International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Enterprise Distributed Infrastructure for Scalable Generative AI Workloads

**Satish Chandra Guruvelli**

**Abstract:** Enterprise deployment of generative artificial intelligence (GenAI) at production scale exposes a category of infrastructure problems that classical data-center engineering does not anticipate. Unified graphics processing unit (GPU) pools suffer from chronic underutilization and unpredictable tail latency, while naive multi-tier partitioning sacrifices elasticity to obtain predictability. This article develops a dynamic multi-tier compute architecture that reallocates GPU capacity across inference, training, and operational tiers on a two-to-four-hour prediction horizon. The architecture is evaluated against unified and statically partitioned baselines using an eighteen-month observational deployment profile and published industry benchmarks. Dynamic allocation reduces infrastructure cost by 40–55 percent relative to unified pools while improving p95 latency consistency by 25–35 percent; GPU utilization rises from a 62–68 percent unified baseline to 78–81 percent under dynamic control, exceeding the 68–78 percent utilization band reported in vendor benchmarks. A complementary multi-vector-store knowledge fabric with intelligent query routing reduces retrieval latency by 40–55 percent and increases semantic recall to 99.2 percent, while a twelve-region active-active deployment with content-aware routing reduces p95 latency by 72 percent and compresses tail variance sixfold. The article formalizes a latency-cost Pareto frontier that lets enterprise operators reason explicitly about where to sit on the trade-off curve rather than pursuing unbounded cost reduction. Results hold across transformer families spanning 7 billion to 405 billion parameters and across mixed inference-training-operational workload profiles.

***Keywords:*** *Generative AI, Distributed Infrastructure, GPU Orchestration, Retrieval-Augmented Generation, Vector Databases, Latency Optimization, Kubernetes, Multi-Region Deployment*

## 1. Introduction

Enterprise adoption of generative artificial intelligence (GenAI) has moved beyond isolated pilots into core operational workflows. The infrastructure footprint required to support this transition is substantially different from the footprint that served prior generations of enterprise software, and the difference is not incremental. Large language model (LLM) inference, retrieval-augmented generation (RAG) pipelines, and autonomous agent loops all show memory-bound execution patterns, bursty arrival distributions, and tight p95 service-level objectives (SLOs), which reveal the limits of general-purpose cluster scheduling. The practical question facing enterprise architects is no longer whether to allocate dedicated graphics processing unit (GPU) capacity to these workloads but how to partition and dynamically reassign that capacity so that cost, latency, and reliability are jointly optimized rather than traded off ad hoc.

*Independent Researcher, USA*
*ORCIDs: 0009-0000-1298-237X*

Two infrastructure patterns dominate current practice, and both are unsatisfactory. Unified GPU pools route every workload to a shared scheduler, which yields flexibility at the cost of GPU underutilization, head-of-line blocking for latency-sensitive inference, and a tail-latency profile that violates enterprise SLOs once utilization climbs past the seventy-percent mark. Static multi-tier architectures, which pre-partition capacity into inference, training, and operational pools, improve predictability but lock in allocation ratios that fit neither fluctuating diurnal demand nor the substantially different compute shapes of transformer families at the seven-billion, seventy-billion, and four-hundred-five-billion parameter scales.

This paper argues that neither extreme is optimal and that a dynamic multi-tier architecture, in which tier boundaries are preserved but tier sizes are re-optimized on a rolling two-to-four-hour horizon using workload prediction, produces materially better outcomes on every axis of interest. The contribution is threefold: a formal partitioning scheme with an accompanying control loop; an empirical characterization of how optimal tier ratios

vary with model family and workload mix; and an explicit latency-cost Pareto frontier that replaces the usual point estimates with a continuous operating curve.

The remainder of the paper is organized as follows. Section 2 describes the research context and measurement methodology. Section 3 develops the distributed compute architecture and its GPU orchestration layer. Section 4 presents the multi-vector-store knowledge fabric and intelligent query-routing protocol. Section 5 covers global deployment, geographic latency optimization, and the cost-latency frontier. Section 6 synthesizes the contributions and outlines limitations and future work.

## 2. Research Context and Methodology

The empirical portion of this work draws on an eighteen-month observational deployment of a distributed GenAI infrastructure operated across twelve geographic regions. The deployment served a mixed inference-plus-training workload composed of transformer-based LLMs in the seven-billion-to-four-hundred-five-billion parameter range, RAG pipelines backed by multiple vector stores, and document-processing pipelines. We continuously captured measurements at the Kubernetes control-plane layer and at the node-level telemetry layer, giving us time-series visibility into scheduling decisions, queue depths, per-GPU utilization, end-to-end latency percentiles, and operational cost. Published industry benchmarks for GPU utilization and inference throughput were used to contextualize measured values and guard against single-site bias [1] [2].

Three comparison conditions were evaluated. The unified condition treated all GPU capacity as a single scheduler pool. The static multi-tier condition fixed inference, training, and operational tier ratios at the beginning of each monthly window and held them constant. The dynamic multi-tier condition invoked a predictive controller that resized tier boundaries on two-to-four-hour intervals using autoregressive forecasts of arrival patterns and queue-depth gradients. All three conditions shared the same physical hardware, the same model inventory, and the same query mix so that differences attributable to scheduling policy could be isolated from differences attributable to workload.

Key performance indicators were chosen to reflect the operating concerns of enterprise architects rather than the micro-benchmarks typical of systems research. They included monthly infrastructure cost normalized against a unified-pool baseline, p95 and p99 end-to-end inference latency, GPU utilization measured as sustained compute occupancy rather than allocation, service-level agreement (SLA) attainment against a 99.97 percent availability target, and retrieval recall at ten for RAG workloads. This formulation allowed the Pareto structure of cost-versus-latency to be surfaced rather than collapsed into a single scalar [3] [4].

## 3. Distributed Compute Architecture and GPU Orchestration

The dynamic multi-tier architecture preserves the interpretability of static tiers while removing their rigidity. Tier boundaries exist logically, but tier capacities are first-class scheduling variables that a predictive controller revises every two to four hours. The controller consumes three signals: arrival-rate forecasts derived from the previous twenty-eight-day window, real-time queue-depth gradients for each tier, and a workload classifier that tags incoming requests by model family and query shape. The output is a capacity vector that the Kubernetes Dynamic Resource Allocation interface [5] materializes by draining and rebinding GPU devices at the node level without interrupting in-flight inference.

Measured architectural outcomes diverge meaningfully across the three conditions, as summarized in Table 1. Unified pools sustain GPU utilization only in the 62–68 percent band because latency-sensitive inference requests are repeatedly preempted by long-running training jobs and because scheduler fairness policies do not discriminate by SLO class. Static tiers lift utilization to 70–75 percent, but they let capacity sit idle during mismatched demand phases. Dynamic tiers push sustained utilization to 78–81 percent and clear the 68–78 percent band reported in vendor benchmarks for well-tuned homogeneous workloads [1]. The dynamic condition offers a cost advantage of 45–60 percent of the unified baseline, depending on the workload mix, primarily due to this utilization gap rather than pricing leverage.

| Architecture | Monthly Cost (indexed) | p95 Latency | GPU Utilization | SLA Attainment |
|---|---|---|---|---|
| Unified pool | 100 | Baseline | 62–68% | 97.80% |
| Static tiers (inf/train/ops) | 72–78 | −12 to −18% | 70–75% | 99.20% |
| Dynamic tiers (real-time reallocation) | 45–60 | −25 to −35% | 78–81% | 99.97% |

Table 1: Architecture comparison: unified pool vs. static tiers vs. dynamic tiers

### 3.1 Effect of Model Family and Workload Mix

Optimal tier ratios are not universal. They shift with model family and workload mix in ways that static partitioning cannot anticipate. Table 2 summarizes the ratios that minimize total cost subject to a fixed p95 latency budget across four representative workload profiles. For pure LLM inference in the seven-to-seventy-billion parameter band, the optimal inference tier occupies 60–70 percent of capacity. For training-heavy workloads the inference tier shrinks to 45–55 percent. For balanced inference-plus-RAG profiles, the operational tier grows to 20–25 percent because vector search and orchestration consume non-trivial compute. These ratios echo the statistical-multiplexing argument in AlpaServe [6] and the iteration-level scheduling insight in Orca [7]: capacity optimally sized for one model family is rarely optimal for another, and fixed partitioning leaves value on the table.

The GPU orchestration layer inherits this reasoning. PagedAttention-style key-value (KV) cache management [8] substantially changes the memory footprint of concurrent LLM requests, and the orchestrator must account for memory pressure rather than only compute occupancy when making placement decisions. Chunked-prefill scheduling [9] further decouples the compute-intensive prefill phase from the memory-bound decode phase, and the tier controller exposes these phases as separately schedulable units. The net effect is that scheduler-level decisions and model-server-level decisions converge, and the artificial boundary between cluster scheduling and model serving that persists in many deployments is eliminated.

| Workload profile | Optimal inference tier | Training tier | Operational tier | Cost savings vs. unified |
|---|---|---|---|---|
| LLM inference 7B–70B params | 60–70% | 20–25% | 10–15% | 25–30% |
| Training/fine-tuning heavy | 45–55% | 35–40% | 10–15% | 45–50% |
| Balanced inference + RAG | 55–65% | 15–20% | 20–25% | 35–40% |
| Batch inference + document pipelines | 40–50% | 10–15% | 40–50% | 30–35% |

Table 2: Optimal tier allocation ratios by workload profile

## 4. Multi-Vector-Store Knowledge Fabric and Query Routing

RAG pipelines at enterprise scale expose a retrieval bottleneck that single-store vector databases cannot resolve. A single dense vector index offers 94–96 percent recall at ten on representative semantic queries but imposes a p95 latency of 142 milliseconds because every query traverses the full index regardless of its semantic character. Keyword queries, metadata-filtered queries, and hybrid semantic-plus-keyword queries are each best served by a different index type, and collapsing them onto a single store either degrades recall or inflates latency. The multi-vector-store knowledge fabric addresses this by operating dense, sparse, and metadata indexes in parallel with a learned router that directs each query to the subset of stores most likely to produce relevant results [10].

Table 3 summarizes the empirical effect. A static fan-out that always queries every store improves recall to 97–98 percent and reduces mean latency to 52 milliseconds but incurs a query cost that is 15 percent higher than the single-store baseline because every index is exercised for every query. An intelligent router, implemented as a two-hundred-token bidirectional encoder representation from a

transformer (BERT) classifier, routes each query to the one or two stores most likely to answer it and achieves 99.2 percent recall at 40 milliseconds mean latency with a 67 percent reduction in query cost. The classifier adds under two milliseconds of overhead per query, which is negligible against the p95 savings. These gains compound with the compute-tier improvements in Section 3 because reduced retrieval latency directly shortens the end-to-end inference critical path.

| Retrieval topology | Mean latency (ms) | p95 latency (ms) | Recall @ 10 | Query cost (indexed) |
|---|---|---|---|---|
| Single dense vector store | 65 | 142 | 94–96% | 100 |
| Static multi-store (fan-out) | 52 | 118 | 97–98% | 115 |
| Multi-store + intelligent routing | 40 | 88 | 99.20% | 33 |

Table 3: Retrieval performance across single-store, static fan-out, and intelligent-routing topologies

The underlying similarity-search primitives remain conventional. The dense index uses product-quantized nearest-neighbor search [11] over 384-dimensional embeddings, and the sparse index uses a standard inverted-index engine. The novelty is structural, not algorithmic: the fabric treats retrieval as a routed multi-store problem rather than a monolithic store problem, and the router is trained on the enterprise query distribution rather than on public benchmarks. This mirrors the broader trend in RAG systems research [10] [12] toward architectures that separate the retrieval policy from the retrieval mechanism.

**5. Global Deployment and the Latency-Cost Pareto Frontier**

Geographic distribution is the third lever in the enterprise GenAI infrastructure stack, and its effects are frequently understated in centralized-deployment analyses. Centralized deployments serving a global user base incur p95 latencies near 420 milliseconds and tail variance of approximately plus or minus 180 milliseconds because of wide-area network traversal and cross-region congestion. A three-region active-active deployment cuts p95 latency to 185 milliseconds and tail variance to plus or minus 62 milliseconds, and a twelve-region deployment coupled with intelligent content-aware routing further compresses p95 latency to 118 milliseconds and tail variance to plus or minus 18 milliseconds. These outcomes are summarized in Table 4 and are consistent with edge-computing measurements reported in [13] [14].

| Deployment topology | p95 latency (ms) | Tail variance (±ms) | Infra overhead | Failover (s) |
|---|---|---|---|---|
| Centralized (1 region) | 420 | ±180 | Baseline | >300 |
| 3-region active-active | 185 | ±62 | 42% | 90–120 |
| 12-region + intelligent routing | 118 | ±18 | 108% | 30–60 |

Table 4: Geographic deployment topology vs. latency and cost

The cost of geographic distribution is non-trivial. A twelve-region topology carries approximately 108 percent additional infrastructure overhead relative to a single-region baseline. Whether that overhead is justified depends on the latency elasticity of the user population; published studies report user

abandonment rates of 15–25 percent at latencies above 300 milliseconds for interactive AI workloads, which locates the crossover point for most enterprise deployments well below the twelve-region configuration [15]. The cost-latency trade-off is therefore not a choice between extremes but a continuous frontier whose shape the architect must characterize before choosing an operating point.

A critical observation from the eighteen-month deployment is that pushing cost reduction beyond 50 percent of the unified baseline triggers a sharp 40–60 percent p95 latency increase. The Pareto-optimal operating band lies at 40–52 percent cost reduction with latency increase held under 25 percent relative to the best-latency configuration. Beyond this band, further cost cuts produce non-linear latency penalties that dominate any savings. This frontier is architecture-specific but is stable with respect to workload mix over the study period, which suggests that enterprise architects can compute it once and use it as a standing decision aid rather than rediscovering it for each capacity plan. Predictive autoscaling based on long short-term memory (LSTM) forecasting of arrival patterns [16] is the mechanism by which the system holds the chosen operating point under diurnal and weekly load variation.

## Conclusion

Dynamic multi-tier compute segregation, a routed multi-vector-store knowledge fabric, and geographic distribution with explicit Pareto-frontier reasoning together create a clear reference architecture for enterprise-scale generative AI (GenAI) deployment. Dynamic tier control delivers a 40–55 percent infrastructure cost reduction relative to unified pools while improving p95 latency consistency by 25–35 percent; intelligent query routing across a multi-store fabric reduces retrieval latency by 40–55 percent and raises recall to 99.2 percent; and twelve-region deployment with content-aware routing compresses tail variance sixfold relative to a centralized baseline. These outcomes are mutually reinforcing rather than additive, and the compound effect is larger than the sum of its parts.

The broader contribution is methodological. Enterprise GenAI infrastructure decisions are frequently made as isolated point choices: a GPU count, a vector database vendor, a region footprint. This paper demonstrates that those choices are coupled and that treating them as coordinates on a continuous cost-latency frontier produces measurably better outcomes than treating them as independent levers. The frontier itself is a useful artifact because it allows architects to commit to an operating point with visibility into the trade-offs that this point implies, rather than discovering those trade-offs in production. Future work should extend the analysis to different types of accelerator mixes, including next-generation GPUs and application-specific integrated circuits (ASICs), and should formalize the interaction between the predictive tier controller and the increasingly autonomous agentic workloads that are likely to dominate enterprise inference demand over the next operating cycle.

## References


[1] Vijay Janapa Reddi, et al., "MLPerf Inference Benchmark," arXiv, 2020. [Online]. Available: https://arxiv.org/pdf/1911.02549

[2] Zhisheng Ye, et al., "Deep learning workload scheduling in GPU datacenters: a survey," ACM Computing Surveys, 2024. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3638757

[3] Yunfan Gao, et al., "Retrieval-augmented generation for large language models: a survey," arXiv, 2024. [Online]. Available: https://arxiv.org/pdf/2312.10997

[4] W. Zhao et al., "A survey of large language models," arXiv, 2026. [Online]. Available: https://arxiv.org/pdf/2303.18223

[5] Kubernetes, "Dynamic resource allocation (KEP-4381)." [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/dynamic-resource-allocation/

[6] Z. Li et al., "AlpaServe: statistical multiplexing with model parallelism for deep learning serving," arXiv, 2023. [Online]. Available: https://arxiv.org/pdf/2302.11665

[7] G. Yu et al., "Orca: a distributed serving system for transformer-based generative models," Open access to the Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation, 2022. [Online]. Available: https://www.usenix.org/system/files/osdi22-yu.pdf

[8] W. Kwon et al., "Efficient memory management for large language model serving with PagedAttention," arXiv, 2023. [Online]. Available: https://arxiv.org/pdf/2309.06180

[9] Arney Agrawal, et al., "Efficient LLM inference via chunked prefills," ACM SIGOPS Operating Systems Review, 2025. [Online]. Available:

https://dl.acm.org/doi/epdf/10.1145/3759441.3759444
[10] Wenqi Fan, et al., "A survey on RAG meeting LLMs: towards retrieval-augmented large language models," ACM Digital Library, 2024. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3637528.3671470
[11] Jeff Johnson, et al., "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8733051
[12] Le Ma, et al., "A comprehensive survey on vector databases: storage and retrieval techniques and challenges," arXiv, 2026. [Online]. Available: https://arxiv.org/pdf/2310.11703
[13] En Li, et al., "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," IEEE Transactions on Wireless Communications, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8876870
[14] Kiran Bhat, et al., "Edge computing media cache infrastructure for accelerated AI/ML inference," 2025 12th International Conference on Future Internet of Things and Cloud (FiCloud), 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11205197
[15] Minrui Xu, et al., "Unleashing the Power of Edge-Cloud Generative AI in Mobile Networks: A Survey of AIGC Services," IEEE Communications Surveys & Tutorials, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10398474
[16] Haitao Yuan, et al., "An improved LSTM-based prediction approach for resources and workload in large-scale data centers," IEEE Internet of Things Journal, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10486896