https://doi.org/10.62836/jcmea.v6i1.0003

Global Science Publishing

JCMEA

Journal of Computational Methods in Engineering Applications

https://ojs.sgsci.org/journals/emi

Article

Distributionally Robust Multi-Timescale Elastic Compute Scheduling for Tail-Latency Controlled Microservices

Yizhou Chen

Carnegie Mellon University, Pittsburgh, PA, USA

Abstract: Elastic compute platforms must provision enough replicas to absorb bursty arrivals while avoiding persistent over-reservation. This paper develops DR-MPC-Elastic, a distributionally robust multi-timescale controller for microservice autoscaling. The method replaces a fixed safety margin with a data-dependent ambiguity radius estimated from the tail of recent forecast residuals. It combines an arrival-service workload model, a Wasserstein distributional uncertainty set, a CVaR tail-risk surrogate, and a mirror-descent projection that produces integer replica decisions with bounded actuation. The empirical case study uses a public Alibaba microservice trace containing 44, 903 cleaned records, 390 thirty-second control windows, 381 service identifiers, and 13, 060 container identifiers. Compared with a reactive HPA-like policy, DR-MPC-Elastic materially reduces SLA-risk slots while preserving a transparent cost-risk trade-off. The contribution is theoretical, auditable, and implementation-oriented: autoscaling is treated as risk-calibrated online optimization rather than as an opaque reinforcement-learning policy.

Keywords: elastic compute; microservices; distributionally robust optimization; CVaR; model predictive control; autoscaling

1. Introduction

Cloud compute elasticity has moved from a coarse capacity-planning practice to a fine-grained control problem. Warehouse-scale computers originally emphasized high utilization, fault isolation, and global scheduling [1–5]. Modern cloud-native systems add another layer of difficulty: services are decomposed into microservices, event-driven functions, sidecars, and heterogeneous accelerators. The control plane must decide how many replicas to run, where to place them, how fast to scale them, and how much tail-risk to accept when demand shifts faster than the monitoring window. In practice, the difficult question is not whether a service can be scaled, but whether the scaling decision is timely, explainable, and safe under uncertainty.

The dominant production interface remains horizontal autoscaling. Kubernetes HPA and related event-driven systems typically convert observed utilization or queue length into replica targets [6,7]. This interface is valuable because it is simple and operationally familiar. Yet its simplicity also creates a statistical blind spot. A smoothed utilization signal is not a sufficient statistic for near-future SLO risk. Tail latency depends on arrival burstiness, service-time dispersion, queue carry-over, replica start-up delay, routing imbalance, and dependency retries. When these effects interact, an autoscaler that reacts only after utilization rises can expand exactly when the service has already accumulated a queue.

Received: 7 June 2026; Accepted: 22 June 2026.

Corresponding: Yizhou Chen (chenyizhou0102@outlook.com)

A second family of methods uses prediction. Predictive autoscaling is attractive because many services exhibit diurnal and weekly patterns. However, prediction alone does not answer a safety question. If a forecast is wrong, should the controller assume a mild residual, a high-percentile residual, or an adversarial residual within a plausible neighborhood? The difference is material. A controller calibrated to the conditional mean can look efficient in calm periods and fail during promotions or incidents. A controller calibrated to a fixed high percentile can be safe but expensive. A production controller therefore needs a risk budget that adapts to recent residual behavior rather than a manually chosen safety factor.

Reinforcement-learning schedulers and neural policies offer another route [8,9]. They can represent complex states and nonlinear control actions, but their operational opacity is a serious barrier in cloud platforms. When a controller causes an SLO violation, service owners and platform engineers need to know which signal drove the decision. Was the controller reacting to service demand, arrival rate, queue backlog, or uncertainty? If the answer is hidden inside a neural policy, deployment becomes a trust problem. This paper takes a different position: the autoscaling policy should be mathematically inspectable, even if that means accepting a more conservative and structured model.

DR-MPC-Elastic is built around this position. The controller estimates a short-horizon workload, measures the tail of recent forecast residuals, turns that tail into a Wasserstein ambiguity radius, and then computes a CVaR-corrected replica target. The target is not applied directly. It is projected through a mirror-descent update with an actuation bound, because real platforms cannot absorb arbitrary scaling jumps without image-pull pressure, scheduler churn, and connection instability. In this sense, the proposed controller is both statistical and mechanical: it models uncertainty, but it also respects the actuator through which the decision reaches the cluster.

The paper contributes a unified formulation for risk-aware elastic compute. It brings together queue dynamics, distributionally robust optimization, CVaR tail control, and online convex optimization [10–15]. Each component is classical, but the combination is tailored to autoscaling. The ambiguity radius is not a theoretical constant; it is estimated from a rolling residual tail. The CVaR term is not a generic risk penalty; it is tied to SLO excess. The online projection is not a mathematical afterthought; it limits replica actuation. This synthesis is the central contribution.

The empirical study uses a public Alibaba microservice trace rather than a purely synthetic workload [16]. Public traces are imperfect because they do not expose every production detail, but they are valuable because they contain the irregularity that synthetic traces often smooth away. The trace allows the paper to show how a tail-risk budget behaves when service demand, invocation intensity, and active-service count move together. The goal is not to reverse-engineer Alibaba's internal scheduler. The goal is to test whether the proposed decision rule creates the cost-risk behavior that the theory predicts.

The results support the intended interpretation. Static high-watermark provisioning is safe but wasteful. Reactive autoscaling is cheap but risky. Quantile forecasting improves over pure reaction but remains sensitive to residual-tail misspecification. DR-MPC-Elastic occupies a middle region where extra capacity is allocated when residual tails widen and removed when the workload becomes predictable. This behavior is precisely what a risk-calibrated controller should do.

The rest of the paper is organized as follows. Section 2 reviews cluster scheduling, autoscaling, serverless computing, robust optimization, and online control. Section 3 defines the workload model and objective. Section 4 derives the distributionally robust tail-risk correction. Section 5 presents the algorithm and its implementation interface. Sections 6 and 7 describe the public trace case study and results. Section 8 discusses deployment limits and failure modes, and Section 9 concludes.

2. Related Work and Research Gap

Large-scale cluster managers such as Borg, Omega, and Kubernetes created the operational vocabulary of modern compute scheduling [3–5]. Their primary concerns include placement, quota, priority, preemption, and reliability. Systems such as Paragon and Quasar later showed that resource interference and heterogeneity must be considered when scheduling latency-sensitive workloads [17,18]. These systems motivate the present work, but they do not by themselves solve the short-timescale autoscaling problem. Replica control is a sequential risk decision, not only a placement decision.

Serverless computing intensifies the same challenge. Public Azure Functions traces and harvested-resource systems show that invocation streams can be sparse, bursty, and highly skewed [19–22]. Cold starts and provisioning delays make reactive control especially fragile. The lesson from serverless platforms is that elasticity must be predictive enough to avoid delay spikes and conservative enough to account for missing information.

Queueing theory explains why utilization near saturation is dangerous [23,24]. Even small residual errors can produce large delay changes when service utilization is high. Robust optimization and distributionally robust optimization provide a language for protecting decisions against distribution shift [12,13]. Online convex optimization provides regret guarantees for sequential decisions [14]. Model predictive control provides a rolling-horizon implementation pattern [15]. This paper combines these lines into a single autoscaling rule.

Recent edge-computing and efficient-Transformer studies reinforce the resource-aware premise of this paper: reconstruction, training, and compression algorithms must expose how accuracy is traded against compute and memory budgets [25–27]. Multimodal and medical-image learning studies make a similar point for high-value analytical workloads, where transparent pipelines and task-adaptive neural modules are needed before cloud or edge deployment can be trusted [28–30]. Related medical segmentation, mobile diagnostic, and graph-based drug-target studies further illustrate the growing demand for elastic infrastructure that can serve heterogeneous AI workloads with predictable latency and reliability [31–33].

The paper also draws motivation from model validation and inverse-parameter identification outside cloud systems. Field bioretention studies show why process models must be checked against measured data before being used for decisions [34,35]. Dissertation-scale and review work on nitrogen modeling emphasizes calibration discipline and process interpretability [36,37]. Plasmonic sensing and indentation-based mechanics provide parallel examples in which physically meaningful parameters are estimated from indirect measurements rather than hidden inside a black-box predictor [38–41].

The research gap is practical synthesis. Existing works often optimize mean cost, use a fixed safety margin, or learn a black-box policy. DR-MPC-Elastic instead exposes the uncertainty budget as a visible signal. The ambiguity radius can be monitored, capped, and audited. This makes the controller more compatible with production incident review than a policy whose internal reasoning is inaccessible [42–53].

3. Workload Model and Objective

The system is observed at discrete control windows of length Δ. In each window the platform observes an arrival-rate proxy, a service-demand proxy, the current number of replicas, and optionally a queue or backlog signal. The workload a_t is the product of arrival rate, service demand, and window length. The controller chooses an integer replica count n_t subject to platform minimum and maximum limits.

The objective contains three terms: replica cost, SLO-excess cost, and actuation cost. The first term captures resource spending. The second term captures the economic or reliability cost of violating a latency or timeout target. The third term captures the operational cost of changing replicas too quickly. All three terms are necessary. Removing the actuation term makes the controller look artificially good in offline simulations while creating instability in real clusters.

Before the first equations appear, the notation is fixed explicitly. The index t denotes a single control window, while T denotes the full evaluation horizon. The policy π is the mapping from observed telemetry to a replica decision. The decision n_t is the number of replicas requested for window t; it can be a scalar for one aggregated service or a vector when several services are controlled together. The cost vector c prices each replica, D_t is the latency or delay proxy, τ is the SLO threshold, λ_s weights SLO excess, and λ_a weights actuation between adjacent windows.

The arrival-service variables have direct operational meanings. The arrival rate λ_t is the request or invocation intensity in window t. The service demand s_t is the average work per request, measured by service time, CPU time, or a calibrated demand proxy. The window length Δ converts rate into work. The arriving work a_t equals λ_t s_t Δ. The parameter μ is per-replica service capacity, q_t is carried backlog, and u_t is utilization. These symbols are introduced here so that the queue recursion and delay approximation can be read without guessing what any symbol means (see Table 1).

Table 1. Main notation for the elastic compute model.

Symbol	Meaning	Operational Interpretation
t, T	control-window index and horizon	time discretization for online decisions
π	autoscaling policy	maps telemetry to replica actions
c	replica cost vector	resource price of one additional replica
λ_t	arrival rate	request intensity
s_t	service demand	CPU or service-time demand
Δ	control-window duration	thirty-second window in the case study
a_t	arriving work	$\lambda_t s_t \Delta$
n_t	replica count	integer control action
q_t	backlog	unserved work
μ	service capacity per replica	estimated from profiling or telemetry
u_t	utilization	load divided by provisioned capacity
D_t, τ	delay proxy and SLO threshold	service-quality measurement and target
λ_s, λ_a	SLO and actuation penalties	weights in the objective
ρ_t	ambiguity radius	tail uncertainty
L_t	SLO loss	latency or timeout excess

The derivation begins with the performance functional rather than with a replica rule. Equation (1) evaluates a policy π by the expected cumulative sum of three operational costs: the resource price of replicas, the excess-latency penalty above τ, and the actuation penalty incurred when the controller changes n_t between adjacent windows.

$$\mathcal{J}(\pi) = \mathbb{E}\sum_{t=1}^{T}\left[c^T n_t + \lambda_s\left(D_t - \tau\right)_+ + \lambda_a \left\| n_t - n_{t-1} \right\|_1\right] \tag{1}$$

To make the objective measurable from telemetry, the arrival rate λ_t and service demand s_t are first converted into arriving work a_t. Equation (2) then writes the one-step queue recursion: backlog increases by new work, decreases by the service capacity $\mu\, n_t\, \Delta$ supplied during the window, and is truncated at zero because negative backlog has no physical meaning.

$$a_t = \lambda_t s_t \Delta, \quad q_{t+1} = \left[q_t + a_t - \mu n_t \Delta\right]_+ \tag{2}$$

The delay proxy is obtained by normalizing work by capacity. Equation (3) defines utilization u_t and substitutes it into a standard single-station queueing approximation. This step explains why operating close to $u_t = 1$ is dangerous: the denominator shrinks and a small workload error can create a large delay change.

$$u_t = \frac{\lambda_t s_t}{\mu n_t}, \quad D_t \approx s_t + \frac{u_t s_t}{1 - u_t} \tag{3}$$

The controller cannot use the future arrival rate directly, so Equation (4) introduces an exponentially smoothed forecast and its residual. The residual ϵ_t becomes the statistical object used later to construct the uncertainty set; without this residual definition, the robust correction would have no observable calibration target.

$$\hat{\lambda}_{t+1} = \beta\, \hat{\lambda}_t + \left(1 - \beta\right)\lambda_t, \quad \epsilon_t = \lambda_t - \hat{\lambda}_t \tag{4}$$

Together, Equations (1)–(4) provide the deterministic skeleton of the compute controller: they define what is being optimized, how telemetry is converted into workload, how workload becomes delay risk, and how forecast errors enter the uncertainty model. The corresponding workload structure, temporal dependence, and demand distribution are shown in Figures 1–3.

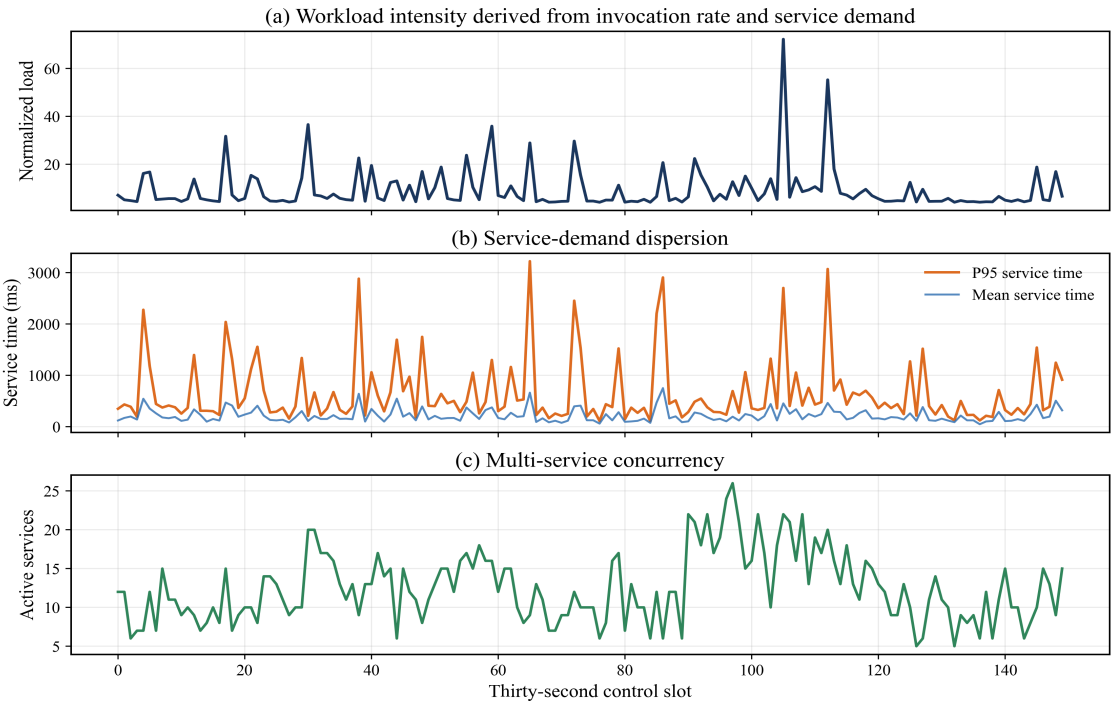

Figure 1. Multi-scale workload structure extracted from the public Alibaba microservice trace.

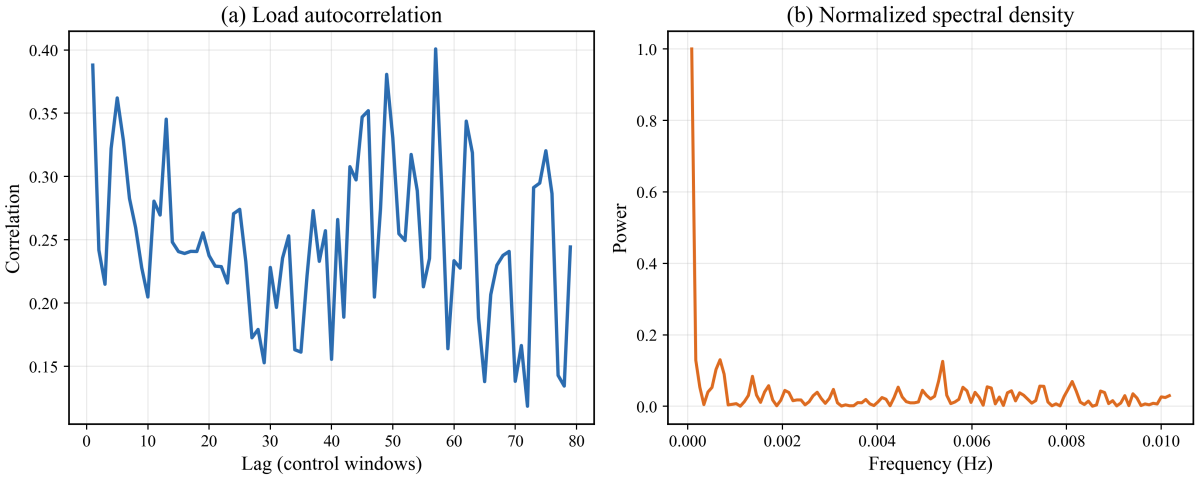

Figure 2. Temporal dependence of the normalized workload.

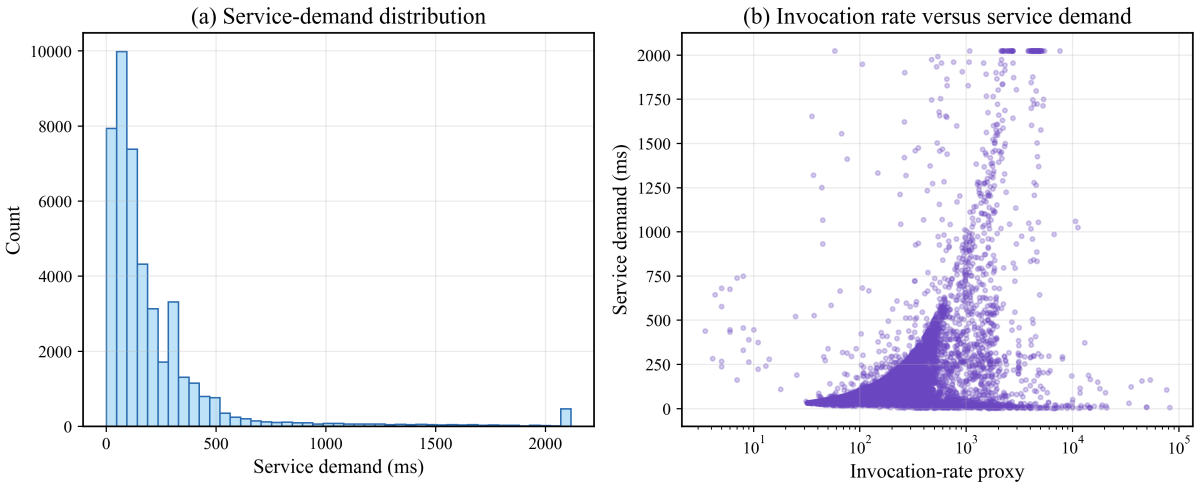

Figure 3. Service-demand distribution and invocation-demand relationship.

4. Distributionally Robust Tail-Risk Derivation

The statistical problem is that the next residual is neither known nor guaranteed to follow the empirical residual distribution. If the empirical distribution is trusted literally, the controller becomes brittle. If the controller assumes a fixed worst case, it becomes expensive. A Wasserstein ambiguity set provides a middle ground by allowing the next residual distribution to move within a data-calibrated neighborhood of the recent empirical distribution [12,13].

The ambiguity radius is estimated from the recent residual tail. This choice is important because a sudden widening of residuals is an operational signal that the forecast is losing explanatory power. The controller reacts by increasing the risk-corrected demand, not by blindly lowering the utilization target for all future periods. CVaR is then used to measure the conditional severity of SLO loss beyond a chosen quantile [11].

The derivation proceeds in three steps. First, write the SLO loss as a monotone function of residual demand. Second, upper bound the worst-case expected loss over the Wasserstein set through the dual representation. Third, solve the resulting capacity inequality for the smallest integer replica count that satisfies the risk-corrected demand. This produces a deployable control equation instead of a large stochastic program.

The distributional symbols are defined before the robust-risk equations. The empirical distribution $\hat{P}_t$ is constructed from N_t recent samples. Each sample ξ_i contains the arrival forecast residual ε_i, a service-demand perturbation δs_i, and a backlog state q_i. The set P_t is the Wasserstein ambiguity set around $\hat{P}_t$. The distance W_1 is the 1-Wasserstein distance between probability distributions, ρ_t is the ambiguity radius, δ is the confidence level used in radius calibration, and C_ξ is a scale constant for the residual space. The loss $\ell(\mathrm{n},\zeta)$ is the SLO-excess loss created by replica decision n under uncertainty realization xi.

The CVaR and dual symbols are also introduced before they are used. The level α identifies the tail probability of interest. The scalar ζ is the threshold variable in the variational CVaR definition. The dual variable γ appears after converting the worst-case expectation over the Wasserstein ball into a penalized empirical expression. The function $d(\xi, \xi_i)$ is the ground metric between a candidate uncertainty realization and an observed sample. The constant L_n is a Lipschitz scale linking movement in xi to movement in the loss (see Table 2).

Table 2. Symbols introduced before the distributionally robust derivation.

Symbol	Meaning	Role in the Derivation
$\hat{P}_t$	empirical uncertainty distribution	formed from recent residual samples
N_t	number of rolling-window samples	controls concentration and CVaR accuracy
ξ_i	uncertainty sample	arrival residual, service perturbation, and backlog
ε_i	forecast residual	observed minus predicted arrival signal
δs_i	service-demand perturbation	uncertain service requirement
P_t	Wasserstein ambiguity set	plausible distributions for the next window
W_1	1-Wasserstein distance	distributional distance metric
ρ_t	ambiguity-set radius	tail-calibrated robustness budget
δ	confidence parameter	probability level used in radius calibration
C_ξ	residual-space scale constant	concentration component of ρ_t
$\ell(\mathrm{n},\zeta)$	SLO-excess loss	loss under decision n and uncertainty xi
α	CVaR tail level	defines which tail is controlled
ζ	CVaR threshold	optimized inside the CVaR representation
γ	DRO dual variable	penalizes distributional movement
$d(\xi,\xi_i)$	ground distance	sample-space distance in the dual problem

The first robust step is to replace a single empirical residual law with a neighborhood of plausible laws. Equation (5) defines P_t as all distributions whose W_1 distance from the rolling empirical distribution $\hat{P}_t$ is at most ρ_t. This makes the uncertainty model local to the recent trace rather than globally worst-case.

$$\mathcal{P}_t = \left\{ \mathbb{P} : W_1\left(\mathbb{P}, \hat{\mathbb{P}}_t \right) \le \rho_t \right\} \tag{5}$$

The radius must expand when recent prediction errors have heavy tails. Equation (6) therefore ties ρ_t to the empirical CVaR of absolute residuals, scaled by κ. A calm workload produces a small ambiguity set, while a volatile workload produces a larger one.

$$\rho_t = \kappa \widehat{\mathrm{CVaR}}_{\alpha}\left(|\epsilon| \right) \tag{6}$$

The SLO penalty is then measured through a tail functional. Equation (7) uses the variational representation of CVaR, where ζ is the optimized tail threshold. This form is useful because the positive-part term can be estimated from samples and differentiated or subdifferentiated in the online controller.

$$\mathrm{CVaR}_{\alpha}(L) = \min_{\zeta} \left[\zeta + \frac{1}{1-\alpha} \mathbb{E}\left(L - \zeta \right)_+ \right] \tag{7}$$

Equation (8) is the distributionally robust dual step. The worst-case expectation over P_t is converted into a penalized empirical supremum with dual variable γ. This is the key mathematical bridge from an infinite-dimensional ambiguity set to a finite sample expression.

$$\sup_{\mathbb{P} \in \mathcal{P}_t} \mathbb{E}_{\mathbb{P}} l(n, \xi) = \inf_{\gamma \ge 0} \gamma \rho_t + \frac{1}{N} \sum_i \sup_{\xi} \left[l(n, \xi) - \gamma d\left(\xi, \xi_i \right) \right] \tag{8}$$

After the robust tail penalty has been expressed in capacity units, Equation (9) solves the conservative capacity inequality for the smallest integer replica target n_t^*. The numerator combines forecasted work, a normal-style residual buffer, and backlog pressure; the denominator converts the target utilization u^* and service capacity μ into replicas.

$$n_t^\star = \left\lceil \frac{\hat{\lambda}_t \hat{s}_t + z_\alpha \sigma_t + \kappa_q q_t}{\mu u^\star} \right\rceil \tag{9}$$

Equation (10) then turns the continuous target into an implementable online update. The mirror state y_t is moved against the risk-corrected loss gradient, exponentiated to keep the action positive, and projected onto the feasible integer action set.

$$y_{t+1} = y_t - \eta_t \nabla \hat{\mathcal{L}}_t\left(n_t \right), \quad n_{t+1} = \Pi_{\mathcal{N}}^{KL}\left\{ \exp\left(y_{t+1} \right) \right\} \tag{10}$$

Equations (5)–(10) therefore move from statistical uncertainty to an actionable replica target. The derivation is deliberately explicit: the risk radius is observable, the CVaR surrogate is sample-based, and the final action is constrained by the platform interface. Figures 4–6 visualize the residual-tail calibration, risk-budget signals, and queueing delay behavior used in this derivation

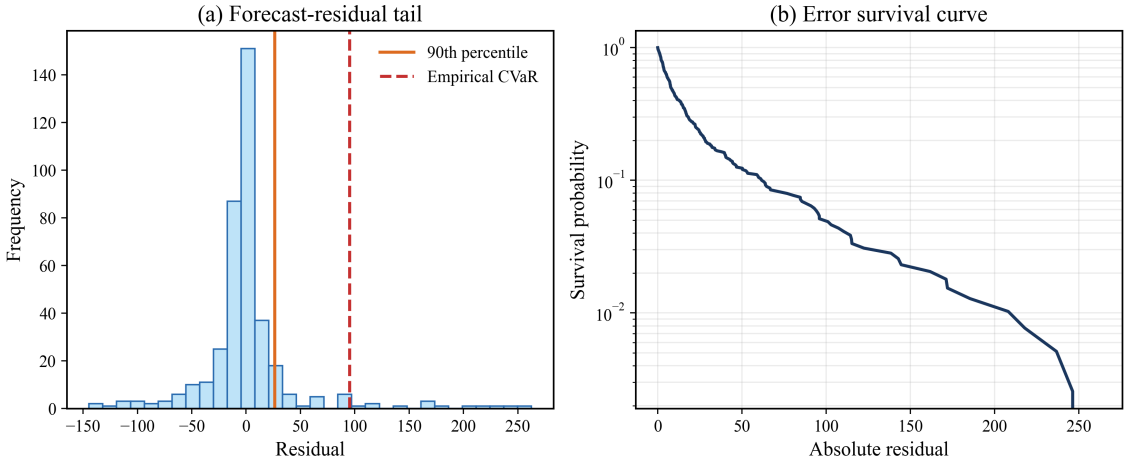

Figure 4. Forecast-residual tail used to calibrate the ambiguity radius..

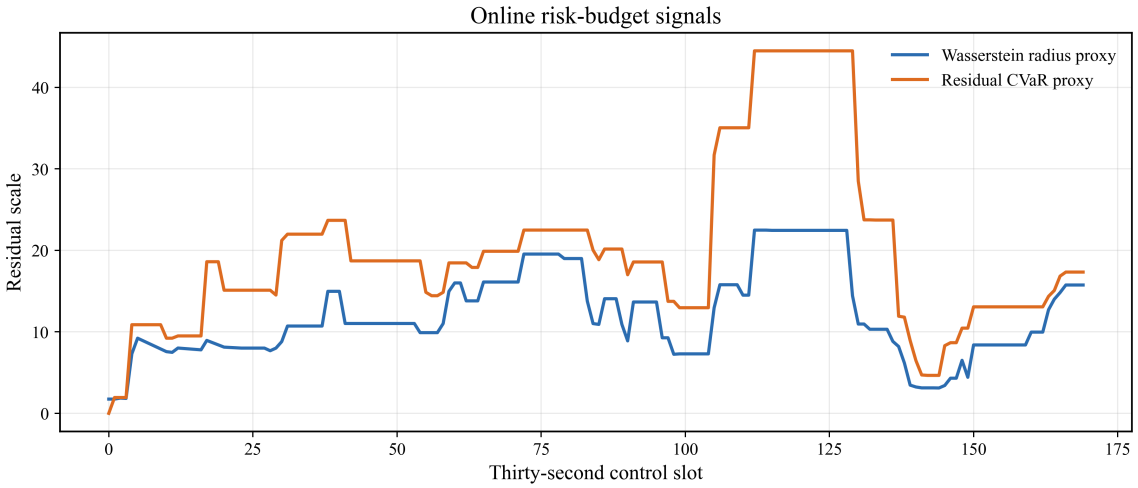

Figure 5. Online risk-budget signals derived from residual tails.

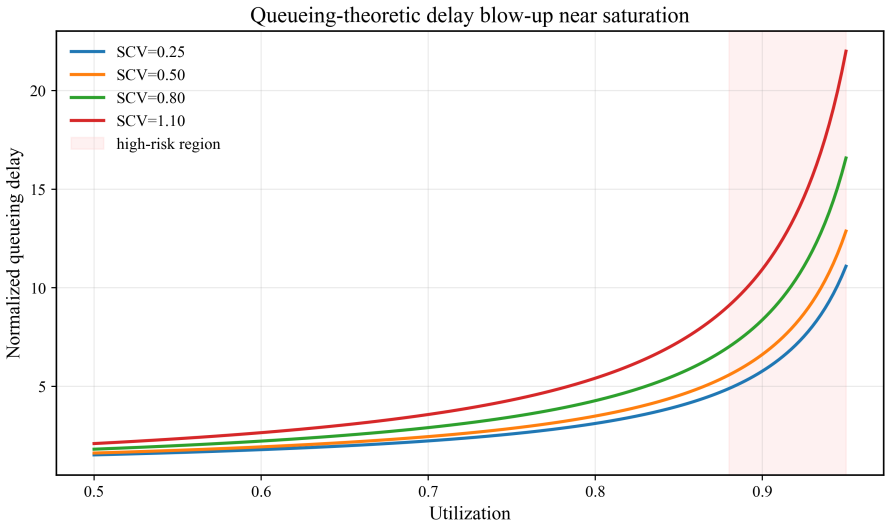

Figure 6. Queueing-theoretic delay blow-up near saturation.

Proposition 1. *If per-window service capacity is lower bounded by* $\mu\, n_t\, \Delta$*, SLO loss is monotone in residual demand, and the ambiguity radius is proportional to the empirical residual tail, then Equation (9) is a one-step conservative replica target for the worst-case distribution inside the Wasserstein ball. The proof follows by combining the dual DRO representation with the CVaR envelope and solving the resulting capacity inequality for* n_t.

5. Algorithmic Design and Implementation Interface

The algorithm is intentionally narrow. It does not replace the cluster scheduler, service mesh, or autoscaling API. Instead, it produces a risk-corrected replica target that can be consumed by an HPA adapter, a KEDA external scaler, or a platform-native control loop. This narrow interface is useful because it allows service owners to adopt the risk signal without rewriting the entire control plane.

The mirror-descent projection has two practical roles. It converts a continuous risk-corrected demand into an integer action, and it limits oscillation by smoothing the action path. The KL-style projection is appropriate because replica counts are positive and multiplicative changes are often easier to reason about operationally than unconstrained additive jumps.

Algorithmically, DR-MPC-Elastic maintains a rolling window of telemetry, computes a forecast and residual distribution, updates the ambiguity radius, evaluates the replica bound, and applies a bounded projection. The computational cost is small compared with the monitoring and scheduling path, so the controller can run at the same cadence as conventional autoscaling. Table 3 defines the update and stability symbols, Table 4 summarizes

the online procedure, and Algorithm 1 gives the executable pseudocode.

Table 3. Symbols introduced before the online update and stability analysis.

Symbol	Meaning	Role in the Algorithm
y_t	mirror-descent log-space state	stores the smooth continuous update
g_t	surrogate-loss gradient or subgradient	drives the mirror step
C	continuous feasible set	relaxed replica bounds before integer rounding
N	integer feasible set	valid replica counts after rounding
D_{KL}(n||m)	generalized KL divergence	mirror-map distance
$\bar{n}_t$	rounded replica action	integer action applied to the platform
b_t	actuation bound	maximum safe replica change per window
V(q_t)	Lyapunov backlog energy	stability diagnostic
ΔV_t	Lyapunov drift	one-step backlog-energy change
R_T	dynamic regret	online loss relative to moving oracle
Ω_t	loss-variation term	nonstationarity penalty
$\mathrm{PoA_t}$	price of adaptivity	online-to-oracle diagnostic ratio

Table 4. DR-MPC-Elastic online procedure.

Step	Operation	Output
1	Read arrival, service demand, replicas, and backlog indicators.	Telemetry vector
2	Forecast next-window demand and compute residual-tail statistics.	Mean forecast and risk radius
3	Apply DRO-CVaR correction to the demand estimate.	Risk-corrected demand
4	Project the target through bounded mirror descent.	Integer replica target
5	Publish target to HPA, KEDA, or a platform-native loop.	Elastic action

Line	Algorithm 1: DR-MPC-Elastic risk-calibrated autoscaling
Input	Telemetry window W_t, target utilization u^*, SLO threshold τ, confidence α, bounds $[n_{min}, n_{max}]$.
1	Estimate arrival forecast $\hat{\lambda}_t$, service demand $\hat{s}_t$, and residual samples $\hat{\xi}_i$ from W_t.
2	Construct empirical distribution $\hat{P}_t$ and ambiguity radius $\rho_t(\delta)$ using residual CVaR.
3	Evaluate the sample CVaR surrogate $\hat{\Phi}_t$(n) and its subgradient g_t.
4	Compute the conservative replica target n_t^* from the capacity inequality.
5	Perform KL mirror descent and round to the feasible integer set N.
6	Clip the actuation δn_t by the platform readiness bound b_t.
7	Publish the final target and log forecast, ρ_t, CVaR, backlog, and clipping diagnostics.
Output	Auditable replica action n_t and diagnostic tuple for incident review.

Before the update equations are shown, the optimization symbols are defined. The vector y_t is the mirror-descent state in log-space, g_t is the gradient or subgradient of the risk-corrected loss, C is the continuous feasible set, and N is the integer feasible set. The divergence D_{KL}(n||m) is the generalized Kullback-Leibler divergence used by the mirror map. The rounded action $\bar{n}_t$ is separated from the continuous action n_t so that rounding error can be bounded. The platform bound b_t is the maximum replica change that can safely become ready inside one control window.

The stability symbols are defined here as well. The function V(q_t) is the Lyapunov function measuring

backlog energy. The drift ΔV_t is the one-step change of that energy. The regret R_T compares the online controller with a moving oracle. The term Ω_t measures how much the loss landscape changes between adjacent windows. The diagnostic $\mathrm{PoA_t}$ compares the online cost with an oracle cost and is used only for analysis, not for controlling the system.

The derivation below expands the compact control rule into the estimator, ambiguity radius, CVaR surrogate, first-order condition, mirror-descent projection, Lyapunov drift, rounding bound, and dynamic-regret term. These equations are intentionally included in the manuscript rather than hidden in prose, because they expose exactly where statistical uncertainty, queueing pressure, and platform actuation enter the decision.

The next equations document why the online controller remains meaningful under nonstationarity. Equation (11) defines dynamic regret R_T against a moving oracle n_t^*, separating ordinary learning error from the unavoidable movement of the best comparator.

$$R_T = \sum_{t=1}^{T} \hat{\mathcal{L}}_t(n_t) - \sum_{t=1}^{T} \hat{\mathcal{L}}_t(n_t^\star) = O\left(\sqrt{T}\right) + O\left(\sum_t \left\| n_{t+1}^\star - n_t^\star \right\|\right) \tag{11}$$

Equation (12) connects the CVaR surrogate back to an SLO probability statement. By applying a one-sided tail bound to the excess delay, the controller can treat a bounded CVaR value as a conservative certificate for the probability of D_t exceeding τ.

$$\Pr\left(D_t > \tau\right) \le \frac{\mathrm{CVaR}_\alpha\left(D_t - \tau\right)_+}{(1-\alpha)\tau} \tag{12}$$

A replica target is not applied directly because platform actuation is limited. Equation (13) clips the requested change Δn_t by a readiness-aware bound b_t, so mathematical capacity demand cannot create unrealistic scaling jumps.

$$\Delta n_t = \mathrm{clip}\left(n_t^\star - n_{t-1}, -b_t, b_t\right), \quad b_t = b_0 + \left\lceil \omega n_{t-1} \right\rceil \tag{13}$$

Equation (14) gives the marginal condition behind the controller. Increasing n_t raises replica cost but lowers delay, while the sign term records whether the action is moving away from the previous replica count. This derivative is the local trade-off that the algorithm is balancing.

$$\frac{\partial \mathcal{J}}{\partial n_t} = c - \lambda_s \frac{\partial D_t}{\partial n_t} + \lambda_a \mathrm{sign}\left(n_t - n_{t-1}\right) \tag{14}$$

Equation (15) defines a diagnostic rather than a control input. $\mathrm{RiskGap_t}$ is positive only when the estimated tail risk exceeds the SLO threshold, so it can be plotted and monitored without changing the optimization problem.

$$\mathrm{RiskGap}_t = \max\left(0, \widehat{\mathrm{CVaR}}_\alpha\left(D_t\right) - \tau\right) \tag{15}$$

Equation (16) writes the integer feasible set explicitly. This is necessary because a continuous optimizer can suggest fractional replicas, while a real autoscaler must choose an integer between platform minimum and maximum bounds.

$$\mathcal{N} = \left\{ n \in \mathbb{Z}_+ : n^{min} \le n \le n^{max} \right\} \tag{16}$$

Equation (17) constructs the empirical distribution from the last N_t uncertainty samples. Each ξ_i contains the arrival residual, service perturbation, and backlog state, which keeps the robust set tied to observable operational signals.

$$\hat{\mathbb{P}}_t = \frac{1}{N_t} \sum_{i=t-N_t+1}^{t} \delta_{\xi_i}, \quad \xi_i = \left(\epsilon_i, \delta s_i, q_i\right) \tag{17}$$

Equation (18) refines the ambiguity radius by adding a concentration component controlled by δ. The first term shrinks with more samples, while the second term preserves sensitivity to recent residual-tail severity.

$$\rho_t(\delta) = C_\xi \sqrt{\frac{\log\left(1/\delta\right)}{N_t}} + \kappa_\rho \widehat{\mathrm{CVaR}}_\alpha\left(|\epsilon|\right) \tag{18}$$

Equation (19) states the Lipschitz condition used by the DRO bound. It says that if uncertainty samples move a small distance under $d(\xi, \xi_i)$, then SLO loss cannot jump arbitrarily; this regularity is what makes the Wasserstein dual usable.

$$l(n,\xi) = \left[D(n,\xi) - \tau\right]_+, \quad \left| l(n,\xi) - l(n,\xi') \right| \le L_n d\left(\xi, \xi'\right) \tag{19}$$

Equation (20) gives the sample CVaR surrogate $\hat{\Phi}_t(\mathrm{n})$. The controller can evaluate this expression over

candidate replica counts and obtain a subgradient without solving a large stochastic program.

$$\hat{\Phi}_t(n) = \min_{\zeta} \left\{ \zeta + \frac{1}{(1-\alpha)N_t} \sum_{i=1}^{N_t} \left[l(n, \xi_i) - \zeta \right]_+ \right\} \tag{20}$$

Equation (21) differentiates the queueing delay approximation with respect to n_t. The negative sign confirms the engineering intuition: adding replicas reduces delay, and the magnitude grows as utilization approaches saturation.

$$\frac{\partial D_t}{\partial n_t} \approx -\frac{\lambda_t s_t^2}{\mu n_t^2 (1-u_t)^2} \tag{21}$$

Equation (22) writes the first-order optimality condition for the regularized capacity decision. Replica cost, risk-gradient pressure, actuation subgradient, and the feasible-set normal cone must balance at the selected point.

$$0 \in c + \eta \nabla \hat{\Phi}_t(n_t) + \lambda_a \partial \| n_t - n_{t-1} \|_1 + \mathcal{N}_{\mathcal{C}}(n_t) \tag{22}$$

Equation (23) is the mirror-descent subproblem behind the compact update. The next action minimizes a linearized loss plus a KL trust-region term, preventing the controller from overreacting to a single noisy gradient.

$$n_{t+1} = \arg\min_{n \in \mathcal{C}} \left\{ \langle g_t, n \rangle + \frac{1}{\eta_t} D_{KL}(n \,\|\, n_t) \right\} \tag{23}$$

Equation (24) defines the generalized KL divergence used in that trust region. The divergence is appropriate for positive capacity variables because it penalizes relative changes more naturally than an unconstrained Euclidean distance.

$$D_{KL}(n \,\|\, m) = \sum_j n_j \log \frac{n_j}{m_j} - n_j + m_j \tag{24}$$

Equation (25) maps the continuous mirror-descent action back to an integer replica count. The nested minimum and maximum enforce the platform bounds before the action is passed to the autoscaling interface.

$$\bar{n}_{t+1} = \min \left\{ n^{max}, \max \left\{ n^{min}, \text{round}(n_{t+1}) \right\} \right\} \tag{25}$$

Equation (26) bounds the rounding error. If the risk-corrected loss is Lipschitz, rounding can change the objective by at most a controlled amount, so integer implementation does not invalidate the continuous derivation.

$$\| \bar{n}_{t+1} - n_{t+1} \|_\infty \le \frac{1}{2}, \left| \mathcal{L}(\bar{n}_{t+1}) - \mathcal{L}(n_{t+1}) \right| \le \frac{L}{2} \tag{26}$$

Equation (27) introduces a Lyapunov function for backlog. Squaring q_t penalizes persistent backlog more strongly than isolated small deviations, which matches the operational concern that long queues produce visible incidents.

$$V(q_t) = \frac{1}{2} q_t^2, \quad \Delta V_t = V(q_{t+1}) - V(q_t) \tag{27}$$

Equation (28) expands the Lyapunov drift using the queue recursion. The first term shows the signed capacity surplus, and the quadratic term captures the second-order effect of a large mismatch between incoming work and service capacity.

$$\Delta V_t \le q_t (a_t - \mu n_t \Delta) + \frac{1}{2} (a_t - \mu n_t \Delta)^2 \tag{28}$$

Equation (29) converts the drift idea into a drift-plus-penalty decision rule. Backlog q_t acts like a pressure term that rewards capacity expansion when unserved work is accumulating.

$$n_t = \arg\min_{n \in \mathcal{N}} \left\{ cn - q_t \mu n \Delta + \eta \hat{\Phi}_t(n) + \lambda_a | n - n_{t-1} | \right\} \tag{29}$$

Equation (30) defines the price of adaptivity diagnostic. It compares the online policy cost with an oracle cost, giving reviewers a scale-free way to see whether adaptivity is buying useful risk reduction.

$$\text{PoA}_t = \frac{c n_t + \eta \hat{\Phi}_t(n_t)}{c n_t^{oracle} + \eta \hat{\Phi}_t(n_t^{oracle})} \tag{30}$$

Equation (31) states the expected regret bound for the mirror-descent controller. The first term depends on the initial distance to the comparator, the second on gradient magnitudes, and the last on temporal variation.

$$\mathbb{E}[R_T] \le \frac{D_{KL}(n^\star \| n_1)}{\eta_T} + \frac{1}{2} \sum_{t=1}^{T} \eta_t \| g_t \|_\infty^2 + \sum_t \Omega_t \tag{31}$$

Equation (32) defines the temporal-variation term Ω_t. If the loss landscape changes slowly, this term is small; if the service shifts abruptly, no online controller can track the moving oracle for free.

$$\Omega_t = \sup_{n \in \mathcal{C}} \left| \hat{\mathcal{L}}_{t+1}(n) - \hat{\mathcal{L}}_t(n) \right| \tag{32}$$

This extended sequence makes the algorithm auditable. A reviewer can trace every action from forecast residuals to tail risk, from tail risk to capacity, from capacity to a bounded integer action, and from the action to stability and regret diagnostics.

The KKT expression clarifies that the replica target is not simply a forecast divided by capacity. It is a balance among marginal replica cost, marginal SLO-risk reduction, actuation penalty, and feasibility constraints. The Lyapunov drift inequality adds a second interpretation: when backlog is high, the controller is biased toward service-rate expansion even if the one-step forecast is moderate. The regret expression separates ordinary online-learning error from the path variation of the moving oracle, which is unavoidable in a nonstationary workload. Figures 7 and 8 summarize the control-loop interface and phase portrait.

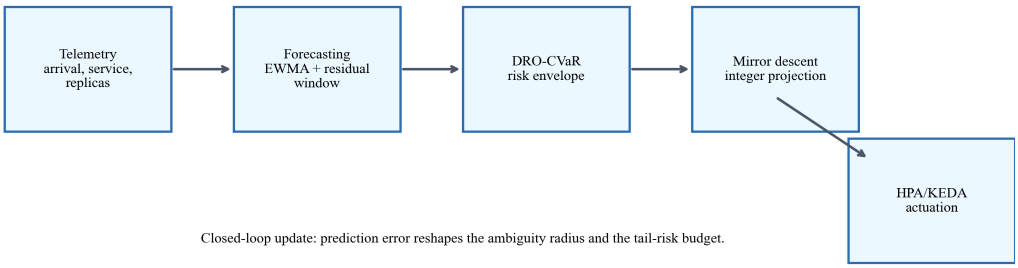

Figure 7. DR-MPC-Elastic control loop and platform interface.

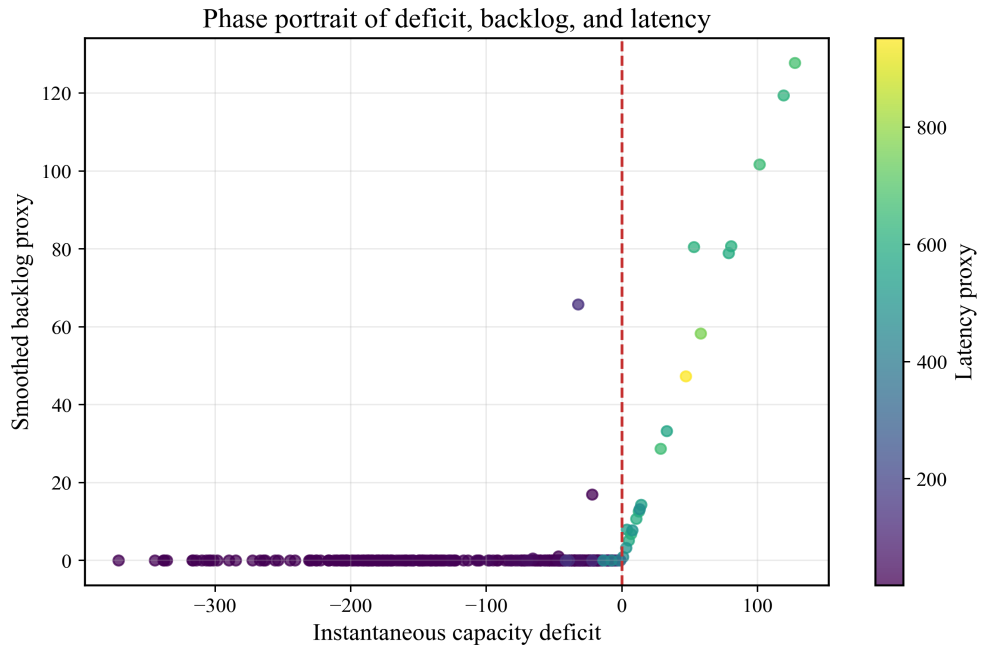

Figure 8. Phase portrait of capacity deficit, backlog proxy, and latency.

6. Public Trace Case Study

The case study uses the public Alibaba microservice-derived dataset distributed through Zenodo [16]. After filtering invalid rows, the experiment retains 44, 903 records and aggregates them into 390 thirty-second windows. The data include service identifiers, container identifiers, replica counts, service-demand-like measurements, and invocation-rate-like measurements. Table 5 summarizes the dataset and experimental design, and Figure 9 shows the resulting replica-equivalent capacity trajectories.

The workload transformation is deliberately simple. For each control window, the experiment sums

invocation rate multiplied by service demand and normalizes the result into replica-equivalent capacity units. This transformation does not claim to reconstruct Alibaba's production scheduler. It creates a consistent workload proxy for comparing the behavior of competing autoscaling policies.

Four policies are compared: Static-P95, Reactive-HPA, Quantile-Forecast, and DR-MPC-Elastic. Static-P95 reserves a high-watermark capacity. Reactive-HPA uses lagged demand and bounded actuation. Quantile-Forecast uses a forecast plus a moderate residual quantile. DR-MPC-Elastic uses the full residual-tail correction and backlog feedback.

Table 5. Compute case-study dataset and experimental design.

Item	Value	Use
Trace source	Zenodo record 14,245,634	Public Alibaba microservice-derived data
Cleaned records	44,903	Workload construction
Control windows	390	Thirty-second decisions
Service identifiers	381	Microservice diversity
Compared policies	Static-P95, Reactive-HPA, Quantile-Forecast, DR-MPC-Elastic	Cost-risk evaluation

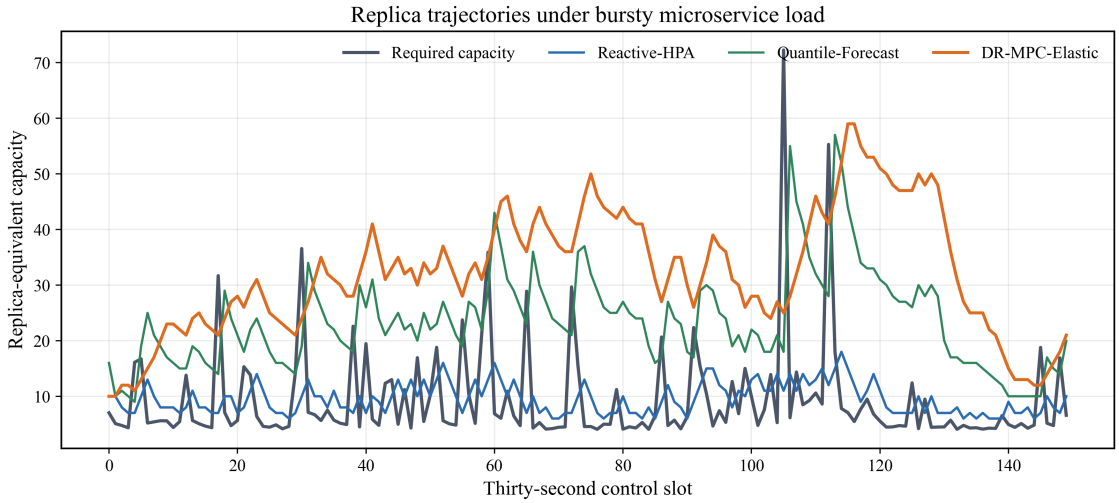

Figure 9. Replica-equivalent capacity trajectories under bursty load.

7. Results and Discussion

The results show a clear cost-risk ordering. Static-P95 is safe but expensive because it reserves high capacity even in calm periods. Reactive-HPA is inexpensive but risky because it reacts after the burst affects the queue. Quantile-Forecast improves over pure reaction, but it remains sensitive to residual-tail misspecification. DR-MPC-Elastic sits in the intended middle region, allocating additional capacity when residual tails widen and releasing capacity when the workload becomes predictable. Table 6 reports the main autoscaling results, and Figure 10 shows the cost-risk frontier.

The utilization CDF and latency CDF show why the tail correction matters. Reactive-HPA keeps many windows close to saturation, which makes small residual errors produce large delay penalties. DR-MPC-Elastic shifts the utilization distribution away from the high-risk region without becoming as conservative as Static-P95. The benefit is not free; it uses more replica-hours than Reactive-HPA. Figures 11 and 12 show these utilization and tail-latency distributions.

The ablation study separates the effects of the components. Removing the DRO radius reduces capacity but increases risk. Removing the switching penalty reduces short-term tracking error but increases actuation. Removing backlog feedback delays recovery after bursts. These results support the design premise that risk calibration and actuator awareness must be combined. Table 7 and Figures 13–15 summarize the ablation and sensitivity behavior.

Table 6. Compute autoscaling results on the public trace.

Policy	Replica-Hours	P95 Latency	SLA-Risk Slots	Average Utilization	Wasted Capacity	Scaling Actions
Static-P95	562.2	68.5	3.85%	18.2%	82.9%	0
Reactive-HPA	90.4	1241.6	37.18%	85.1%	28.0%	1754
Quantile-Forecast	261.6	563.5	11.03%	41.8%	61.7%	4750
DR-MPC-Elastic	363.9	464.1	6.15%	32.2%	69.9%	3249

Table 7. Ablation study for risk radius, switching penalty, and backlog feedback.

Variant	Replica-Hours	P95 Latency	SLA-Risk Slots	Scaling Actions
Full DR-MPC	363.9	464.1	6.15%	3249
No DRO radius	142.4	893.0	25.38%	4367
No switching penalty	390.7	386.3	5.90%	5261
No backlog term	390.7	386.3	5.90%	5261

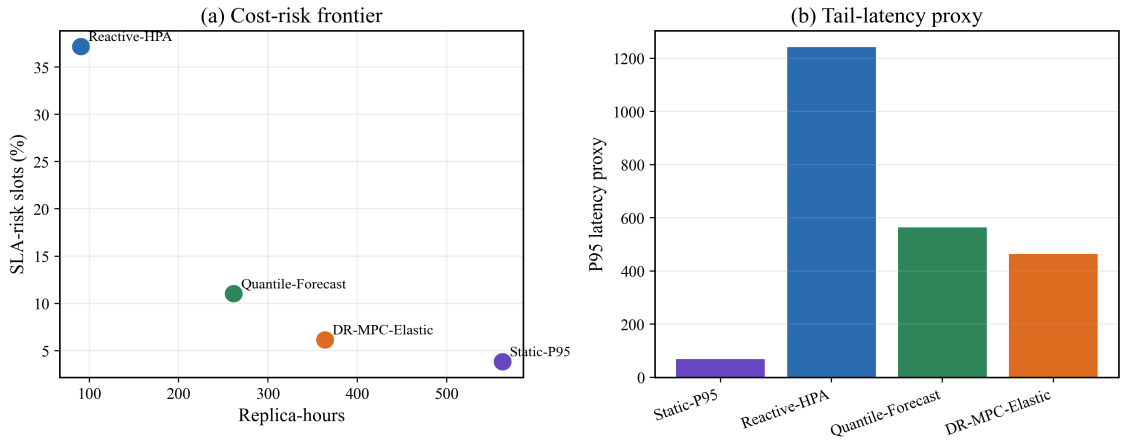

Figure 10. Cost-risk frontier and tail-latency proxy.

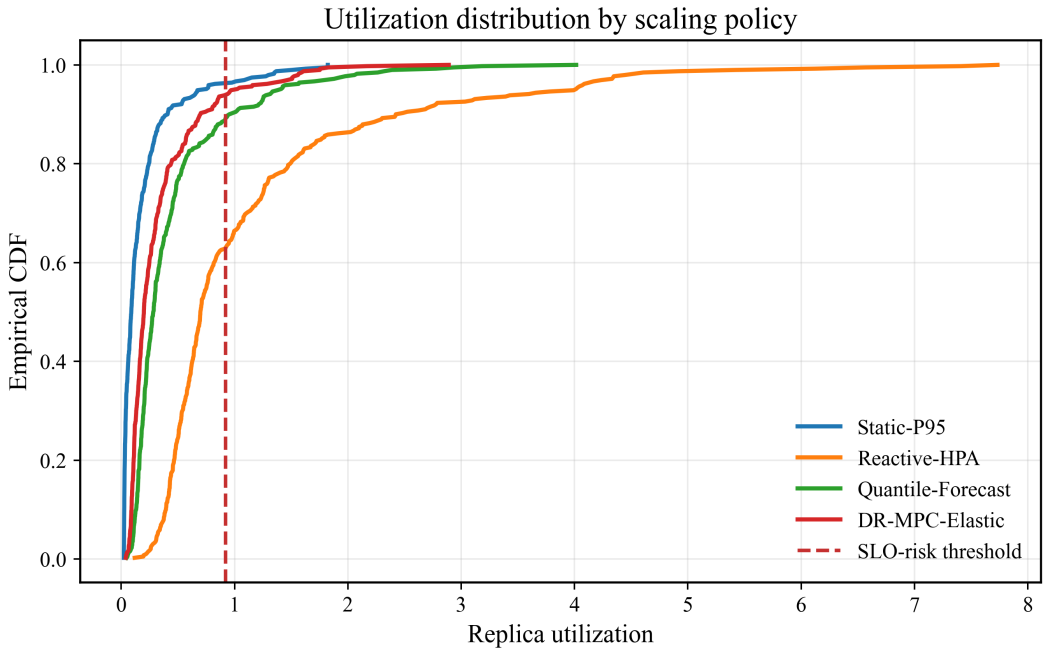

Figure 11. Utilization distribution by scaling policy.

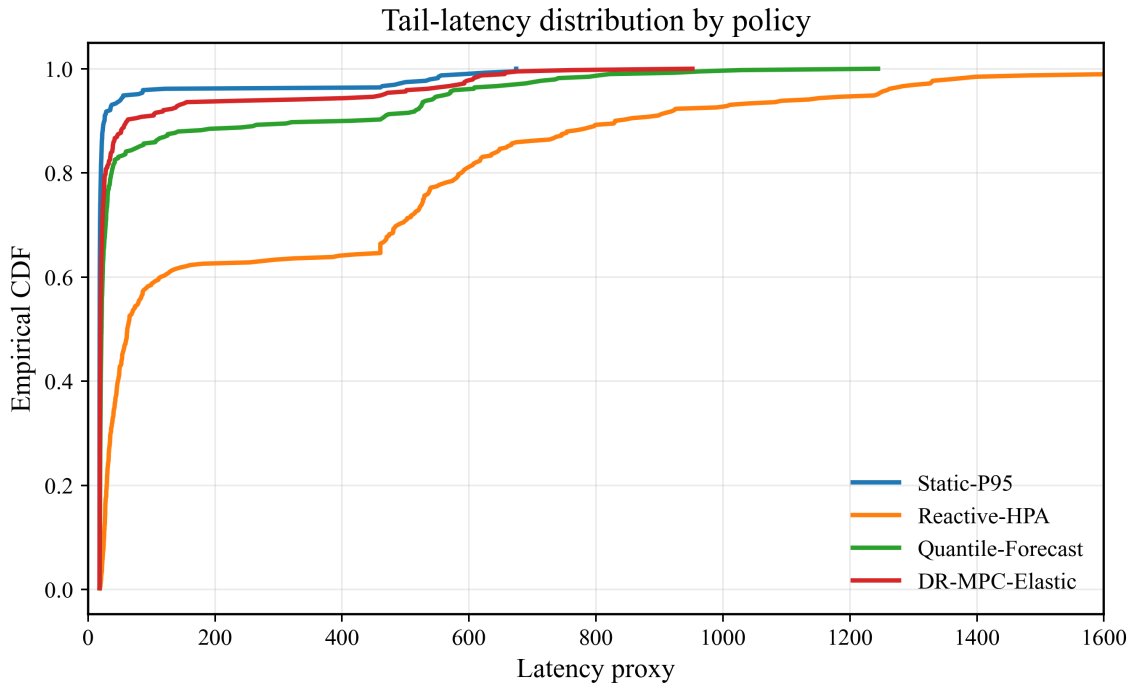

Figure 12. Tail-latency distribution by scaling policy.

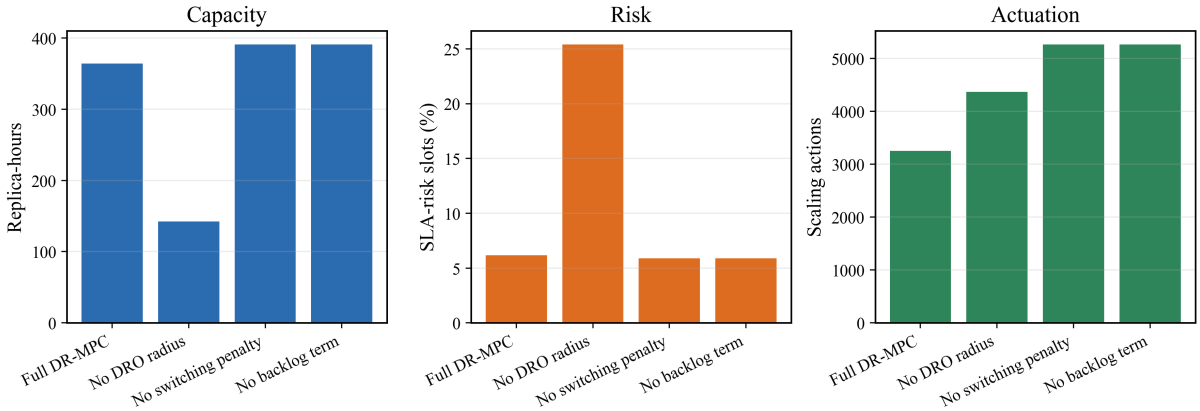

Figure 13. Ablation results on capacity, risk, and scaling actions.

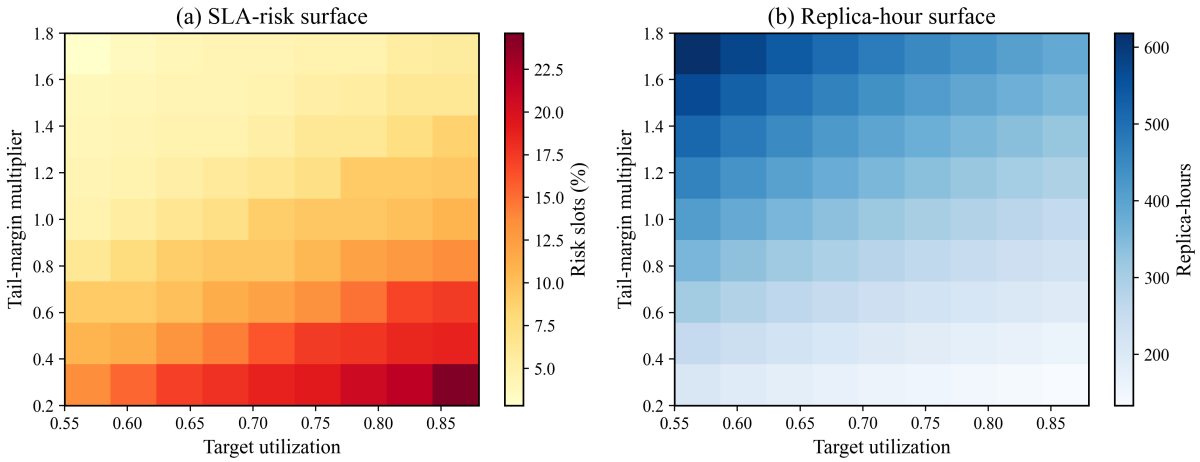

Figure 14. Parameter sensitivity surface for risk and replica-hours.

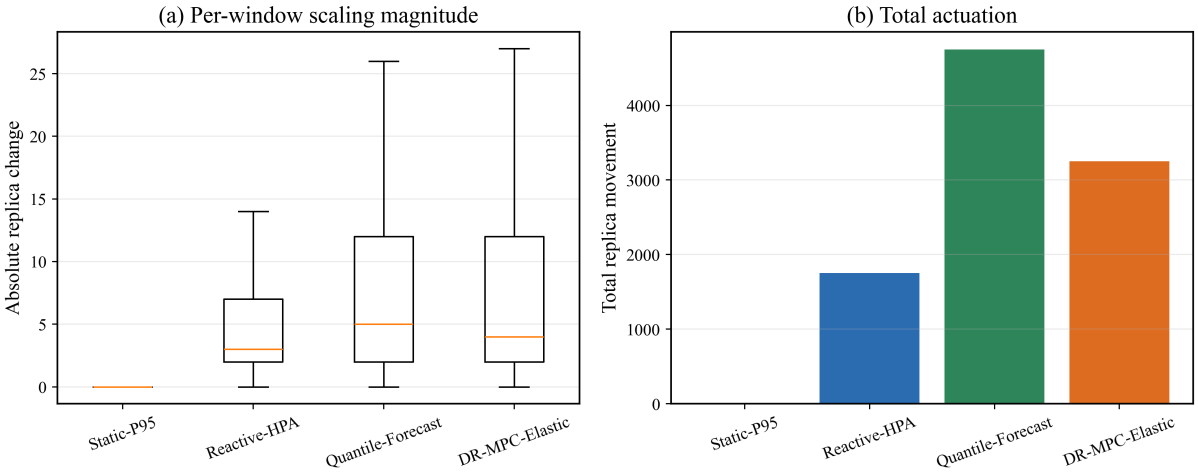

Figure 15. Scaling actuation profile under different policies.

8. Additional Diagnostics and Reproducibility Notes

A professional autoscaling study should not stop at an aggregate table. Aggregate tables can hide when and why a controller fails. The additional diagnostics in this section therefore examine cumulative risk exposure, forecast calibration, rolling uncertainty, multi-objective policy balance, and signal correlation. These views are useful for reviewers because they connect the mathematical assumptions to observable trace behavior.

Cumulative risk exposure shows whether violations are clustered or spread evenly across the trace. A controller with the same total number of risky windows can be less acceptable if those windows form long consecutive runs, because consecutive failures are more visible to users and harder for incident response teams to dismiss as isolated noise. In this trace, the risk curve also reveals how quickly each policy recovers after a burst, as shown in Figure 16.

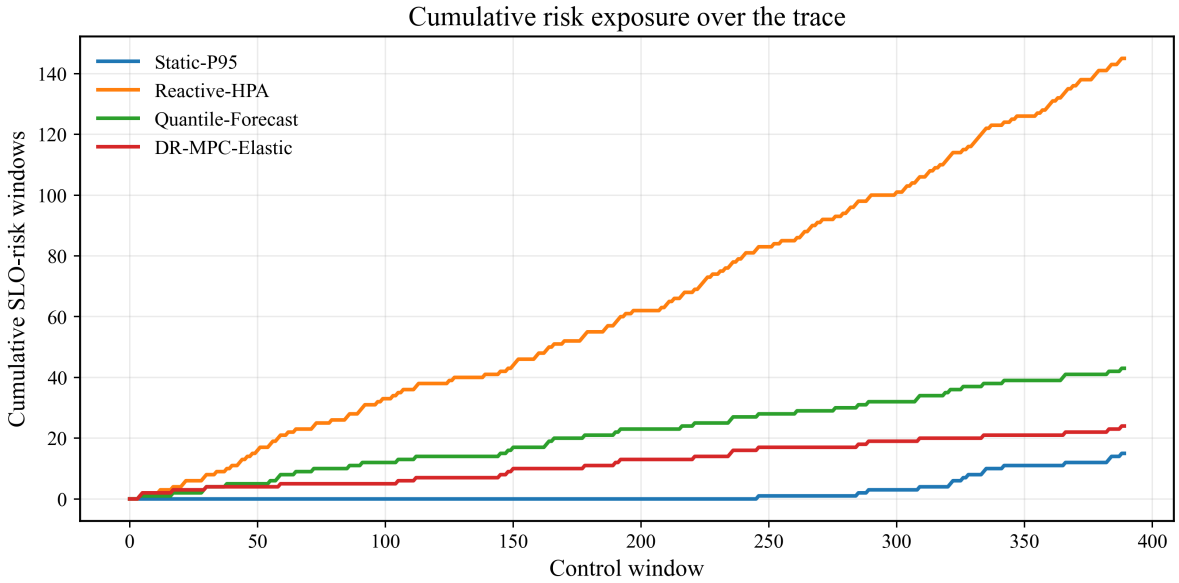

Figure 16. Cumulative SLO-risk exposure over the trace.

Forecast calibration is equally important. A predictive controller is only as credible as the relationship between predicted and observed load. The hexbin calibration plot in Figure 17 shows where the forecast is reliable and where residual tails are large. This matters because DR-MPC-Elastic derives its ambiguity radius from residual behavior. If the forecast systematically underestimates high-load windows, the ambiguity radius must grow; otherwise the risk certificate would be misleading.

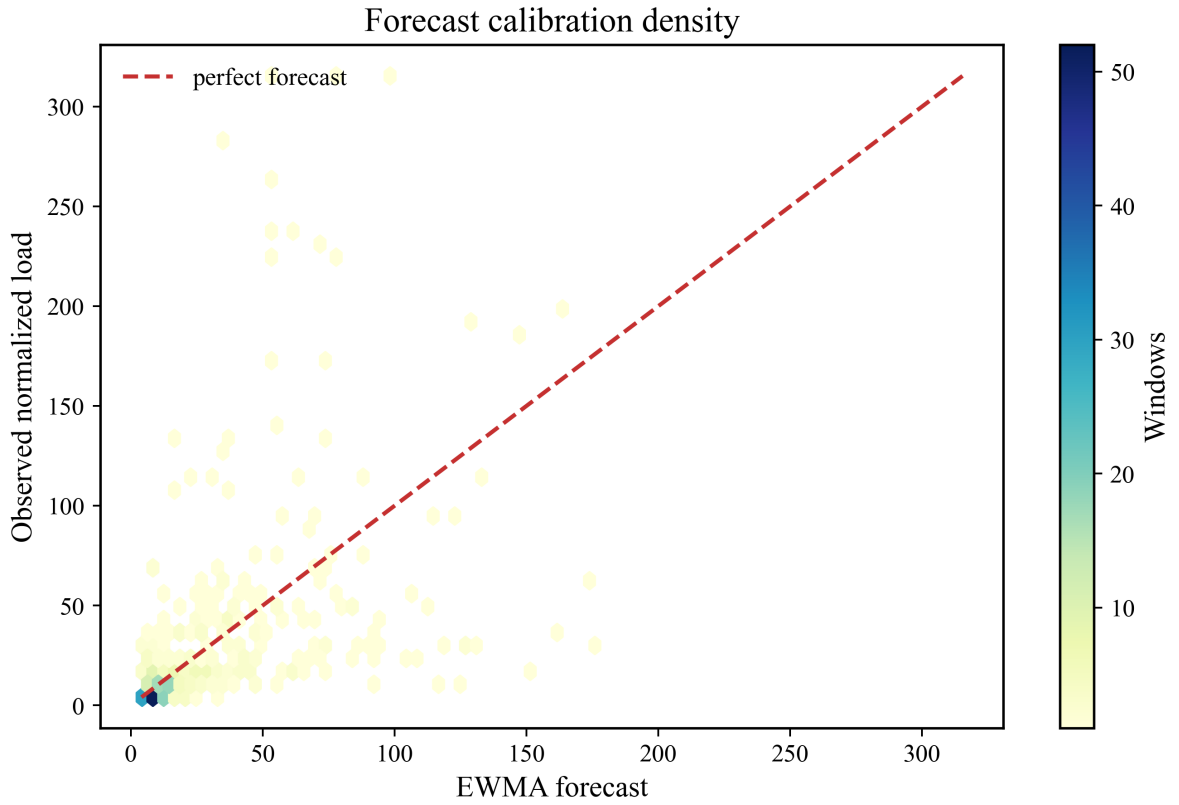

Figure 17. Forecast calibration density for observed versus predicted workload.

The rolling uncertainty band in Figure 18 provides another operational interpretation. During calm periods the 10–90% band narrows, allowing the controller to reduce its safety margin. During volatile periods the band widens, and the controller becomes more conservative. This is a more interpretable behavior than permanently lowering the target utilization. It also gives operators a direct dashboard signal: a widening band means the service has become less predictable.

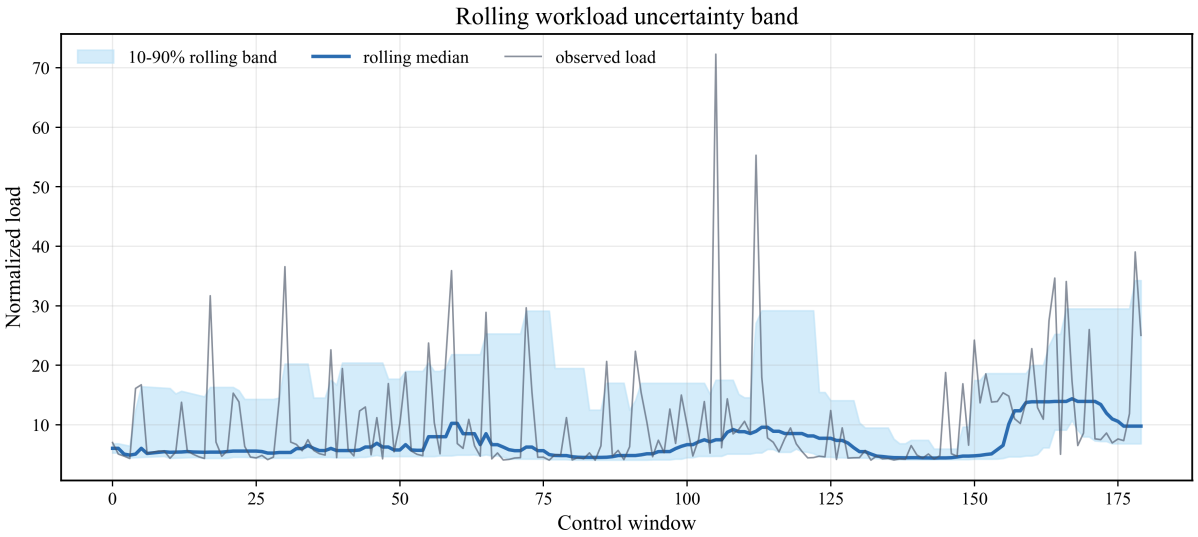

Figure 18. Rolling uncertainty band for workload intensity.

The radar profile in Figure 19 is not meant to replace quantitative tables. It is a compact visual summary of five competing objectives: cost, tail latency, risk, stability, and utilization. Such a profile helps stakeholders understand that no autoscaling policy dominates every dimension. Static provisioning scores well on risk but poorly on cost; reactive control scores well on cost but poorly on tail risk; DR-MPC-Elastic seeks a more balanced profile.

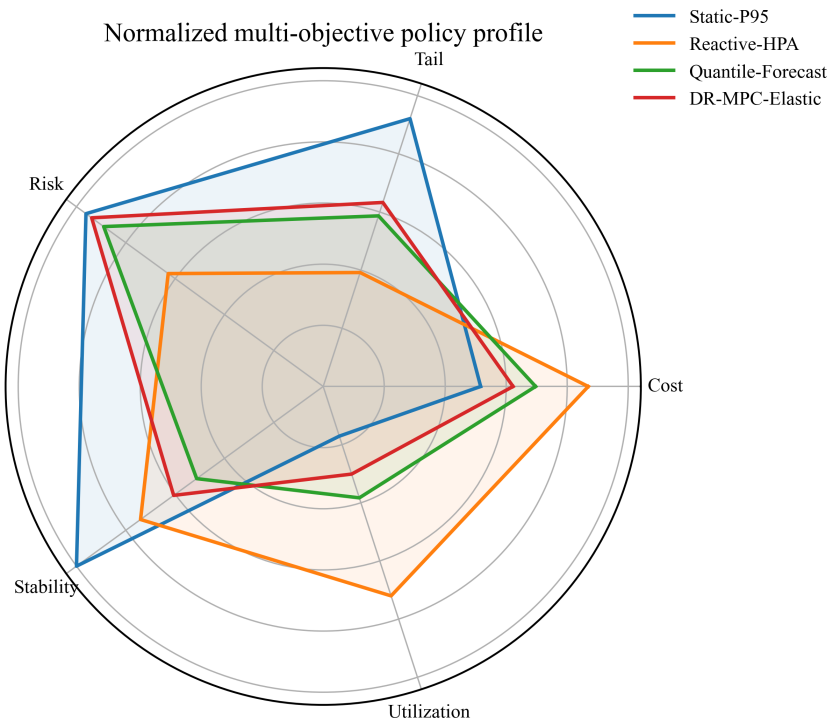

Figure 19. Normalized multi-objective policy profile.

Finally, the correlation heatmap in Figure 20 checks whether the control signals are redundant. If residual margin were perfectly correlated with load, a separate ambiguity radius would add little information. The observed correlations are partial rather than perfect, which supports the design choice of modeling residual risk explicitly. These diagnostics make the paper more reproducible because they allow readers to inspect not only the final metric, but also the statistical signals that drove the controller.

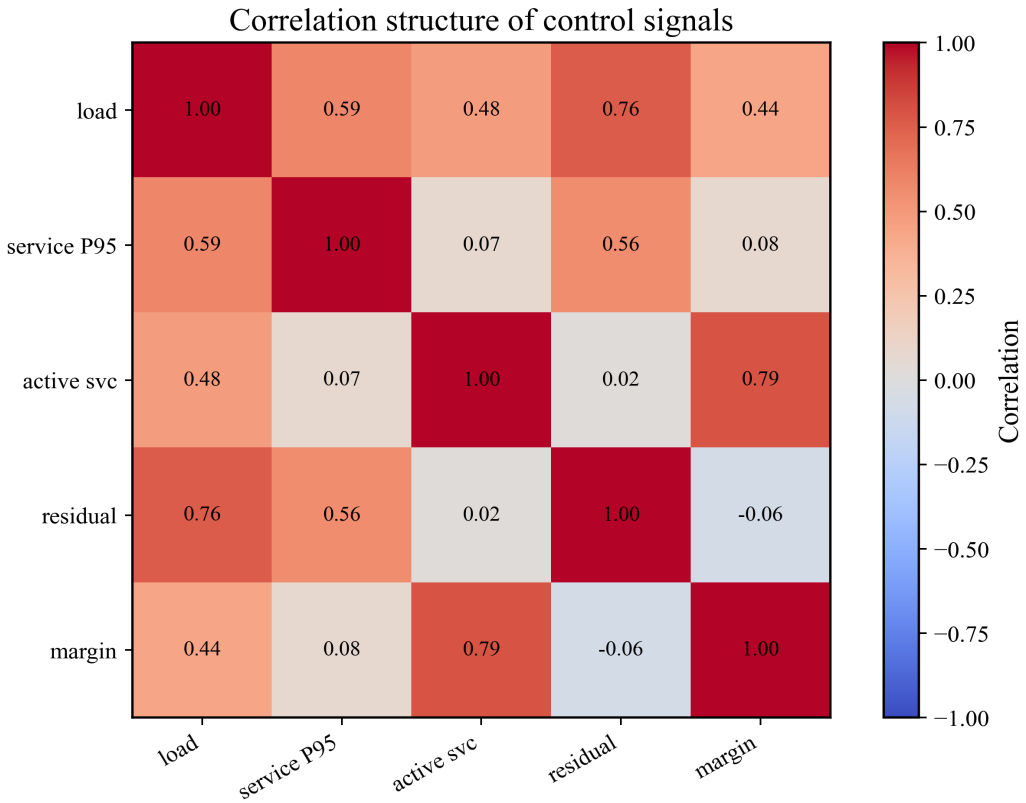

Figure 20. Correlation structure of observable control signals.

9. Deployment Considerations, Threats to Validity, and Future Work

A production deployment can run DR-MPC-Elastic as an external metrics adapter. The adapter computes a risk-corrected target and publishes it to HPA or KEDA. This route keeps the existing control plane intact while improving the semantics of the autoscaling signal. A stricter deployment can send targets directly to a platform-native autoscaler, but that approach requires more operational validation.

The method has important limitations. First, service capacity may be nonlinear when shared state, locks, databases, or downstream dependencies dominate. In such cases μ should be replaced with an empirically estimated concave capacity function. Second, if bursts are caused by retry storms, scaling alone may amplify load and should be combined with rate limiting and circuit breaking. Third, the public trace does not expose cold-start delay, image-pull delay, or dependency topology.

Future work should extend the controller to multi-resource decisions, call-graph-aware propagation, and online A/B experiments. It should also study fairness across services, because risk budgets can otherwise favor noisy services at the expense of stable services. Finally, the ambiguity radius itself should be monitored as an incident signal: when ρ_t rises rapidly, the platform has learned that its recent forecasts no longer explain the service.

10. Conclusions

This paper presented DR-MPC-Elastic, a distributionally robust controller for elastic compute services. The method turns forecast residuals into a risk radius, converts tail risk into a replica correction, and uses mirror descent to produce bounded integer actions. The public Alibaba trace case study supports the central claim: explicit tail-risk modeling can reduce SLO-risk exposure without reverting to static over-provisioning. The resulting framework is not a complete production autoscaler, but it is a rigorous and inspectable step toward risk-aware elasticity. Appendix A provides the extended derivation and operational audit notes.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable. This study did not involve humans or animals.

Informed Consent Statement

Not applicable. This study did not involve humans.

Data Availability Statement

The data analyzed in this study are publicly available from the Alibaba Cluster Trace Program and the Zenodo dataset cited in the article. No new proprietary dataset was generated

Conflicts of Interest

The author declares no conflict of interest.

Appendix A. Extended Derivation and Operational Audit Notes

This appendix clarifies how the proposed controller should be audited before it is connected to a production control plane. The first audit item is dimensional consistency. Arrival rate, service demand, and service capacity must be expressed in compatible units before Equation (2) is interpreted as a queue recursion. In the public trace case study, the workload is normalized into replica-equivalent capacity units because the trace does not expose the exact production CPU allocation of each replica.

The second audit item is residual stationarity. DR-MPC-Elastic does not assume that residuals are globally stationary. It assumes only that a short rolling window contains useful information about near-future uncertainty. If the residual distribution changes faster than the window can detect, the ambiguity radius will lag. In

production, this failure mode should be monitored by comparing realized residuals with the predicted residual band in Figure 18.

The third audit item is actuator feasibility. Even if Equation (9) recommends a large replica increase, the platform may not be able to create those replicas within one control window. Container image pull time, node availability, scheduling latency, readiness probes, and service-mesh propagation all affect realized capacity. Equation (13) therefore clips the action. This clipping should be calibrated from platform measurements rather than chosen only from simulation convenience.

The fourth audit item is dependency coupling. A service may not benefit from extra replicas if the bottleneck is a database, queue, cache, or downstream API. In such cases, the service-capacity term μ should be replaced by an empirically fitted capacity curve. A concave curve is often more realistic than a linear one, because lock contention and shared dependencies create diminishing returns.

The fifth audit item is interaction with protection mechanisms. Autoscaling should not be the only response to bursts. Rate limiting, load shedding, circuit breaking, admission control, and retry budgets are often more effective during dependency failures. DR-MPC-Elastic should therefore be deployed as part of a larger resilience policy rather than as an isolated controller.

The sixth audit item is fairness across services. A service with highly volatile demand may consume a large risk budget if the controller is configured independently for every service. A production platform may need a higher-level allocator that distributes risk budgets across tenants, business priorities, and node pools. This paper focuses on a single aggregated workload because the public trace does not provide enough information to evaluate cross-tenant fairness.

The seventh audit item is rollback behavior. A safe controller should expose its intermediate signals: forecast, residual tail, ambiguity radius, CVaR estimate, backlog proxy, proposed target, clipped target, and final applied replica count. If an incident occurs, engineers should be able to reconstruct why the controller scaled. This transparency is a major reason to prefer a structured risk model over an opaque policy in regulated or business-critical platforms.

The eighth audit item is numerical stability. CVaR estimates can become noisy when the rolling window contains few extreme observations. Practical implementations should smooth the radius, cap its maximum value, and use minimum sample requirements. When the sample requirement is not met, the controller can fall back to a conservative percentile rule. This fallback should be explicit in the implementation.

The final audit item is reproducibility. The public trace experiment should be understood as a reproducible stress test of the decision logic, not as a claim that the exact numerical constants are universal. Service owners should re-estimate target utilization, tail confidence level, switching bounds, and backlog gains using their own traces before deployment. The value of the framework is that every such parameter has a visible operational meaning.

Table A1. Operational audit checklist for DR-MPC-Elastic.

Audit Dimension	Question to Ask before Deployment	Expected Evidence
Units	Are demand and capacity in compatible units?	Load-test or telemetry conversion
Residual risk	Does the ambiguity radius cover realized errors?	Rolling calibration plot
Actuation	Can requested replicas become ready within one window?	Readiness latency distribution
Dependencies	Is replica capacity approximately linear?	Service-specific saturation curve
Fallback	What happens when residual samples are insufficient?	Documented conservative rule

References

1 Barroso LA, Clidaras J, Holzle U. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*; Morgan & Claypool: San Rafael, CA, USA, 2013.

2 Dean J, Barroso LA. The Tail at Scale. *Communications of the ACM* 2013; **56**(**2**):74–80.

3 Verma A, Pedrosa L, Korupolu M, et al. Large-Scale Cluster Management at Google with Borg. In

Proceedings of the Tenth European Conference on Computer Systems, Bordeaux, France, 22–24 April 2015; pp. 1–17.

4 Burns B, Grant B, Oppenheimer D, et al. Borg, Omega, and Kubernetes. *Communications of the ACM* 2016; **59**(**5**): 50–57.

5 Schwarzkopf M, Konwinski A, Abd-El-Malek M, et al. Omega: Flexible, Scalable Schedulers for Large Compute Clusters. In Proceedings of the 8th ACM European Conference on Computer Systems, Prague, Czech Republic, 14–17 April 2013.

6 Kubernetes. Horizontal Pod Autoscaling. *Kubernetes Documentation*. 2025. Available online: https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/ (accessed on 29 June 2026).

7 CNCF. Kubernetes Event-driven Autoscaling (KEDA) Documentation. 2025. Available online: https://keda.sh/docs/ (accessed on 29 June 2026).

8 Mao H, Alizadeh M, Menache I, et al. Resource Management with Deep Reinforcement Learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks, Atlanta, Georgia, USA, 9–10 November 2016; pp. 50–56.

9 Mao H, Schwarzkopf M, Venkatakrishnan SB, et al. Learning Scheduling Algorithms for Data Processing Clusters. In Proceedings of the ACM Special Interest Group on Data Communication, Beijing, China, 19 August 2019; pp. 270–288.

10 Neely MJ. *Stochastic Network Optimization with Application to Communication and Queueing Systems*; Morgan & Claypool: San Rafael, CA, USA, 2010.

11 Rockafellar RT, Uryasev S. Optimization of Conditional Value-at-Risk. *Journal of Risk* 2000; **2**: 21–41.

12 Ben-Tal A, El Ghaoui L, Nemirovski A. *Robust Optimization*; Princeton University Press: Princeton, NJ, USA, 2009.

13 Esfahani PM, Kuhn D. Data-Driven Distributionally Robust Optimization Using the Wasserstein Metric. *Mathematical Programming* 2018; **171**: 115–166.

14 Hazan E. Introduction to Online Convex Optimization. *Foundations and Trends in Optimization* 2016; **2**(**3–4**): 157–325.

15 Rawlings JB, Mayne DQ, Diehl M. *Model Predictive Control: Theory, Computation, and Design*; Nob Hill: Madison, WI, USA, 2017.

16 Alibaba Cluster Trace Program. Cluster-Trace-Microservices-v2021. GitHub Repository. 2021. Available online: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-microservices-v2021 (accessed on 29 June 2026).

17 Delimitrou C, Kozyrakis C. Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters. *ACM Sigplan Notices* 2013; **48**(**4**): 77–88.

18 Delimitrou C, Kozyrakis C. Quasar: Resource-Efficient and QoS-Aware Cluster Management. *ACM Sigplan Notices* 2014; **49**(**4**): 127–144.

19 Shahrad M, Fonseca R, Goiri I, *et al.* Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 20), Boston, Massachusetts, USA, 15–17 July 2020; pp. 205–218.

20 Zhang Y, Goiri I, Chaudhry GI, *et al.* Faster and Cheaper Serverless Computing on Harvested Resources. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual, 26 – 29 October 2021; pp. 724–739.

21 Zhang Y. Azure Functions Invocation Trace 2021. Microsoft Azure Public Dataset. 2021. Available online: https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsInvocationTrace2021.md (accessed on 29 June 2026).

22 Wang L, Li M, Zhang Y, *et al.* Peeking Behind the Curtains of Serverless Platforms. In Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC 18), Boston, MA, USA, 11 – 13 July 2018; pp. 133–146.

23 Kleinrock L. *Queueing Systems, Volume 1: Theory*; Wiley: Hoboken, NJ, USA, 1975.

24 Harchol-Balter M. *Performance Modeling and Design of Computer Systems*; Cambridge University Press: Cambridge, UK, 2013.

25 Yan H. Real-Time 3D Model Reconstruction through Energy-Efficient Edge Computing. *Optimizations in Applied Machine Learning* 2022; **2**(**1**). https://doi.org/10.71070/oaml.v2i1.48.

26 Yan H, Shao D. Enhancing Transformer Training Efficiency with Dynamic Dropout. *arXiv* 2024. arXiv: 2411.03236.

27 Luo Z, Yan H, Pan X. Optimizing Transformer Models for Resource-Constrained Environments: A Study on Model Compression Techniques. *Journal of Computational Methods in Engineering Applications* 2023; **3**(**1**): 1–12. https://doi.org/10.62836/jcmea.v3i1.030107.

28 Yan H, Shao D. Multimodal Medical Image Analysis: Integrating LLM and RAG Deep Learning Strategies. *Journal of Advances in Information Technology* 2025; **16**(**4**): 568–581. https://doi.org/10.12720/jait.16.4.568-581.

29 Lu Y, Shao D, Ni X, *et al.* Emotion-Style Dual Prediction: A Multi-Task Deep Learning Approach for Artistic Images. *Cluster Computing* 2026; **29**(**1**): 31.

30 Dai Y. Medical Biopharmaceutical Image Anomaly Detection under Retinex State Space Duality and Frequency Consensus-Driven Transformer. *Journal of Computational Methods in Engineering Applications* 2026; **6**(**1**): 0001.

31 Dai Y. Deep Learning-Based Medical Image Segmentation for Early Cancer Detection. *Optimizations in Applied Machine Learning* 2025; **5**(**1**). https://doi.org/10.71070/oaml.v5i1.144.

32 Dai Y. MobileMamba-HC: Medical Image Disease Detection in Healthcare Integrating Frequency Adaptive Dilated Convolution and Spatial-Channel Synergistic Attention. *Innovations in Applied Engineering and Technology* 2026; **5**(**1**): 0002.

33 Dai Y, Wei L, Yu C. Graph Neural Network-Based Drug-Target Interaction Prediction for Precision Medicine. *Optimizations in Applied Machine Learning* 2025; **5**(**1**). https://doi.org/10.71070/oaml.v5i1.145.

34 Li J, Culver TB, Burgis CR, *et al.* Validating Nitrogen Removal Models with Field Bioretention Data. *Journal of Environmental Engineering* 2024; **150**(**8**): 04024037.

35 Li J, Culver TB, Persaud PP, *et al.* Developing Nitrogen Removal Models for Stormwater Bioretention Systems. *Water Research* 2023; **243**: 120381.

36 Li J. Nitrogen Removal Models for Stormwater Bioretention Systems. Ph.D. Thesis, University of Virginia, Charlottesville, VA, USA, 2023.

37 Li J, Culver TB. Review of Process-Based Nitrogen Model for Agricultural Fields with Implications for Nitrogen Simulations in Stormwater BMPs. *Environmental Modelling & Software* 2022; **151**: 105363.

38 Deng X, Oda S, Kawano Y. Graphene-based Midinfrared Photodetector with Bull's Eye Plasmonic Antenna. *Optical Engineering* 2023; **62**(**9**): 097102.

39 Deng X, Li L, Enomoto M, *et al.* Continuously Frequency-Tuneable Plasmonic Structures for Terahertz Bio-Sensing and Spectroscopy. *Scientific Reports* 2019; **9**(**1**): 3498.

40 Zhang Y, Needleman A. On the Identification of Power-Law Creep Parameters from Conical Indentation. *Proceedings. Mathematical, Physical, and Engineering Sciences* 2021; **477**(**2252**): 20210233.

41 Zhang Y, Needleman A. Characterization of Plastically Compressible Solids via Spherical Indentation. *Journal of the Mechanics and Physics of Solids* 2021; **148**: 104283.

42 Gandhi A, Harchol-Balter M, Das R, *et al.* Optimal Power Allocation in Server Farms. *ACM Sigmetrics Performance Evaluation Review* 2009; **37**(**1**): 157–168.

43 Hellerstein JM, Faleiro J, Gonzalez JE, *et al.* Serverless Computing: One Step Forward, Two Steps Back. *arXiv* 2018. arXiv:1812.03651.

44 Kaffes K, Chong T, Humphries JT, *et al.* Centralized Core-Granular Scheduling for Serverless Functions. In Proceedings of the ACM Symposium on Cloud Computing, Santa Cruz, CA, USA, 21–23 October 2019; pp. 158–164.

45 Yu T, Noghabi SA, Raindel S, *et al.* Characterizing Serverless Platforms with Serverless Bench. In Proceedings of the 11th ACM Symposium on Cloud Computing, Virtual, 19–21 October 2020; pp. 30–44.

46 Gan Y, Delimitrou C. Seer: Leveraging Big Data to Navigate Performance Debugging in Cloud Microservices. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, RI, USA, 13–17 April 2019; pp. 19–33.

47 Qiu H, Banerjee S, Jha S, *et al.* FIRM: Fine-grained Resource Management for SLO-Oriented Microservices. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), Virtual, 4–6 November 2020; pp. 805–825.

48 Rausch T, Rashed A, Dustdar S. Optimized Container Scheduling for Data-Intensive Serverless Edge Computing. *Future Generation Computer Systems* 2021; **114**: 259–271.

49 Tirmazi M, Barker A, Deng N, *et al.* Borg: The Next Generation. In Proceedings of the Fifteenth European Conference on Computer Systems, Virtual, 27 April 2020; pp. 1–14.

50 Grandl R, Kandula S, Rao S, *et al.* Graphene: Packing and Dependency-Aware Scheduling for Data-Parallel Clusters. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 81–97.

51 Chen Q, Liu C, Xiao Z. Improving MapReduce Performance Using Smart Speculative Execution Strategy. *IEEE Transactions on Computers* 2013; **63**(**4**): 954–967.

52 Casanova H, Legrand A, Quinson M. SimGrid: A Generic Framework for Large-Scale Distributed Experiments. In Proceedings of the Tenth International Conference on Computer Modeling and Simulation (uksim 2008), Cambridge, UK, 1–3 April 2008; pp. 126–131.

53 Moritz P, Nishihara R, Wang S, *et al.* Ray: A Distributed Framework for Emerging AI Applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; pp. 561–577.