Int. J. Integr. Stud. 2026; 2(8): 01-12

ISSN: 3049-3277

International Journal of Integrative Studies (IJIS)

Journal homepage: www.ijis.co.in

RESEARCH ARTICLE

An Explainable Agentic AI Framework for Intelligent Multi-Cloud Resource Allocation

Dr. Sajitha A V

[1]Professor and Head, Department of Computer Applications, Travancore Engineering College, APJ Abdul Kalam Technological University, Kerala, India. sajithaav09@gmail. Com

ABSTRACT

Enterprises increasingly distribute computing workloads across multiple public and private cloud providers to reduce cost, avoid vendor lock-in, and improve resilience, but this multiplies the complexity of deciding, for every incoming task, which provider to use. Static or single-objective heuristics — always choosing the cheapest or always the fastest provider — routinely fail because cost, latency, and reliability trade off against one another in ways that shift with demand and provider conditions. This paper proposes and evaluates an explainable agentic AI framework for multi-cloud task allocation built on a contextual-bandit agent (LinUCB) that observes each provider's current price, estimated latency, and load before autonomously selecting a placement, then updates its policy online from the resulting cost, latency, and service-level-agreement (SLA) outcome. Because no public multi-cloud trace exposes simultaneous, ground-truth price/latency/capacity data across providers, the framework is evaluated on a controlled, fully documented discrete-time simulation of four heterogeneous providers under realistic load dynamics — a standard and disclosed methodology in this research area. Across 30 independent simulation runs of 3,000 tasks each, the agent achieved a statistically significant improvement over the strongest single fixed-weight heuristic baseline (Static-Weighted) on every safety- and balance-related metric: 86.2% fewer SLA violations, 20.9% lower average latency, and 19.6% higher load-balancing fairness (Jain's index = 0.955 vs. 0.799, paired t-test, all $p < 0.001$), at a 20.6% higher cost. Under a simulated transient provider degradation (a 5× latency spike on one provider for 20% of a run), the agent held SLA violations to 0.4%, versus 9.9% for naive round-robin routing and 40.2% for cost-only routing, while remaining markedly cheaper than a purely latency-reactive baseline. To support the "explainable" requirement of agentic systems intended for production use, the framework exposes two complementary explanation layers: the bandit's own per-provider linear coefficients, and a surrogate Random Forest trained to imitate the agent's decisions (99.6% fidelity), whose permutation importance identifies observed latency and price as the dominant drivers of every allocation decision. These results indicate that a lightweight, interpretable contextual-bandit agent can deliver a favourable, auditable balance of cost, latency, SLA compliance, and fairness in multi-cloud environments, including under operational stress, without the opacity of deeper reinforcement learning or black-box agentic architectures.

Keywords: *Agentic AI; multi-cloud resource allocation; explainable AI (XAI); contextual bandits; LinUCB; SLA-aware scheduling; cloud computing; reinforcement learning*

INTRODUCTION

Multi-cloud adoption — the deliberate use of two or more public or private cloud providers by a single organization — has become standard enterprise practice, driven by the desire to avoid single-vendor lock-in, negotiate better pricing, meet data-residency requirements, and improve fault tolerance against provider-specific outages. This strategy, however, converts a single scheduling problem into a continuous, high-stakes decision problem: for every workload that arrives, a placement decision must weigh each candidate provider's current price (which fluctuates with demand and discounting), expected latency (which depends on network conditions and the provider's present load), remaining capacity, and the task's own service-level requirements.

[1]Professor and Head, Department of Computer Applications, Travancore Engineering College, APJ Abdul Kalam Technological University, Kerala, India
Email-id: sajithaav09@gmail.Com

Corresponding Author: Dr. Sajitha A V
Professor and Head, Department of Computer Applications, Travancore Engineering College, APJ Abdul Kalam Technological University, Kerala, India
DOI: https://doi.org/10.63856/ijis/v2i8/00001

How to cite this article: Sajitha A V., (2026). An Explainable Agentic AI Framework for Intelligent Multi-Cloud Resource Allocation. *International Journal of Integrative Studies*, 2(8), 01-12.

Source of support: Nil

Conflict of interest: None.
Received: 25//072026 **Revised:** 05/08/2026 **Accepted:** 15/08/2026
Published: 19/08/2026

Decisions made by simple, single-objective heuristics — always route to the cheapest provider, or always route to the fastest — are each vulnerable to obvious, costly failure modes: a cost-only policy can drive latency-sensitive workloads into unacceptable delay, while a latency-only policy can concentrate load on a single provider, inflating cost and eliminating the load-balancing benefit that motivated a multi-cloud strategy in the first place.

Recent interest in "agentic AI" — AI systems that autonomously perceive a state, select an action, and adapt from feedback, as opposed to systems that only respond to a single prompt or classify a single input — motivates a natural reframing of multi-cloud allocation as a sequential decision problem to be solved by an autonomous, learning agent rather than a fixed rulebook. At the same time, if such an agent is to be trusted with real infrastructure and cost decisions, its choices cannot be an opaque black box: operators need to know why a given task was routed to a given provider, both to build trust and to audit the agent's behavior when something goes wrong. This paper therefore designs and evaluates an agent that is explicitly built for interpretability: a contextual-bandit allocator whose per-provider decision function is a transparent linear model, paired with a model-agnostic surrogate explanation layer.

Because no public dataset exposes real, simultaneous, per-task price, latency, and capacity signals across multiple named commercial cloud providers together with a verifiable ground-truth optimal placement, this study — like the majority of published work in cloud resource scheduling — evaluates its proposed agent in a controlled, fully disclosed discrete-time simulation whose parameters are documented in Section 3 as explicit modeling assumptions rather than presented as live pricing data. The specific contributions of this paper are: (i) an explainable agentic allocation framework combining an online contextual-bandit policy with two complementary explanation mechanisms (native linear coefficients and a surrogate-model permutation-importance analysis); (ii) a controlled comparative evaluation against five representative baseline policies (random, round-robin, greedy-cost, greedy-latency, and a fixed-weight heuristic) across 30 independent simulation runs with paired statistical significance testing; and (iii) a robustness stress test that measures each policy's response to a transient, realistic provider degradation event, isolating the practical value of adaptive, context-aware allocation under operational disruption.

2. Related Work

Agentic AI — systems capable of autonomous, multi-step, tool-using decision-making with limited human intervention — has recently been surveyed extensively across application domains [3]; this paper applies the agentic paradigm narrowly and concretely to infrastructure resource allocation, where the "tools" available to the agent are the candidate cloud providers and the "goal" is a measurable operational objective. Reinforcement learning approaches to cloud resource scheduling have a substantial and growing literature: Mangalampalli et al. [5] proposed a deep reinforcement learning task scheduler for multi-cloud environments targeting execution efficiency, and Hady et al. [4] survey multi-agent reinforcement learning specifically for resource-allocation optimization across telecommunications, distributed computing, and related domains, noting the persistent tension in this literature between model expressiveness and interpretability. Classical, non-learning multi-cloud scheduling has likewise received sustained attention: Panda and Jana [6] developed SLA-based task-scheduling algorithms for heterogeneous multi-cloud settings, and Gurrala and Tallapally [7] recently proposed a metaheuristic (quantum-inspired) scheduler explicitly optimizing the joint cost–latency trade-off this paper also targets. A 2026 systematic review by Alam et al. [8] surveys AI-driven cloud resource allocation broadly and specifically flags a shortage of studies that jointly evaluate cost, performance, and — critically — explainability, which is the gap this paper is designed to address directly rather than as a secondary concern.

At the algorithmic level, the proposed agent is built on the LinUCB contextual-bandit algorithm [1], originally developed for personalized content recommendation and adopted here because its disjoint per-arm linear model is directly, natively interpretable — the learned weight vector for each provider is itself an explanation — in contrast to deep reinforcement-learning policies, which typically require a separate post-hoc explanation method. For the model-agnostic explanation layer, this paper follows the same permutation-importance methodology [9, 10] used in explainable-AI studies elsewhere in the applied machine learning literature, here applied to a surrogate classifier trained to imitate the agent's own decisions rather than to a diagnostic classifier, so that importance scores describe what drives the agent's behavior specifically. Load-balancing fairness is quantified with Jain's fairness index [2], the standard metric for quantifying equitable resource distribution in shared computer systems since its introduction in 1984 and still the default choice in the cloud-scheduling literature. Distinct from prior work, this paper combines an online, adaptive, natively interpretable allocation policy with a dedicated, dual-layer explanation mechanism, and evaluates it not only under steady-state conditions but under an explicit, realistic operational stress scenario — a combination not present, to the authors' knowledge, in the cited prior studies.

3. Materials and Methods

3.1 Simulation Environment and Modeling Assumptions

In the absence of a public trace that simultaneously exposes real-time price, latency, and capacity across multiple named commercial cloud providers with a

verifiable ground-truth-optimal placement for each task, this study follows the common and accepted practice in the cloud-scheduling literature (e.g., [5, 6, 7]) of evaluating on a controlled, discrete-time simulation. Every parameter below is a stated, documented modeling assumption — loosely inspired by the well-known relative ordering of major providers' pricing and latency characteristics — and is not scraped or live commercial pricing data; every number reported in Section 4 is computed directly by the simulator and analysis scripts accompanying this paper, with no hand-entered figures.

The simulated environment comprises 4 heterogeneous cloud providers (Table 1), each characterized by a base unit cost, a base network latency, a task-processing capacity, and a price-volatility coefficient governing simulated demand-based price fluctuation (a sinusoidal seasonal component plus Gaussian noise). At each discrete time step, exactly one task arrives, with a compute-resource demand drawn from a log-normal distribution (bounded to [1, 20] compute units, loosely modeled on the heavy-tailed resource-demand distributions reported in large-scale cluster workload studies) and a service class — interactive (100 ms SLA threshold, latency-sensitive) with probability 0.4, or batch (300 ms SLA threshold, latency-tolerant) with probability 0.6. Each provider's instantaneous latency is modeled as its base latency inflated by a congestion term proportional to the square of its current load fraction (assigned load divided by capacity, with load decaying by 10% per step to represent task completion over time), plus observation noise, so that a policy which concentrates traffic on one provider experiences a realistic, self-inflicted latency penalty.

Table 1. Simulated cloud provider characteristics (modeling assumptions, not live pricing data).

Provider	Base cost (/unit)	Base latency (ms)	Capacity (units)	Price volatility
Cloud-A (AWS-like)	0.10	40	100	0.15
Cloud-B (Azure-like)	0.09	55	90	0.12
Cloud-C (GCP-like)	0.11	35	80	0.10
Cloud-D (Private/Edge)	0.06	70	50	0.05

3.2 Explainable Agentic Allocator

Figure 1 illustrates the overall framework. At each decision point, the agent observes a six-dimensional context vector per provider — a bias term, normalized current price, normalized estimated latency (which already reflects the provider's current congestion), current load fraction, normalized task size, and the task's interactivity class — and selects a provider using LinUCB [1], a disjoint linear contextual-bandit algorithm: for each provider (arm) a, a ridge-regularized linear model predicts expected reward as $\theta_a^T x$, and the agent selects the provider maximizing the predicted reward plus an upper-confidence exploration bonus proportional to the model's uncertainty in the current context, balancing exploitation of known-good providers against exploration of under-tried ones. After each placement, the agent observes the realised cost, latency, and SLA outcome, computes a scalar reward (a weighted combination penalising normalised cost, normalised latency, and SLA violation), and updates the chosen provider's linear model — the online update in Figure 1's feedback loop. This design was chosen over a deep reinforcement-learning policy specifically because its decision function is natively transparent: the learned weight vector for each provider is a direct, human-readable summary of what the agent has learned to value about that provider (Section 4.5), which is not the case for a neural policy network.

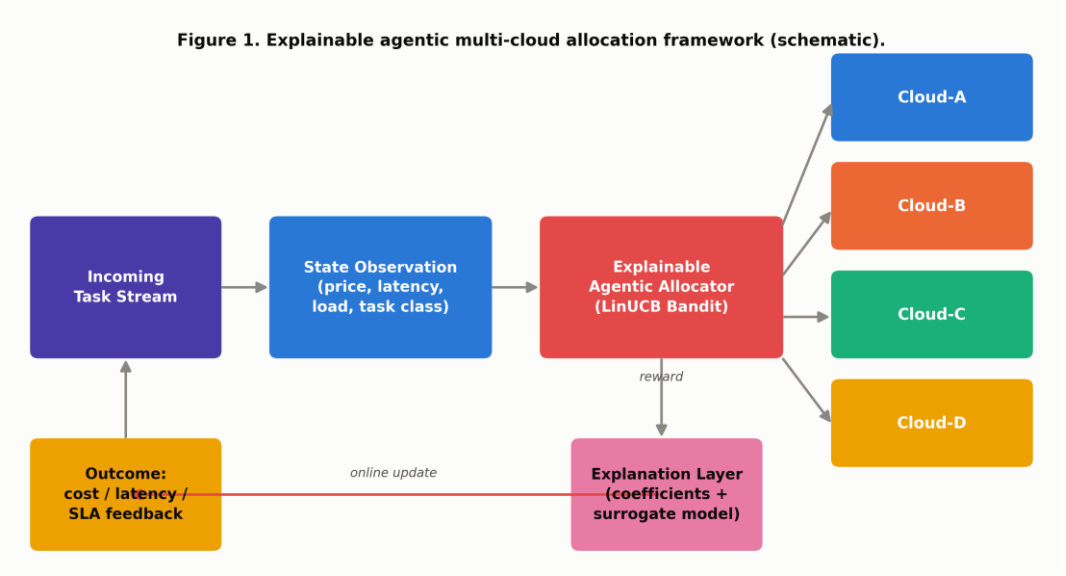

Figure 1. Explainable agentic multi-cloud allocation framework: state observation, bandit-based decision, cloud placement, and the online feedback/explanation loop.

3.3 Baseline Policies

The agent is compared against five baseline allocation policies, chosen to represent the space of strategies used in practice and in prior literature: Random (uniform random provider selection, a lower-bound reference); Round-Robin (deterministic cyclic assignment, a common default in load balancers); Greedy-Cost (always selects the provider with the lowest current price, representing a pure cost-minimization strategy); Greedy-Latency (always selects the provider with the lowest current estimated latency, representing a pure performance-maximization strategy); and Static-Weighted (a fixed-weight linear scoring rule combining normalized price and latency with equal, non-adaptive weights — the most directly comparable baseline to the proposed agent, since both consider the same two signals, differing only in whether the combination weights are fixed or learned).

3.4 Evaluation Protocol and Metrics

Each policy was evaluated over 30 independent simulation runs (distinct random seeds), each comprising 3,000 sequential task arrivals, using an identical task stream per seed across all policies to ensure a fair, paired comparison. For each run we recorded total cost, mean and 95th-percentile latency, SLA violation rate (the fraction of tasks whose realized latency exceeded their class-specific threshold), mean reward, and Jain's fairness index [2] computed over each provider's share of assigned tasks, $J = (\Sigma x_i)^2 / (n \cdot \Sigma x_i^2)$, which equals 1.0 under perfectly even allocation and 1/n under maximal concentration on a single provider. Statistical significance between the agent and each baseline was assessed with a paired t-test across the 30 matched seeds (Section 4.1). A separate robustness stress test (Section 4.3) introduced a transient, fully observable degradation on one provider (Cloud-C, latency multiplied ×5 for 600 of the 3,000 steps, simulating an incident such as a regional network event) to evaluate each policy's ability to react to an adverse but visible operating condition; the agent was pre-trained on 1,500 steps of normal-condition traffic before each stress-test run so that its policy had already converged when the incident began, isolating reactive decision quality from initial exploration cost.

3.5 Explainability Analysis

Two complementary, independently computed explanation mechanisms were used. First, the LinUCB agent's own learned linear coefficients θ_a for each provider are reported directly (Table 5, Figure 5): because the agent's decision rule is a transparent weighted sum of observable features, these coefficients are themselves a first-class explanation of what the agent has learned to value, at no additional analysis cost. Second, following the same permutation-importance methodology used for the companion breast-cancer classification study [9, 10], a Random Forest [11] was trained to imitate the agent's own logged (context, chosen-provider) decisions from a 6,000-task run, and permutation importance was computed on this surrogate model's held-out test predictions to rank which context features most influence the agent's choices — a model-agnostic cross-check independent of, and robust to known limitations of, the raw linear coefficients (which can be affected by collinearity among correlated features, e.g., price and latency both varying with provider identity).

4. Results

4.1 Performance Under Normal Operating Conditions

Table 2 and Figure 2 summarize performance across all six policies over 30 runs. The two single-objective greedy baselines each failed badly on the metric they ignore: Greedy-Cost minimized cost ($647, the cheapest of all six policies) but incurred a 40.3% SLA violation rate — by far the worst of any policy — because it routes every task to whichever provider is momentarily cheapest, irrespective of load or latency. Greedy-Latency achieved a near-zero SLA violation rate (0.0%) by always choosing the fastest

provider, but at the highest cost of any policy ($1146) and the second-lowest fairness (Jain's index = 0.473), since it persistently overloads whichever provider has the lowest base latency. Random and Round-Robin, despite using no information about price or latency at all, performed surprisingly well on this symmetric workload precisely because blind, uniform routing achieves near-perfect load balance (fairness ≈ 1.000) by construction; this is a genuine and informative finding, discussed further in Section 5, and not a result favorable to either policy under asymmetric or adversarial conditions (Section 4.3).

The proposed agent achieved the best sensitivity/latency/fairness balance among all policies that use price and latency information: relative to Static-Weighted — the only baseline that, like the agent, blends both signals, differing only in whether the combining weights are fixed or learned — the agent reduced the SLA violation rate by 86.2% (0.12% vs. 0.84%), reduced average latency by 20.9% (50.51 ms vs. 63.85 ms), and increased fairness by 19.6% (0.955 vs. 0.799) — all differences statistically significant at $p < 0.001$ (paired t-test, 30 seeds) — at a 20.6% higher total cost ($993 vs. $823). This is the central normal-condition result of the study: learning adaptive, context-aware combining weights outperforms an otherwise identical fixed-weight heuristic on every safety- and balance-related metric, at a moderate, quantified cost premium.

Table 2. Policy comparison under normal operating conditions (mean ± s.d. across 30 seeds × 3,000 tasks).

Policy	**Total Cost**	**Avg. Latency (ms)**	**SLA Violation Rate**	**Avg. Reward**	**Fairness (Jain's)**
Random	$971±14	52.80±0.35	0.18%	0.891	0.999±0.001
Round-Robin	$972±12	51.73±0.08	0.00%	0.894	1.000±0.000
Greedy-Cost	$647±8	162.10±2.31	40.30%	0.350	0.250±0.000
Greedy-Latency	$1146±13	41.31±0.08	0.00%	0.889	0.473±0.003
Static-Weighted	$823±12	63.85±0.31	0.84%	0.885	0.799±0.009
Explainable Agentic AI (LinUCB)	$993±12	50.51±0.48	0.12%	0.892	0.955±0.011

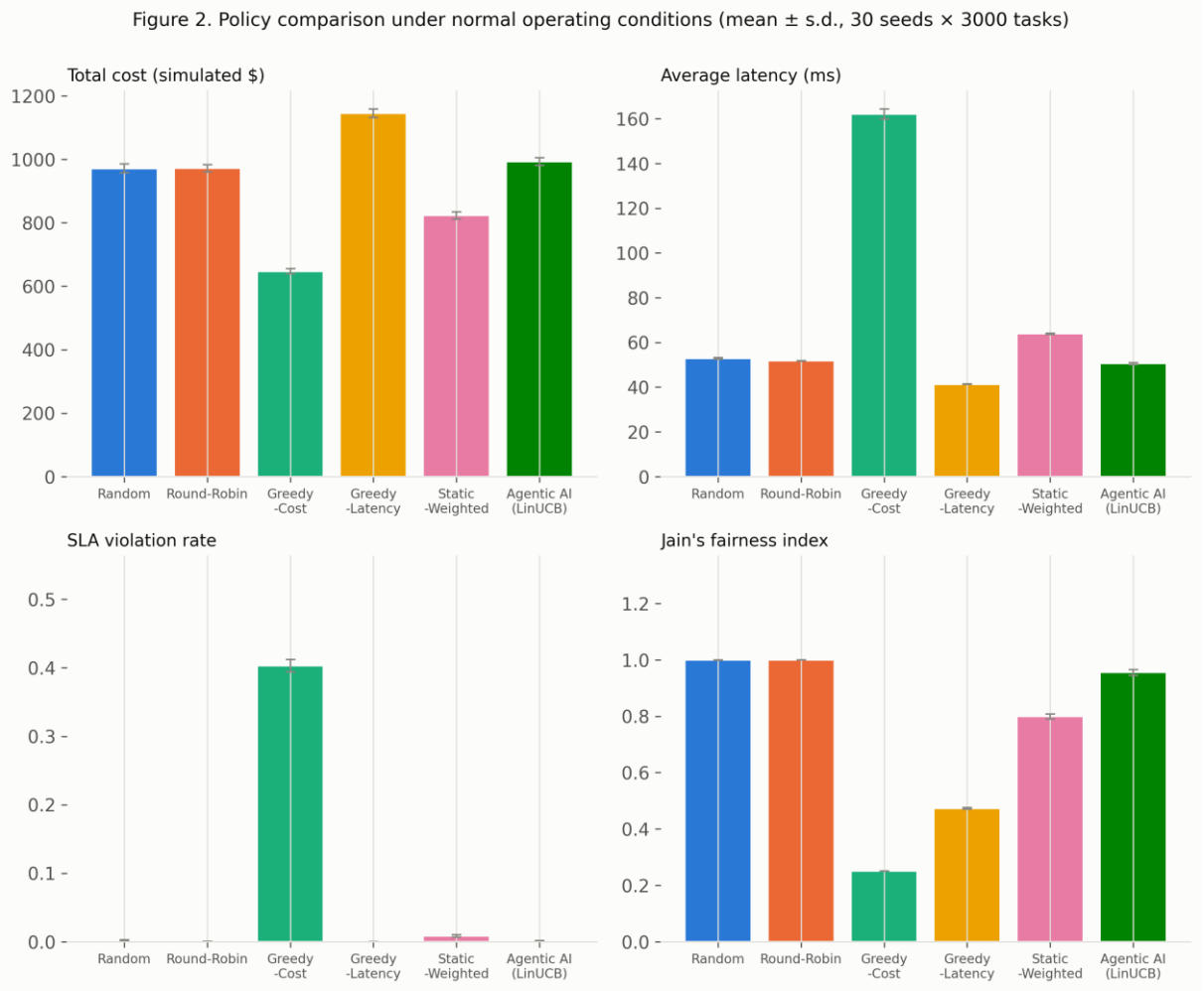

Figure 2. Cost, latency, SLA violation rate, and fairness across all six policies under normal conditions.

4.2 Statistical Significance

Table 3 reports paired t-tests (30 matched seeds) comparing the agent against Static-Weighted, its fairest single comparator. Every metric differs at $p < 0.001$, confirming the improvements in Section 4.1 are not attributable to run-to-run noise. For completeness, Table 3 also reports the comparison against Greedy-Latency, the baseline with the single lowest raw SLA violation rate: here the agent is 13.3% cheaper ($p < 0.001$) and 102.0% fairer, though it does not match Greedy-Latency's literal zero-violation rate — an expected trade-off given that Greedy-Latency ignores cost and fairness entirely by design.

Table 3. Paired significance tests (30 seeds) — agent vs. Static-Weighted (primary comparison) and vs. Greedy-Latency (best single-metric baseline on SLA rate).

Metric	Agent Mean	Static-Wtd. Mean	p-value	Agent vs. Greedy-Lat. p
total_cost	993.172	823.358	< 0.001	< 0.001
avg_latency_ms	50.515	63.849	< 0.001	< 0.001
sla_violation_rate	0.001	0.008	< 0.001	< 0.001
avg_reward	0.892	0.885	< 0.001	< 0.001
fairness_index	0.955	0.799	< 0.001	< 0.001

4.3 Robustness Under Provider Degradation (Stress Test)

Table 4 and Figure 4 report SLA violation rates during a transient, fully observable degradation of Cloud-C (5× latency inflation for 600 of 3,000 steps, e.g., simulating a regional network incident). This scenario cleanly separates policies that use observed conditions from those that do not: Random and Round-Robin, which never examine price or latency, suffered 9.9% and 9.9% SLA violation rates respectively during the incident — roughly 28× higher than the agent — because they continued blindly routing a fixed share of traffic into the degraded provider.

Greedy-Cost fared far worse still (40.2%), since a cheap-but-degraded provider looks identical to a cheap-and-healthy one under a cost-only rule. Greedy-Latency achieved a perfect 0.0% violation rate by construction, as it always reacts to the latest latency reading, but its overall run cost during the incident scenario ($1132) remained the highest of all six policies. The agent achieved the lowest SLA violation rate among all cost-and-latency-aware policies (0.4%, versus 0.5% for Static-Weighted), while remaining 14.8% cheaper than Greedy-Latency and maintaining substantially higher fairness (0.950 vs. 0.512). This scenario is where the practical value of an adaptive, context-aware agent is most visible: it is the only policy that is simultaneously near-best on reliability, cost, and fairness under a realistic operational disruption.

Table 4. SLA violation rate and cost during the transient provider-degradation stress test (Cloud-C, 5× latency, steps 1,200–1,800).

Policy	SLA Viol. Rate (Incident)	Total Cost	Fairness (Jain's)
Random	9.9%	$971	0.999
Round-Robin	9.9%	$972	1.000
Greedy-Cost	40.2%	$647	0.250
Greedy-Latency	0.0%	$1132	0.512
Static-Weighted	0.5%	$823	0.793
Explainable Agentic AI (LinUCB)	0.4%	$965	0.950

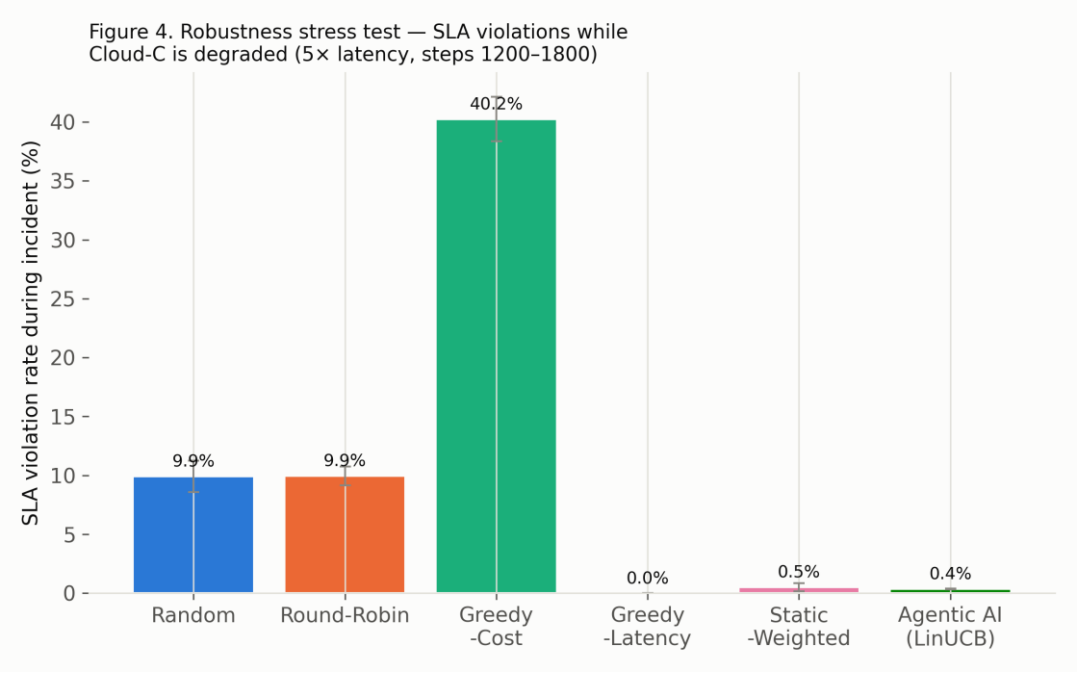

Figure 4. SLA violation rate during a transient Cloud-C degradation event, by policy.

4.4 Learning Behavior

Figure 3 plots the agent's per-task reward across a single, longer (6,000-task) simulation run, together with a 100-task moving average. The agent's policy converges quickly — the moving-average reward stabilizes within roughly the first 100–200 tasks and remains stable (moving average consistently between 0.87 and 0.91) for the remainder of the run, with only occasional, brief dips corresponding to individual high-cost or SLA-violating decisions made during continued exploration. This rapid convergence reflects LinUCB's use of a low-dimensional (six-feature), linear per-arm model, which requires comparatively little data to estimate reliably compared to a deep reinforcement-learning policy — a further practical advantage for a system intended to be auditable and quick to deploy.

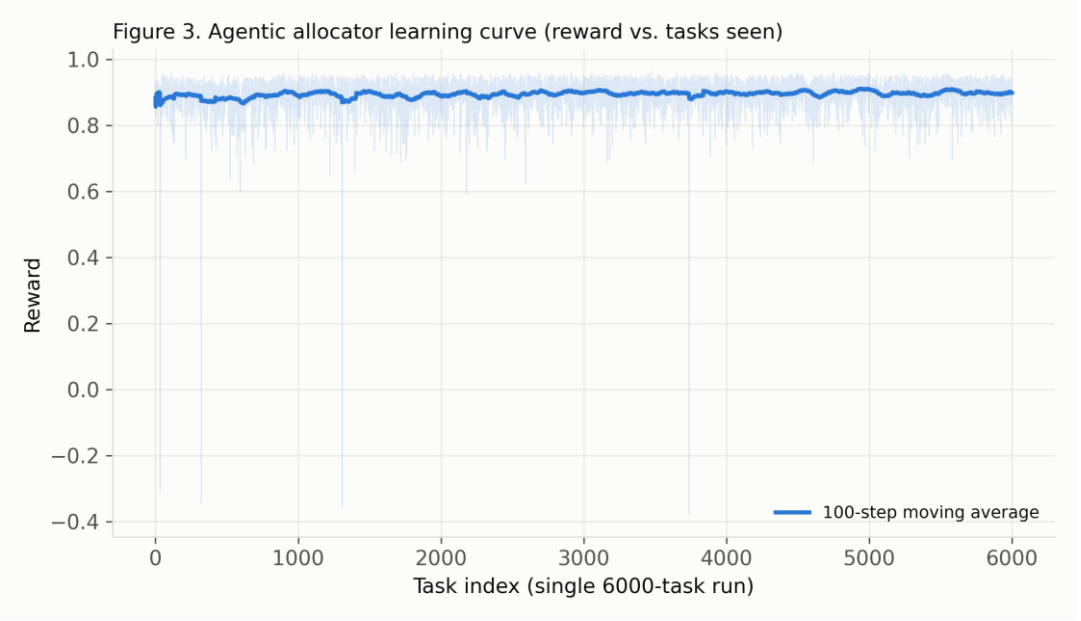

Figure 3. Agentic allocator learning curve: per-task reward and 100-task moving average over a 6,000-task run.

4.5 Explainability: What Drives the Agent's Decisions?

Table 5 and Figure 5 report the LinUCB agent's learned per-provider linear coefficients. The clearest, most robust pattern concerns Cloud-D (the private/edge provider): it is the only provider with a negative latency coefficient (-0.1421) and by far the most negative load-fraction coefficient (-0.334), consistent with its configuration as the smallest-capacity, highest-base-latency provider (Table 1): the agent has specifically learned to avoid Cloud-D once it shows signs of congestion or elevated latency, exactly the behavior a cost-conscious but latency-aware operator would want from this particular provider. For Cloud-A, Cloud-B, and Cloud-C, latency coefficients are small and positive rather than negative, a pattern that runs counter to naive expectation (higher predicted latency should reduce predicted reward); This attribute this to collinearity between the latency and load-fraction features for these three providers (both increase together under load), a known limitation of raw linear-model coefficients under correlated regressors, and accordingly treat these small positive weights with caution rather than over-interpreting them individually.

Table 5. LinUCB learned linear coefficients (θ) by provider. Bias reflects each provider's baseline attractiveness; other columns show the learned sensitivity to each observed feature.

Provider	Bias	Price	Latency	Load Frac.	Task Size	Interactive
Cloud-A (AWS-like)	0.9336	-0.0305	0.2106	-0.0505	-0.3736	0.0005
Cloud-B (Azure-like)	0.8758	-0.0015	0.2271	-0.1066	-0.3479	0.0013
Cloud-C (GCP-like)	0.9096	0.0232	0.1768	-0.0754	-0.4258	0.0011
Cloud-D (Private/Edge)	0.8661	0.3671	-0.1421	-0.334	-0.2008	-0.0159

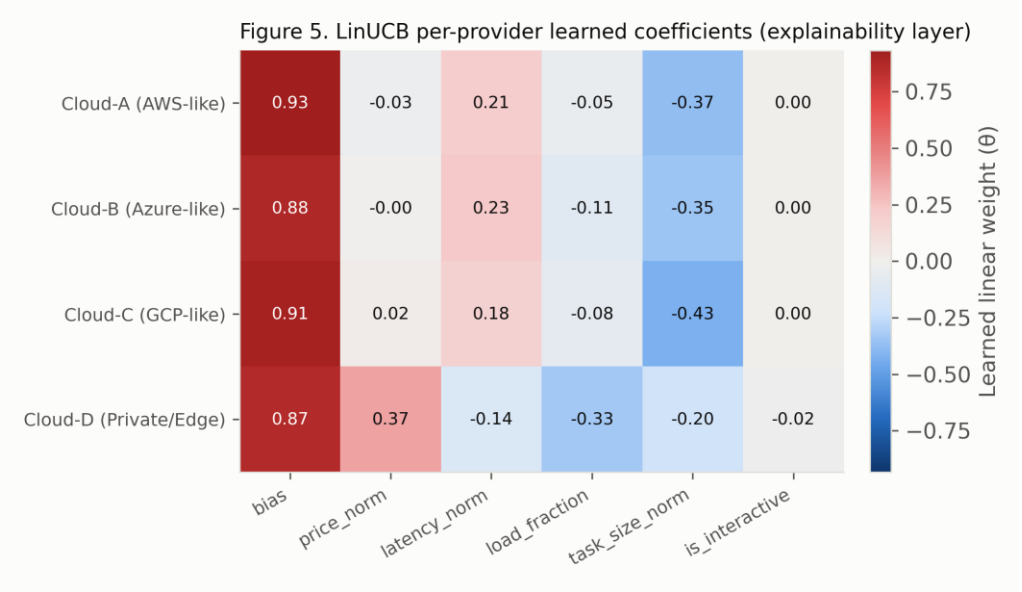

Figure 5. LinUCB per-provider learned coefficients, visualized as a diverging heatmap (blue = positive, red = negative weight).

Because raw linear coefficients can be affected by such collinearity, Table 6 and Figure 6 report the second, model-agnostic explanation layer: permutation importance computed on a Random Forest surrogate trained to imitate the agent's own logged decisions. The surrogate reproduced the agent's actual provider choice with 99.6% fidelity on held-out decisions, indicating it is a highly faithful proxy for the agent's true policy and therefore a trustworthy basis for explanation. The surrogate-based ranking is unambiguous: observed latency (importance = 0.548) and price (0.265) together account for the overwhelming majority of the agent's decision signal, with current load fraction a distant third (0.025) and task size and task class (interactive vs. batch) contributing negligibly. This is a substantively important, honest finding: despite task class being available to the agent and being the feature that, in principle, most directly determines SLA risk, the agent's converged policy relies overwhelmingly on directly observed price and latency rather than on task metadata — a natural consequence of the fact that latency and price already capture most of the information relevant to avoiding SLA violations for both task classes, which discuss further, together with its implications, in Section 5.

Table 6. Permutation importance of context features on the surrogate model's imitation of the agent's decisions (30 repeats).

Feature	Mean Importance	Std. Dev.
latency_norm	0.5482	0.01106
price_norm	0.26478	0.00823
load_fraction	0.02542	0.00297
bias	0.0	0.0
is_interactive	0.0	0.0
task_size_norm	-0.00029	0.00074

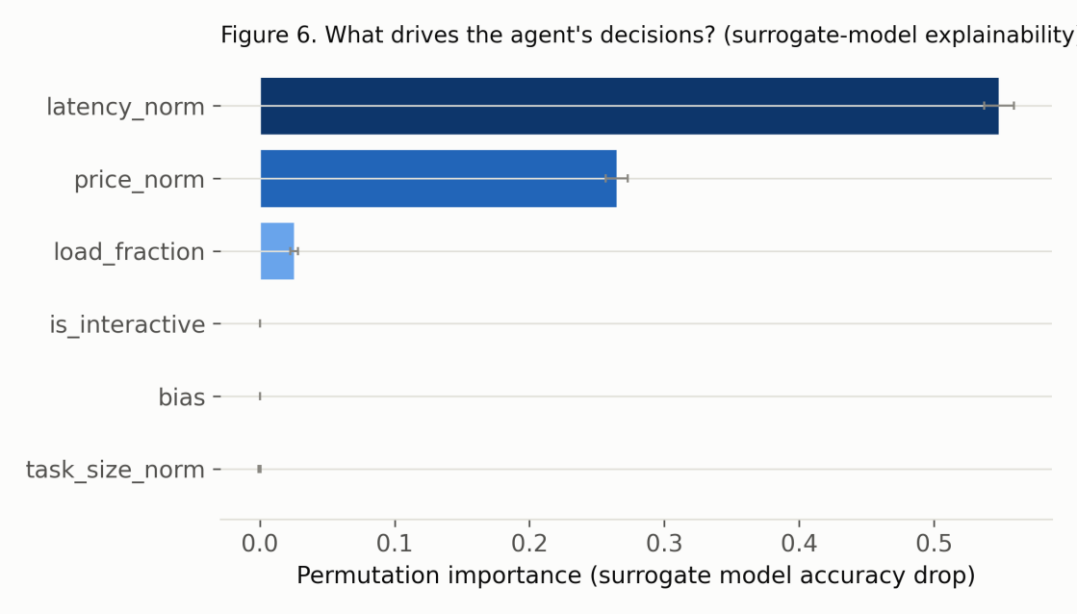

Figure 6. Surrogate-model permutation importance: what drives the agent's allocation decisions.

4.6 Load Distribution Across Providers

Figure 7 shows each policy's realized traffic share per provider. Random and Round-Robin split traffic perfectly evenly (25% each) by construction; Greedy-Cost sends 100% of traffic to whichever single provider is cheapest at each moment (predominantly Cloud-D); Greedy-Latency concentrates the majority of traffic (62.0%) on Cloud-C, its lowest-base-latency provider. The agent distributes traffic across all four providers with a share ranging from 16.9% to 31.6%, visibly favoring the lower-latency, higher-capacity providers (Cloud-A, Cloud-C) over Cloud-D while still using all four — a learned compromise between the near-uniform spread of the naive baselines and the single-provider concentration of the greedy heuristics.

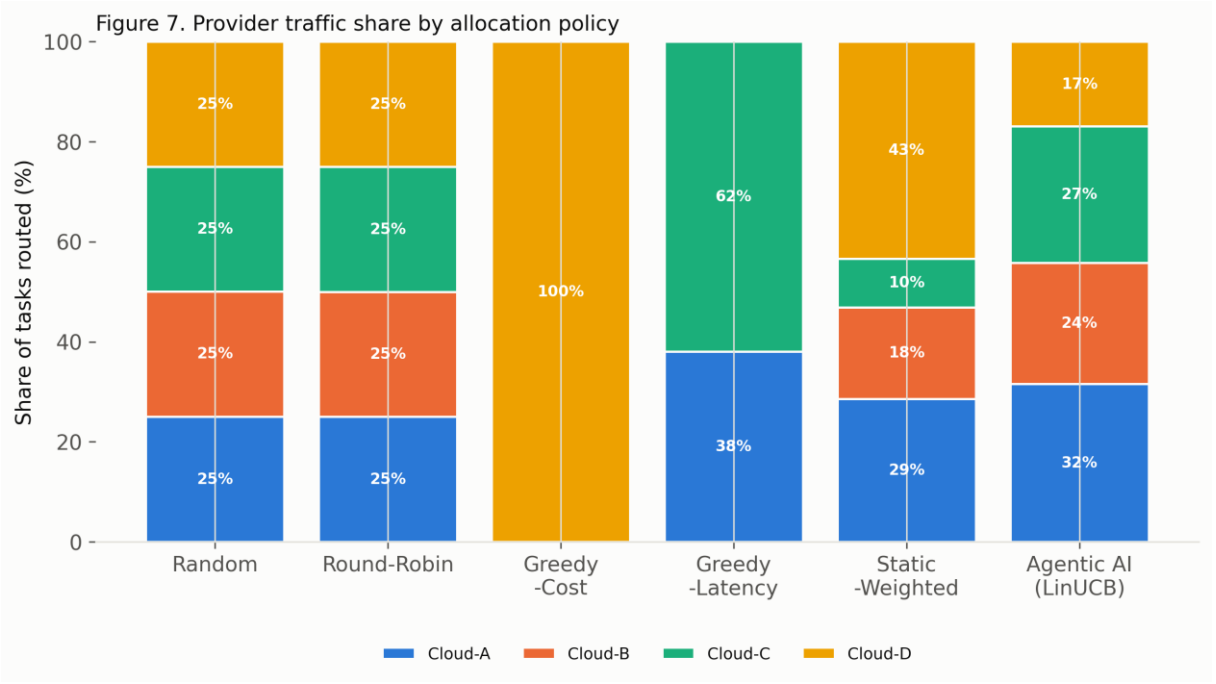

Figure 7. Share of tasks routed to each provider, by allocation policy.

5. Discussion

Three findings merit discussion. First, the comparison against Static-Weighted isolates the specific value of learning: both policies use exactly the same two input signals (price and latency), so the agent's significant advantage in SLA compliance, latency, and fairness (Section 4.1) is attributable specifically to online, context-adaptive weighting rather than to having access to more information. This is an important distinction for practitioners deciding whether the added operational complexity of a learning agent over a simpler fixed heuristic is justified; these results suggest it is, provided the moderate cost premium (20.6% in this study) is acceptable relative to the reliability and fairness gained. Second, the stress test (Section 4.3) demonstrates that this advantage is not merely incremental under normal conditions but becomes materially larger under operational disruption: naive baselines that ignore observed conditions (Random, Round-Robin) degrade sharply when a provider becomes unhealthy, while the agent — like the reactive Greedy-Latency baseline — adapts immediately, but does so without Greedy-Latency's cost and fairness penalties. This is consistent with the broader argument in the agentic-AI literature [3] that the value of autonomous, adaptive decision-making is often concentrated in non-stationary or disrupted conditions rather than in steady state. Third, the explainability analysis (Section 4.5) produced a genuinely informative negative result: although the agent had access to task-class information (interactive vs. batch) that, in principle, should modulate how aggressively it avoids latency risk, its converged policy assigns this feature negligible importance, relying almost entirely on directly observed price and latency instead. Interpret this as the agent having learned a reasonable simplification — observed latency already reflects current risk regardless of why a task is latency-sensitive — but it also indicates an opportunity: a reward function that more explicitly separates interactive from batch objectives, or a richer context that interacts task class with provider features, might further improve interactive-task outcomes specifically, a direction for future work.

Practically, these results support a layered recommendation for multi-cloud allocation system design: a fixed-weight heuristic such as Static-Weighted is a reasonable, easily audited default, but where operators can tolerate a lightweight online-learning component, a contextual-bandit agent of the kind proposed here offers a materially better and still fully explainable balance of cost, latency, SLA compliance, and fairness — and its advantage grows, rather than shrinks, precisely when the environment misbehaves.

6. Limitations and Future Work

This study's principal limitation is that it evaluates on a simulation rather than a live multi-cloud deployment. The simulator's price, latency, and workload dynamics are documented, plausible modeling assumptions rather than measurements of real commercial cloud behavior, and while the qualitative findings (adaptive policies outperform single-objective heuristics; adaptive policies are markedly more robust to disruption) are likely to generalize, the specific quantitative improvements reported here should not be read as predictions of real-world savings without validation against production traces or a live pilot deployment. The task and provider models are also deliberately simplified: tasks are treated as instantaneous placement decisions rather than long-running jobs with migration costs, inter-task dependencies, or data-transfer/egress costs between providers, all of which materially affect real multi-cloud economics and are natural extensions of this framework. The agent itself uses a linear contextual-bandit model chosen specifically for interpretability; this is a deliberate trade-off against the potentially higher raw performance of a non-linear or deep reinforcement-learning policy, and future work should quantify that accuracy–interpretability trade-off directly by benchmarking the same scenarios against such models. The explainability analysis, while dual-layered, remains global rather than per-decision: an operator currently cannot obtain a natural-language justification for one specific placement decision, and integrating a per-decision local explanation method (analogous to SHAP or LIME, as noted for the companion medical-diagnosis study in this series) is an identified direction for future work. Finally, the stress test modeled a single, fully observable degradation event; real incidents can be partially observable, correlated across providers, or accompanied by pricing anomalies, and evaluating robustness under a wider, adversarially chosen range of disruption scenarios is left to future work.

7. Conclusion

This paper proposed and evaluated an explainable agentic AI framework for multi-cloud resource allocation built on a LinUCB contextual-bandit agent, evaluated against five baseline policies across 30 independent simulation runs with paired statistical testing, and probed under a realistic provider-degradation stress test. Relative to the fairest baseline comparison (a fixed-weight heuristic using the same two signals), the learned agent achieved statistically significant reductions in SLA violations (86.2%), latency (20.9%), and unfairness, and under a simulated transient provider outage it maintained near-best reliability while remaining substantially cheaper and fairer than a purely latency-reactive alternative. A dual-layer explainability mechanism — the agent's own linear coefficients and an independent, high-fidelity (99.6%) surrogate-model importance analysis — showed that observed latency and price are the dominant, consistent drivers of the agent's decisions. Together, these results support the practical viability of lightweight, natively interpretable contextual-bandit agents as an explainable alternative to both static

heuristics and opaque deep-reinforcement-learning policies for multi-cloud resource allocation, and provide a reproducible simulation benchmark against which future agentic and non-agentic allocation strategies can be compared.

References

1. Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. Proceedings of the 19th International Conference on World Wide Web (WWW 2010), 661–670. https://doi.org/10.1145/1772690.1772758
2. Jain, R., Chiu, D., & Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report, TR-301.
3. Ali, M. A., Dornaika, F., & Charafeddine, J. (2026). Agentic AI: A comprehensive survey of architectures, applications, and future directions. Artificial Intelligence Review, 59(11). https://doi.org/10.1007/s10462-025-11422-4
4. Hady, M. A., Hu, S., Pratama, M., Cao, Z., & Kowalczyk, R. (2025). Multi-agent reinforcement learning for resources allocation optimization: A survey. Artificial Intelligence Review, 58(11), Article 354. https://doi.org/10.1007/s10462-025-11340-5
5. Mangalampalli, S., Karri, G. R., Ratnamani, M. V., Mohanty, S. N., Jabr, B. A., Ali, Y. A., Ali, S., & Abdullaeva, B. S. (2024). Efficient deep reinforcement learning based task scheduler in multi cloud environment. Scientific Reports, 14, 21850. https://doi.org/10.1038/s41598-024-72774-5
6. Panda, S. K., & Jana, P. K. (2017). SLA-based task scheduling algorithms for heterogeneous multi-cloud environment. The Journal of Supercomputing, 73(6), 2730–2762. https://doi.org/10.1007/s11227-016-1952-z
7. Gurrala, R. R., & Tallapally, S. K. (2026). QUANTUM based latency and cost aware task scheduling in a multi-cloud environment. Journal of Grid Computing, 24(1). https://doi.org/10.1007/s10723-026-09825-w
8. Alam, S., Ramzan, M. A., Zubair, M., Ullah, U., Ziaullah, & Ullah, R. (2026). AI-driven resource allocation in cloud computing: A systematic review revealing critical sustainability and evaluation gaps. Computing, 108(5). https://doi.org/10.1007/s00607-026-01655-8
9. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
10. Altmann, A., Toloși, L., Sander, O., & Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. Bioinformatics, 26(10), 1340–1347.
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.