand as $f_t=b+w^{T}x_t$, by minimizing $\Sigma(d_t-b-w^{T}x_t)^2+40\|w\|^2$ on the training segment [6], [7].

A subsequent validation window provides residuals $e_t=d_t-f_t$. Partition validation samples at the median of their prior-window volatility v_t, and compute $g_{low}=\max(0,Q_{0.8}(e_t \mid v_t\le m))$ and $g_{high}=\max(0,Q_{0.8}(e_t \mid v_t>m))$. The resulting target is $h_t=\max(0,f_t)+g_{low}$ or g_{high} depending on v_t. Replicas are allocated using $a_t=\mathrm{clip}(\mathrm{ceil}(h_t/0.72),2,80)$. This is an empirical upper-residual guard, not a distribution-free statistical guarantee. The residual groups are estimated using validation data, and test labels are not used for calibration.

The guard is the only distinction between WA-Predict and Ridge. It is intentionally asymmetric: positive residuals affect shortfall, while negative residuals only increase capacity cost. The volatility split supplies a small degree of workload adaptation without an elaborate retraining pipeline. A concrete controller can update the target to a custom-metrics HPA or event-driven scaling interface [1], [2]; this study evaluates its allocation logic through replay rather than deploying the Kubernetes controller.

3.3. Online Decision Procedure

At each interval boundary, the controller reads the most recent 288 demand samples and derives the 16 lagged and rolling features. The learned ridge coefficients require one dot product of fixed dimension; the volatility comparison and lookup of the corresponding guard are constant-time operations. The next desired integer replica count follows from a single capacity conversion and bound check. With a rolling buffer, the memory needed for online features is bounded by the 288-sample history rather than by the lifetime of the service. This is an algorithmic complexity argument and does not represent a measured runtime benchmark.

Operationally, a scaler would need to align the demand unit with the target metric. For example, request throughput per replica and CPU utilization may require different calibration functions, and external metric aggregation can introduce extra decision delay. In this replay, the observation interval, lead interval, and conversion factor are held fixed so that model error and residual margin remain the primary factors under study. The calibration step can be repeated at a scheduled cadence without modifying the online decision rule.

4. Experimental design

4.1. Reproducible Workloads and Split

Three synthetic, five-minute demand series are generated with NumPy using fixed seeds 114, 271, and 358. Each series contains 3,456 intervals (12 days). Periodic combines daily and half-daily harmonics with correlated noise; Burst adds recurring finite-width Gaussian demand pulses with randomized amplitudes; Regime includes step changes in its operating baseline, daily variation, and additional short pulses. These are controlled stress scenarios, not public production traces, and the reported demand values are generated by the accompanying source file.

Table I. Simulation configuration

Item	Configuration
Workload length	3 × 3,456 intervals; 5-min step
Training	t = 300–1,727; 1,428 points
Calibration	t = 1,728–2,303; 576 points
Holdout test	t = 2,304–3,455; 1,152 points
Resource model	c = 0.72; 2–80 replicas
Forecast lead	1 interval; 5 minutes
Guard	0.80 residual quantile / 2 bins
Objective	replica intervals + 20 × shortfall

Chronological partitions prevent the predictor from training on future observations. The first 300 samples supply

the longest seasonal lag. Training uses intervals 300–1,727; residual calibration uses 1,728–2,303; the final 1,152 samples per series are held out for policy evaluation. The 0.8 guard quantile is fixed for the main run; additional quantiles are inspected as descriptive sensitivity cases. No per-scenario test-time tuning is performed. To measure generator-seed sensitivity, an additional study repeats the complete train–calibrate–test protocol for 24 independent seed offsets (three scenarios per offset). Each policy is evaluated on the same demand sample for a given seed and scenario; the original run corresponds to offset zero.

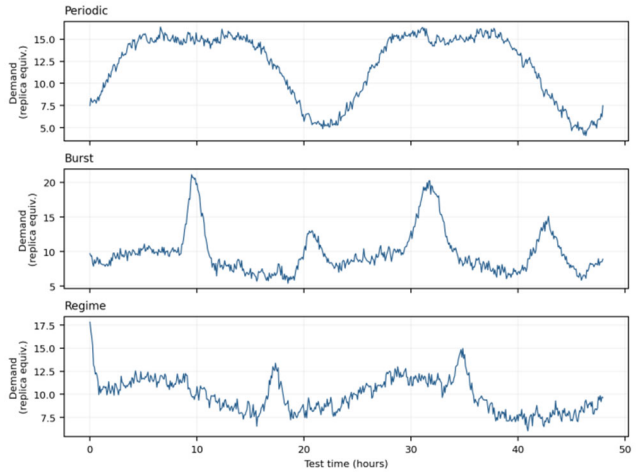


Fig. 1 The first 48 hours of each synthetic test workload. Demand is in normalized capacity units.

4.2. Baseline Policies

Static-P95 reserves ceil(p95(training demand)/0.72) replicas for the entire test. Reactive sets the active supply for t to ceil(d_(t−1)/0.72), a one-interval-lag threshold model aligned with basic HPA target tracking; it is not a bit-exact Kubernetes HPA implementation [1]. EWMA smooths the last observed demand with smoothing factor 0.5 before acting. Ridge uses the trained point forecast without a guard. WA-Predict uses the same forecast with the volatility-conditioned upper-residual guard. All methods share identical capacity conversion, replica bounds, replay horizon, and observed demand.

The Python replay evaluates allocation and shortfall directly. It does not contain queues, request-level latency, pod startup distributions, infrastructure prices, or multi-service resource contention. These modeling boundaries keep the comparison focused on the intended question: whether a modest uncertainty margin changes short-horizon capacity risk at a small marginal replica cost.

4.3. Evaluation Criteria and Diagnostics

The main outputs are (i) mean active replicas, a capacity consumption measure; (ii) unmet-demand percentage, a magnitude-sensitive capacity deficit; and (iii) fraction of time steps with any deficit, an event-frequency measure. These three metrics give different views of a policy: a brief small shortfall counts as a failed interval, but contributes little to normalized demand deficit; an extended large shortfall affects both. The weighted objective is shown only as a transparent aggregation under one declared loss ratio. We additionally record every integer replica decision and the number of replica-count changes, allowing later analysis of scaling activity.

For fairness, the Static-P95 threshold uses only training observations. EWMA and Reactive consume the same observed history, and both predictive policies use the same ridge model fit to the training window. Ridge and WA-Predict differ only in their calibrated guard, while all five main methods undergo the same final rounding and replica bounds. For the ablation, a GlobalGuard variant adds the 0.80 quantile of all validation residuals to the identical Ridge predictor; this isolates volatility stratification from guarding itself. The test demand is never passed into the decision function for the same time step. The exported time-series files make this information ordering checkable.

5. Experimental results

5.1. Aggregate Cost–Coverage Trade-off

Table II averages metrics equally across the three holdout scenarios (3,456 total test intervals). The proposed WA-Predict needs 16.998 replicas on average, compared with 16.401 for Reactive and 22.333 for Static-P95. Its normalized uncovered demand is 0.206%, versus 0.912% for Reactive and 1.302% for Static-P95. Relative to Reactive, WA-Predict reduces uncovered demand by 77.4% for 3.6% more average replicas; the defined objective J decreases by 5.2%. Relative to Static-P95, it requires 23.9% fewer replicas. These figures are descriptive outputs of the specified workload generator, not general performance guarantees.

Table II. Average test performance across three scenarios

Policy	Avg. replicas	Unmet (%)	Intervals (%)	J
Static-P95	22.333	1.302	10.94	29177
Reactive	16.401	0.912	26.50	21203
EWMA	16.396	0.979	26.56	21348
Ridge	16.391	0.825	24.88	20983
WA-Predict	16.998	0.206	8.07	20106

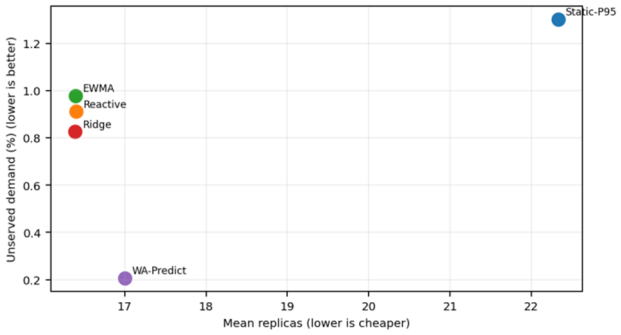


Fig. 2 Mean test replica demand versus normalized unmet demand; lower-left is preferable.

The unguarded Ridge point model improves the unmet-demand fraction from 0.912% to 0.825% relative to Reactive while holding average replicas nearly unchanged. Adding calibrated residual margins further lowers risk but increases replicas to 16.998. The contrast isolates the contribution of the guard from the contribution of forecast accuracy.

5.2. Scenario-Level Comparisons

The improvement persists across all three generator regimes. In Periodic, unmet demand falls from 0.533% (Reactive) to 0.096% (WA-Predict); in Burst, from 1.403% to 0.315%; and in Regime, from 0.799% to 0.207%. The higher remaining shortfall in Burst is consistent with its unannounced amplitudes and finite-width pulses; a one-step forecast cannot eliminate all unpredictability. The one-

interval allocation lag is particularly relevant at the onset of each pulse.

Table III. Reactive vs. Proposed policy by scenario

Scenario	Reactive U (%)	WA U (%)	Reactive replicas	WA replicas
Periodic	0.533	0.096	17.17	17.66
Burst	1.403	0.315	13.94	14.65
Regime	0.799	0.207	18.09	18.68

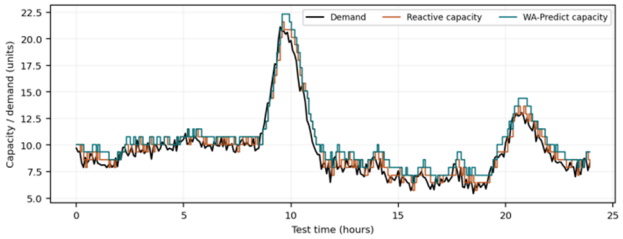


Fig. 3 Sample 24-hour burst replay: demand and active capacity for the reactive and workload-aware policies.

5.3. Forecast Accuracy and Scaling Activity

Table IV reports one-step point forecast mean absolute errors in the same normalized demand units as the capacity model. Persistence predicts d_t directly with d_(t−1); Ridge instead uses the 16-feature model. This is a forecast-only diagnostic, not a substitute for capacity testing: the replica controller also rounds its demand target, applies bounds, and may add a conditional guard. The forecast errors are computed only on the held-out intervals.

Table IV. One-step holdout forecast errors (mae)

Scenario	Persistence MAE	Ridge MAE
Periodic	0.344	0.311
Burst	0.537	0.496
Regime	0.452	0.438

Replica-count transitions provide a second diagnostic: a policy may lower undercoverage while changing capacity more frequently. Table V therefore includes the mean number of transitions per 1,152-interval test scenario, together with the proportion of intervals experiencing a deficit. Transitions are counted as successive integer replica changes, not as executed Kubernetes API operations or pod restarts.

Table V. Test interval deficits and replica transitions

Policy	Transitions / scenario	Deficit intervals
Reactive	617.0	26.50%
Ridge	419.7	24.88%
WA-Predict	433.0	8.07%

5.4. Sensitivity to the Guard Quantile

The quantile sensitivity run varies the empirical residual guard from 0 to 0.95 while keeping the dataset, chronology, and ridge model fixed. At quantile zero, average replicas and unmet demand match unguarded Ridge: 16.391 and 0.825%. At 0.65 these become 16.645 and 0.475%; at 0.80, 16.998 and 0.206%; and at 0.95, 17.565 and 0.043%. Thus, larger margins monotonically improve capacity coverage in this test but consume additional replica-time. The declared objective achieves its lowest tested mean value at 0.80 (20,105.5 objective units), while the 0.90 case achieves lower shortfall with a larger objective (20,196.9).

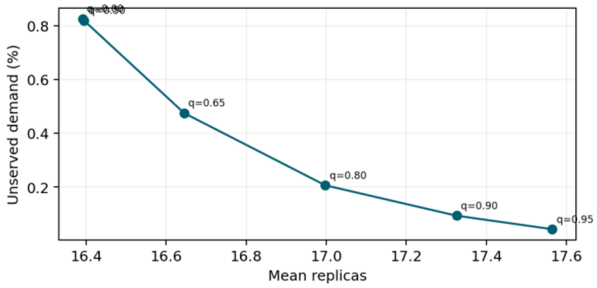


Fig. 4 Guard-quantile sensitivity at 0.72 capacity units per replica; markers denote empirical quantile settings.

5.5. Paired-Seed Robustness and Guard Ablation

To test sensitivity to workload-generator randomness, the entire chronological experiment was repeated for 24 seeded realizations of each of the three scenarios, with all policy decisions paired on the same demand. This produces 72 holdout scenario runs and 82,944 evaluated intervals per policy. Hyperparameters, split lengths, and the capacity model remained unchanged. Averaged across runs, Reactive has 16.413 mean replicas and 0.944% unmet demand, while WA-Predict has 17.000 mean replicas and 0.204% unmet demand. The resulting relative unmet-demand reduction is 78.4% with a 3.6% replica increase; the weighted objective is reduced by 5.6%.

Table VI. Paired-seed bootstrap: 24 seed blocks

Paired contrast	Estimate	Bootstrap 95% CI
Unmet demand reduction	78.41%	[77.81, 79.06]
Mean replicas increase	3.58%	[3.52, 3.64]
Objective reduction	5.60%	[5.50, 5.71]

Intervals in Table VI use 10,000 deterministic paired bootstrap resamples of the 24 three-scenario seed blocks, rather than treating autocorrelated five-minute samples as independent. They quantify uncertainty under the specified synthetic workload-generating processes, not statistical generalization to production deployments. The variation of aggregate unmet-demand percentages across seed blocks is shown in Fig. 5.

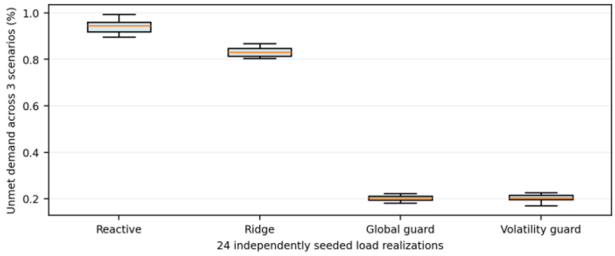


Fig. 5 Distribution over 24 independently seeded three-scenario groups. Boxes show median and quartiles, with whiskers covering 1.5 times the interquartile range.

The guard ablation distinguishes the sources of gain. Averaged over the 72 runs, unguarded Ridge gives 16.409 replicas and 0.833% unmet demand; a single global 0.80 validation-residual guard gives 17.004 and 0.202%; the proposed two-bin volatility guard gives 17.000 and 0.204%. The two guarded variants have almost identical weighted objectives (20,110.08 and 20,110.01 units, respectively). Thus, a calibrated positive-residual margin explains most of the observed benefit, while splitting by local volatility adds little measurable advantage in these workload processes. A

global margin is a credible minimal-complexity option when volatility adaptation is unnecessary.

6. Discussion

6.1. Interpretation and Deployment Path

The main effect is not a large forecast-model advantage or a reliable advantage of volatility splitting; it is an allocation decision designed around positive error. A simple ridge forecast approximates short-term trends, while calibrated safety margins allocate enough headroom to limit transient undercoverage. In a cloud-native deployment, the same decision rule could consume CPU, request-rate, or queue-depth measurements through a monitoring adapter and publish desired replicas to an external scaling controller [1]–[5]. The prediction lead should match the deployment-specific time to obtain usable capacity, and the guard should be recalibrated when measurement semantics or utilization targets change.

A cost-efficient policy is not the policy with minimum replicas at all times. It trades marginal capacity for lower uncovered demand. The weight 20 used in the objective is an explicit modeling decision, so the cost ranking can change under another service valuation. The underlying results therefore report raw replica and shortfall metrics alongside J instead of presenting only a weighted score. For practical applications, a service-level objective must be linked to measured response-time and error-rate behavior rather than inferred from this capacity proxy [30]–[32].

6.2. Scope of Evidence

The experiment is a discrete-time, single-service simulation on generated loads. The seeds, generator, temporal split, training method, calibration rule, policy implementations, and result files are included for repetition. There is no live Kubernetes deployment or measured billing, and the capacity-to-quality relationship is a stated assumption. The analysis also omits network delays, fixed pod startup overhead beyond one interval, downscale stabilization, shared cluster constraints, and latency queues. These effects are important for a subsequent deployment study; the present numerical findings are specific to the defined replay system [42-50].

6.3. Conditions for Use

The model is best viewed as a resource-allocation component whose inputs, retraining cadence, and objective must be set to the target application. If usable capacity takes two or more collection intervals to arrive, the forecast horizon must be extended and the baselines re-run at the same delay. If services run on shared nodes, node packing and resource interference require an additional cluster-level scheduling layer. If replicas have non-additive throughput, the constant 0.72 conversion must be replaced with a measurement-derived performance curve. None of these engineering extensions alters the core separation of demand forecasting and positive-residual capacity protection [53-58].

A final practical consideration is the stability of the calibrated residual distribution. A shift in service release, traffic mix, or time-of-day structure can change the meaning of the two guard bins. Monitoring signed forecast errors in production would allow the reserve to be refreshed periodically while keeping the lightweight linear model. The paired-seed results strengthen repeatability within these three generator families; they do not estimate how often live recalibration would be needed or quantify out-of-distribution performance.

7. Conclusion

A compact workload-aware predictive scaling rule combines regularized linear forecasting with a conditional positive-residual guard. In the original three synthetic test scenarios, it reduces normalized unmet demand by 77.4% relative to a one-step reactive allocation model with a 3.6% increase in average replicas; against fixed 95th-percentile allocation, it uses 23.9% fewer replicas. A further 24-seed, three-scenario-per-seed study yields a 78.4% unmet-demand reduction at 3.6% greater mean replicas. An unconditional residual guard performs comparably, indicating that empirical one-sided forecast calibration is the principal useful component under these workload conditions. The result demonstrates a concrete, reproducible trade-off between capacity protection and replica-time cost, with code, generated data, and detailed metrics supplied as accompanying experimental artifacts.

References

[1] Ping, W., & Fan, H. (2023). Beyond isolated ratios: Accrual-revenue-cash flow indicators and interaction-aware learning for detecting financial reporting irregularities in U.S. public companies. World Journal of Economics and Business Research, 1(1), 46-55.

[2] Zhang, C., & Wang, Y. (2024). A data-driven multi-objective production scheduling framework for battery pack manufacturing under material arrival uncertainty. AI and Data Science Journal, 1(1), 86-93.

[3] Wang, J., Liu, J., Ge, Y., & Jiang, B. (2023). Risk-aware ensemble learning for credit default prediction under economic uncertainty. World Journal of Economics and Business Research, 1(1), 18-32.

[4] Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). HeteroGCL: A heterogeneous graph contrastive learning framework for scalable and sustainable cryptocurrency AML. Applied Sciences, 16(6), 2860.

[5] Ping, W., & Fan, H. (2024). Disclosure quality and valuation risk in level 3 investments: A disclosure-implied risk framework for U.S. business development companies. Social Science and Management, 1(1), 78-87.

[6] Qin, Y., Rhee, M., Teng, D., Zhang, C., & Zi, B. (2026). Neuro-symbolic constraint verification for LLM-driven internet finance transaction execution. Scientific Reports.

[7] Guo, Z., Chen, T., & Shang, W. (2022). Class-conditional intermediate-domain adversarial adaptation for cross-device acoustic scene classification. Journal of Computer Science and Electrical Engineering, 4(1), 26-35.

[8] Shen, A., & Huang, S. (2022). A systems-aware framework for intelligent large-scale recommendation models. Journal of Computer Science and Electrical Engineering, 4(1), 36-44.

[9] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value Shapley propagation. World Journal of Economics and Business Research, 1(1), 33-45.

[10] Shen, A., & Liu, W. (2026). Budget-aware adaptive test-time compute with hierarchical self-verification for semantic ID generative recommendation. Computers.

[11] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. Innovation and Technology Studies, 3(1), 27-41.

[12] Fang, Q. (2024). Bias-aware large language models for equitable educational feedback and assessment. Trends in Social Sciences and Humanities Research, 2(1), 41-50.

[13] Lin, Y., Wang, Y., & Bian, M. (2024). XOR-EW: An explainable and cost-aware artificial intelligence framework for operational risk early warning in financial institutions. AI and Data Science Journal, 1(1), 94-102.

[14] Li, P., Feng, E., Wang, G., & Mo, T. (2024). Hybrid deep reinforcement learning and integer optimization for resource scheduling in heterogeneous AI computing clusters. Innovation and Technology Studies, 1(1), 48-56.

[15] Shen, A. (2023). Compute-efficient scaling of large sequential recommendation models under dynamic user behavior. World Journal of Information Technology, 1(1), 62-71.

[16] Wang, G., Li, P., Feng, E., Shen, A., & Rhee, M. (2026). A dual-process memory-augmented large language model agent for explainable sequential recommendation. IEEE Access.

[17] Wang, B., Zhao, W., & Wang, Z. (2024). DAS-GNN: A scalable disagreement-aware graph neural framework for anomaly detection in large-scale networks. AI and Data Science Journal, 1(1), 67-75.

[18] Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

[19] Cao, J., Zheng, W., Ge, Y., & Wang, J. (2025). DriftShield: Autonomous fraud detection via actor-critic reinforcement learning with dynamic feature reweighting. IEEE Open Journal of the Computer Society, 6, 1166-1177.

[20] Wang, Y., Bian, M., & Lin, Y. (2022). An explainable temporal graph neural network for disruption prediction and risk propagation in multi-tier supply chains. Journal of Computer Science and Electrical Engineering, 4(1), 17-25.

[21] Mo, T., Wang, G., Feng, E., & Li, P. (2023). Execution-feedback-guided context selection for large language model-based software testing and debugging. World Journal of Information and Knowledge Management, 1(1), 55-64.

[22] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34-48.

[23] Shang, W., & Chen, T. (2024). PrefixKD-Edge: Real-time locomotion-intent prediction with prefix self-distillation on lightweight temporal networks. Innovation and Technology Studies, 1(1), 30-38.

[24] Zou, J., Qin, Y., & Rhee, M. (2024). Beyond model scale: A task-aware reliability benchmark for large language models in clinical decision support. Innovation and Technology Studies, 1(1), 39-47.

[25] Chen, T., Shang, W., & Guo, Z. (2024). XSLA: An SLA-aware heterogeneous XPU scheduling framework for edge AI inference. World Journal of Information Technology, 2(2), 48-59.

[26] Mai, N. T., Fang, Q., & Cao, W. (2025). Measuring student trust and over-reliance on AI tutors: Implications for STEM learning outcomes. International Journal of Social Sciences and English Literature, 9(12), 11-17.

[27] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early warning of liquidity stress in U.S. banks with seasonally aligned frailty hazards and delayed-feedback conformal alarms. Journal of Trends in Finance and Economics, 1(1), 85-100.

[28] Chen, J., Liang, Y., Liu, J., & Zhou, M. (2026). Temporal transformer with conditional tabular GAN for credit card fraud detection: A sequential deep learning approach. Mathematics, 14(7), 1183.

[29] Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

[30] Bian, M., Wang, Y., & Lin, Y. (2024). Explainable supplier risk prediction for critical telecommunications infrastructure procurement: An empirical XGBoost-SHAP framework. Social Science and Management, 1(1), 64-77.

[31] Mai, N. T., Cao, W., & Fang, Q. (2025). A study on how LLMs (e.g., GPT-4, chatbots) are being integrated to support tutoring, essay feedback and content generation. Journal of Computing and Electronic Information Management, 18(3), 43-52.

[32] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints. Mathematics, 14(12), 2137.

[33] Wang, Y., & Zhang, C. (2023). AERIS: Adaptive executor-scoped resource management for privacy-preserving data processing in Apache Spark. Eurasia Journal of Science and Technology, 5(2), 25-33.

[34] Zhang, F., Shen, A., & Huang, S. (2026). TRIAD: Co-designing mixed-precision quantization, operator fusion, and on-chip memory reuse for real-time vision-language-action inference on automotive NPUs. Results in Engineering, 113673.

[35] Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access, 12, 116546-116557.

[36] Zhang, F., & Yang, J. (2024). Robustness-aware model compression for multimodal perception under challenging conditions. AI and Data Science Journal, 1(1), 103-111.

[37] Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A multi-agent LLM framework with an evidence-based debate mechanism for financial statement fraud detection. Mathematics, 14(15), 2695.

[38] Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610.

[39] Fang, Q. (2024). TRUE-EWS: Trustworthy early warning with equalized conformal guarantees for equitable student support in U.S. education. Educational Research and Human Development, 1(2), 44-54.

[40] Jiao, Y., & Liang, Y. (2024). An integrated process-mining and explainable machine-learning framework for bottleneck and internal-control risk detection in the multi-entity financial close. Social Science and Management, 1(3), 96-107.

[41] Fan, H., Jiao, Y., Wang, M., Chen, J., & Yang, S. (2026). Fraud learns too: Continual graph learning under strategic adversarial drift in dynamic networks. Scientific Reports.

[42] Qin, Y., Zou, J., & Rhee, M. (2022). An AI-driven intelligent transaction execution framework for internet finance platforms: Cost-sensitive real-time integrity interdiction. Eurasia Journal of Science and Technology, 4(2), 36-44.

[43] Zhu, R., & Yue, X. (2024). An explainable machine learning framework for predicting dividend sustainability and financial risk of US REITs under interest-rate shocks. Journal of Trends in Finance and Economics, 1(2), 48-57.

[44] Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency

volatility prediction. IEEE Open Journal of the Computer Society, 7, 226-243.

[45] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access.

[46] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access.

[47] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63-72.

[48] Zhang, F., & Yang, J. (2022). Accuracy-aware model compression for real-time visual perception under resource constraints. Journal of Computer Science and Electrical Engineering, 4(1), 45-52.

[49] Liang, Y., & Jiao, Y. (2024). Real-time financial fraud detection using a feature-light temporal dynamic graph neural network with contrastive learning. Journal of Trends in Finance and Economics, 1(1), 76-84.

[50] Lin, Y., Yue, X., Bian, M., Zhu, R., & Teng, D. (2026). Subgraph-level temporal graph transformer with laundering-motif prototypes for anti-money laundering on blockchains. IEEE Access.

[51] Fang, Q., & Liu, W. (2025). HARLA-ED: Resolving information asymmetry and enhancing algorithmic symmetry in intelligent educational assessment via hybrid reinforcement learning. Symmetry, 18(1), 58.

[52] Zhang, H. (2024). Graph-aware multi-task learning for early prediction of timing violations and routing congestion in digital IC physical design. World Journal of Information Technology, 2(1), 13-20.

[53] Feng, E., Li, P., Mo, T., & Wang, G. (2022). PRISM: Prediction-guided risk-aware deep reinforcement learning for energy-aware dynamic resource scheduling in distributed cloud data centers. Journal of Computer Science and Electrical Engineering, 4(1), 53-61.

[54] Fan, H., Liang, Y., Jiao, Y., Ping, W., & Lin, Y. (2026). An explainable temporal machine learning framework for financial fraud risk prediction in listed institutions. Highlights in Business, Economics and Management, 70, 173-185.

[55] Rhee, M., Zou, J., & Qin, Y. (2024). CLARITY: A closed loop framework for duplicate aware failure prediction and budgeted automated remediation in cloud infrastructure. AI and Data Science Journal, 1(1), 76-85.

[56] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports.

[57] Teng, D. (2025). RSRBench: Reproducibility and performance benchmarking for regional seismic risk workflows. Academic Journal of Architecture and Civil Engineering, 3(1), 40-52.

[58] Guo, Z., Fang, Q., Yang, J. S., & Rhee, M. (2026). Closed-loop audio neuroregulation for ADHD: Graph-based attention decoding and rhythm entrainment. IEEE Access.