QoS-Aware Brokering for LLM-as-a-Service

SLA-Constrained Routing with Drift Detection

Okba Tibermacine[1]

ENSIA, Algiers, Algeria

Chouki Tibermacine[2]

IRISA, Univ. Southern Brittany, Vannes, France

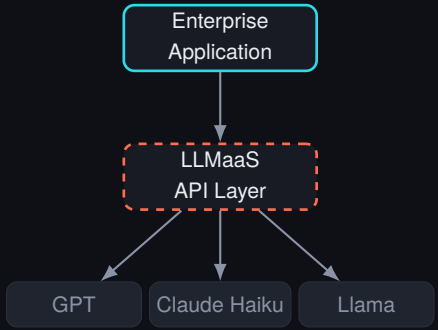
IEEE International Conference on Web Services (ICWS) 2026, Sydney, Australia

Outline

1. Context & State of the Art
2. Problem Statement
3. Solution: QoS-Aware Broker
4. Implementation & Experimentation
5. Conclusion

Motivation: The Rise of LLMaaS

- Enterprises increasingly consume LLMs as third-party APIs (LLM-as-a-Service) instead of hosting GPU clusters
- Providers: OpenAI, Anthropic, Mistral, DeepSeek, Google, Meta $\rightarrow$ heterogeneous portfolio of endpoints
- Adopted in critical workflows: customer support, financial summarization, medical triage



- These workflows demand robust Service-Oriented Architecture (SOA) guarantees…
- …but LLMaaS endpoints expose no formal QoS contract

Two Core Challenges

1. Non-Deterministic Latency & Cost

LLM inference is auto-regressive: execution time and financial cost scale with the (unknown) number of generated tokens — they cannot be known before completion.

2. Temporal API Drift

Providers silently update models (quantization, alignment shifts). Factual reliability or formatting compliance can degrade suddenly, without warning or documentation.

Result: state-of-the-art predictive routers exhibit **SLA violation rates exceeding 60%**

State of the Art: Predictive LLM Routing

Cascading & Classifier Routers

- **FrugalGPT** — sequential cascade until a confidence threshold is met
- **RouteLLM** — lightweight classifier predicts if a cheap model suffices
- **LLMRouter** — KNN / SVM / MLP classifiers over prompt embeddings

Adjacent Efforts

- **MESS+** — stochastic optimization for probabilistic SLA satisfaction
- DRL routers — optimize long-term average reward
- **Llumnix, SABER, PreServe** — engine-level GPU / KV-cache scheduling
- **SELA** — edge/cloud split routing

Common trait

All learn a static predictive mapping prompt $\rightarrow$ model, optimizing *unconstrained expected utility*.

The Gap: Classifiers vs. True SLA Enforcers

	Predictive Routers	Our Broker
Optimization target	Unconstrained utility	Hard SLA constraint satisfaction
Adaptivity	Static learned weights	Real-time online estimators
Drift awareness	None	Asynchronous CUSUM detection
Recovery mechanism	None	ϵ-greedy re-admission
Domain heritage	ML / systems	Services Computing / SOA + control theory

No prior work jointly combines

(i) strict multi-dimensional SLA pruning, (ii) online EWMA QoS tracking, (iii) CUSUM drift detection on black-box LLM APIs, and (iv) ϵ-greedy recovery.

Problem Formalization: Dynamic API State

Model pool $\mathcal{M} = \{m_1, \ldots, m_n\}$. Each m_i exposes a time-varying, non-deterministic state vector:

$$L_i(q_t) \approx TTFT_i + \frac{\mathbb{E}[N_{out}(q_t)]}{TPS_i} \quad \text{(latency)}$$

$$C_i(q_t) = N_{in}(q_t) \cdot P_{in}^{(i)} + \mathbb{E}[N_{out}(q_t)] \cdot P_{out}^{(i)} \quad \text{(cost)}$$

$$Q_i(q_t) \in [0, 1] \quad \text{(quality)}$$

- N_{in}: prompt token length (known); $\mathbb{E}[N_{out}]$: heuristic estimate of output length
- Unlike REST APIs, none of these quantities are constant across calls

Problem Formalization: SLA-Constrained Objective

Each query arrives with a strict SLA tuple:

$$\mathcal{SLA} = \langle L_{max},\, C_{max},\, Q_{min} \rangle$$

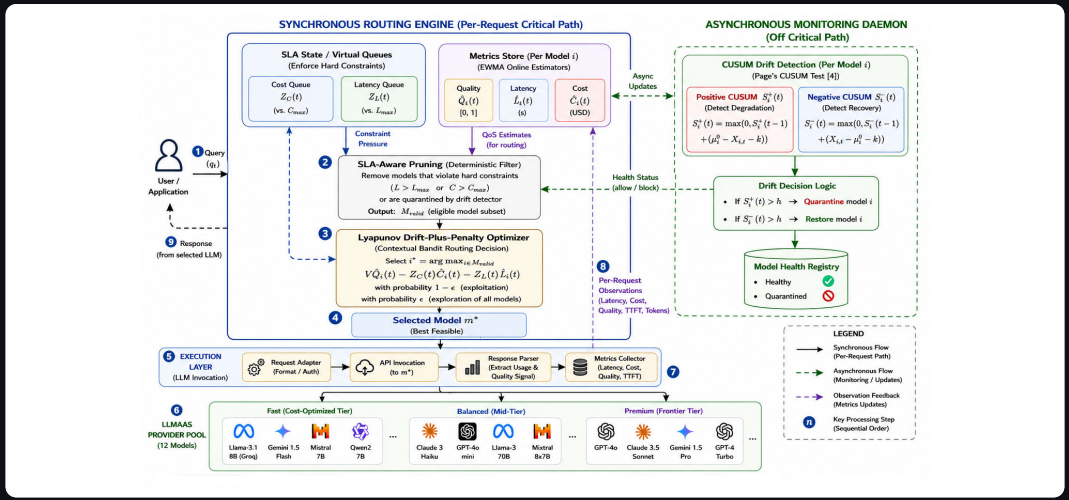
Broker objective

Select $m^* \in \mathcal{M}$ maximizing a multi-objective utility over quality, cost and latency strictly subject to:

- all bounds in $\mathcal{SLA}$ being respected, and
- m^* not currently undergoing unmitigated temporal drift

$\Rightarrow$ reframe LLM routing as a Stochastic Constrained Contextual Bandit, not a classification problem

System Architecture



Synchronous Routing Engine (steps 1–4) || **Asynchronous Monitoring Daemon** (off critical path)

Real-Time State Tracking: EWMA

- Execution modules asynchronously write observations (TTFT, tokens, latency) to a thread-safe `ModelMetricsStore` (`threading.RLock()`)
- Each API's state is tracked with an Exponentially Weighted Moving Average:

$$EWMA_t = \lambda X_t + (1 - \lambda)\, EWMA_{t-1}$$

- Recent observations weighted more $\rightarrow$ rapid detection of latency spikes / rate limits, without over-reacting to isolated network noise

Asynchronous CUSUM Drift Detection

- Baseline $\mu_0^{(i)}$ locked after a $W = 100$-query calibration phase (Welford's online mean/variance)

$$S_t^+ = \max\bigl(0,\ S_{t-1}^+ + (\mu_0^{(i)} - X_t - k)\bigr) \quad \Rightarrow \quad \text{alert if } S_t^+ > h$$

$$S_t^- = \max\bigl(0,\ S_{t-1}^- + (X_t - \mu_0^{(i)} - k)\bigr) \quad \Rightarrow \quad \text{re-admit if } S_t^- > h$$

- k: allowance parameter (min. shift magnitude) — suppresses false alarms from noise
- Runs in the Asynchronous Monitoring Daemon: quarantine / restore models *without* blocking the request stream

Lyapunov Drift-Plus-Penalty Optimization

Virtual queues track accumulated SLA violation (Lindley's equation):

$$Z_C(t+1) = \max[0,\, Z_C(t) + C_{i(t)} - C_{max}], \qquad Z_L(t+1) = \max[0,\, Z_L(t) + L_{i(t)} - L_{max}]$$

Minimize Lyapunov drift $\Delta(\boldsymbol{Z}(t))$ *minus* V·expected quality reward $\Rightarrow$ deterministic routing rule:

Routing decision

$$i^* = \arg\max_{i \in \mathcal{M}_{valid}} \Big(V \cdot \hat{Q}_i(t) \;-\; Z_C(t) \cdot \hat{C}_i(t) \;-\; Z_L(t) \cdot \hat{L}_i(t) \Big)$$

- V: trade-off knob between utility and SLA stability; queues adapt online
- Over RL:
  - Model-agnostic, no prior knowledge of provider distributions needed
  - Gives a deterministic low-overhead decision at every single request

Routing Algorithm & ϵ-Greedy Exploration

1. **Prune**: keep models with $S_i^+ < h \rightarrow \mathcal{M}_{valid}$
2. Draw $u \sim U(0,1)$
3. If $u < \epsilon$: explore — probe a random (possibly quarantined) model
4. Else: exploit — apply the Lyapunov rule over $\mathcal{M}_{valid}$
5. **Execute** query, observe (l_t, c_t, q_t)
6. **Update** EWMA & CUSUM statistics for m^*
7. **Update** virtual queues Z_C, Z_L (Lindley)

Why ϵ-greedy?

Quarantined models would otherwise starve (never re-observed). Forced probing feeds S_t^-, guaranteeing asymptotic convergence under non-stationary API drift.

Implementation & Open-Source Artifact

- Implemented as a non-blocking routing plugin for the open-source **LLMRouter** framework
- Concurrency-safe `ModelMetricsStore` (`RLock`) supports hundreds of concurrent requests
- Synchronous critical path: $\mathcal{O}(N)$ in pool size — CUSUM cost fully offloaded to the async daemon

Artifact availability

Source code, calibration data & figure-reproduction scripts:
`doi.org/10.5281/zenodo.19183702`

- `custom_routers/qos_aware_router/` — QoS-aware router extension
- `calibrate_apis.py`, `evaluate_broker.py` — evaluation scripts

Scan for code, data & scripts

Experimental Setup

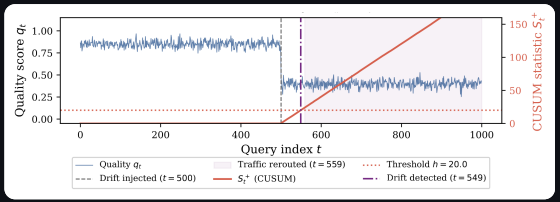
Real-world microbenchmark (50 queries/provider)

Model	Lat (s)	TTFT (s)	Cost ($)
GPT-4o	4.23±2.09	0.83±0.48	0.00186
Claude-3-Haiku	2.90±2.36	1.09±1.16	0.00024
Llama-3.1-8B	0.40±0.15	0.13±0.03	0.00002

Simulated scale

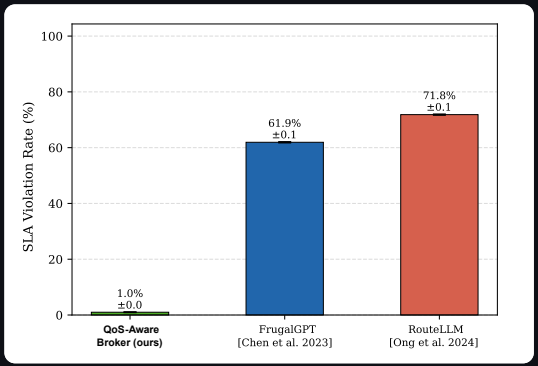
- 12-model pool: Fast / Balanced / Premium tiers, anchored to microbenchmark variance
- 100,000 queries per seed, 10 seeds for significance
- Quality modeled as $\mathcal{N}(\mu, \sigma^2)$
- Baselines: **FrugalGPT**, **RouteLLM**

Experiment 1 — CUSUM Drift Response



- 1,000-query stream, single endpoint
- Calibration: first 100 queries, $\mu_0 \approx 0.85$
- At t=500: injected drift drops true quality to 0.40
- S_t^+ stays ≈ 0 pre-drift, then rapidly accumulates
- Breaches h=20.0 $\rightarrow$ model quarantined automatically, before human intervention

Experiment 2 — SLA Violation Enforcement



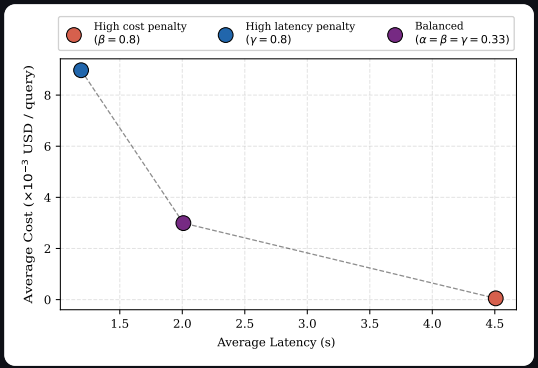
Constraints: $L_{max} = 2.0$s, $C_{max} = \$0.001$ — 100k queries $\times$ 10 seeds

- RouteLLM: $71.8\% \pm 0.1$ violations
- FrugalGPT: $61.9\% \pm 0.1$ violations
- **Our broker:** $1.0\% \pm 0.0$

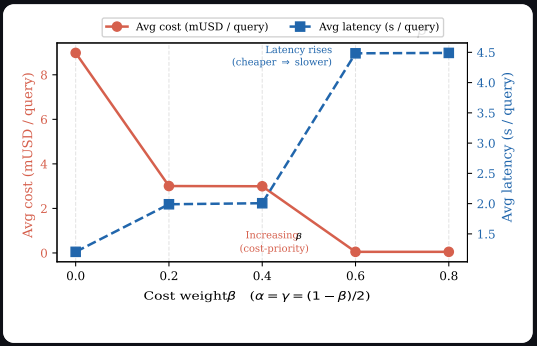
Residual 1.0%

Unavoidable physical variance from auto-regressive token generation exceeding the heuristic $\mathbb{E}[N_{out}]$ — not a routing failure.

Experiment 3 & 4 — Pareto Front & Ablation



γ=0.8 $\Rightarrow$ low-latency tier $\quad$ β=0.8 $\Rightarrow$ cost-minimizing tier



Sweeping $\beta \in [0, 0.8]$: monotonic cost ↓ / latency ↑ — no chaotic oscillations

Discussion: Strengths & Limitations

Strengths

- Near-zero overhead sync path, $\mathcal{O}(N)$
- CUSUM fully asynchronous
- ϵ-greedy resolves non-stationary bandit starvation, enables automatic recovery

Limitations

- Relies on accurate $\mathbb{E}[N_{out}]$ estimation
- Underestimation can bypass pre-execution pruning
- Open-ended generation (code, long-form) remains hard to bound *a priori*

Conclusion & Future Work

Summary

A QoS-Aware Broker that reframes LLM routing as SLA-constrained optimization — Lyapunov Drift-Plus-Penalty + CUSUM drift detection — cutting SLA violations from 61.9–71.8% to **1.0%** at production scale.

Future work

- Best-of-N routing via test-time compute scaling
- Extend to Agent-as-a-Service orchestration (e.g., MoMA)
- Gray-box integration with live KV-cache migration (Llumnix)

Thank You

Questions?

Okba Tibermacine[1]

okba.tibermacine@ensia.edu.dz

Chouki Tibermacine[2]

chouki.tibermacine@irisa.fr

Code, data & scripts: doi.org/10.5281/zenodo.19183702