Accelerate your generative AI inferencing by choosing Google Kubernetes Engine

Compared to a competing Amazon EKS solution, the Google solution—with its optimized GKE Inference Gateway—delivered greater token throughput and lower latency

- **15.7% greater output token throughput***
- **92.8% lower mean time to first token (TTFT)***
- **62.6% lower mean inter-token latency***

*GKE with GKE Inference Gateway on the Llama 3.1-8B Instruct model vs. Amazon EKS on the Llama 3.1-8B Instruct model

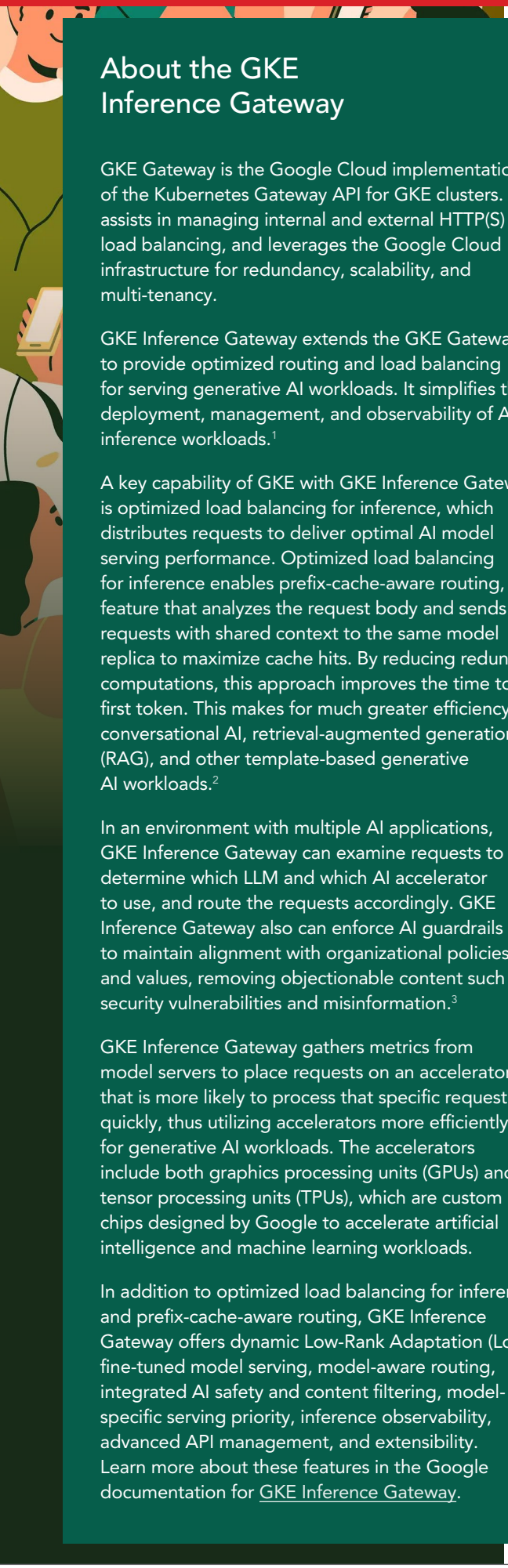
Executive summary

Google Kubernetes Engine (GKE) with GKE Inference Gateway outperformed Amazon Elastic Kubernetes Service (EKS) in inference performance tests using the Kubernetes inference-perf tool on the Llama 3.1-8B-it model. The two cloud environments differed only how the solutions distributed inference requests over their eight NVIDIA A100 40GB GPUs. GKE used the intelligent inference-aware GKE Inference Gateway instead of a standard HTTP load balancer.

The Google solution delivered greater throughput and better latency. Specifically, GKE with GKE Inference Gateway on Google Cloud delivered approximately 15.7 percent higher token throughput, a 92.8 percent reduction in time-to-first-token (dramatically faster perceived response start), a 62.6 percent reduction in inter-token latency (smoother streaming), and improved tail-latency behavior under increasing requests-per-second (RPS) rates.

Why does faster inference matter?

Companies using generative AI can choose from many platforms to carry out inferencing tasks. If you're deploying interactive AI models such as chatbots, creative tools, or customer support systems, a platform that delivers better performance means that users perceive a system as more responsive and faster. Faster inference can also improve scalability for demanding applications and lower operational costs by maximizing hardware efficiency and throughput, allowing the system to process more requests in less time.



About the GKE Inference Gateway

GKE Gateway is the Google Cloud implementation of the Kubernetes Gateway API for GKE clusters. It assists in managing internal and external HTTP(S) load balancing, and leverages the Google Cloud infrastructure for redundancy, scalability, and multi-tenancy.

GKE Inference Gateway extends the GKE Gateway to provide optimized routing and load balancing for serving generative AI workloads. It simplifies the deployment, management, and observability of AI inference workloads.[1]

A key capability of GKE with GKE Inference Gateway is optimized load balancing for inference, which distributes requests to deliver optimal AI model serving performance. Optimized load balancing for inference enables prefix-cache-aware routing, a feature that analyzes the request body and sends requests with shared context to the same model replica to maximize cache hits. By reducing redundant computations, this approach improves the time to first token. This makes for much greater efficiency in conversational AI, retrieval-augmented generation (RAG), and other template-based generative AI workloads.[2]

In an environment with multiple AI applications, GKE Inference Gateway can examine requests to determine which LLM and which AI accelerator to use, and route the requests accordingly. GKE Inference Gateway also can enforce AI guardrails to maintain alignment with organizational policies and values, removing objectionable content such as security vulnerabilities and misinformation.[3]

GKE Inference Gateway gathers metrics from model servers to place requests on an accelerator that is more likely to process that specific request quickly, thus utilizing accelerators more efficiently for generative AI workloads. The accelerators include both graphics processing units (GPUs) and tensor processing units (TPUs), which are custom chips designed by Google to accelerate artificial intelligence and machine learning workloads.

In addition to optimized load balancing for inference and prefix-cache-aware routing, GKE Inference Gateway offers dynamic Low-Rank Adaptation (LoRA) fine-tuned model serving, model-aware routing, integrated AI safety and content filtering, model-specific serving priority, inference observability, advanced API management, and extensibility. Learn more about these features in the Google documentation for GKE Inference Gateway.

We tested an AI inference application using eight NVIDIA A100 40GB GPUs on two cloud platforms to determine how the extra functionality offered by GKE affected performance:

- **Google Kubernetes Engine (GKE)** in conjunction with the GKE Inference Gateway
- **Amazon Elastic Kubernetes Service (EKS)** with its standard HTTP application load balancer

We used the Kubernetes inference-perf benchmark to evaluate both solutions for the Llama 3.1-8B Instruct LLM, capturing several performance and latency metrics. To learn about these metrics, see the Glossary of inference performance metrics at the end of this document.

We found that GKE with GKE Inference Gateway outperformed Amazon EKS on all measures, delivering higher throughput, lower latency, and better stability under load. These results show that choosing GKE with GKE Inference Gateway can empower businesses to innovate faster and provide highly responsive AI-driven services.

About our testing

We used the Shared prefix use case in the Kubernetes inference-perf benchmark to evaluate both solutions for the Llama 3.1-8B Instruct LLM.

To test prefix-aware routing, the solutions in this use case process artificially generated requests that begin with correlated text. Prefix-aware routing is most beneficial in applications where the requests tend to begin with similar text, such as systems that answer questions about long documents or manuals where users frequently ask multiple, related questions. The inference-perf tool generates requests with prefixes it draws from a statistical model.

We used cloud-specific vLLM tuning sets. For more detail on our testing, see the science behind the report.

What we found

Higher output token throughput and lower mean latency

As Figure 1 shows, GKE with GKE Inference Gateway exhibits substantially lower mean normalized time per output token (TPOT), or latency, than Amazon EKS across multiple requests-per-second points. For example, at roughly 100 RPS, GKE delivers mean latency of 18.8 ms vs. 31.4 ms for Amazon EKS. At higher load points where Amazon EKS latency increases sharply (116 to 169 ms), GKE remains much lower (roughly 22 to 70 ms). GKE maintains consistently lower average end-to-end token latency and is more stable under load. This indicates that GKE with GKE Inference Gateway can handle more inference requests without large increases in latency.

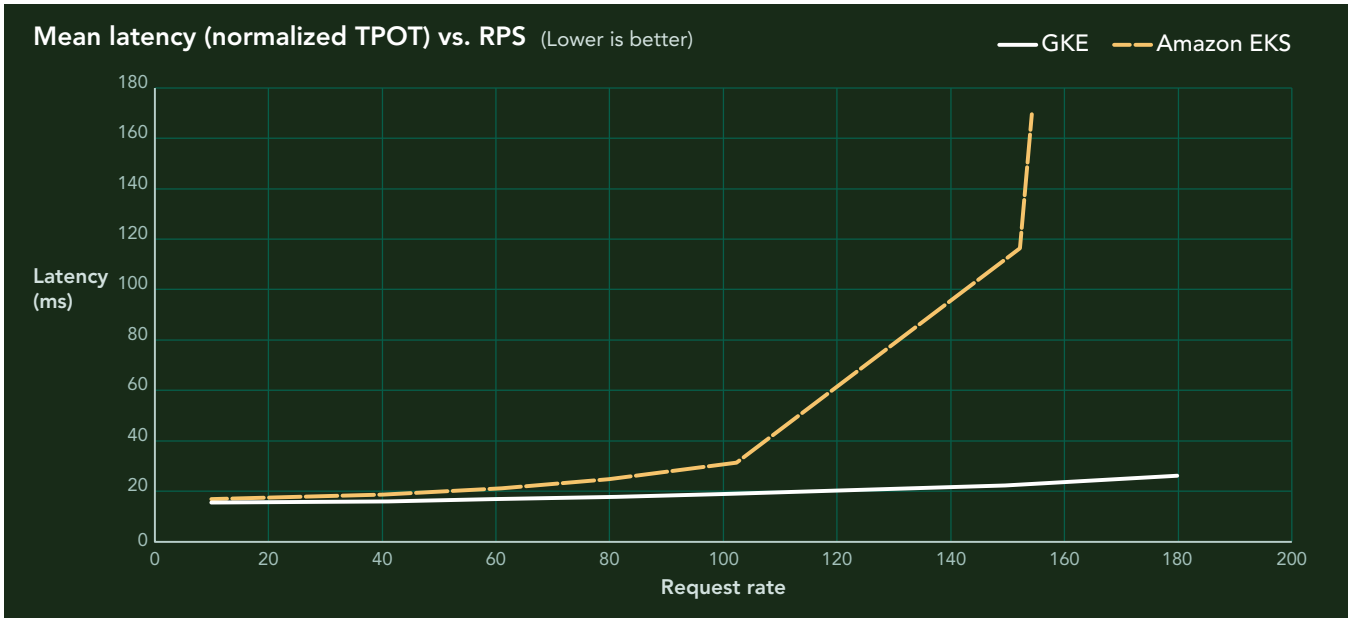


Figure 1: Mean latency (normalized time per output token) of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Lower is better. Source: PT.

The inference-perf tool analyzes the throughput-latency tradeoff (see Figure 1) to compute the request rate that yields the highest output-token throughput before its latency starts to rapidly increase. The metrics we present in the rest of this section and the following section are from that request rate.

As Figure 2 shows, GKE also achieved 15.7 percent higher output token throughput on the same hardware, which means that it processed more requests and tokens per second per deployment. So, by selecting GKE, your organization could gain either increased capacity for the same hardware footprint or reduced hardware needs for the same workload. Table 1 provides a more complete picture of token throughput.

Note: The graphs in this report use different scales to keep a consistent size.
Please be mindful of each graph's data range as you compare.

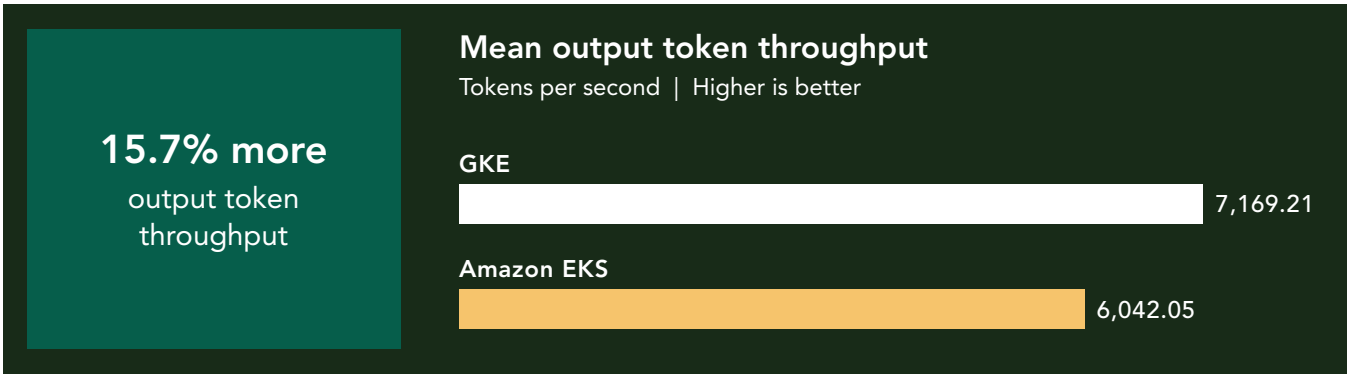


Figure 2: Mean output token throughput of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Source: PT.

Table 1: Mean token throughputs and RPS of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Higher is better. Source: PT.

	Mean token throughput (tokens per second)			Requests/sec
	Input	Output	Total	
GKE	197,868.63	7,169.21	205,037.84	115.68
Amazon EKS	166,639.92	6,042.05	172,681.97	97.51
GKE advantage	**15.78%**	**15.72%**	**15.78%**	**15.70%**

Faster time to first token and lower inter-token latency improve end-user responsiveness

Lower time to first token (TTFT) and lower inter-token latency (ITL) with GKE reflect faster response beginnings and quicker token generation, which directly improve perceived latency for end-users interacting with LLM-based features such as chat responses and streaming outputs.

Table 2 shows several different mean latency metrics. GKE with GKE Inference Gateway delivers substantially lower latencies across every metric—notably a dramatic reduction in time to first token and inter-token latency.

Table 2: Mean latency in ms of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Lower is better. Source: PT.

	Mean latency (ms)			
	Normalized time per output token	Time per output token	Time to first token	Inter-token latency
GKE	35.04	30.29	188.36	30.29
Amazon EKS	128.07	81.03	2,624.73	81.03
GKE advantage	**72.64%**	**62.62%**	**92.82%**	**62.62%**

As Figure 3 shows, GKE delivered dramatically faster mean TTFT, cutting response time by 92.8 percent. Dramatically faster TTFT means responses begin much sooner on GKE, which improves perceived latency for interactive scenarios such as chat or streaming outputs.

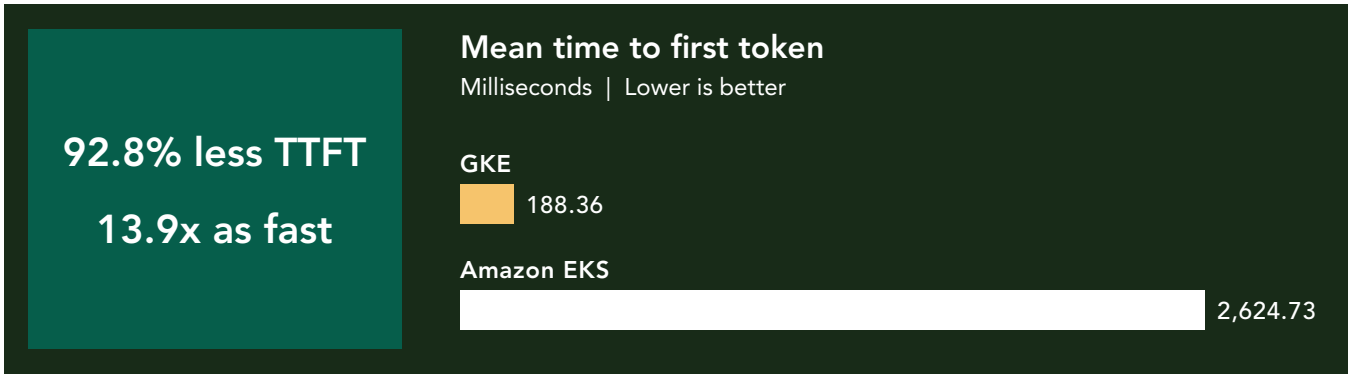


Figure 3: Mean time to first token of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Source: PT.

As Figure 4 shows, mean inter-token latency on GKE was also much better, by 62.6 percent. Lower ITL yields smoother token emission after the first token, improving streaming quality.

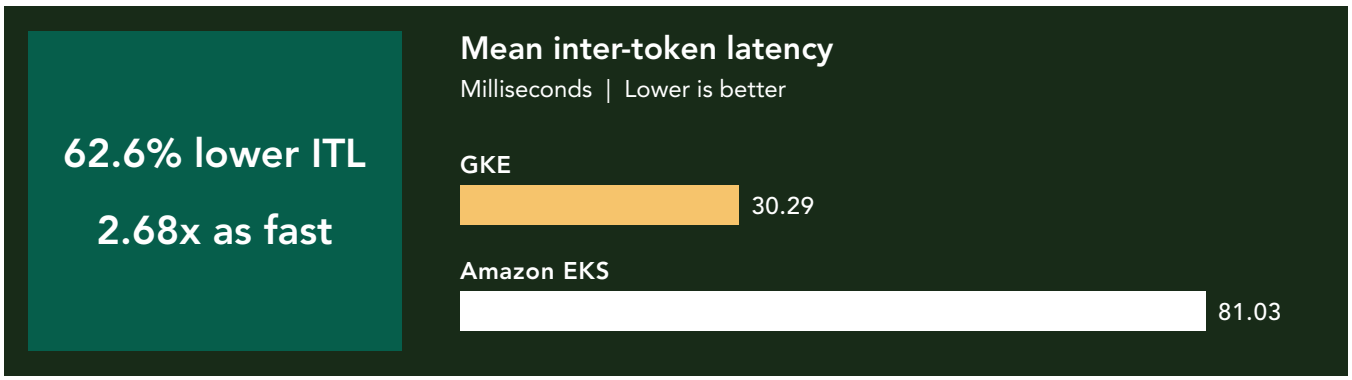


Figure 4: Mean inter-token latency of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Source: PT.

Lower tail latency

The tail latency metric reflects the slowest end-to-end response times a system actually delivers. In other words, it is measured at the highest percentiles of request latency. It indicates how long the slowest few requests take to complete, even when most requests are faster. It's important to know whether some users might experience a substantial delay. For example, saying that a system handled 1,000 requests with 99th percentile latency of 800 ms means that 990 of the requests were faster than 800 ms, but the slowest 10 requests took 800 ms or longer.

Tail latency affects the user experience because even when average latency is low, the users whose requests take the longest experience the application as slow or unresponsive. Tail latency can also affect overall system performance because in distributed systems such as machine learning inference pipelines, one slow component can delay the entire response. Service-level agreements and objectives (SLAs and SLOs) often use tail latency because for some situations it is a more useful gauge of real-world performance than average latency. The inconsistent response times that result from high tail latency are especially problematic for real-time applications such as chatbots, fraud detection, and personalized recommendations.

On the Shared prefix use case, 95th percentile tail latency on GKE was up to 83.9 percent lower than on Amazon EKS with standard HTTP load balancer (see Figure 5). Based on these results, companies using AI workloads where requests begin with similar content would deliver a better experience to most users by choosing GKE with GKE Inference Gateway.

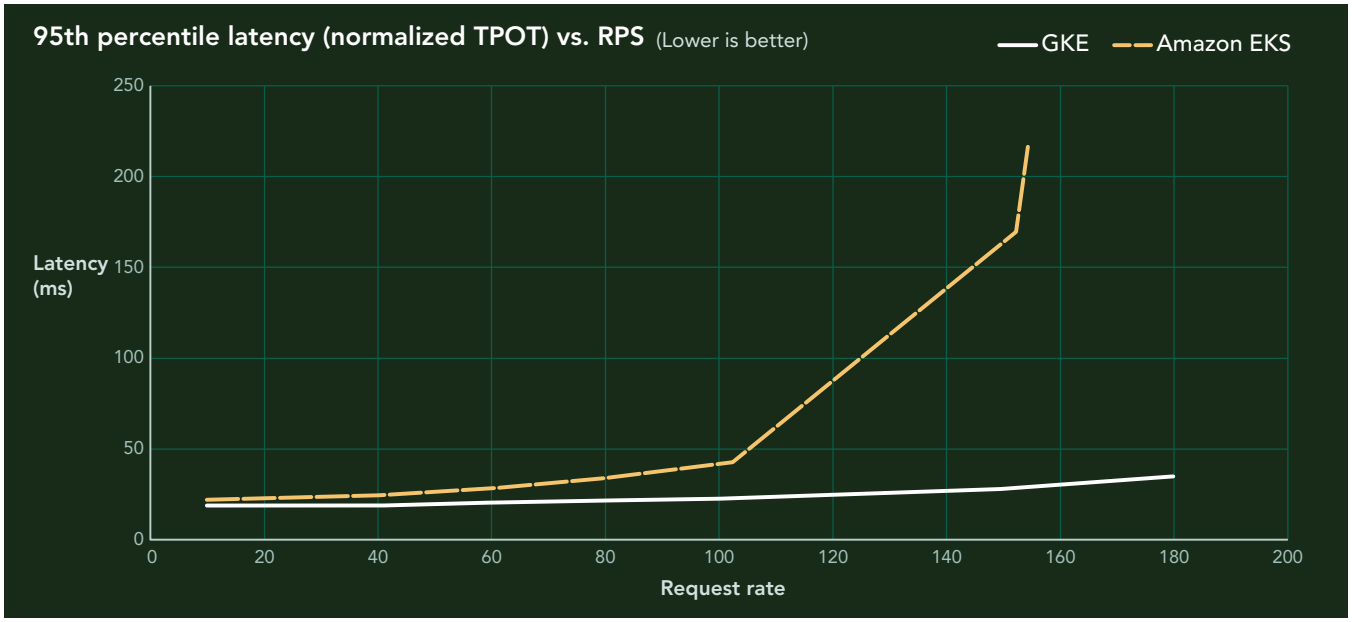


Figure 5: 95th percentile tail latency of GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Lower is better. Source: PT.

As we noted earlier, the inference-perf tool analyzes the throughput-latency tradeoff (see Figure 1) to compute the request rate that yields the highest output-token throughput before its latency starts to rapidly increase. The metrics we present in the rest of this section are from that request rate.

Figure 6 shows the 95th percentile normalized time per output token (NTPOT), the interval between the user sending the request and receiving the last token. The NTPOT for GKE with GKE Inference Gateway was 67.0 percent lower than that for Amazon EKS.

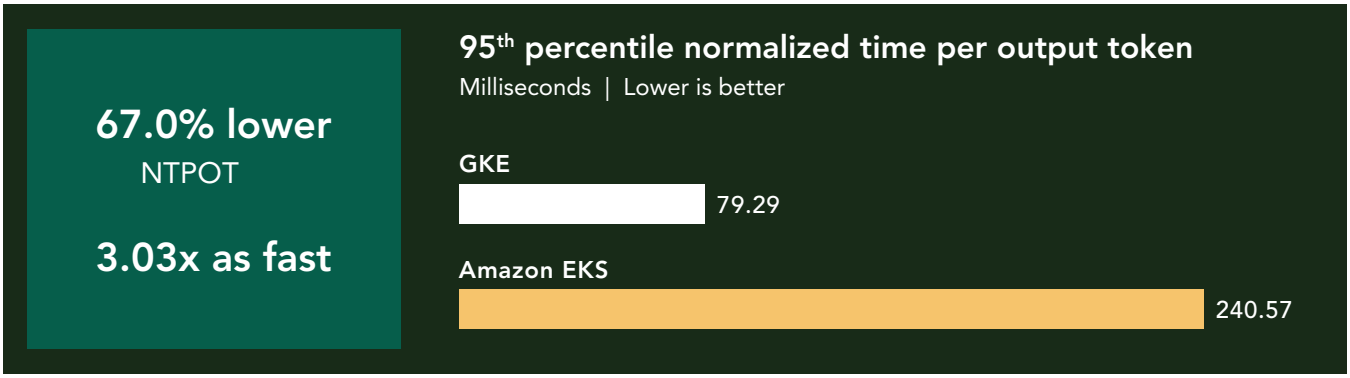


Figure 6: 95th percentile normalized time per output token for GKE with GKE Inference Gateway and Amazon EKS on the Llama 3.1-8B Instruct LLM on the Shared prefix use case. Both solutions used the same hardware. Source: PT.

Glossary of inference performance metrics

The source for the definitions below is Google Cloud Documentation.

Benchmark indicators	Metric	Description
Latency	Time to First Token (TTFT) (ms)	Time it takes to generate the first token for a request.
	Normalized Time Per Output Token (NTPOT) (ms)	Request latency normalized by the number of output tokens, measured as request_latency / total_output_tokens.
	Time Per Output Token (TPOT) (ms)	Time it takes to generate one output token, measured as (request_latency - time_to_first_token) / (total_output_tokens - 1).
	Inter-token latency (ITL) (ms)	Measures latency between two output token generations. Unlike TPOT, which measures latency across the entire request, ITL measures the time to generate each individual output token. These individual measurements are then aggregated to produce mean, median, and percentile values such as p90.
	Request latency (ms)	End-to-end time to complete a request.
Throughput	Requests per second	Total number of requests that you serve per second. Note that this metric might not be a reliable way to measure LLM throughput because it can vary widely for different context lengths.
	Output tokens per second	A common metric that is measured as total_output_tokens_generated_by_server / elapsed_time_in_seconds.
	Input tokens per second	Measured as total_input_tokens_generated_by_server / elapsed_time_in_seconds.
	Tokens per second	Measured as total_tokens_generated_by_server / elapsed_time_in_seconds. This metric counts both input and output tokens, helping you compare workloads with high prefill versus high decode times.

Conclusion

GKE with GKE Inference Gateway delivered superior AI inference performance compared to Amazon EKS using a standard HTTP load balancer on the Llama 3.1 8B Instruct Shared prefix use case. Across identical hardware—eight NVIDIA A100 40GB GPUs—the GKE with GKE Inference Gateway configuration demonstrated clear advantages in both capacity and responsiveness:

- **Higher throughput:** 15.7 percent more tokens processed per second, enabling higher request capacity or reduced hardware needs for the same workload.
- **Much faster time to first token (TTFT):** 92.8 percent shorter wait time, producing dramatically quicker perceived response starts for interactive scenarios.
- **Lower inter-token latency (ITL):** 62.6 percent reduction, resulting in smoother and faster token streaming after the first token.
- **Improved latency stability and tail behavior:** GKE maintained lower mean and high percentile latencies across RPS points and showed fewer extreme slow requests, improving consistency for end users.

These gains are consistent with the features we have discussed—most notably GKE Inference Gateway optimized inference load balancing and prefix cache aware routing—that reduce redundant computation, increase cache hits, and better utilize accelerators. Together, the throughput, latency, and stability improvements translate into a better experience for interactive GenAI applications (chatbots, RAG, streaming responses) and improved infrastructure efficiency for operators.

Companies that rely on workloads where requests commonly share prefixes or benefit from cache locality (for example, document Q&A, multi turn conversations, or template based generation) need high performance. For these workloads, consider GKE with GKE Inference Gateway to improve responsiveness, capacity, and cost efficiency on equivalent GPU hardware.

---

1. Google, "About GKE Inference Gateway," accessed May 5, 2026, https://docs.cloud.google.com/kubernetes-engine/docs/concepts/about-gke-inference-gateway.
2. Google, "About GKE Inference Gateway."
3. McKinsey and Company, "What are AI guardrails?" accessed November 7, 2025, https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-are-ai-guardrails.

This project was commissioned by Google.

Read the science behind the report ▶

Primary contributors

Tech: Dan Sullivan

Writing: Laura W.

Design: Jared White

PM: Claire Ackerman

How we created this report

A PT team, which includes the contributors we've listed and others, created this report and performed the technical work behind it. We used AI to draft portions of the text.

Principled Technologies®

Facts matter.®