# AI System Latency Budget & Reactive Scaling Architecture

Production Performance Reference

Prepared by: Vipin Kumar

## 1. Latency Budget by Pipeline Stage

Target latency thresholds per stage across p50, p95, and p99 percentiles. The total pipeline row accounts for per-stage sums plus overhead and tail stacking.

| Pipeline Stage | p50 | p95 | p99 | Why |
|---|---|---|---|---|
| Embedding | <100 ms | <200 ms | <400 ms | Small work, low variance |
| Retrieval | <200 ms | <400 ms | <800 ms | DB / network adds delay |
| LLM Inference | <1000 ms | <2000 ms | <4000 ms | Heavy compute, dominant latency driver |
| **Total Pipeline Base** | **<1300 ms** | **<2600 ms** | **<5200 ms** | Pure sum of stages |
| **Total Pipeline (With Overhead)** | **<1500 ms** | **<3000 ms** | **<6200 ms** | Base + system overhead (10-20%) + tail effects |

**Key Rules**

- Sequential: total latency = sum(stages) + overhead
- Parallel: total latency = max(stage) + overhead
- Typically, p95 ≈ 2x p50 (normal variance under load)
- Typically, p99 ≈ 2x p95 (tail amplification due to retries, cold starts, queueing)
- If p95 > 3x p50 → tail latency problem exists
- LLM Inference drives ~70–80% of total latency → optimize p95/p99 first
- p99 is early signal of system instability before error rate increases

## 2. Latency Budget Framework

The framework behind how budget numbers are derived, validated, and applied. Percentiles do not compose linearly - always measure end-to-end in addition to per-stage.

| Category | Formula / Value | Why | Example | Budget Standard |
|---|---|---|---|---|
| **Stage Budgeting** | p50 / p95 / p99 per stage | Break system into controllable parts | Retrieval p95 = 400ms | Define per stage SLOs |
| **Total (Base)** | Sum of stage latencies | Ideal sequential execution (no overhead) | 1300ms | Baseline reference |

| Category | Formula / Value | Why | Example | Budget Standard |
|---|---|---|---|---|
| **Overhead** | Network + serialization + orchestration | Adds variability, impact increases at higher percentiles | API gateway + JSON = 150ms | Typically: p50+~10%, p95+~15%, p99+~20% |
| **Total (With Overhead)** | Sum of stages + system overhead (10–20%) + tail effects | Accounts for real system overhead and variability | 1300ms → 1500ms | Typically: p50+~10%, p95+~15%, p99+~20% |
| **Total (actual)** | Measured end-to-end latency | Percentiles do not compose linearly | Measured end-to-end p95 ≠ sum of stage p95 | Always validate separately |
| **Dominant Stage** | LLM (~70–80%) | Drives overall latency and tail behavior | 2000ms of 2600ms | Optimize p95/p99 first |
| **Tail Latency Rule** | p99 >> p95 | Compounding: retries, queueing, cold starts | p95 = 2600ms → Typically p99 ≈ 5200ms+ (tail amplification) | Typically: p99 ≈ 2x p95 (tail amplification) |
| **Early Signal** | p99 increase | Indicates instability before failures | p99 rises before error rate | Monitor proactively |
| **Parallel Rule** | max(stage) + overhead | Slowest parallel stage dominates execution time | max (400, 2000) + overhead ≈ 2200-2400ms (dominated by LLM) | Budget around max stage |
| **Sequential Rule** | sum(stages) + overhead | Execution latency accumulates step-by-step | 100+200+1000 + overhead ≈ 1450–1550ms | Sum of all stages |
| **Note:** Budgeted latency represents estimated planning targets. Actual latency must be validated using end-to-end measurement, as percentile values do not compose linearly. | | | | |

---

## 3. Metric Priority by Environment

The most important latency percentile shifts as you move from development toward production, reflecting increasing traffic and real user impact.

| Environment | Key metric | Why it matters |
|---|---|---|
| **DEV** | **p50** | Focus on correctness, latency outliers are acceptable. |
| **QA** | **p95** | Tests realistic load, catches slow paths early. |
| **SIT** | **p95** | Components tested together - integration bottlenecks surface. |

| Environment | Key metric | Why it matters |
|---|---|---|
| **UAT** | **p95 + p99** | Real business users testing - tail latency starts affecting user experience. |
| **STG** | **p95 + p99** | Production mirror - tail latency predicts real user impact. |
| **PRD** | **p99** | 1 in 100 users at scale = thousands of people. Primary signal for SLO violations, every outlier impacts real users at scale. |

---

## 4. Key Design Rules

- In this reference model, p95 ≈ 2× p50
- p99 ≈ 2× p95 (tail amplification; may exceed under stress)
- If p95 > 3× p50 → investigate tail latency issues
- LLM Inference drives ~70–80% of total latency → optimize p95/p99 first
- Sequential pipelines → total latency = sum(stages) + overhead
- Parallel pipelines → total latency = max(stage) + overhead

---

## 5. Reactive Scaling Architecture

The diagram below shows the full reactive architecture for the AI service, including per-stage latency budgets, reactive responses, Kubernetes HPA configuration, and rollback decision logic.

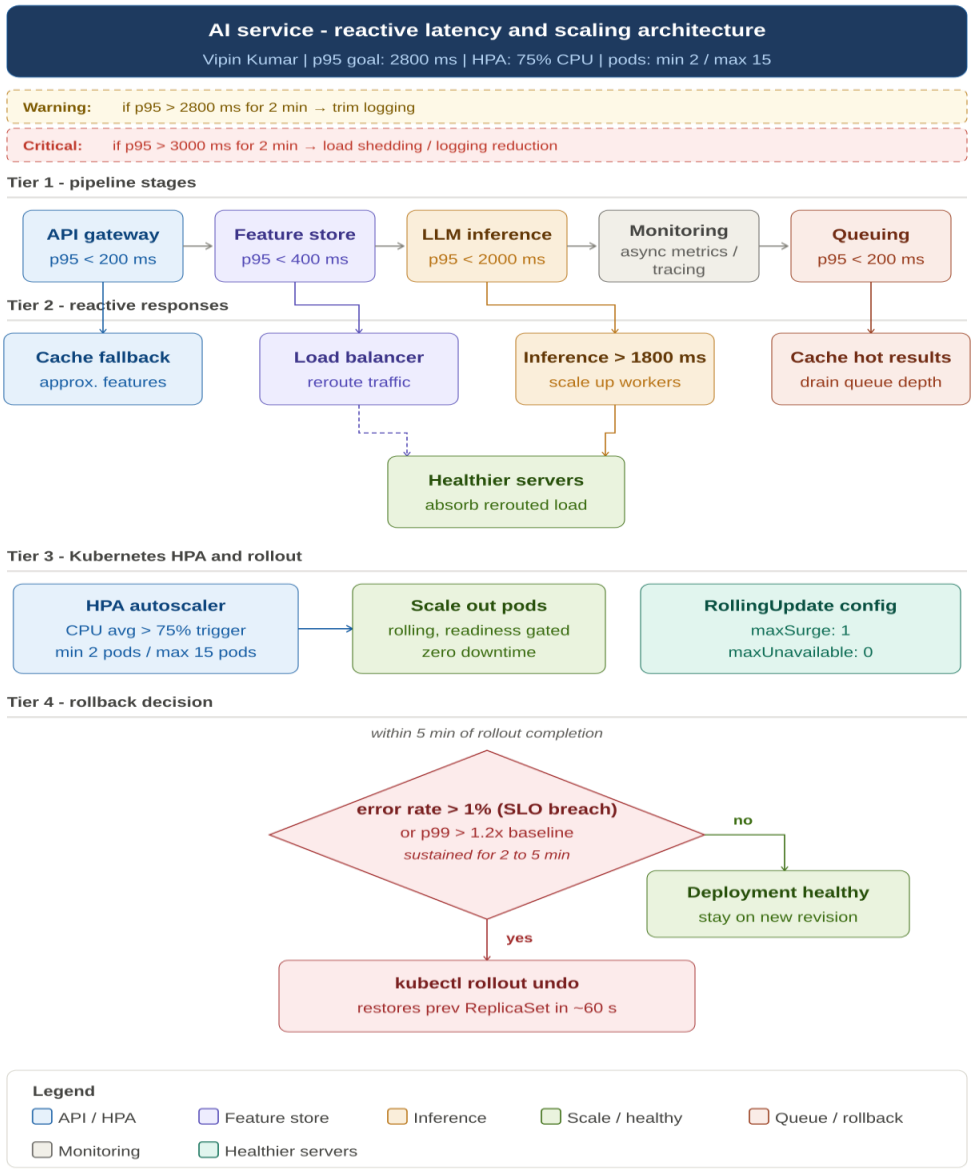


Figure 1. AI Service – Reactive Latency and Scaling Architecture