Uncertainty-Aware Resource Orchestration for Agentic AI Systems

Masoud Shokrnezhad
ICTFICIAL Oy, Finland
masoud.shokrnezhad@ictficial.com

Tarik Taleb
Ruhr University Bochum, Germany
tarik.taleb@ruhr-uni-bochum.de

***Abstract*—Agentic artificial intelligence (AI) systems powered by large language models (LLMs) execute as autonomous multi-step pipelines that dynamically invoke tools, integrate observations into memory, and iterate until a task is resolved or a resource budget is exhausted. Unlike classical workflows with fixed graphs, the aggregate compute, bandwidth, and end-to-end (E2E) latency demands of such pipelines are unknown at dispatch time: they emerge at runtime from intermediate observations and stochastic token sampling, yielding heavy-tailed, multi-modal random variables. This structural unpredictability forces infrastructure orchestrators into a fundamental dilemma: under-provisioning risks mid-execution throttling and deadline violations, while over-provisioning wastes scarce edge or cloud capacity. Existing schedulers require a fully known pipeline graph, while edge resource-management methods rely on parametric or stationary demand models, and neither fits the structurally unknown, non-stationary profiles of agentic pipelines. We formalize this setting as the uncertainty-aware agentic orchestration problem, where the goal is to reserve, at dispatch time, compute and bandwidth allocations that cover actual pipeline demands, and to verify that the pipeline can meet its deadline, each with a prescribed probability. We then propose Conformal Reservation for Orchestration (CRO), an online algorithm that translates a sliding window of past execution traces into dispatch-time resource allocations with finite-sample, distribution-free coverage guarantees via conformal prediction. Simulations over heterogeneous task mixes with multi-modal and heavy-tailed profiles show that CRO is the only evaluated method that simultaneously satisfies target probabilistic coverage across all risk levels while, among coverage-feasible methods, attaining the lowest over-reservation, roughly halving the resource waste of the Chebyshev-bound baseline.**

***Index Terms*—agentic AI, conformal prediction, resource provisioning, online resource allocation, large language models**

I. Introduction

Agentic artificial intelligence (AI) systems powered by large language models (LLMs) have rapidly matured into autonomous pipelines [1] that, following a reasoning and acting (ReAct)-style loop [2], iteratively select and invoke tools (web searches, code interpreters, database queries, and sub-agent calls), incorporate each observation into memory, and repeat until the task is resolved or a budget is exhausted [3], [4]. Critically, the full execution trace, including which tools are invoked, how many steps the pipeline runs, the per-step compute and bandwidth demands, and the end-to-end (E2E) latency, is not known at task-dispatch time: it emerges at runtime from intermediate observations and stochastic token sampling, rendering the aggregate resource consumption a heavy-tailed, multi-modal[1], and task-dependent random variable. For infrastructure orchestrators, this creates a fundamental challenge: resources must be reserved *before* execution begins, yet under-provisioning risks mid-execution throttling and deadline violations, while over-provisioning wastes scarce edge or cloud capacity. The central problem we address is therefore how an orchestrator can reserve compute and bandwidth at dispatch time with provable statistical guarantees on resource coverage and deadline satisfaction, without prior knowledge of the pipeline structure or its distribution.

Two complementary strands of related work approach resource provisioning under uncertainty, downstream of service discovery and selection [5]. The first strand, stochastic workflow scheduling for cloud systems, addresses directed acyclic graph (DAG)-structured workloads. Ye *et al.* [6] formulate stochastic multi-workflow scheduling in clouds and derive cost-minimizing decisions under a normal approximation ($\rho = \mathbb{E}[t] + \mathrm{std}(t)$) for task execution times. EPOSS [7] fits per-task Gamma distributions via maximum likelihood estimation (MLE) and enforces probabilistic deadline constraints, scaling to large infrastructure-as-a-service (IaaS) cloud workloads. DEFT [8] trains a mixture-of-experts (MoE) deep reinforcement learning (DRL) policy that encodes workflow DAGs and virtual machine (VM) states for deadline-aware scheduling.

The second strand, resource management for mobile edge computing (MEC) under uncertainty, addresses joint offloading and bandwidth allocation in hierarchical edge networks via Lyapunov optimization and stochastic programming [9], [10], and game-theoretic multi-agent learning for AI-generated content (AIGC) services [11]. At the intersection, E^3-Agent [12] uses an LLM meta-controller to adapt routing for edge generative inference under distribution shift, LLM-based orchestrators combine hierarchical planning with continual reinforcement learning or conversational multi-domain agents [13], [14], and Amayuelas *et al.* [15] study self-allocation of computational tasks in multi-agent LLM systems.

The shared limitation of existing work is a structural mismatch with the agentic setting: classical schedulers assume a known pipeline DAG [6]–[8]; MEC methods rely on parametric or stationary demand models that do not capture multi-modal, heavy-tailed agentic resource profiles [9], [10]; LLM-based orchestrators adapt policies online without dispatch-

[1]Throughout this paper, *multi-modal* denotes a probability distribution with multiple modes (peaks), not multimodal foundation models that jointly process text, vision, and other data modalities.

time coverage guarantees [12]–[14]; and the only distribution-free bounds available, namely the raw empirical quantile [16] and Chebyshev's inequality, trade one deficiency for another: the former lacks finite-sample coverage guarantees, while the latter over-covers by a large margin.

This paper fills the gap with three contributions. **(i)** We formalize the *uncertainty-aware agentic orchestration problem (**P**)*, casting compute, bandwidth, and deadline requirements as probabilistic chance constraints over stochastic pipeline totals. **(ii)** We propose Conformal Resource Orchestrator (CRO), a conformal prediction-based online reservation algorithm that translates a sliding window of past execution traces into dispatch-time allocations with finite-sample, distribution-free coverage guarantees, requiring no parametric assumption and incurring only the cost of sorting the calibration window at each dispatch. **(iii)** Through simulation over heterogeneous task mixes with multi-modal and heavy-tailed resource profiles, we demonstrate that CRO is the only method that simultaneously satisfies probabilistic coverage at all target risk levels and, among coverage-feasible methods, attains the lowest over-reservation, roughly halving the resource waste of the Chebyshev bound.

The remainder of this paper is organized as follows. Section II formulates the system model and problem. Section III presents CRO and its coverage guarantee. Section IV provides simulation results, and Section V concludes.

II. System Model and Problem Formulation

A. Agentic AI System Model

We consider an agentic system $\mathcal{A}$ composed of three tightly coupled components: an LLM backbone $\mathcal{M}$ responsible for reasoning and decision-making, a memory module $\mathcal{H}$ maintaining the evolving interaction context, and a finite tool set $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ of executable capabilities (e.g., web search, code interpreter, database query, sub-agent call) [3], [17]. Upon receiving a task q, the agent executes a *ReAct-style* loop [2]: at each discrete step k, the LLM observes the current context $s_k = (q, \mathcal{H}_k)$ and selects an action $a_k \in \mathcal{T} \cup \{\texttt{stop}\}$. If $a_k = t_j$, tool t_j is invoked with LLM-generated parameters, produces an observation o_k, and o_k is appended to $\mathcal{H}$ before the next step. The loop terminates when the agent selects `stop` or an execution budget is exhausted [4]. The resulting execution trace defines a *pipeline* $\mathcal{G} = (v_1, v_2, \ldots, v_N)$: a sequential chain of N tool invocations where each step v_k consumes the full accumulated context $\mathcal{H}_k$ produced by all preceding steps. Critically, this chain is *not known before execution begins*: the agent's action at each step is conditioned on intermediate observations, so both the sequence of invoked tools and the total length N are determined at runtime. Figure 1 illustrates the full system model, from agent components and the ReAct loop through to resource consumption, uncertainty sources, and the orchestration objective.

B. Resource Consumption Model

Each tool invocation at step k consumes two classes of infrastructure resources.

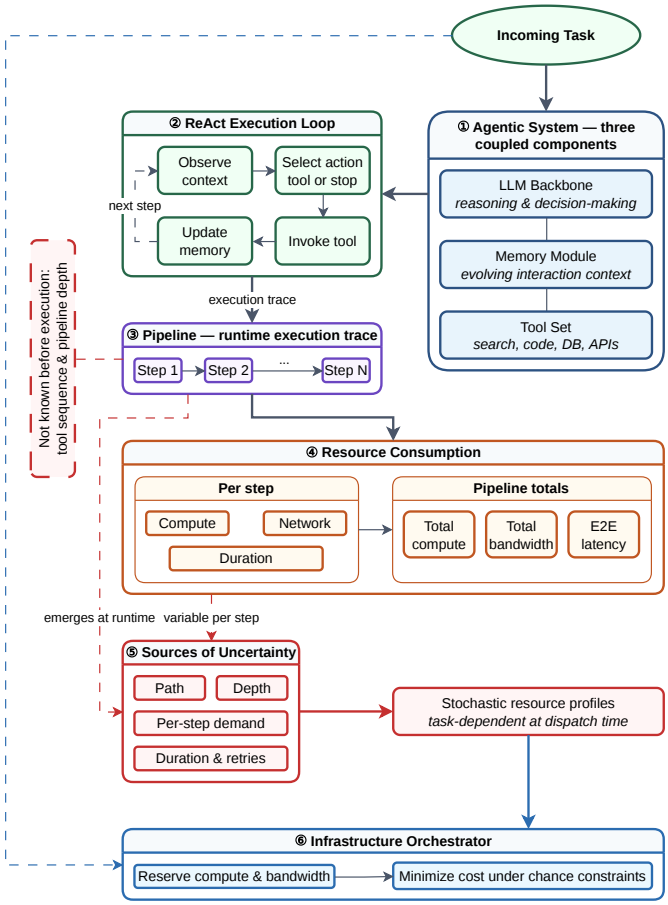

Fig. 1. System model overview. The agentic system $\mathcal{A}$ (LLM backbone, memory, tool set) executes a ReAct loop that produces a runtime pipeline $\mathcal{G} = (v_1, \ldots, v_N)$ whose depth N and tool sequence are unknown at dispatch. Each step incurs compute and network demands that aggregate into total consumption C, B, and latency L. Four structural sources (path, depth, per-step demand, duration) make these quantities stochastic, motivating the uncertainty-aware orchestration problem (3).

Compute. Every step requires LLM inference over the accumulated context $\mathcal{H}_k$ and, if $a_k = t_j$, execution of tool t_j (e.g., running a code interpreter, querying a vector index, or calling a remote LLM). Let $c_k \geq 0$ denote the compute demand at step k, measured in normalized units (e.g., graphics processing unit (GPU)-seconds). The value of c_k depends on both the length $|\mathcal{H}_k|$ (which governs LLM inference cost) and the content of $\mathcal{H}_k$ (which determines the invoked tool and its input parameters, governing tool-side execution cost) [12].

Network. Tools that access remote services (representational state transfer (REST) application programming interfaces (APIs), cloud LLM endpoints, retrieval databases) require network bandwidth. Let $b_k \geq 0$ denote the average bandwidth demand at step k (Mbps). Steps invoking local tools have $b_k \approx 0$; remote tools incur variable b_k depending on payload and response size [3].

The total resource consumption over a pipeline execution is

$$C = \sum_{k=1}^{N} c_k, \qquad B = \sum_{k=1}^{N} b_k, \tag{1}$$

and the E2E pipeline latency is

$$L = \sum_{k=1}^{N} \tau_k, \tag{2}$$

where $\tau_k \geq 0$ is the wall-clock duration of step k.

C. Sources of Uncertainty

Because $\mathcal{G}$ emerges at runtime, the quantities C, B, and L in (1)–(2) are *random variables* from the perspective of the infrastructure orchestrator. Four structural sources drive this uncertainty:

1) **Path uncertainty.** The action a_k selected by the LLM at each step depends on s_k and on stochastic token sampling, making $\mathcal{G}$ unpredictable a priori. A step may branch to any $t_j \in \mathcal{T}$, or terminate, based on the content of intermediate observations.
2) **Depth uncertainty.** The total number of steps N is unknown at dispatch time; it depends on task complexity, the quality of intermediate results, and whether failure-recovery loops (reflection, retry) are triggered [3]. Unlike classical workflow scheduling [6], [8], the chain length is not known at submission time.
3) **Per-step demand uncertainty.** Even when tool t_j is invoked, its compute demand c_k and bandwidth demand b_k vary with the LLM-generated input parameters, output length, and the current context size $|\mathcal{H}_k|$ [12]. Output length is unpredictable, and its effect cascades: a longer observation at step k expands $|\mathcal{H}_{k+1}|$, affecting c_{k+1}.
4) **Duration uncertainty.** The execution time τ_k varies due to remote API round-trip latency, code execution complexity, and LLM inference time that scales with context length [4]. Failures and retries convert reliability events into additional demand spikes.

Together, these sources yield stochastic resource profiles $C \sim p_C(\cdot \mid q)$, $B \sim p_B(\cdot \mid q)$, and $L \sim p_L(\cdot \mid q)$, where the distributions are task-dependent and partially known at task-dispatch time.

D. Problem Formulation

At the moment a task q arrives, the orchestrator must reserve compute capacity ρ_c (GPU-seconds) and network bandwidth ρ_b (Mbps) for the forthcoming pipeline execution. Under-provisioning risks deadline violations and mid-execution throttling; over-provisioning wastes scarce edge or cloud resources. Let $\alpha_c > 0$ and $\alpha_b > 0$ be the unit costs of compute and bandwidth, D the task deadline, and $C_{\max}$, $B_{\max}$ the available capacity at the serving node. The *uncertainty-aware orchestration problem* is:

$$(\mathbf{P}): \min_{\rho_c,\, \rho_b} \quad \alpha_c \rho_c + \alpha_b \rho_b \quad \textit{s.t.} \tag{3a}$$

$$\Pr(L \leq D) \geq 1 - \epsilon, \tag{C1}$$

$$\Pr(C \leq \rho_c) \geq 1 - \delta, \tag{C2}$$

$$\Pr(B \leq \rho_b) \geq 1 - \delta, \tag{C3}$$

$$0 \leq \rho_c \leq C_{\max}, \quad 0 \leq \rho_b \leq B_{\max}. \tag{C4}$$

where $\epsilon \in (0, 1)$ is the maximum tolerated deadline-violation probability and $\delta \in (0, 1)$ is the maximum probability that actual resource demand exceeds the reserved allocation. Constraint (C1) requires that the pipeline completes before deadline D with probability at least $1-\epsilon$, tolerating a small risk of deadline violation rather than enforcing an infeasible worst-case guarantee. Constraints (C2) and (C3) require that the reserved compute and bandwidth allocations cover the actual pipeline demand with probability at least $1 - \delta$; any excess demand beyond ρ_c or ρ_b would cause throttling or queuing at runtime. Constraint (C4) enforces that reservations do not exceed the available node capacity.

Remark 1. *Problem* (3) *is challenging for three reasons: (i) the distributions p_C, p_B, and p_L are not available in closed form and must be estimated from execution traces; (ii) C, B, and L are positively correlated through the shared random depth N, so the constraints cannot be decoupled; and (iii) the reservation decision must be made online at low overhead, ruling out exhaustive Monte Carlo rollouts at runtime.*

III. Proposed Method

A. Overview

The three challenges identified in Remark 1 call for a method that is simultaneously non-parametric, statistically rigorous, and computationally lightweight. We propose the *CRO*, which casts the probabilistic constraints of problem (3) as *conformal prediction intervals* computed from a sliding window of past execution traces. By construction, CRO inherits the finite-sample, distribution-free coverage guarantee of conformal prediction [16], [18]: no assumption on the shape of p_C, p_B, or p_L is needed, and the guarantee holds for any window size $n \geq 1$, as made precise below.

B. Calibration Window and Nonconformity Scores

Let $\mathcal{D}_n = \{(C^{(i)}, B^{(i)}, L^{(i)})\}_{i=1}^{n}$ be the *calibration window*: the n most recently completed pipeline traces, each recording realized total compute, bandwidth, and latency for one task as defined in (1)–(2). The window is capped at a configurable size W (i.e., $n \leq W$), and traces within the window are treated as exchangeable with the arriving task, which holds when the task distribution is approximately stationary over the window horizon. For each trace i, the *nonconformity score* for compute is

$$s_C^{(i)} = C^{(i)} - \hat{C}, \tag{4}$$

where $\hat{C} = \frac{1}{n}\sum_{j=1}^{n} C^{(j)}$ is the rolling sample mean[2]. Scores $s_B^{(i)} = B^{(i)} - \hat{B}$ and $s_L^{(i)} = L^{(i)} - \hat{L}$ are defined analogously. A large positive score flags a pipeline whose actual demand exceeded the typical level, capturing the tail behavior driven by depth and per-step demand uncertainty (sources 2–3 in Section II-C).

[2]Any predictor $\hat{C}$ trained on a disjoint fitting set can replace the rolling mean without affecting the coverage guarantee; the mean is used here for $O(1)$ online updates.

C. Conformal Reservations and Feasibility Check

When a new task q arrives, CRO uses the scores in $\mathcal{D}_n$ to compute reservations (ρ_c, ρ_b) specifically for q before dispatching it.

Resource reservations. For target violation levels δ (resource overflow) and ϵ (deadline miss), let $k_C = \lceil (1-\delta)(n+1) \rceil$ and $k_L = \lceil (1-\epsilon)(n+1) \rceil$. If $k_C = n+1$ or $k_L = n+1$ (i.e., $n < (1-\delta)/\delta$ or $n < (1-\epsilon)/\epsilon$), that order statistic among n scores does not exist: we set the corresponding quantile to $+\infty$ and DISPATCH returns INFEASIBLE [18]. Otherwise

$$\hat{q}_C = \mathrm{Quantile}\Big(\{s_C^{(i)}\}_{i=1}^{n};\ k_C/n \Big), \tag{5}$$

with $\hat{q}_B$ and $\hat{q}_L$ at ranks k_C and k_L. The unclamped reservations are

$$\rho_c = \hat{C} + \hat{q}_C, \tag{6}$$

$$\rho_b = \hat{B} + \hat{q}_B. \tag{7}$$

Feasibility check. E2E latency L cannot be reserved directly, so CRO checks whether its $(1-\epsilon)$ conformal bound fits the deadline:

$$\hat{L} + \hat{q}_L \le D. \tag{8}$$

DISPATCH also returns INFEASIBLE if any quantile is infinite or if $\rho_c > C_{\max}$ or $\rho_b > B_{\max}$. Clamping to $C_{\max}$ would replace $\{C \le \hat{C} + \hat{q}_C\}$ by $\{C \le C_{\max}\}$, which need not hold with probability $1-\delta$. An infeasible task may be deferred, re-routed, or rejected by operator policy.

D. Online Window Update and Complexity

After each pipeline completes, CRO appends (C, B, L) to $\mathcal{D}$ and drops the oldest entry once $|\mathcal{D}|$ exceeds W (circular buffer). Rolling means are updated in $O(1)$; the conformal quantiles require sorting n scores, costing $O(W \log W)$ per dispatch. For practical window sizes ($W \le 10^3$) this is sub-millisecond, making CRO suitable for online orchestration without blocking the dispatch path. Algorithm 1 summarizes the full procedure.

E. Coverage Guarantee

Theorem 1 (Marginal coverage)**.** *Suppose the traces in $\mathcal{D}_n \cup \{(C, B, L)\}$ are exchangeable, where (C, B, L) denotes the random totals of the incoming task q ($(n+1)$-th task). If $k_C \le n$ and $k_L \le n$,*

$$\begin{aligned} \Pr\big(C \le \hat{C} + \hat{q}_C\big) &\ge 1-\delta, \\ \Pr\big(B \le \hat{B} + \hat{q}_B\big) &\ge 1-\delta, \\ \Pr\big(L \le \hat{L} + \hat{q}_L\big) &\ge 1-\epsilon. \end{aligned} \tag{9}$$

Whenever DISPATCH *returns a finite reservation (ρ_c, ρ_b), these bounds are unclamped ($\rho_c = \hat{C} + \hat{q}_C \le C_{\max}$, $\rho_b = \hat{B} + \hat{q}_B \le B_{\max}$) and $\hat{L} + \hat{q}_L \le D$, so $\{L \le \hat{L} + \hat{q}_L\} \subseteq \{L \le D\}$.*

Proof sketch. Let $C_{(k_C)}$ be the k_C-th smallest compute value in the calibration window. Exchangeability makes the incoming value C equally likely to have any of the $n+1$ ranks. Thus,

$$\Pr(C \le C_{(k_C)}) \ge \frac{k_C}{n+1} \ge 1-\delta.$$

Algorithm 1 Conformal Resource Orchestrator (CRO)

procedure DISPATCH($\mathcal{D}_n, D, \epsilon, \delta, C_{\max}, B_{\max}$)
  $n \leftarrow |\mathcal{D}_n|$; $k_C \leftarrow \lceil (1-\delta)(n+1) \rceil$; $k_L \leftarrow \lceil (1-\epsilon)(n+1) \rceil$
  if $k_C > n$ or $k_L > n$ **then**
    return INFEASIBLE ▷ quantile undefined
  end if
  Compute rolling means $\hat{C}, \hat{B}, \hat{L}$ from $\mathcal{D}_n$
  Compute scores $s_C^{(i)}, s_B^{(i)}, s_L^{(i)}$ for all i via (4)
  $\hat{q}_C \leftarrow \mathrm{Quantile}\big(\{s_C^{(i)}\};\ k_C/n\big)$ ▷ cf. (5)
  $\hat{q}_B \leftarrow \mathrm{Quantile}\big(\{s_B^{(i)}\};\ k_C/n\big)$
  $\hat{q}_L \leftarrow \mathrm{Quantile}\big(\{s_L^{(i)}\};\ k_L/n\big)$
  $\rho_c \leftarrow \hat{C} + \hat{q}_C$; $\rho_b \leftarrow \hat{B} + \hat{q}_B$
  if $\hat{L} + \hat{q}_L > D$ or $\rho_c > C_{\max}$ or $\rho_b > B_{\max}$ **then**
    return INFEASIBLE ▷ deadline or capacity
  end if
  return $(\rho_c,\ \rho_b)$
end procedure

procedure ONCOMPLETE($C, B, L, \mathcal{D}, W$)
  Append (C, B, L) to $\mathcal{D}$
  if $|\mathcal{D}| > W$ **then** drop oldest entry
  end if
end procedure

Subtracting the common predictor $\hat{C}$ does not change the order of the calibration values. Hence $\hat{C} + \hat{q}_C = C_{(k_C)}$, yielding the compute bound; the same argument gives $\hat{B} + \hat{q}_B = B_{(k_C)}$. If $k_C = n+1$, the required calibration order statistic is unavailable and DISPATCH returns INFEASIBLE. A finite return uses these unclamped bounds: replacing $\hat{C} + \hat{q}_C$ by $C_{\max}$ could invalidate the coverage claim. Applying the same rank argument to latency gives $\Pr(L \le \hat{L} + \hat{q}_L) \ge 1-\epsilon$. Finally, a finite return satisfies (8), so $\{L \le \hat{L} + \hat{q}_L\} \subseteq \{L \le D\}$. □

Remark 2. *Theorem 1 provides separate (*marginal*) guarantees for compute, bandwidth, and latency, as required by* (C2)–(C3)*. For a finite return, the latency bound also implies* (C1)*. It does not guarantee that compute and latency are simultaneously within their bounds. The union bound gives*

$$\Pr(C \le \rho_c \text{ and } L \le \hat{L} + \hat{q}_L) \ge 1 - \delta - \epsilon.$$

The tighter product $(1-\delta)(1-\epsilon)$ would require independence, which is not expected because both quantities depend on the random pipeline depth N. Using a rank below k_C loses the finite-sample guarantee. We claim no universal nonparametric optimality; among the evaluated methods satisfying (C2)*, CRO yields the least waste in Section IV.*

IV. SIMULATION RESULTS

A. Setup

Synthetic trace model. Because real-world agentic execution logs at scale are not yet publicly available, we construct a synthetic trace generator that faithfully captures all four

TABLE I
SIMULATION PARAMETERS ($j \in \{\text{LT, MD, HV}\}$)

Parameter (symbol)	Light	Medium	Heavy
System constants			
Compute capacity ($C_{\max}$)		200 GPU-s	
Bandwidth capacity ($B_{\max}$)		100 Mbps	
Task deadline (D)		30 s	
Per-type trace model			
Depth (N_j)	$1{+}\text{Poi}(3)$	$1{+}\text{Poi}(6)$	$1{+}\text{NB}(2, \frac{1}{6})$
Compute shape (κ_c^j)	2.0	2.5	3.0
Compute scale (θ_c^j)	0.5	1.0	1.5
Bandwidth shape (κ_b^j)	1.5	2.0	2.5
Bandwidth scale (θ_b^j)	0.3	0.6	0.9
Latency slope (s_j)	0.30	0.50	0.80
Latency noise rate (e_j)	0.20	0.40	0.70
Experiment settings			
Calibration window (n)		10–200 (default 100)	
Risk level (δ)		0.05–0.30	
Deadline tolerance (ϵ)		0.10	
Test tasks / trial		1 000	
Monte Carlo trials		400	

uncertainty sources of Section II-C. Each pipeline belongs to one of three task types (*light*, *medium*, *heavy*) sampled with equal probability. For light and medium tasks the pipeline depth follows $N \sim 1 + \text{Poisson}(\lambda_j)$, matching the standard stochastic workflow assumption [6]. For heavy tasks we use $N \sim 1 + \text{NegBin}(r{=}2,\ p{=}1/6)$, which gives the same mean $\mathbb{E}[N] = 10$ but variance $\text{Var}[N] = 60$ (versus 10 for Poisson), modeling the high variability in pipeline depth caused by reflection and retry loops [3]. Per-step demands are drawn independently as $c_k \sim \Gamma(\kappa_c^j, \theta_c^j)$ and $b_k \sim \Gamma(\kappa_b^j, \theta_b^j)$ (shape and scale), and step latency follows $\tau_k = s_j c_k + \text{Exp}(e_j)$, where s_j is a type-specific slope linking inference time to compute load and e_j is the rate of an independent exponential noise term. Here $j \in \{\text{light, medium, heavy}\}$ indexes the task type; all per-type parameters are listed in Table I. The compound Gamma-NegBin distribution of C for heavy tasks is positively skewed and heavy-tailed (strictly harder than a single Gamma), and the equal-type mixture creates a multi-modal marginal that no single parametric family can approximate well.

Baselines. We compare CRO against four complementary baselines:

(i) **EmpQ** [16]: uses the raw empirical $(1{-}\delta)$-quantile with no conformal correction, and the standard non-conformity quantile before the finite-sample $+1$ adjustment of [18] is applied. It isolates the exact contribution of the conformalization step in CRO.

(ii) **Gaussian** [6]: sets $\rho_c = \hat{C} + \Phi^{-1}(1{-}\delta)\,\hat{\sigma}_C$, the Normal-approximation quantile, corresponding to $\rho = \mathbb{E}[t] + \text{std}(t)$ in Ye et al. [6]. It fails when C is non-Gaussian.

(iii) **Parametric** [7]: fits a Gamma distribution via MLE on the calibration window and uses the theoretical $(1{-}\delta)$-quantile, following the approach of EPOSS [7] adapted online. It fails when the distribution is heavier-tailed than Gamma.

(iv) **Chebyshev**: uses the Cantelli one-sided bound $\rho_c = \hat{C} + \sqrt{(1{-}\delta)/\delta}\,\hat{\sigma}_C$, which is distribution-free and always guarantees coverage, but is far more conservative than CRO.

Metrics. (1) *Empirical coverage* $\hat{p}_C = \Pr(C \le \rho_c)$ and $\hat{p}_B = \Pr(B \le \rho_b)$ on held-out test traces; (2) *normalized over-reservation* $\mathbb{E}[(\rho_c{-}C)^+]/\bar{C}$; (3) *deadline feasibility accuracy*: whether check (8) correctly identifies $\Pr(L{>}D) \le \epsilon$.

B. Results

Coverage calibration. Fig. 2(a)–(b) sweeps $\delta \in \{0.05, \ldots, 0.30\}$ at fixed $n = 100$. CRO tracks the ideal diagonal ($y = 1{-}\delta$) throughout: at $\delta = 0.10$ it achieves $\hat{p}_C = 0.903$, confirming Theorem 1. EmpQ runs nearly parallel to CRO but sits consistently below the diagonal ($\hat{p}_C = 0.894$ at $\delta = 0.10$). The gap is small at $n = 100$ because the conformal correction shifts only one quantile rank; yet it is systematic and grows at smaller n (see below), confirming that the $+1$ adjustment is non-trivial for finite samples. Gaussian exhibits a crossing pattern: it over-covers at $1{-}\delta \le 0.80$ and under-covers at $1{-}\delta \ge 0.85$, reaching $\hat{p}_C = 0.919$ at target 0.95, a violation of constraint (C2) exactly when service level agreements (SLAs) are most stringent. Parametric under-covers monotonically ($1.8\% - 4.3\%$ deficit), with the Gamma MLE fit consistently missing the compound Gamma-NegBin upper tail. Chebyshev always exceeds the diagonal ($\hat{p}_C = 0.973$ at $\delta = 0.10$), proving coverage but at the cost of a normalized over-reservation of 3.58, nearly $2\times$ that of CRO (Fig. 2(c)).

Resource efficiency and window size. Fig. 2(c) shows that among the five compared methods only CRO simultaneously meets the coverage constraint and minimizes waste. At $\delta = 0.10$: EmpQ (1.69), Gaussian (1.67), and Parametric (1.50) all appear cheaper than CRO (1.81), but they achieve this by under-reserving, violating constraint (C2). Chebyshev (waste $= 3.58$) is the only other method that provably meets the constraint, but at $2\times$ the over-reservation of CRO. The 0.14 additional normalized waste of CRO over EmpQ is the finite-sample correction cost. Among methods that meet (C2), CRO has the least waste here; we do not claim optimality over all nonparametric procedures. Fig. 2(d) shows the finite-sample advantage most dramatically. At $n = 10$, CRO already achieves $\hat{p}_C = 0.911$; EmpQ drops to 0.838, a 7.3 percentage-point deficit, because without the conformal correction there is no finite-sample guarantee on the empirical quantile. Gaussian (0.860) and Parametric (0.849) also start well below target and never recover: both plateau permanently below 0.90. Chebyshev, conversely, over-covers at all n ($\hat{p}_C > 0.94$), confirming that its conservatism is not a data-scarcity artifact.

Deadline feasibility check. At $\epsilon = 0.10$, CRO's feasibility test (8) achieves perfect accuracy across all 400 trials: it correctly flags heavy-type pipelines as infeasible despite $\bar{L} < D$, because NegBin-depth tail latency drives $\Pr(L{>}D)$ above 10%. All quantile-based baselines (EmpQ, Gaussian, Parametric, Chebyshev) also achieve perfect feasibility accuracy here, confirming that tail-aware latency reservation is necessary and sufficient for the deadline check, but only CRO simultaneously provides the finite-sample guarantee on C and B with minimal waste. The plotted grid has $k_C, k_L \le n$; the capacity test is inactive except in fewer than 2% of $n{=}10$ trials.

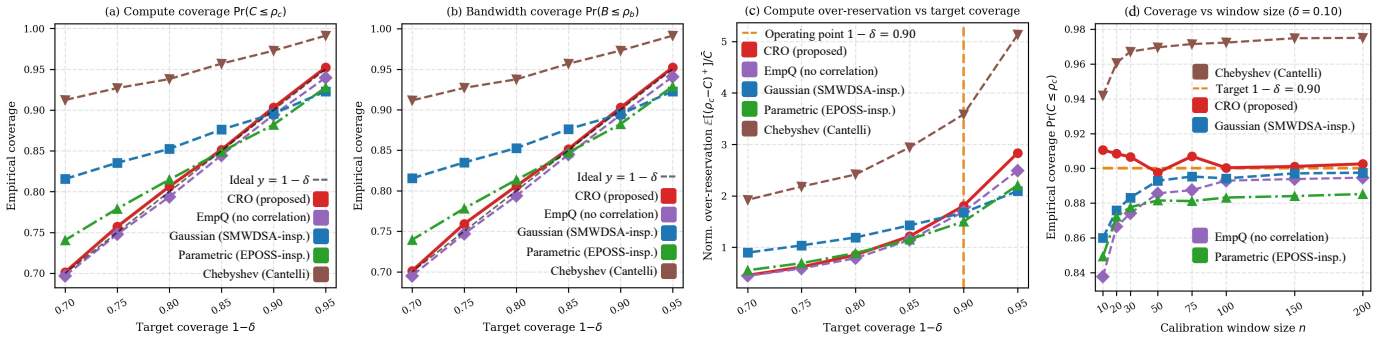

Fig. 2. Simulation results (n=100 unless noted, ϵ=0.10, 400 trials). (a) Compute coverage $\Pr(C \le \rho_c)$ vs. target $1-\delta$; (b) bandwidth coverage $\Pr(B \le \rho_b)$ vs. target; (c) normalized compute over-reservation vs. target coverage; (d) compute coverage vs. calibration window size at δ=0.10. CRO tracks the ideal diagonal in (a)–(b) and meets the 0.90 target in (d) for all $n \ge 10$; EmpQ, Gaussian, and Parametric under-reserve; Chebyshev over-covers at nearly 2× the waste of CRO in (c).

V. Conclusion

This paper addressed the problem of dispatch-time resource reservation for agentic AI pipelines, whose aggregate compute, bandwidth, and latency demands are unknown at dispatch time and emerge at runtime as heavy-tailed, multi-modal random variables. We formalized this as the uncertainty-aware agentic orchestration problem and proposed CRO, an online algorithm grounded in conformal prediction. CRO maintained a sliding window of past execution traces and derived per-task resource reservations with finite-sample, distribution-free coverage guarantees, requiring no parametric assumption on the pipeline distribution and incurring only $O(W \log W)$ overhead per dispatch. Simulations over heterogeneous task mixes confirmed that CRO was the only evaluated method that simultaneously satisfied target probabilistic coverage across all risk levels while, among coverage-feasible methods, attaining the lowest over-reservation, roughly halving the resource waste of the Chebyshev-bound baseline and remaining reliable from as few as ten calibration traces.

Several directions remain open for future work. First, extending CRO to jointly reserve resources for batches of concurrent tasks would reduce over-reservation through cross-task diversification. Second, incorporating distribution-shift detection into the window update would strengthen the exchangeability assumption under non-stationary workloads. Third, validating the approach on real-world agentic logs and integrating it with production orchestration platforms such as Kubernetes, as well as with LLM-based network orchestrators [13], [14], represent important steps toward practical deployment. Fourth, coupling CRO with GenAI-driven service discovery [5] and with key-value objectives extracted from service descriptions [1] would close the loop from service intent to coverage-guaranteed allocation.

Acknowledgment

This work was, in part, supported by BMFTR, Germany (6GEM+, Grant 16KIS2411), and the EU Horizon Europe programme (6G-Path, Grant 101139172).

References

[1] M. Shokrnezhad *et al.*, "KPI2KVI: A Multi Agent Workflow for Calculating Key Value Indicators from Service Descriptions," in *2026 Global Information Infrastructure and Networking Symposium (GIIS)*, Apr. 2026, pp. 1–6.
[2] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," Mar. 2023, arXiv:2210.03629 [cs.CL].
[3] A. Plaat *et al.*, "Agentic Large Language Models, a Survey," *Journal of Artificial Intelligence Research*, vol. 84, Dec. 2025.
[4] C. Yu *et al.*, "A Survey on Agent Workflow – Status and Future," in *2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, May 2025, pp. 770–781, iSSN: 2769-3554.
[5] M. Farhoudi *et al.*, "Service Registration, Indexing, Discovery, and Selection: An Architectural Survey Toward a GenAI-Driven Future," *IEEE Access*, vol. 13, pp. 209 680–209 722, Dec. 2025.
[6] L. Ye *et al.*, "Dynamic Scheduling Stochastic Multiworkflows With Deadline Constraints in Clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 4, pp. 2594–2606, Oct. 2023.
[7] G. R. Russo *et al.*, "Efficient Probabilistic Workflow Scheduling for IaaS Clouds," Dec. 2024, arXiv:2412.06073 [cs.DC].
[8] Y. Shen *et al.*, "Deft Scheduling of Dynamic Cloud Workflows with Varying Deadlines via Mixture-of-Experts," Jun. 2026, arXiv:2606.01162 [cs.AI].
[9] S. Xia *et al.*, "Distributed Computing and Networking Coordination for Task Offloading Under Uncertainties," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5280–5294, May 2024.
[10] A. Du *et al.*, "Online Service Placement, Task Scheduling, and Resource Allocation in Hierarchical Collaborative MEC Systems," *IEEE Transactions on Services Computing*, vol. 18, no. 2, pp. 983–997, Mar. 2025.
[11] H. Liu *et al.*, "Joint Communication and Computation Scheduling for MEC-enabled AIGC Services: A Game-Theoretic Stochastic Learning Approach," *IEEE Internet of Things Journal*, pp. 1–1, 2026.
[12] R. Bao *et al.*, "E^3-Agent: An Executable and Evolving Agent for Resource Management of Edge Generative Inference," May 2026, arXiv:2605.27428 [cs.LG].
[13] M. Shokrnezhad *et al.*, "An Autonomous Network Orchestration Framework Integrating Large Language Models with Continual Reinforcement Learning," *IEEE Communications Magazine*, vol. 63, no. 8, pp. 78–84, Aug. 2025.
[14] M. Shokrnezhad *et al.*, "Conversational Orchestration for Organic 6G," *IEEE Network*, pp. 1–8, 2026.
[15] A. Amayuelas *et al.*, "Self-Resource Allocation in Multi-Agent LLM Systems," Apr. 2025, arXiv:2504.02051 [cs.MA].
[16] A. N. Angelopoulos *et al.*, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *Foundations and Trends in Machine Learning*, vol. 16, no. 4, pp. 494–591, 2023.
[17] T. Schick *et al.*, "Toolformer: Language Models Can Teach Themselves to Use Tools," *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 539–68 551, Dec. 2023.
[18] V. Vovk *et al.*, *Algorithmic Learning in a Random World*. New York: Springer, 2005.