Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 5, No. 4, 2026

Multi-Objective Optimization in Cloud-Native Intelligent Systems

Marius Schottlender[1], Arya Barlocker[1], Eric Li[2]
[1]Arizona State University, Tempe, USA
[2]The University of Texas at Austin, Texas, USA
*Corresponding author: Eric Li: lison88033@gmail.com

Abstract: Multi-objective optimization has become one of the central design principles for cloud-native intelligent systems because the dominant workloads of the current cycle of distributed computing-large language model inference, LLM-assisted scheduling, observability-driven AIOps, privacy-preserving federated learning, and edge/IoT reasoning-no longer admit a single "best" objective. Instead, practitioners must optimize under persistent trade-offs among latency, throughput, service-level objective compliance, energy, carbon emissions, cloud expenditure, fairness, privacy, trust, robustness, and remediation time. Recent work makes this shift explicit. In the LLM-serving literature, DeepServe models joint SLO-cost scheduling as a contextual bandit; BOute co-optimizes heterogeneous model routing and GPU deployment with multi-objective Bayesian optimization; ECCOS combines predictive quality-cost estimation with constrained optimization; and the Liao preprint framed as prompt-level cost prediction and SLO awareness presents CAPS, a bi-objective carbon-aware scheduler for online LLM inference. In adjacent domains, cloud autoscaling has been reformulated as risk-constrained reinforcement learning, microservice rate limiting as deep RL under throughput-latency tension, and federated cloud analytics as a joint optimization of accuracy, communication cost, trust, and privacy. The first experiment studies policy search for multi-pool LLM serving under cost, carbon, and SLO-met goodput. The second study autoscales under workload drift and compares a threshold policy, a latency-only reactive controller, and a risk-aware controller. The experiments are not benchmark replications; they are explanatory simulations designed to make the trade-offs in the reviewed literature concrete. They show that Pareto-oriented search can recover better budgeted goodput than simple scalarization under a carbon budget, and that risk-aware capacity control can achieve substantially lower tail latency than a latency-only controller at a modest capacity premium. These findings align with the broader literature's movement away from single-objective heuristics and toward closed-loop, multi-objective, and observability-aware decision systems.

Keywords: Multi-objective optimization; cloud-native intelligent systems; LLM inference scheduling; cloud resource management

1. Introduction

Cloud-native intelligent systems now span far more than containerized deployment of machine learning models. They include heterogeneous LLM-serving stacks, serverless inference, multi-tenant GPU schedulers, agentic control planes, observability-assisted diagnosis, federated anomaly detection, and edge/IoT schedulers that push inference and control across cloud, edge, and device layers. A recent research agenda on cloud-native and distributed support for LLMs argues that such systems must jointly address latency, cost, quality of service, fairness, isolation, admission control, and energy efficiency under bursty, heterogeneous workloads, and that observability is not simply a monitoring afterthought but a prerequisite for learning-based control [1-3].

That perspective is historically significant because classical cloud scheduling literature often optimized a single primary criterion-makespan, utilization, response time, or cost-using heuristics or scalar reward functions. In contrast, today's intelligent cloud workloads create objective vectors rather than scalar objectives. An online LLM-serving platform may need to maximize SLO-met goodput while minimizing cost, carbon, and tail latency. A microservice diagnosis system may need to maximize localization accuracy while minimizing mean time to remediation and false positives. A federated anomaly-detection fabric may need to preserve privacy and Byzantine robustness while minimizing WAN communication and preserving accuracy. Even edge systems, which have long wrestled with delay-energy trade-offs, now increasingly treat these trade-offs as first-class optimization targets rather than engineering side constraints [4].

The shift is especially visible in LLM systems. Foundational serving work such as vLLM with PagedAttention, DistServe's prefill/decode disaggregation, ServerlessLLM, and Sarathi-Serve each solved critical single-system bottlenecks-memory fragmentation, phase interference, cold starts, and throughput-latency trade-offs-but they largely optimized one or two closely coupled system metrics at a time. The newer literature reviewed here builds on those system primitives and makes the optimization problem more explicit: routing across heterogeneous model fleets, coordinating heterogeneous GPUs, trading carbon intensity against SLO attainment, reasoning about multi-tenant fairness, and coupling autoscaling or token-level scheduling with workload forecasts [5].

This review therefore asks four questions. First, what are the dominant objective sets now being optimized in cloud-

native intelligent systems? Second, what algorithmic forms are emerging-analytical optimization, constrained solvers, Bayesian optimization, reinforcement learning, contextual bandits, and LLM-based decision agents-and where do they fit best? Third, how do observability, privacy, trust, and security alter the optimization landscape beyond latency and cost? Fourth, what is still missing for the field to mature from promising prototypes to reproducible, comparable, and operationally safe systems? The answers matter because the reviewed literature is no longer just about improving one system metric. It is about governing cloud-native intelligence under permanent tension between competing goals [6].

2. Taxonomy and problem formulation

A useful unifying view is to treat cloud-native intelligent systems as closed-loop decision systems that operate over distributed state, make repeated placement or control decisions, and optimize multiple objectives subject to hard constraints. Let x denote the decision variables. Depending on the system, x may represent prompt-to-model routing, GPU pool assignment, batching policy, token slicing, prefill/decode separation, autoscaling action, rate-limit threshold, federated aggregation weights, or remediation action. The objective vector typically has the form

$$\min F(x) = \begin{bmatrix} f_{\text{latency}}(x),\ f_{\text{cost}}(x),\ f_{\text{carbon}}(x), \\ f_{\text{risk}}(x),\ f_{\text{privacy-loss}}(x),\ -f_{\text{goodput}}(x) \end{bmatrix}$$

subject to SLO, trust, privacy, safety, and resource constraints. Pareto dominance then becomes the natural language of solution quality: a decision dominates another when it is no worse in every objective and strictly better in at least one. This is the underlying logic behind contemporary constrained schedulers, carbon-aware routing, and privacy-cost-aware federated systems, even where the paper's surface formulation differs. ECCOS explicitly casts multi-LLM serving as a constrained optimization problem with quality and workload constraints; BOute uses multi-objective Bayesian optimization to co-optimize routing and deployment; CAPS jointly optimizes goodput and per-request carbon cost; LarS defines a bi-objective cloud scheduling problem over cost and success rate; and the resource-management agenda paper argues that future systems must jointly optimize latency, cost, QoS, fairness, and sustainability [7].

The literature can be organized into four interdependent layers. The first is the execution layer, where systems decide how to place, batch, split, and schedule computation across heterogeneous accelerators and sometimes across cloud and edge. The second is the control layer, where autoscaling, admission control, and rate limiting adjust system boundaries under drift and bursts. The third is the observability and diagnosis layer, which fuses metrics, logs, traces, and service dependencies to identify anomalies, localize root causes, and in some cases trigger remediation. The fourth is the trust, privacy, and defense layer, where federated learning, differential privacy, Byzantine resilience, and attack mitigation turn security and governance into optimization variables rather than post-processing checks. The most important trend is that these layers are converging: observability informs scheduling, scheduling affects security posture, privacy affects communication cost, and diagnostic models influence control actions [8].

The reviewed evidence also shows that optimization granularity is shrinking. Earlier cloud schedulers often operated at the VM or service level. Newer systems act at the request, prompt, token, or expert level. The research agenda paper explicitly frames elastic LLM scheduling around heterogeneity in prompt length, generation length, model structure, adapter activation, expert routing, and KV-cache pressure, and emphasizes that traditional resource abstractions are too coarse for modern LLM serving. BOute also highlights the need to co-optimize query routing and model deployment, while the MoE scheduling paper organizes decisions into request-, engine-, and expert-level scheduling. This change in granularity matters because multi-objective optimization becomes more expressive as the system can move from "which cluster?" to "which prompt on which model on which hardware at what time under which carbon signal?" [10]

Before turning to the literature in detail, one more taxonomic point is necessary. "Multi-objective" does not imply the same solver everywhere. Some papers keep a scalar reward but encode multiple objectives inside it. Some impose hard constraints and optimize one principal objective. Some use explicit Pareto-style search. Some use contextual bandits or RL to optimize long-run vector trade-offs. Some use LLMs not as workloads but as decision agents. The literature is therefore best understood not as one algorithmic family but as a design space in which objective multiplicity is the common feature, while the optimization mechanics differ by time scale, safety requirements, observability quality, and deployment cost [11].

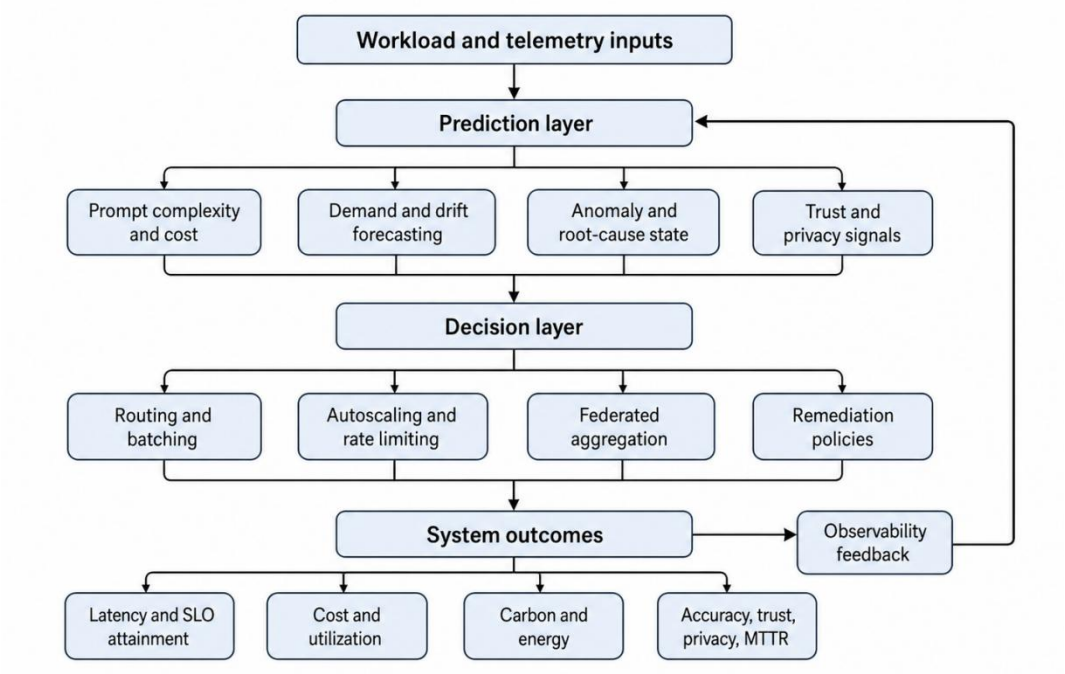

Fig. 1. Common closed-loop architecture for multi-objective optimization in cloud-native intelligent systems. The taxonomy is synthesized from the reviewed scheduling, observability, federated learning, and cloud-control literature [9].

3. Literature review

The fastest-moving body of work is LLM-serving and inference scheduling. DeepServe is emblematic because it does not treat SLO compliance and cost efficiency as separate post hoc criteria; the abstract describes a contextual-bandit scheduler that profiles prompt length, KV-cache hit ratio, and predicted generation length, then adaptively chooses batch size, concurrency, cache eviction policy, and warm-standby strategy. On ShareGPT and BurstGPT traces with LLaMA-2-13B and Mixtral-8x7B on NVIDIA A100 GPUs, the paper reports reductions of P99 latency by 38-62 percent, GPU-utilization gains of 14-23 percent, and per-request cost reductions up to 27 percent relative to its baselines. This is important not only because the reported gains are strong, but because the work operationalizes a live trade-off between tail latency and cost in a serverless multi-tenant setting rather than optimizing one and auditing the other later [12].

BOute pushes the optimization frontier further by performing algorithm-system co-design across heterogeneous LLMs and heterogeneous GPUs. Its core observation is that cost-efficient serving depends jointly on routing easier queries to cheaper models and on choosing hardware deployments that exploit heterogeneous GPU resources. It formulates this as a multi-objective Bayesian optimization problem over routing strategy and deployment configuration and reports up to 157 percent improvement over prior systems under identical cost budgets and quality requirements, or average cost reductions of 38 percent while preserving the same targets. That result is notable because Bayesian optimization is not usually the first method associated with cloud serving, yet the search space here is expensive, structured, and non-convex-exactly the conditions where surrogate-based search can outperform manually tuned heuristics [13].

ECCOS occupies a complementary point in the design space. It models multi-LLM serving as a constrained optimization problem with explicit response-quality and workload constraints. Its two-stage structure-a multi-objective predictor followed by a constrained optimizer-is especially revealing for the field as a whole. The predictor estimates capability and output-length-related cost, while the optimizer uses Lagrangian dual methods to find assignments that satisfy system-level quality and load constraints. ECCOS also contributes the QAServe dataset, a query-model-performance dataset built from QA and reasoning tasks, and reports up to 6.30 percent improvement in success rate and 10.15 percent cost reduction with less than 0.5 percent added decision overhead relative to response time. The larger point is methodological: in many cloud-native multi-objective systems, progress will come from better estimators of future state as much as from better solvers [14].

The Liao et al. preprint-titled as a study of prompt-level cost prediction and SLO awareness-follows that same logic but with a sustainability twist. Its abstract presents CAPS, a carbon-aware prompt scheduler that uses a lightweight prompt-complexity predictor, real-time grid carbon intensity, GPU energy profiles, and per-tenant SLO tiers to route requests among a low-latency pool, a low-carbon pool, and a delay-tolerant batch pool. In its trace-driven evaluation, CAPS reduces carbon emissions per 1K generated tokens by 26.8 percent relative to round robin while matching or exceeding an SLO-aware baseline in SLO attainment. This paper matters because it marks the expansion of the cloud-native objective set from latency, cost, and throughput to include carbon at online-

serving timescales rather than only in training-job placement [15].

Work on MoE and hierarchical offloading extends multi-objective optimization to finer scheduling granularity. Sun et al. propose three-level scheduling for MoE reasoning-request, engine, and expert levels-with request-level strategies such as shortest-job-first and priority-aware aging, engine-level dispatch that considers prefix load and KV-cache utilization, and expert-level hotspot relief. Across more than 100 experiments, the framework reports up to 17.8 percent TTFT reduction and 13.3 percent TPOT reduction relative to vLLM. Wang et al. similarly argue for hybrid edge-cloud LLM serving and use hierarchical scheduling plus dynamic attention sparsification; despite being listed as a withdrawn ICLR 2026 submission, the abstract reports up to 1.86 times throughput improvement against a cloud-only baseline. Together, these papers show that the optimization locus is shifting from service-level placement to phase- and component-level orchestration [16].

A related thread uses LLMs as schedulers rather than only as scheduled workloads. Pei et al.'s cloud-task scheduling paper presents LarS, which combines DRL-derived high-quality trajectories with LLM fine-tuning and explicitly formulates a two-objective optimization over cost minimization and QoS-compliant success rate maximization. The reward function combines cost and QoS, and the generalization evaluation compares a fine-tuned LLaMA-2 scheduler with random, round-robin, earliest, and DQN baselines across environments with 8, 20, and 40 VMs. The paper reports that the LLM scheduler consistently yields the highest success rate across environments and nearly 20 percent lower cost than the closest naive method in the 8-VM case while maintaining lower response time [17].

Jadhav et al. examine a similar question in HPC scheduling rather than cloud task placement. Their ReAct-style LLM scheduler uses scratchpad memory and constraint enforcement to balance makespan, wait time, resource use, and fairness across seven real-world HPC scenarios. The interesting result is not simply that LLM-based reasoning can compete with FCFS, SJF, and OR-Tools on multi-objective workloads, but that the paper explicitly identifies the real-time deployment cost of reasoning overhead as a limiting factor. This is a useful corrective to overenthusiastic interpretations of LLM-driven orchestration: interpretability and generalization are valuable, but they themselves become objectives that may trade off against decision latency and controller cost [18].

The same multi-objective logic is visible outside datacenter-scale LLM serving. Khan et al.'s Scientific Reports paper on LLM-enabled adaptive scheduling in IoT sensing uses LLM reasoning to reduce redundant sensing and optimize network resource use. The paper reports improvement of 57.8-60 percent in its MTP metric, decreases in median delay between 26 and 60 percent, and an energy-efficient operating point with tight confidence intervals. The result is significant because it demonstrates that LLM-assisted decision systems are not only cloud residents; they increasingly mediate cross-layer optimization at the edge, where communication cost, energy, and delay are tightly coupled [19].

The analytics and control side of the literature shows a parallel shift. Gelenbe's edge paper is conceptually foundational because it argues against purely heuristic or simulation-only approaches and derives an explicit mathematical allocation of job fractions across edge servers to minimize average delay and average energy consumption. That analytical strand remains important even in a field dominated by learning-based control, because it gives interpretable structure for trade-off surfaces and can serve as a sanity bound for learned policies. Bergquist, Gelenbe, Nasereddin, and Sigman then move from edge task placement to network-access control by studying Quasi-Deterministic Transmission and showing that traffic shaping can mitigate massive access without increasing end-to-end delay while reducing gateway queue length; the same paper also proposes an adaptive non-deterministic transmission policy with only one buffer at the gateway. These are not LLM papers, but they are part of the same multi-objective lineage: balance delay, protection, and resource efficiency under bursty access conditions [20].

Cloud control papers are increasingly explicit about stability and risk as objectives in their own right. Huang et al. formulate elastic scaling as a risk-constrained reinforcement-learning problem and stress that the controller must jointly optimize response performance, resource utilization, and instability cost under workload drift. Lyu et al. similarly cast microservice rate limiting as a deep-RL problem that dynamically balances system throughput and service latency; their arXiv abstract reports 23.7 percent throughput improvement and 31.4 percent P99 latency reduction against fixed-threshold methods, plus large reductions in degradation incidents and manual interventions in production. These results extend the interpretation of "multi-objective optimization" from placement and batching into safety-aware control: not only "how much capacity?" but "how much capacity without oscillation, collapse, or chronic overreaction?" [21]

The observability and diagnosis literature broadens the objective vocabulary even further. Wang et al. propose an LLM-enhanced observability framework that integrates OpenTelemetry telemetry with a domain-adapted LLM for multimodal analysis of metrics, logs, and traces. Their reported results include an F1 score of 0.95 and an 84.2 percent reduction in mean time to remediation compared with rule-based baselines. Huang et al. propose structure-aware unified modeling for root cause localization, explicitly incorporating service dependency graphs into representation learning and anomaly attribution, improving over DeepTraLog, TraceGra, MADMM, and SRdetector with 0.912 accuracy and 0.941 AUC in the reported table. These papers are crucial because they show that multi-objective optimization in cloud-native intelligence is not only about serving AI quickly; it is also about running AI systems reliably, explainably, and recoverably [22].

Anomaly detection work in microservices reinforces that point. Zhang et al. propose unsupervised anomaly detection through cross-service temporal contrastive learning, explicitly modeling service-call relations and multi-scale temporal dynamics. The preprint emphasizes that the method avoids additional labeling cost and does not depend on manual thresholds; the reported table shows its F1 score improving as

the model stabilizes at lower learning rates, reaching 0.910 in the best listed configuration. Liu et al.'s cost-sensitive Mamba paper tackles long-tail, low-frequency microservice faults by injecting asymmetric business costs into the loss function and using a state-space sequence model to better capture long-range dependencies. Zhang et al.'s protocol anomaly paper similarly combines GRU sequence modeling, reconstruction error, and contrastive learning over status-code sequences, with the reported sensitivity study indicating accuracy, F1, and AUC approaching or exceeding 0.99 at an appropriate temperature coefficient. In all three cases, "accuracy" alone is insufficient; researchers are optimizing class imbalance, false-negative cost, interpretability, and operational usefulness [23].

Fault diagnosis under limited labels and data imbalance provides a further example. Huang et al.'s self-supervised industrial IoT fault diagnosis paper uses a Transformer-based temporal encoder, multi-view consistency constraints, and sequence reconstruction. Its sensitivity study explicitly examines rising normal-to-abnormal sample ratios and shows that F1 falls under more severe imbalance but remains relatively high even at a 5:1 ratio, while the comparative table reports 0.907 F1 for the proposed method compared with lower scores for GNN, GAT, SimCLR, Transformer, 1DCNN, and BiLSTM baselines. The optimization target here is not just classification accuracy; it is stable performance under the kinds of label sparsity that real cloud-native and industrial systems routinely face [24].

Trust, privacy, and federated deployment transform the objective landscape again. Li et al.'s HierFedDP introduces a three-tier hierarchical federated learning system with local differential privacy and an edge aggregation frequency parameter, reporting 49 percent WAN communication reduction while maintaining comparable detection performance. Cost-TrustFL addresses multi-cloud federated learning under egress fees and malicious clients; it combines approximate Shapley-based lightweight reputation with cost-aware aggregation and reports 86.7 percent accuracy under 30 percent malicious clients while reducing communication cost by 32 percent. TrustGraph-DFL removes the central server entirely, using consistency-weighted neighborhood aggregation in decentralized federated learning and achieving 89-93 percent accuracy under several attack types while keeping false positive rates below 10 percent even at 50 percent Byzantine fraction in the reported stress test. These systems make it impossible to define "optimal" without specifying at least accuracy, communication cost, attack resilience, and trust calibration [25].

Security work beyond federation reaches the same conclusion. DISFIDA addresses distributed self-supervised federated intrusion detection for IoT and IoV environments, explicitly motivated by the cost and confidentiality constraints of highly distributed healthcare and vehicular systems. Adaptive attack mitigation for IoV flood attacks combines attack detection and mitigation to protect gateway QoS during flooding, and the QDTP line of work shows that access-shaping can jointly protect delay and queue stability. Meanwhile, the 2026 Information Fusion survey on LLM-based agent security organizes attacks and defenses around evaluation criteria, transferability, robustness, stealthiness, and defense quality. Taken together, these papers imply that security objectives are no longer separable from scheduling and control. A scheduling policy that ignores trust, poisoning, or attack surface is not merely incomplete; it may be dominated by a slightly slower but materially safer policy [26].

Table I synthesizes the literature reviewed in this section and groups it by decision scope, optimized objectives, and algorithmic style. The classification is derived from the sources cited in the surrounding discussion [27].

Area	Representative systems	Main objectives	Dominant optimization style	Reported evidence
Serverless multi-tenant LLM inference	DeepServe	P99 latency, GPU utilization, cost, SLO compliance	Contextual bandits with online profiling	Strong P99 and cost gains on ShareGPT/BurstGPT
Heterogeneous model and GPU serving	BOute	Quality, latency, performance per budget, serving cost	Multi-objective Bayesian optimization	Large cost-efficiency gains under fixed budgets
Multi-LLM routing	ECCOS	Response quality, cost, workload feasibility	Predictor plus constrained optimization	Higher success rate and lower cost with negligible scheduling overhead
Carbon-aware online LLM serving	CAPS	Goodput, carbon, SLO attainment	Prompt prediction plus bi-objective online routing	Lower carbon per 1K tokens with preserved SLO attainment
MoE and hybrid edge-cloud scheduling	Multi-Layer Scheduling; hierarchical offloading	TTFT, TPOT, throughput, utilization	Hierarchical scheduling and phase/component optimization	Lower TTFT/TPOT and higher throughput
LLMs as schedulers	LarS; LLM-HPC scheduling; LLM-AS	Cost, QoS, success rate, fairness, network performance	DRL-assisted fine-tuning; ReAct reasoning; LLM-guided control	Better generalization, good constraint satisfaction, but nontrivial reasoning cost
Cloud control	Risk-constrained autoscaling; RL rate limiting	Latency, throughput, resource usage, stability	Reinforcement learning with safety/risk structure	Better throughput/tail latency and more stable scaling
Observability and diagnostics	LLM-RCA; structure-aware RCA; contrastive anomaly detection	F1/AUC, MTTR, false alarms, detection cost	Graph modeling, multimodal learning, LLM reasoning	Faster remediation and better anomaly localization
Privacy, trust, and defense	HierFedDP; Cost-TrustFL; TrustGraph-DFL; DISFIDA	Accuracy, communication cost, privacy, Byzantine resilience	Hierarchical FL, trust-weighted aggregation, self-supervision	Lower WAN cost, higher attack resilience, practical multi-cloud

Area	Representative systems	Main objectives	Dominant optimization style	Reported evidence
				robustness

4. Comparative synthesis

Several robust cross-paper patterns emerge from the literature. The first is that prediction quality is now a structural part of optimization quality. DeepServe depends on a request profiler. ECCOS depends on a capability-cost predictor. CAPS depends on prompt-complexity and carbon prediction. Risk-constrained scaling depends on high-quality state estimation under drift. Structure-aware RCA depends on orienting message passing around service dependencies rather than generic feature fusion. These systems do not optimize on raw state alone; they optimize on predicted state. That means errors in workload forecasting, prompt complexity prediction, causal graph estimation, or trust scoring can dominate solver performance. In practice, the literature suggests that improvements in lightweight predictive models may yield larger system-level gains than marginal improvements in the downstream optimizer [28].

The second pattern is granularity inversion: decision-making is moving downward from clusters and VMs to prompts, tokens, prefill/decode phases, experts, and service graph neighborhoods. This is visible in the research agenda's emphasis on KV-cache pressure and phase-specific control, in DistServe's phase disaggregation, in Sarathi-Serve's chunked-prefill scheduling, in MoE request/engine/expert scheduling, and in the token-level resource-slicing line referenced by newer 2026 preprints. Finer granularity raises controller complexity, but it also exposes objective trade-offs that were previously hidden by coarse abstractions. Inference carbon cost, for example, is not just a function of which region runs a service. It can differ by prompt type, execution pool, queueing regime, and generated-token count [29].

The third pattern is the broadening of the objective set. Cost and latency remain central, but the current literature increasingly includes carbon, communication overhead, privacy loss, trust, MTTR, false positive rate, robustness under attack, and model quality. This is not a cosmetic broadening. It changes algorithm choice. A scalarized cost-latency scheduler can be effective if objectives are smooth and safe to trade. Once privacy or attack resilience enters the problem, hard constraints and conservative control become more attractive. Once MTTR and RCA quality are part of the score, diagnostic pipelines need graph structure and multimodal telemetry. Once carbon enters online inference, low-latency pools and delay-tolerant pools must coexist. Multi-objective optimization in cloud-native intelligence is therefore not simply "more metrics." It is a change in the system contract [30].

The fourth pattern is observability becoming part of control rather than a separate operations toolchain. The research agenda paper explicitly argues that learning-based resource management requires fine-grained telemetry and that observability closes the loop. The RCA and anomaly-detection papers reviewed here make the same point from the opposite direction: without service dependencies, multi-source observability, and temporal context, anomaly signals fragment across logs, traces, and metrics. The most consequential implication is architectural. Future cloud-native intelligent systems will increasingly treat observability pipelines as control-plane inputs for scheduling, autoscaling, and remediation, not just dashboards for operators [31].

A fifth pattern is more cautionary: evaluation remains fragmented. DeepServe evaluates on ShareGPT and BurstGPT; ECCOS uses QAServe; CAPS uses public conversation traces plus carbon-intensity data; LarS uses synthetic or simulator-defined VM environments; the MoE scheduler uses its own experimental workload suite; trust-aware federated systems use CIFAR-10, FEMNIST, or CICIDS2017; microservice diagnosis systems use different anomaly datasets and report different metrics. As a result, the field currently has transferability problems. It is difficult to compare absolute gains across papers because the objective vectors, workloads, and feasibility constraints differ. The literature is rich in local improvements, but still weak in benchmark harmonization [32].

Table II condenses the objective spaces that recur across the literature and the metrics typically used to operationalize them. This table is synthesized from the reviewed sources rather than copied from any single paper [33].

Objective family	Typical metrics	Typical decision variables	Representative domains
Performance and QoS	TTFT, TPOT, P95 latency, response time, SLO attainment, success rate, goodput	Routing, batching, prefill/decode placement, engine assignment, autoscaling	LLM serving, cloud-task scheduling, HPC scheduling
Economic efficiency	Cost per request, rental cost, egress cost, average execution cost	GPU type, model size, pool selection, inter-cloud aggregation frequency	Multi-LLM serving, cloud scheduling, multi-cloud FL
Sustainability	Energy consumption, carbon per token/request, power consumption	Carbon-aware placement, delay-tolerant batching, edge allocation fractions	LLM serving, edge computing, IoT and distributed scheduling
Reliability and diagnosis	F1, AUC, MTTR, root-cause ranking quality, false alarms	Feature fusion, graph depth, service attribution, remediation policies	Microservices, AIOps, industrial IoT
Robustness and security	Accuracy under attack, false positive rate, poisoning resistance, queue stability	Trust weights, privacy parameters, attack-mitigation actions, traffic shaping	Federated learning, IoT/IoV defense, agentic systems
Governance and privacy	Differential privacy budget, communication overhead, trust/reputation scores	Aggregation topology, local/edge/cloud update cadence, defense thresholds	Hierarchical FL, multi-cloud learning, privacy-preserving analytics

5. Illustrative experiments

The experiments in this section are author-run explanatory simulations designed to make the literature's trade-offs concrete.

They are not reproductions of any published benchmark. The first experiment borrows the structure of the LLM-serving literature: heterogeneous execution pools, SLO-sensitive requests, cost and carbon signals, and policy search under multiple objectives. The second borrows the structure of the autoscaling literature: workload drift, reactive versus risk-aware control, and the tension among tail latency, cost, and stability. The design principles are inspired by the reviewed works on BOute, ECCOS, CAPS, LarS, risk-constrained scaling, and NSGA-II-style multi-objective search, but the numerical results are original to this review [34].

5.1 Experiment A

The simulated workload contained 450 requests with bursty arrivals, variable prompt and generation lengths, and mixed SLO tiers. Three execution pools represented fast, balanced, and eco-oriented GPU resources with different service rates, cost rates, and time-varying carbon intensities. Each policy decided online which pool should serve each request. The optimization targets were average cost per request, average carbon per request, and SLO-met token goodput. We compared extreme greedy policies, a weighted-sum policy-selection scheme, random-search-derived Pareto selection, and an NSGA-II search over routing-policy parameters.

Table III shows the main results. Two conclusions are immediately visible. First, single-objective greediness is brittle. The latency-first policy achieved near-perfect SLO attainment and low P95 latency, but at the highest cost and carbon. The cost-first policy minimized cost and carbon but collapsed goodput and SLO attainment. Second, Pareto-aware search recovered more useful middle-ground policies. The NSGA-II knee policy improved dramatically over cost-first behavior while keeping cost and carbon far below the latency-first extreme. The highest-performance Pareto policy nearly matched the weighted-sum best-goodput point, but did so at slightly lower cost and carbon. These results mirror the literature's central claim that cloud-native serving is rarely well handled by extreme objectives alone.

Method	SLO-met token goodput	Avg. cost / request	Avg. carbon / request	SLO attainment	P95 latency
Greedy latency-first	122.051	4.052	0.112	0.991	2.989
Greedy cost-first	3.126	3.246	0.048	0.049	226.946
Weighted-sum frontier best-goodput point	122.737	3.527	0.073	0.993	4.508
NSGA-II knee policy	80.858	3.354	0.057	0.776	7.690
NSGA-II highest-goodput policy	122.737	3.517	0.072	0.993	4.635

Table IV compares search-method quality rather than single policies. The approximate hypervolume of the NSGA-II frontier exceeded the weighted-sum selection set, and under a mid-range carbon budget the best NSGA-II policy also delivered higher SLO-met token goodput than the alternatives. The improvement is not enormous, but it is meaningful. It suggests that simple scalarization can miss useful regions of non-convex resource-allocation space, which is precisely the motivation behind explicit multi-objective methods in papers such as BOute and related serving optimizers.

Search method	Returned non-dominated policies	Approx. hypervolume	Best goodput under carbon budget
Weighted-sum selection	15	0.933	103.291
Random-search Pareto selection	134	0.964	105.536
NSGA-II	64	0.970	109.143

The interpretation is straightforward. In a cloud-native serving problem with heterogeneous pools and SLO tiers, the decision maker should not ask for "the cheapest" or "the fastest" policy. The correct question is which Pareto point satisfies the deployment's governance preference. A carbon-constrained academic cluster, a latency-sensitive enterprise copilot, and a public multi-tenant API would rationally choose different operating points even if they run the same models on the same hardware.

5.2 Experiment B

The second simulation considered autoscaling under drifting workload with random bursts. A service cluster had to track changing demand while minimizing tail latency, instance cost, and oscillation. We compared three controllers: a utilization-threshold controller, a latency-only reactive controller, and a risk-aware controller with variance-sensitive forecasting and damped capacity adjustments. The workload and controller structure were chosen to reflect the design themes of risk-constrained elastic scaling and adaptive rate limiting from the reviewed literature [35].

The results in Table V reinforce the argument that stability must be treated as a co-equal objective. The utilization-threshold controller virtually eliminated SLO violations, but at the highest average instance count. The latency-only controller was cheapest in capacity terms, but its P95 latency and oscillation rate were dramatically worse. The risk-aware controller occupied the most attractive middle point: it used substantially fewer instances than the utilization-threshold policy, kept oscillation very low, and held P95 latency close to target. This is consistent with the reviewed literature's claim that learning- or risk-aware control becomes valuable when workload drift makes purely reactive policies unstable or expensive.

Controller	P95 latency	SLO violation rate	Mean instances	Oscillation
Utilization threshold	0.835	0.000	3.938	0.041
Latency-only reactive	6.679	0.260	2.788	1.186
Risk-aware heuristic	1.540	0.045	2.902	0.041

The broader lesson from both experiments is the same. In cloud-native intelligent systems, a controller that looks superior under one metric can be decisively inferior once neighboring

objectives are included. This is why the reviewed literature keeps moving toward constrained optimization, Pareto search, and risk-aware control instead of relying on single-objective heuristics.

6. Conclusion and open questions

The literature on multi-objective optimization in cloud-native intelligent systems has matured enough to reveal a coherent field, but not yet enough to be fully standardized. The most important conclusion of this review is that the field is no longer organized around single-system tricks. It is organized around objective conflict: latency versus cost, cost versus carbon, performance versus fairness, privacy versus communication, robustness versus efficiency, and diagnosis quality versus remediation time. Across LLM serving, cloud scheduling, observability, edge control, and federated analytics, the state of the art now couples predictive models with constrained or learning-based decision layers and closes the loop through telemetry [36].

Several open questions remain. The first is benchmark comparability. The literature still lacks a standard suite that jointly evaluates latency, quality, cost, carbon, and safety across realistic cloud-native workloads. The second is solver safety. LLM-based schedulers and adaptive controllers can generalize impressively, but their own reasoning overhead, calibration drift, and failure modes are still underexplored. The third is cross-layer optimization. Most papers optimize one layer at a time-serving, autoscaling, diagnosis, or federation-yet production systems increasingly need these layers to coordinate. The fourth is governance-aware optimization. Privacy budgets, trust scores, and attack resilience are entering objective functions, but the community still lacks common ways to compare them with cost or QoS in operational terms. Finally, bibliographic volatility remains a nontrivial issue: several 2026 items are preprints, some records are duplicated in the source list, and at least two papers show title-abstract evolution. That makes careful source normalization part of the research task itself [37].

Even with those limitations, the direction of travel is clear. Cloud-native intelligent systems will be governed by multi-objective controllers that are prediction-driven, observability-aware, heterogeneity-conscious, and explicit about safety, sustainability, and trust. The key architectural challenge is not finding one universal optimizer. It is composing the right predictive models, constraints, and control policies so that each deployment can operate on the Pareto surface that matches its real-world priorities.

References

[1] X. Li, X. Liu, Z. Wang, C.-Y. Hsieh, and Y. Liu, "DeepServe: SLO-Aware and Cost-Aware Elastic Scheduling for Serverless Multi-Tenant LLM Inference," Research Square preprint, Apr. 2026, doi: 10.21203/rs.3.rs-9317057/v1.

[2] E. Gelenbe, "Minimizing Delay and Power Consumption at the Edge," Sensors, vol. 25, no. 2, Art. no. 502, 2025.

[3] J. Bergquist, E. Gelenbe, M. Nasereddin, and K. Sigman, "Mitigating Massive Access with Quasi-Deterministic Transmission: Experiments and Stationary Analysis," Performance Evaluation, vol. 170, Art. no. 102512, 2025.

[4] J. Huang, J. Zhan, Q. Wang, J. Jia, and B. Zhang, "Stable Fault Diagnosis Under Data Imbalance via Self-Supervised Learning in Industrial IoT," Preprints.org preprint, 2026.

[5] M. Wang, D. Ren, and H. Wu, "Optimizing LLM Inference Offloading with Hierarchical Scheduling and Dynamic Sparsification," submitted to ICLR, 2026.

[6] Y. Jiang, F. Fu, and E. Yoneki, "BOute: Cost-Efficient LLM Serving with Heterogeneous LLMs and GPUs via Multi-Objective Bayesian Optimization," arXiv:2602.10729, 2026.

[7] Z. Liu, R. Meng, S.-Y. Huang, and Z. Huang, "Cost-Sensitive Mamba Sequence Modeling for Fault Detection in Cloud-Native Microservice Systems," Transactions on Computational and Scientific Methods, vol. 5, no. 12, 2025.

[8] H. Pei, Y. Gu, Y. Sun, Q. Wang, C. Liu, X. Chen, and L. Cheng, "LLM-based Cost-Aware Task Scheduling for Cloud Computing Systems," Journal of Cloud Computing, vol. 14, Art. no. 81, 2025.

[9] P. Jadhav, H. Jin, E. Deelman, and P. Balaprakash, "Evaluating the Efficacy of LLM-Based Reasoning for Multiobjective HPC Job Scheduling," arXiv:2506.02025, 2025.

[10] Z. Zhang, W. Liu, J. Tao, H. Zhu, S. Li, and Y. Xiao, "Unsupervised Anomaly Detection in Cloud-Native Microservices via Cross-Service Temporal Contrastive Learning," Preprints.org preprint, 2025.

[11] K. Mei, W. Xu, S. Lin, and Y. Zhang, "ECCOS: Efficient Capability and Cost Coordinated Scheduling for Multi-LLM Serving," arXiv:2502.20576, 2025.

[12] J. Liao, F. Chang, Y. Xue, T. Xia, Z. Huang, and Y. Wang, "Multi-Objective Scheduling for Large Language Model Inference with Prompt-Level Cost Prediction and SLO Awareness," Preprints.org preprint, 2026.

[13] C. Wang, T. Yuan, C. Hua, L. Chang, X. Yang, and Z. Qiu, "Integrating Large Language Models with Cloud-Native Observability for Automated Root Cause Analysis and Remediation," Preprints.org preprint, 2025.

[14] M. N. Khan, S. Lee, S. S. Lee, M. Shah, I. Ullah, S. Basheer, and A. K. Bashir, "LLM-Enabled Adaptive Scheduling in IoT Sensing for Optimized Network Performance," Scientific Reports, vol. 16, Art. no. 13007, 2026.

[15] Y. Sun, G. Haffari, M. Xu, R. Buyya, and A. N. Toosi, "Multi-Layer Scheduling for MoE-Based LLM Reasoning," arXiv:2602.21626, 2026.

[16] R. Xu, Y. Yang, J. Qiu, H. Cui, Y. Sun, and Z. Li, "TrustGraph-DFL: Byzantine-Resilient Decentralized Federated Learning via Consistency-Weighted Neighborhood Aggregation," Preprints.org preprint, 2026.

[17] Z. Wang, A. Zhu, Y. Wu, K. Wu, Y. Li, and Y. Xue, "Zero-Shot Anomaly Prediction in Distributed Systems via Meta-Learning," in Proc. 5th Int. Conf. Electronic Communication, Computer Science and Technology, 2025, pp. 272-276.

[18] S. Li, B. Chen, Y. Li, Z. Wang, Y. Xue, and C. Xu, "Privacy-Preserving Anomaly Detection in Cloud Services Using Hierarchical Federated Learning with Differential Privacy," Preprints.org preprint, 2026.

[19] W. Huang, R. Wei, J. Kou, H. Zhuang, X. Yan, and W. Huang, "Stabilizing Cloud Elastic Scaling with Risk-Constrained Reinforcement Learning Under Workload Drift," Preprints.org preprint, 2026.

[20] J. Yang, J. Chen, Z. Huang, C. Xu, C. Zhang, and S. Li, "Cost-TrustFL: Cost-Aware Hierarchical Federated Learning with Lightweight Reputation Evaluation across Multi-Cloud," arXiv:2512.20218, 2025.

[21] Z. Huang, S. Li, C. Xu, B. Chen, Y. Xue, and J. Yang, "Structure-Aware Unified Modeling for Root Cause Localization in Microservice Systems Using Multi-Source Observability Data," Preprints.org preprint, 2026.

[22] C. Zhang, H. Zhu, A. Zhu, J. Liao, Y. Xiao, and Z. Zhang, "Deep Learning Approach for Protocol Anomaly Detection Using Status Code Sequences," Preprints.org preprint, 2026.

[23] Y. Tang, Y. Liu, J. Lan, Z. Yan, and E. Gelenbe, "Security of LLM-Based Agents: Attacks, Defenses, and Applications," Information Fusion, vol. 127, Art. no. 103941, 2026.

[24] E. Gelenbe, B. C. Gül, and M. Nakıp, "DISFIDA: Distributed Self-Supervised Federated Intrusion Detection," Internet of Things, vol. 28, Art. no. 101340, 2024.

[25] M. Xu, J. Wu, S. Song, S. N. Srirama, B. Javadi, R. Ranjan, D. N. Jha, S. Wang, W. Tian, H. Xu, L. Li, Z. Mo, S. Ren, T. Kunz, P. Kochovski, V.

Stankovski, K. Ye, C. Xu, and R. Buyya, "Cloud-Native and Distributed Systems for Efficient and Scalable Large Language Models-A Research Agenda," arXiv:2604.17227, 2026.

[26] N. Lyu, Y. Wang, Z. Cheng, Q. Zhang, and F. Chen, "Multi-Objective Adaptive Rate Limiting in Microservices Using Deep Reinforcement Learning," arXiv:2511.03279, 2025.

[27] K. Zeng, Z. Huang, Y. Yang, R. Meng, S.-Y. Huang, and X. Zhang, "TokenFlow: Token-Level GPU Sharing and Adaptive Scheduling for Multi-Model Concurrent LLM Inference," 2026.

[28] E. Gelenbe and M. Nasereddin, "Adaptive Attack Mitigation for IoV Flood Attacks," IEEE Internet of Things Journal, vol. 12, no. 5, pp. 4701-4714, 2025.

[29] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in Proc. 29th ACM Symp. Operating Systems Principles, 2023, pp. 611-626.

[30] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, "DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving," arXiv:2401.09670, 2024.

[31] Y. Fu, L. Xue, Y. Huang, A.-O. Brabete, D. Ustiugov, Y. Patel, and L. Mai, "ServerlessLLM: Low-Latency Serverless Inference for Large Language Models," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation, 2024, pp. 135-153.

[32] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee, "Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation, 2024, pp. 117-134.

[33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, 2002.

[34] A. P. Guerreiro, C. M. Fonseca, and L. Paquete, "The Hypervolume Indicator: Computational Problems and Algorithms," ACM Computing Surveys, vol. 54, no. 6, 2021.