Feature

Kubernetes meets GenAI: evaluating performance for AI inference workloads

As GenAI inference becomes a dominant workload, researchers are evolving new Kubernetes-native projects to address bottlenecks, delivering the scalability and efficiency AI workflows require.

by Sai Sindhur Malleni, Raúl Sevilla Canavate, Aleksei Vasilevskii, José Castillo Lema, and André Bauer

The rapid rise of generative AI has fundamentally reshaped cloud computing. From sophisticated training pipelines to high-throughput, real-time inference, modern AI workloads demand dynamic access to expensive, heterogeneous resources, particularly GPUs and specialized accelerators. Kubernetes, already the de facto standard for container orchestration, is increasingly being asked to manage these demanding workloads. But can it keep up? This question is particularly pressing as organizations—from research institutions sharing GPU clusters across teams to sovereign AI initiatives building domestic inference capacity—seek open source, vendor-neutral solutions to maximize the value of scarce accelerator resources.

In our new paper, "Evaluating Kubernetes performance for GenAI inference: from automatic speech recognition to LLM summarization," presented at the 17th ACM/SPEC International Conference on Performance Engineering (ICPE '26) in Florence, Italy, we set out to answer that question. The paper received the Best Industry Paper Award at the conference.

Building on our previous work on multicluster networking performance in Kubernetes (presented at ICPE '25), this study shifts the focus to a different frontier: how well the Kubernetes ecosystem can support the unique demands of generative AI inference. While traditional Kubernetes scheduling and resource management APIs work well for microservices, they were not originally designed for fine-grained GPU resource slicing, complex batch scheduling, or the specific traffic patterns of model serving endpoints.

RESEARCH QUARTERLY
VOLUME 8:1

THE CHALLENGE: AI WORKLOADS PUSH KUBERNETES BEYOND ITS COMFORT ZONE

AI inference workloads differ fundamentally from the microservices Kubernetes was originally designed to orchestrate. They require batch scheduling with priorities and preemption, dynamic partitioning of expensive GPU resources, and intelligent request routing that accounts for model-specific characteristics like key-value (KV) cache state. The standard Kubernetes primitives, while extensible, do not address these needs out of the box.

To bridge this gap, we evaluated three complementary Kubernetes-native projects that each address a distinct aspect of the AI inference pipeline:

- Kueue for job queuing and batch scheduling
- Dynamic Accelerator Slicer (DAS) for GPU resource partitioning
- Gateway API Inference Extension (GAIE) for intelligent inference routing

THREE KUBERNETES-NATIVE PROJECTS, ONE COHESIVE PLATFORM

Kueue is an open source project under the Kubernetes Scheduling Special Interest Group that extends native Kubernetes scheduling to support advanced batch and job management. It introduces a hierarchical queuing model with LocalQueues and ClusterQueues, enabling cluster operators to define resource boundaries, priority classes, and admission controls. This is particularly valuable for AI workloads where multiple teams share expensive GPU resources and need fair, predictable scheduling.

The Dynamic Accelerator Slicer (DAS) tackles GPU resource efficiency. In AI inference deployments, statically pre-slicing GPUs often leads to fragmentation and underutilization. DAS provides on-demand, just-in-time GPU slice allocation driven by actual workload requirements. Using NVIDIA MIG (Multi-Instance GPU) technology, it dynamically creates and destroys GPU partitions to match workload demands, improving utilization and reducing idle costs.

The Gateway API Inference Extension (GAIE) brings inference-aware networking to Kubernetes. Built on the standard Gateway API, it introduces domain-specific constructs for generative AI: model versioning, adaptive backend selection, and intelligent, prefix-cache-aware routing. Together with llm-d, a Kubernetes-native, high-performance distributed LLM inference framework, GAIE routes requests to the most suitable backend based on live model and accelerator metrics, including queue length, prefix cache hits, and LoRA adapter availability.

A REAL-WORLD USE CASE: FROM AUDIO TRANSCRIPTION TO SUMMARIES

To evaluate how these three components work together, we designed a multi-stage AI inference pipeline that reflects common industry usage patterns. The workflow processes a dataset of corporate earnings calls through two stages:

1. **Automatic Speech Recognition (ASR)**: Audio files are transcribed using OpenAI's Whisper models (medium and large variants), managed as batch jobs on Kubernetes.
2. **LLM Summarization:** The resulting transcripts are fed to a Qwen3-8B large language model for summarization, served through an OpenAI-compatible API endpoint.

We used the Earnings 22 dataset, comprising 125 financial earnings calls from various global companies, selecting a subset of 32 files for our experiments. All experiments ran on Red Hat OpenShift Service on AWS (ROSA) with a single GPU node featuring 8 NVIDIA A100 GPUs. We used kube-burner for workload orchestration and GuideLLM for benchmarking the LLM inference endpoint.

WHAT WE FOUND

Our evaluation yielded concrete, measurable improvements at each stage of the pipeline.

Scheduling with Kueue

Kueue's priority and preemption mechanisms reduced the total makespan—the time for all 32 transcription jobs to complete—by up to 15% compared to the baseline. More importantly, Kueue delivered deterministic, fair, and reproducible scheduling with negligible admission latency overhead (below 25 ms

Configuration	Makespan	Queue Time	Exec Time	Medium Jobs		Large Jobs	
	(Med, (Min–Max))	(Med / P95)	(Med)	Queue (Med)	Exec (Med)	Queue (Med)	Exec (Med)
Pure Jobs (Baseline)	60.5 (59.1–60.5)	0.0 / 0.0	28.4	0.0	23.6	0.0	35.3
Kueue BestEffortFIFO	62.5 (62.4–65.8)	16.7 / 39.5	9.4	15.9	6.5	17.3	17.8
+ Priority	54.7 (52.2–56.1)	19.5 / 44.5	9.5	39.1	6.7	10.2	17.1
+ Preemption	51.1 (49.2–52.7)	25.0 / 41.9	8.9	36.7	6.4	3.7	15.6
2 ClusterQueues + Borrowing	55.0 (54.2–60.0)	13.7 / 32.4	8.7	10.2	6.7	25.1	18.0

Table 1. *Performance comparison of Kueue Configurations (96 jobs across 3 runs per configuration). All values are in minutes. Makespan median computed from 3 run-level makespans comprising 32 jobs per run; other medians computed from 96 jobs (32 jobs across 3 test runs) from each configuration.*

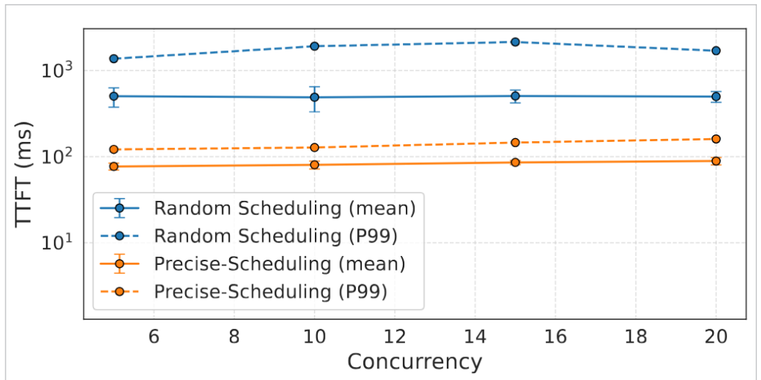

Figure 1. *Comparison of the Time to First Token (TTFT). Error bars indicate 95% confidence interval. Y-axis is depicted in log-scale.*

even at the 99th percentile). The configuration with two ClusterQueues and resource borrowing struck the best balance between fairness and utilization, allowing underutilized GPUs to be dynamically reallocated across queues. As shown in the following table, while the baseline avoids queuing delays, its median execution time is considerably higher (28.4 vs. 8.7–9.5 minutes) because jobs contend for resources without admission control, while Kueue's managed admission trades modest queue wait times for significantly faster and more predictable execution (see **Table 1**).

GPU slicing with DAS

DAS increased the number of concurrently running transcription jobs from 8 (one per physical GPU) to 25 (using MIG slices), reducing mean job completion time by 36%—from 28 to 18 minutes. While individual jobs ran slightly slower on GPU slices compared to full GPUs, the dramatically increased parallelism more than compensated, resulting in significantly better overall throughput. GPU tensor core utilization and DRAM activity both tripled with DAS enabled.

Intelligent routing with llm-d

llm-d's precise-prefix-cache scheduling strategy, which leverages KV-cache events to drive inference requests to replicas with matching prefix tokens, delivered striking improvements over random scheduling thanks to an improved vLLM's KV-cache hit rate. The Time to First Token (TTFT) improved by approximately 6.25 times (from ~500 ms to ~0 ms), and end-to-end request latency dropped by up to 6 seconds at the 99th percentile. Precise-prefix-cache scheduling also exhibited significantly lower variability, suggesting better predictability under high concurrency (see **Figure 1**).

WHAT COMES NEXT

This work demonstrates that Kubernetes, when extended with emerging AI-oriented projects, can effectively serve as a unified substrate for both batch and online GenAI workloads. The complementary strengths of these three components—Kueue for scheduling fairness, DAS for GPU efficiency, and GAIE for intelligent routing—form a cohesive, high-performance platform for demanding AI inference pipelines. These results carry broad implications beyond our specific benchmarks. As GPU resources remain scarce and expensive, the ability to efficiently partition and schedule accelerators through open source Kubernetes-native tooling is critical—whether for research groups sharing a cluster, enterprises optimizing inference costs, or sovereign AI programs building domestic AI infrastructure on transparent, auditable foundations.

Future work will focus on extending the evaluation to multimodal and streaming inference scenarios, integrating emerging components like gang scheduling and Dynamic Resource Allocation (DRA), and positioning this framework as a foundational blueprint for end-to-end AI workload orchestration on cloud-native infrastructure.

LEARN MORE

The full paper is available on the ACM Digital Library. All necessary files and configuration for replicating the test setup can be found on our GitHub repository (red.ht/icpe26-k8s-ai-apis).

About the Authors

Sai Sindhur Malleni is currently a Senior Engineering Manager at NVIDIA, where he leads initiatives focused on establishing Kubernetes as the common substrate for AI training and inference across the industry. His work centers on advancing open source technologies that make GPU infrastructure reliable, elastic, and seamless to operate at scale. Previously he spent over a decade at Red Hat advancing OpenShift performance.

Raúl Sevilla Canavate is a Performance Engineer at Red Hat with a wide experience in Linux, automation, tooling, and performance analysis within the Kubernetes ecosystem for the last eight years. In recent years, he has been focused on pushing Kubernetes and OpenShift to its limits through CNCF's hosted kube-burner project.

Aleksei Vasilevskii is a Senior Systems Software Engineer at NVIDIA with a broad experience optimizing the performance and reliability of high-traffic applications running on Kubernetes/OpenShift. His work centers on maximizing the throughput, resilience, and operational efficiency of Kubernetes-based environments tailored for GPU-accelerated workloads.

José Castillo Lema is a Performance Engineer in the OpenShift Ecosystem Performance/Scale team. He has been designing and implementing IaaS/PaaS solutions, namely OpenStack and Kubernetes/OpenShift, for the last ten years and teaching postgraduate courses for the last ten years.

André Bauer is an Assistant Professor in the Department of Computer Science at the Illinois Institute of Technology. Previously he was a postdoctoral researcher in computer science at the University of Chicago and a researcher at Argonne National Laboratory. His research is on providing intuitive, efficient, and sustainable data science solutions across all disciplines and domains.