BUILT AT AMAZON 2015–2022 AMAZON · ML PLATFORM

Running LLM systems at 2M requests per hour

Architecture, reliability, and cost lessons from operating classification and inference systems at Amazon scale.



Written by Azeruddin Sheikh
BUILT AT AMAZON · 2015–2022

LINKEDIN ↗

Context

Amazon runs machine-learning systems where a slow or wrong answer has an immediate customer and financial cost. I spent seven years as an ML architect and technical lead on systems at that scale, leading a team of more than twenty scientists and engineers. The work spanned classification, document and media extraction, research tooling, and the inference platform underneath them.

This is a sanitized case study. It describes architecture patterns and published metrics, not confidential code, customer data, or internal systems.

The problem

Three pressures arrived at the same time:

Volume. The system had to serve more than two million requests per hour, with a 99.99% availability target. Capacity planning could not be done by adding machines after the fact.

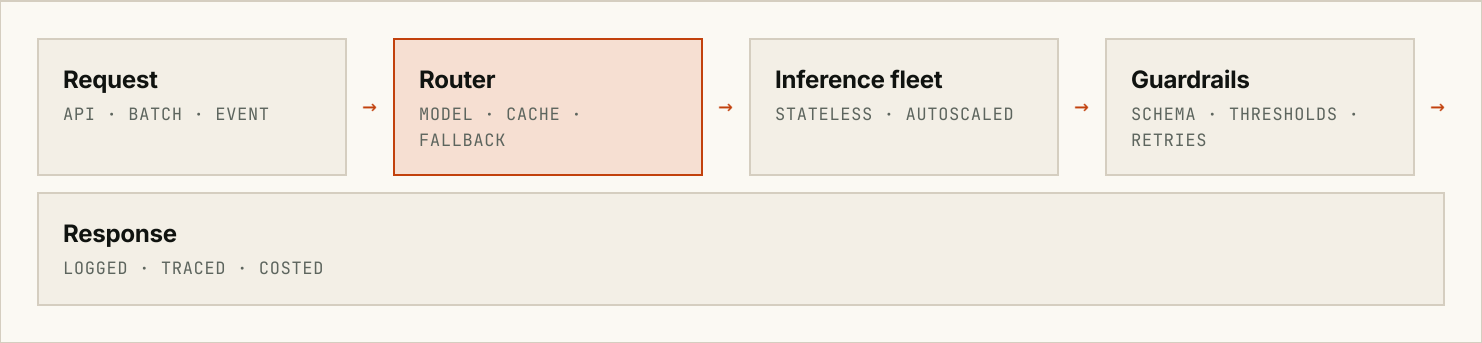
Latency. Inference was the slowest part of the request path. A model upgrade that improved quality but added latency was not automatically a win.

Cost. GPU and CPU capacity were the dominant variable cost. The team was measured on cost per request as well as quality.

The naive answer — a bigger model behind every request — failed all three. The work was to make the system fast, predictable, and cheap enough to operate, without giving up the quality the product depended on.

Architecture

The platform separated the request path from the model path. A thin routing layer decided which model, cache, or fallback should answer each request; the model fleet stayed stateless and horizontally scalable.



SANITIZED PATTERN. NAMES AND TOPOLOGY ARE SIMPLIFIED.

The key decisions:

Route before you generate. Not every request needs the largest model. A classifier, a cache hit, or a smaller model handled the easy majority; the expensive path was reserved for the cases that needed it.

Make the model fleet stateless. Request state lived outside the inference workers. That allowed rolling deployments, autoscaling, and a clean fallback when a worker failed.

Treat evaluation as infrastructure. A model change shipped with an offline evaluation set, a shadow run, and a rollback path. Quality was a release gate, not a postmortem.

Put a cost model next to the latency model. Every architecture decision had a cost per request attached to it. That made tradeoffs visible to engineering and to the business.

What I did

Led a team of 20+ scientists and engineers across model development, inference, and platform work.

Cut inference latency by **80%** while holding service levels, through batching, caching, model routing, and removal of work from the hot path.

Built a classifier over **100,000+ classes** that raised accuracy by **35%** and halved processing time.

Built extraction pipelines for structured content from images and video at **75% higher throughput**.

Reduced time-to-insight on complex research queries by **60%**.

Delivered **$20M+ in savings** through capacity, model, and pipeline decisions.

2M+	99.99%	80%	100K+
REQUESTS/HOUR	AVAILABILITY	LATENCY REDUCTION	CLASSES

Results

MEASURE	BEFORE	AFTER
Inference latency	Baseline	**80% lower**
Classification accuracy	Baseline	**+35%**
Processing time	Baseline	**−50%**
Extraction throughput	Baseline	**+75%**
Delivery savings	—	**$20M+**
Availability at peak	—	**99.99%**

The numbers above are the published outcomes of the work. They are not a guarantee that the same percentages apply to another system; the architecture and the operating model are the transferable part.

What it proves

Production AI is an operations problem as much as a modelling problem. The system that survives is the one where routing, evaluation, fallbacks, and cost are designed in from the beginning — not added after the first outage.

If your AI system is hitting a latency, cost, or reliability wall, that is the shape of problem I work on now. The first step is a two-week diagnostic.

AT A GLANCE

HEADLINE RESULT

99.99%

availability at 2M+ requests/hour

LABEL

Built at Amazon

PERIOD

2015–2022

CONTEXT

Amazon · ML platform

STACK

Distributed inference | Model routing | Feature and retrieval pipelines | Evaluation and guardrails | Cost and capacity modelling

---

Download PDF →

Discuss a similar problem →

2M+	99.99%

REQUESTS PER HOUR	AVAILABILITY
80%	$20M+
LATENCY REDUCTION	SAVINGS

NEXT CASE STUDY

A 1.1M-token memory on an 8 GB laptop

All work →