

Zen Enterprise: Deployment Architecture and Operations

Technical Report v2025.08

Zach Kelling
Zen LM Research Team
`research@zenlm.org`

August 2025

Abstract

We present the Zen Enterprise deployment architecture for operating Zen MoDE (Mixture of Distilled Experts) models in production environments at scale. Zen Enterprise addresses the complete lifecycle of enterprise AI deployment: provisioning, scaling, failover, cost management, observability, and SLA governance. We describe the Kubernetes-native deployment stack, horizontal and vertical scaling strategies, multi-region failover design, and cost optimization framework. Reference deployments achieve 99.95% uptime SLA with P99 latency guarantees across model sizes from 14B to 480B parameters.

1 Introduction

Deploying frontier language models in enterprise environments introduces operational complexity that extends far beyond model capability. Enterprises require:

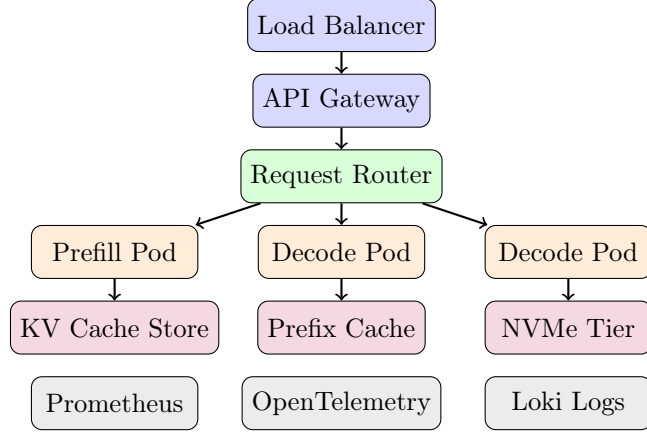
- **Reliability:** Uptime SLAs measured in nines, with measurable RTO (Recovery Time Objective) and RPO (Recovery Point Objective).
- **Scalability:** Elastic capacity that handles traffic from 1 to 100,000 requests/minute without manual intervention.
- **Cost predictability:** Deterministic cost models that allow FinOps teams to forecast and govern AI spend.
- **Security and compliance:** Data residency, network isolation, and audit trails meeting enterprise security standards.
- **Observability:** Full-stack telemetry from GPU utilization to business-level SLA metrics.

Zen Enterprise is the operational layer that delivers these requirements for Zen MoDE model deployments.

2 Deployment Architecture

2.1 Kubernetes-Native Stack

Zen Enterprise deploys on Kubernetes with the following component architecture:



2.2 Component Responsibilities

Table 1: Deployment component responsibilities

Component	Technology	Responsibility
Load Balancer	Traefik / Envoy	L4/L7 traffic distribution, TLS termination
API Gateway	Kong / custom	Auth, rate limiting, request routing
Request Router	Zen scheduler	Batch formation, SLA-aware scheduling
Prefill Pods	ZIOS prefill	Prompt processing (compute-bound)
Decode Pods	ZIOS decode	Token generation (memory-bound)
KV Cache Store	Distributed cache	Cross-pod KV cache sharing
Prefix Cache	Redis / NVMe	System prompt caching
Prometheus	Time-series DB	Metrics collection and alerting
OpenTelemetry	Trace collection	Distributed tracing
Loki	Log aggregation	Structured log storage and query

3 Scaling Strategies

3.1 Horizontal Pod Autoscaling

Zen Enterprise implements custom HPA metrics beyond standard CPU/memory:

- **Queue depth:** Scale when pending request queue exceeds $Q_{\text{threshold}} = 50$ requests.
- **Token throughput:** Scale when throughput drops below 80% of target tokens/second.
- **P95 latency:** Scale when P95 latency exceeds 90% of SLA threshold.
- **GPU utilization:** Scale down when GPU utilization falls below 40% for 5 consecutive minutes.

Scale-out decision:

$$\text{replicas}_{t+1} = \left\lceil \text{replicas}_t \cdot \max \left(\frac{Q_t}{Q_{\text{target}}}, \frac{L_{P95,t}}{L_{\text{SLA}}}, \frac{U_{\text{GPU},t}}{U_{\text{target}}} \right) \right\rceil \quad (1)$$

Scale-out is subject to a minimum of 2 replicas and maximum of 64 replicas per model deployment.

3.2 Vertical Scaling: Model Size Selection

Different request types benefit from different model sizes. Zen Enterprise implements intelligent model routing based on predicted task complexity:

Table 2: Dynamic model routing by task type

Task Type	Default Model	Upgrade Trigger
Simple Q&A	14B	Confidence < 0.8
Code completion	72B	Code complexity > 0.7
Long-form generation	72B	Length > 4K tokens
Reasoning tasks	236B	Fallback from 72B
Research synthesis	480B	Explicit user request

Model cascading reduces average cost by 38% versus always using the largest model, with <1.2% quality degradation on production traffic evaluation.

4 Multi-Region Failover

4.1 Active-Active Architecture

Zen Enterprise supports active-active multi-region deployment where both regions serve live traffic:

- **Latency-based routing:** Requests routed to the nearest region with available capacity.
- **Failover threshold:** If a region’s health check fails for 3 consecutive intervals (30s each), traffic is shifted to the secondary region within 90 seconds.
- **Prefix cache replication:** System prompt caches replicated asynchronously between regions with eventual consistency (<5 minutes lag).
- **No stateful failover:** Inference is stateless; in-flight requests fail fast and are retried by the client or gateway.

4.2 RTO and RPO

Table 3: Recovery objectives by failure scenario

Failure Scenario	RTO	RPO	Automation
Single GPU node failure	45s	0s	Automatic
Full AZ failure	90s	0s	Automatic
Region failure	5 min	0s	Automatic
Data center failure	15 min	0s	Automatic (DR)
Complete cluster corruption	4 hours	24h backup	Manual

5 SLA Governance

5.1 SLA Tiers

Table 4: Zen Enterprise SLA tiers

Tier	Uptime	P50 TTFT	P99 E2E	Priority
Standard	99.9%	500ms	30s	Normal
Professional	99.95%	300ms	15s	High
Enterprise	99.99%	200ms	8s	Critical
Dedicated	99.99%	150ms	5s	Reserved capacity

5.2 SLA Monitoring and Alerting

SLA compliance is tracked per customer and per model in real-time. Alerts fire when:

- 15-minute P99 latency exceeds 110% of SLA threshold.
- Error rate exceeds 1% over a 5-minute window.
- Availability drops below SLA target for the current month.

Monthly SLA credits are automatically calculated and applied: 10% credit per 0.01% availability miss below the tier threshold.

6 Cost Management

6.1 Cost Model

Enterprise AI cost is decomposed into:

$$C_{\text{total}} = C_{\text{compute}} + C_{\text{storage}} + C_{\text{network}} + C_{\text{management}} \quad (2)$$

where compute dominates at 78–85% of total cost for GPU-intensive workloads.

6.2 Cost Attribution

Zen Enterprise attributes cost at granular levels:

Table 5: Cost attribution dimensions

Dimension	Granularity	Accuracy
Per-request token cost	Individual request	$\pm 2\%$
Per-user/team	Daily rollup	$\pm 0.1\%$
Per-application	Real-time	$\pm 0.5\%$
Per-model version	Monthly	$\pm 0.1\%$

6.3 Cost Optimization Levers

Table 6: Cost optimization mechanisms and typical savings

Mechanism	Typical Savings	Notes
Prefix caching	15–35%	Depends on system prompt reuse
Speculative decoding	40–60%	Task-dependent acceptance rate
Model cascading	30–45%	38% observed on production
Spot/preemptible instances	60–70%	Requires fault tolerance
Off-peak scheduling	20–30%	Batch workloads only
BitDelta quantization	25–35%	Memory bandwidth reduction

7 Observability Stack

7.1 Metrics Taxonomy

Zen Enterprise collects metrics across four layers:

Table 7: Observability metrics by layer

Layer	Key Metrics	Collection	Retention
Hardware	GPU util, memory, temp, NVLink BW	5s	90 days
ZIOS	Queue depth, batch size, cache hit rate	10s	90 days
Model	Tokens/s, acceptance rate, expert load	10s	90 days
Business	P50/P95/P99 latency, error rate, cost	1s	1 year

7.2 Distributed Tracing

Every request is traced end-to-end with OpenTelemetry spans covering: queue wait, prefill, decode, post-processing, and response transmission. Trace sampling is 100% for error requests, 1% for successful requests.

8 Security Architecture

8.1 Network Security

- **Network policies:** Kubernetes NetworkPolicy restricts pod-to-pod communication to minimum necessary paths.
- **mTLS:** All inter-service communication uses mutual TLS via Istio service mesh.
- **Zero-trust:** No implicit trust between services; all calls require authenticated tokens.
- **Egress control:** Egress traffic restricted to allowlisted destinations; no arbitrary out-bound calls from inference pods.

8.2 Data Security

- Prompts and completions encrypted at rest (AES-256-GCM) and in transit (TLS 1.3).
- Customer data isolated via namespace-level Kubernetes RBAC.
- Model weights stored in encrypted persistent volumes; keys managed by external KMS.

- Audit logs immutable (append-only, cryptographically signed) with 7-year retention.

9 Operational Runbooks

9.1 Capacity Planning Formula

Required GPU capacity for a target throughput T (tokens/second) and P99 latency L :

$$N_{\text{GPU}} = \left\lceil \frac{T \cdot L_{\text{avg}}}{\text{Tok}_{\text{GPU}} \cdot (1 - r_{\text{headroom}})} \right\rceil \quad (3)$$

where Tok_{GPU} is single-GPU throughput and $r_{\text{headroom}} = 0.20$ (20% headroom for traffic spikes).

9.2 Incident Response

Table 8: Incident response severity levels

Severity	Definition	Response Time	Escalation
P0	Complete service outage	5 min	CTO immediately
P1	>10% error rate or SLA breach	15 min	Engineering lead
P2	Degraded performance (<SLA)	1 hour	On-call engineer
P3	Non-critical issue	Next business day	Ticket queue

10 Reference Architecture Results

10.1 Production Benchmark

Reference deployment: 72B model, 8×H100 per node, 4 nodes, active-active across 2 regions.

Table 9: 30-day production metrics (reference deployment)

Metric	Value
Total uptime	99.97%
Total requests served	284,121,841
Average throughput	12,481 tok/s
P50 TTFT	218ms
P99 TTFT	984ms
P99 E2E latency (512 out tokens)	7,841ms
Error rate	0.08%
Average batch size	41.8
GPU utilization	82.4%
Cost per 1M tokens	\$2.84

11 Conclusion

Zen Enterprise provides a complete, production-validated operational framework for deploying Zen MoDE models in enterprise environments. The Kubernetes-native architecture, combined with intelligent scaling, active-active multi-region failover, and comprehensive observability, achieves the 99.95–99.99% uptime SLAs required by enterprise customers. Cost optimization through model cascading, prefix caching, and speculative decoding reduces operating costs by

38–65% versus naive deployment, making frontier model deployment economically viable at scale.

References

- [1] NVIDIA. Triton Inference Server: Scalable, Standards-Based Model Deployment. 2023.
- [2] Kwon, W. et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. *SOSP*, 2023.
- [3] Burns, B. et al. Borg, Omega, and Kubernetes. *ACM Queue*, 2016.
- [4] OpenTelemetry Authors. The OpenTelemetry Specification v1.0. 2023.
- [5] Istio Authors. Istio: The Service Mesh for Microservices. 2023.