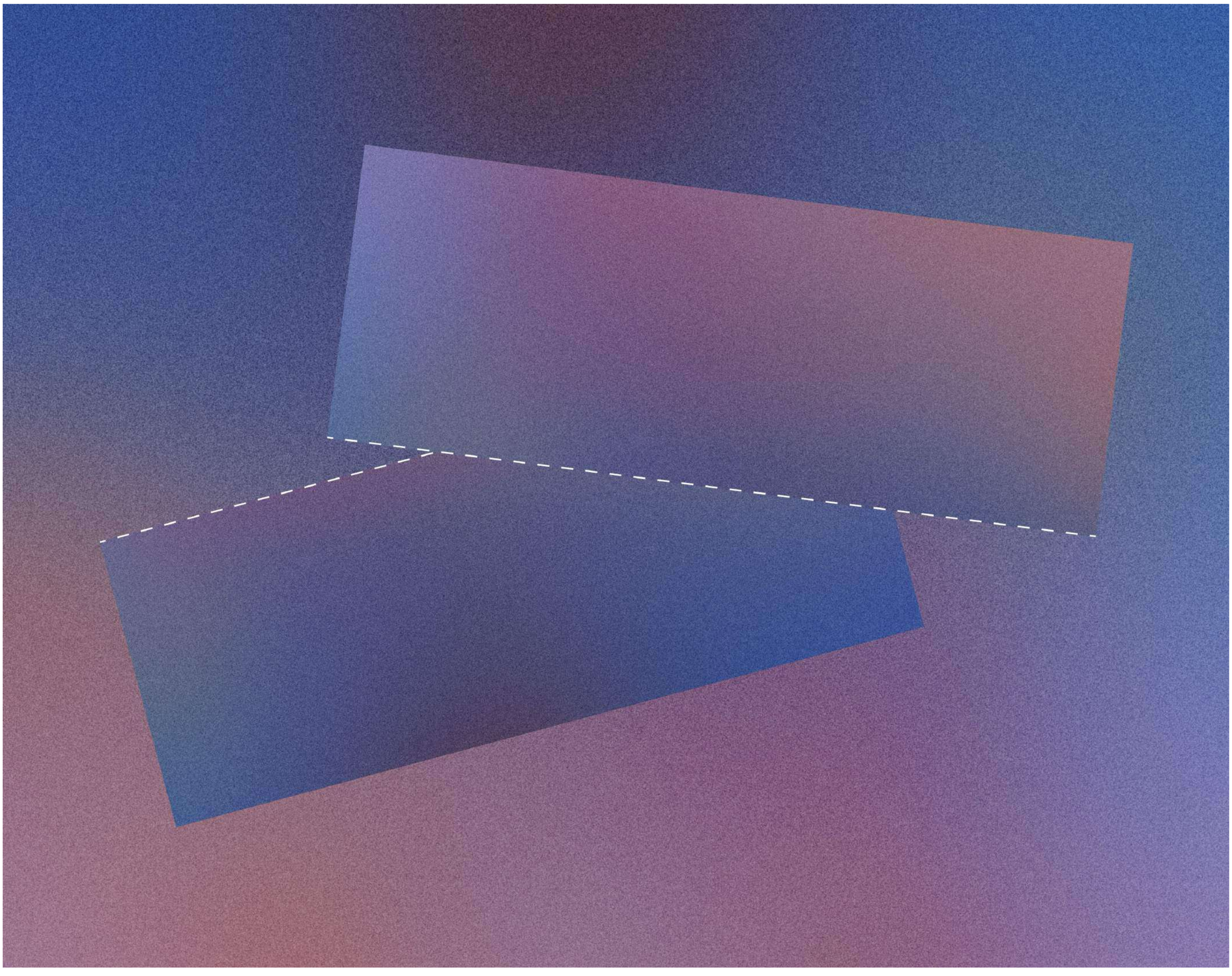
IIElevenLabs

# How we cut RAG latency in half for enterprise-scale conversational systems



# Introduction

At enterprise scale, conversational agents rely on retrieval-augmented generation (RAG) to ensure higher accuracy across large knowledge bases. RAG grounds each model response in the company's private knowledge base - retrieving only the most relevant context for the query. Our system rewrites every query before retrieval, collapsing dialogue history into a precise, auditable prompt that ensures traceability across customer interactions and regulatory audits.

For small datasets, context can be passed directly to the LLM. But for enterprise environments - where knowledge bases can exceed hundreds of millions of documents - RAG becomes the foundation for accuracy, scalability, and control.

Talk to an AI Agent Deployment Expert

01

# Where RAG performance breaks down

The challenge isn’t accuracy - it’s latency. Every request in our architecture includes a query rewriting stage as most user requests reference prior turns, so the system needs to collapse dialogue history into a precise, self-contained query. This step ensures precision but introduces a bottleneck: a dependency on a single, externally hosted LLM for rewriting.

For example:

| If the user asks: | | The system rewrites this to: |
|---|---|---|
| Can we customize those limits based on our peak traffic patterns? | → | Can Enterprise plan API rate limits be customized for specific traffic patterns? |

The rewriting turns vague references like “those limits” into self-contained queries that retrieval systems can use, improving the context and accuracy of the final response.

Our initial retrieval pipeline relied on a single, externally hosted LLM - a design that introduced hard dependencies on provider latency, uptime, and regional throughput limits. As we began scaling deployments across enterprise workloads, this single-threaded dependency became the dominant source of RAG delay. This step alone added 80% of our total RAG latency - unacceptable when operating across thousands of simultaneous conversations.

02

# How we fixed it with model racing

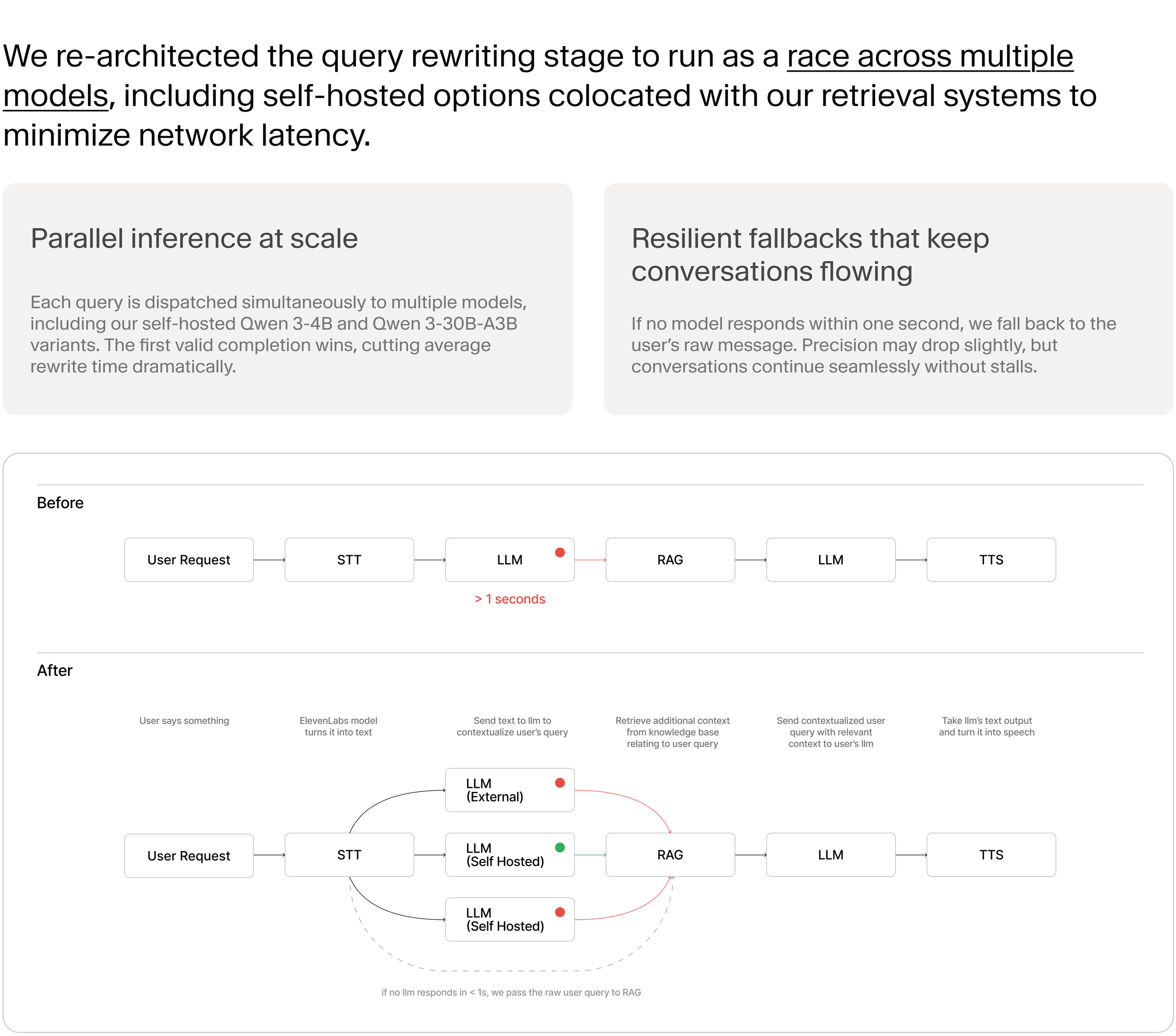
We re-architected the query rewriting stage to run as a race across multiple models, including self-hosted options colocated with our retrieval systems to minimize network latency.

### Parallel inference at scale

Each query is dispatched simultaneously to multiple models, including our self-hosted Qwen 3-4B and Qwen 3-30B-A3B variants. The first valid completion wins, cutting average rewrite time dramatically.

### Resilient fallbacks that keep conversations flowing

If no model responds within one second, we fall back to the user's raw message. Precision may drop slightly, but conversations continue seamlessly without stalls.



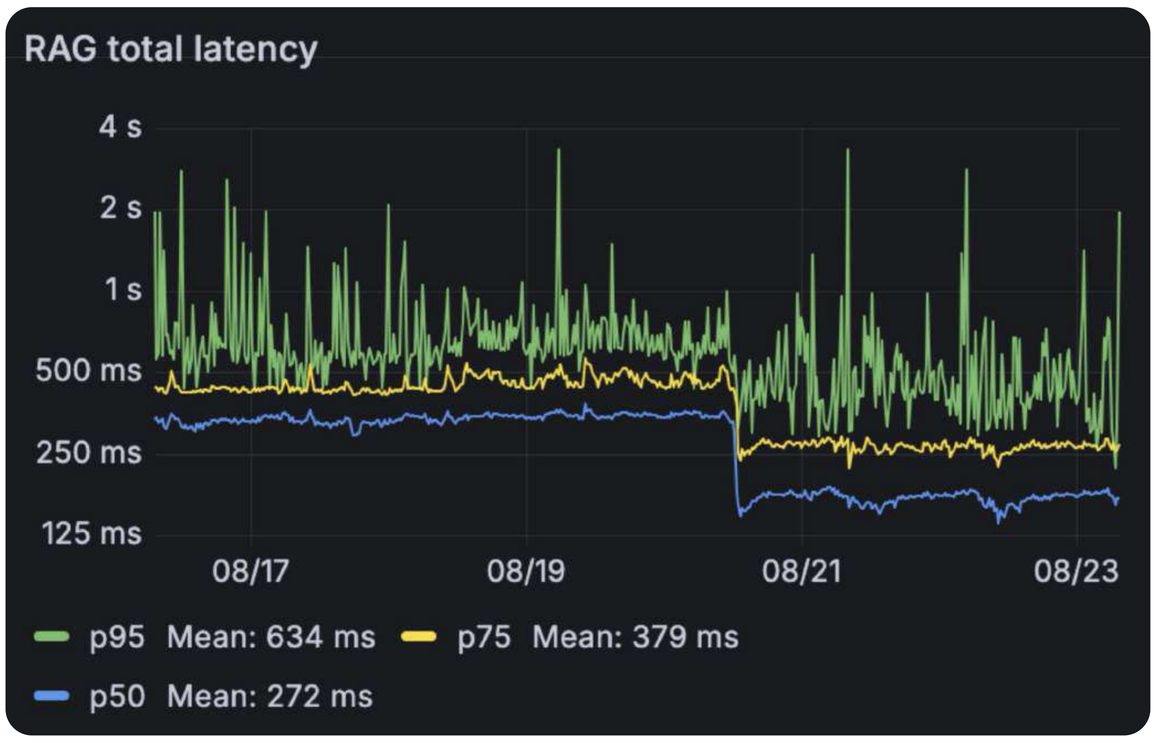

This architecture eliminates a single point of failure, stabilizes response times across regions, and ensures consistent performance even during provider-side slowdowns or outages.

03

# The outcome: sub-200ms RAG latency

This new architecture cut median RAG latency in half, from 326ms to 155ms. Unlike many systems that trigger RAG selectively as an external tool, we run it on every query. With median latency down to 155ms, the overhead of doing this is negligible.

| Before | | After |
|---|---|---|
| Median 326ms | → | Median 155ms |
| p75 436ms | | p75 250ms |
| p95 629ms | | p95 426ms |




Racing also stabilized overall performance. Externally hosted models fluctuate during peak demand, but our internal models stay consistent. Racing them together smoothed latency variance and made outages invisible to end users - conversations continued seamlessly on self-hosted infrastructure.

For example, when one of our LLM providers experienced an outage last month, conversations continued seamlessly on our self-hosted models. Since we already operate this infrastructure for other services, the additional compute cost is negligible.

04

# Why it matters

Latency under 200ms transforms RAG from a bottleneck into a competitive advantage. It enables real-time, compliant, and context-rich conversations over vast internal data sets - without sacrificing performance or control.

Enterprises no longer need to choose between speed and traceability. With RAG latency reduced to near-negligible levels, conversational systems can finally meet the dual mandate of governance and real-time intelligence.

Our team is focused on one goal: building conversational systems that perform predictably, scale globally, and stay compliant in the most regulated industries.

Talk to an AI Agent Deployment Expert