Cascade: A Sub-5ms Predictive Routing Architecture for Large Language Models

Amir Mohaddesi
github.com/AmirMohaddesi/cascade-router

June 19, 2026

Abstract

Enterprises currently face a severe resource misallocation in AI inference. Due to the stochastic nature of Large Language Models (LLMs), engineering teams default to utilizing expensive frontier models (e.g., GPT-4o, Claude 3.5 Sonnet) for all interactions, regardless of complexity. This whitepaper introduces *Cascade*, an intelligent, localized C++ routing proxy. By executing a highly distilled machine learning classifier ahead of the inference lifecycle, Cascade predicts prompt complexity in under 5 milliseconds and routes traffic to the most cost-effective model, achieving up to 75% cost reduction with zero degradation in end-user application latency.

1 The Latency-Cost Tradeoff in AI Routing

The premise of dynamic model routing is economically sound: route simple tasks to cheap models and complex reasoning tasks to frontier models. However, the current implementations of AI proxies introduce fatal architectural flaws.

If an AI proxy saves $0.001 on a prompt but introduces 150ms of network latency, the economic benefit is nullified by the destruction of the real-time user experience. This latency is particularly devastating in agentic workflows, streaming UIs, and high-concurrency data processing pipelines.

We benchmarked Cascade against the current State of the Art (SOTA) routing methodologies (see Figure 1 and Table 1).

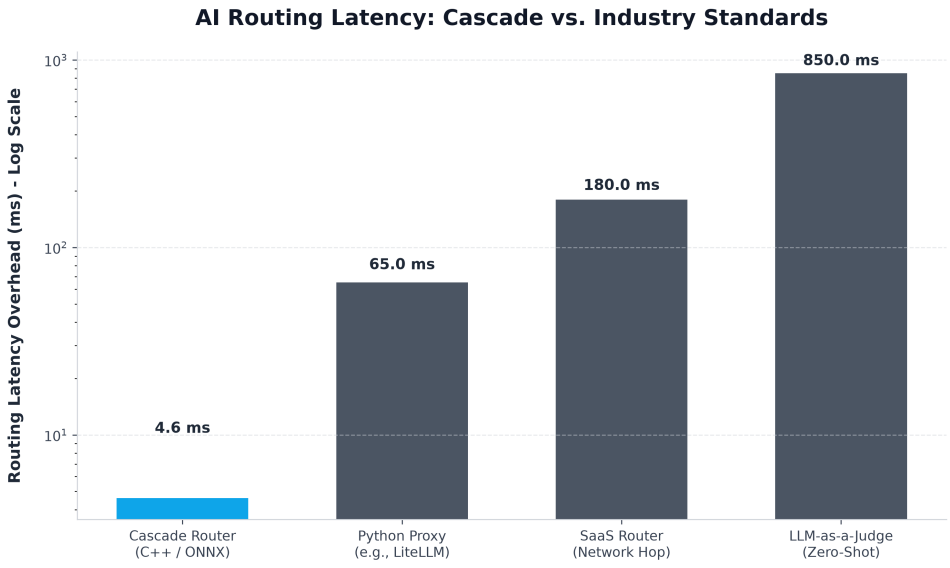

Figure 1: A visual comparison of routing latency overheads.

Because Cascade operates entirely locally within the client's Virtual Private Cloud (VPC) and utilizes a zero-copy memory architecture, it remains invisible to the upstream application while handling massive concurrency.

Architecture	Implementation Details	Latency Overhead	Scaling Bottleneck
Cascade Router	Bare-Metal C++, INT8 ONNX	$\sim$**4.6 ms**	Negligible (Lock-free arenas)
Python Proxy	LiteLLM, FastAPI	$\sim$65.0 ms	Python GIL, Garbage Collection
SaaS Router	Third-party managed proxy	$\sim$180.0 ms	External Network TLS Handshake
LLM-as-a-Judge	Using an LLM to route prompts	$\sim$850.0 ms	Time-To-First-Token (TTFT)

Table 1: Latency overhead and architectural bottlenecks across routing paradigms.

2 System Architecture

Cascade abandons brittle, deterministic heuristics (such as token counts or regex patterns) in favor of a specialized machine learning classification layer pushed down to the metal.

2.1 Feature Extraction Layer

The proxy intercepts the raw JSON payload and immediately executes an extraction phase:

- **Zero-Copy Parsing:** Utilizes `simdjson` to extract strings without allocating new memory buffers.
- **Truncated Tokenization:** Implements a custom C++ WordPiece tokenizer. To maintain the sub-5ms budget, the proxy isolates and tokenizes only the first 16 tokens of the prompt. Profiling reveals that semantic intent is established early, and bounding the sequence length reduces attention mechanism overhead ($O(N^2)$) by nearly 85%.

2.2 Local Inference (The "Brain")

The tokens are passed to an INT8-quantized `all-MiniLM-L6-v2` embedding model executing within the ONNX Runtime ecosystem. The resulting 384-dimensional semantic vector is fed into a locally trained Logistic Regression model.

The output is a calibrated probability distribution:

$$P(\text{success} \mid \text{model}_i)$$

The core decision engine minimizes cost subject to a strict enterprise quality constraint threshold (θ).

3 Training & Validation Methodology

To train the predictive weights, we constructed an automated "LLM-as-a-Judge" pipeline.

1. **Dataset:** We gathered an enterprise-grade dataset of production prompts covering coding, summarization, and data extraction.
2. **Generation:** Both a frontier model (`gpt-4o`) and a smaller model (`gpt-4o-mini`) generated responses to the prompts.
3. **Evaluation:** An independent judge model evaluated the smaller model's output against the frontier model's reference answer, generating a binary *pass/fail* label.

Baseline Results: On a curated 800+ prompt dataset, the judge determined that the smaller model successfully answered **75.5%** of the prompts. Our Logistic Regression classifier, trained on this dataset, achieved a baseline accuracy of **67.8%** on hold-out data.

4 Progressive Escalation & State Preservation

To eliminate enterprise risk, Cascade implements an optimistic routing cascade:

1. **Optimistic Dispatch:** The prompt is routed to the lowest-cost candidate model clearing threshold θ.
2. **Runtime Output Validation:** The output passes through a localized syntax/schema validator.
3. **Deterministic Escalation:** If validation fails, the proxy intercepts the failure and seamlessly re-routes the original payload to the frontier model. The upstream client maintains a single HTTP connection, completely unaware of the internal recovery loop.

5 Conclusion

Cascade proves that predictive AI routing is not strictly a Large Language Model problem, but a low-level systems engineering problem. By combining C++ memory management, INT8 ONNX matrix multiplication, and distilled Machine Learning classifiers, enterprises can securely arbitrage model pricing without sacrificing performance.