The Journal of Open Source Software

Inference Perf: A Benchmarking Tool for GenAI Inference

Ashok Chandrasekar [1], **Sachin Varghese** [2], **Jason Kramberger** [1], **Brendan Slabe** [1], **Chen Wang** [3], and **Yuan Tang** [4]

1 Google, USA **2** Capital One, USA **3** IBM Research, USA **4** Red Hat, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 28 January 2026
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Inference Perf is a generative AI (GenAI) inference performance benchmarking tool aimed at benchmarking and analyzing the performance of inference deployments. It is designed to be model-server agnostic, allowing for apples-to-apples comparisons across different model servers and serving stacks. As part of the inference benchmarking and metrics standardization effort in the Kubernetes `wg-serving` (The Kubernetes Authors, 2024), it seeks to standardize tooling and metrics for measuring inference performance across the Kubernetes and model server communities.

Statement of need

With the rapid adoption of Large Language Models (LLMs) and GenAI, there is a growing need to accurately measure and compare the performance of inference serving systems. Different model servers (e.g., vLLM, TGI, SGLang) and deployment orchestrators (e.g., Kubernetes) introduce substantial variability in performance. Existing tools often lack standardized metrics or GenAI inference specific capabilities like (Grafana Labs, 2021) and (Heyman et al., 2011) or are tightly coupled to specific frameworks like (vLLM Team, 2023), (Hugging Face, 2023b), (NVIDIA, 2024a) where their goal is to provide a tool for developers working on the specific framework to benchmark their system. As a result, it is often hard to reproduce benchmark results across different serving stacks and environments. `inference-perf` addresses this gap by providing a scalable, agnostic, and comprehensive benchmarking suite for GenAI workloads. It supports various real-world and synthetic datasets, different load patterns (e.g., burst, saturation), and integrates with standard cloud-native observability tools like Prometheus allowing it to benchmark both smaller scale systems in development as well as large production-scale deployments orchestrated by Kubernetes. Crucially, it provides a standardized comparison between different model servers and serving stacks across various use cases.

State of the field

There are two kinds of performance benchmarking tools for GenAI inference that are commonly used: 1. Web-based benchmarks like (Grafana Labs, 2021) and (Heyman et al., 2011) 2. Model server benchmarks like (vLLM Team, 2023), (Hugging Face, 2023b) and (NVIDIA, 2024a)

Web-based benchmarks are generic web server benchmarking tools which offer battle-tested way to reliably generate traffic against specific HTTP endpoints. While these can be used to benchmark LLMs and GenAI workloads, they lack the standardized set of metrics that we want to measure with inference often at token level. To measure these token level metrics, streaming request support, tokenizer support and other features specific to the GenAI workload that is being tested are needed. While some of these tools allow extensions, it is restrictive in general and is not ideal for GenAI benchmarking.

Model server benchmarks are geared towards developers of the model server to repeatedly measure performance improvements that are being made to that model server. While these work well for benchmarking GenAI inference, they are very specific to the model servers and don't work well for production workloads where different traffic patterns that simulate real world workloads are needed. Especially to validate autoscaling, load balancing and intelligent routing which are staple features of these production systems.

The main contribution of `infernce-perf` is to provide a standardized model-server agnostic tool that is designed to benchmark production-scale GenAI workloads for various real-world use cases.

Software Design

`inference-perf` is built with a modular architecture comprising several key components:

- **DataGenerator**: Aligns prompt and generation lengths with user input, supporting fixed or variable length tests for use cases like chat completion and summarization including both real world and synthetic datasets.

- **Load Generator**: Generates traffic patterns such as fixed RPS, bursts, or Poisson distributions. It supports multi-process generation for high concurrency which is a critical requirement for benchmarking production-scale systems.

- **Client**: Abstractions for different model servers and protocols (HTTP, gRPC, streaming), ensuring the tool can be extended to support new model servers and protocols. Furthermore, the tool provides native support for the industry-standard OpenAI API, enabling it to benchmark any compatible model server without necessitating modifications.

- **Metrics / Data Collector**: Measures key performance indicators including Time To First Token (TTFT), Time Per Output Token (TPOT), Inter-Token Latency (ITL) and various throughput metrics. It also supports exporting metrics to Prometheus which can be used to visualize metrics using tools like Grafana.

- **Report Generator**: Produces detailed JSON reports with all the metrics collected during benchmarking.

- **Analyzer**: Analyzes the collected metrics and provides insights into the performance of the model server by generating various charts and graphs.

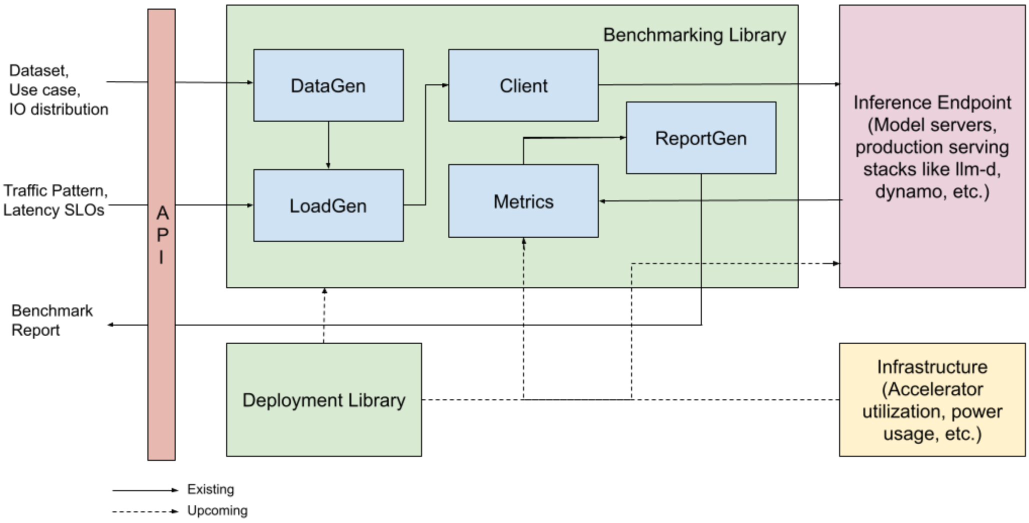

Figure 1: Architecture Diagram

Key Features

- Scalability to support large production deployments with request rate generation up to 10k+ requests per second via a novel multi-process load generator capable of maintaining accurate QPS over longer durations.

- Support for multiple backends including vLLM (Kwon et al., 2023), SGLang (Zheng et al., 2024), and HuggingFace TGI (Hugging Face, 2023a). It is also extensible to support any serving stack which follows the OpenAI API like llm-d (LLM-D Team, 2025b) and NVIDIA Dynamo (NVIDIA, 2024b) which are not model servers, but entire optimized inference stacks with advanced orchestration capabilities.

- Simulation of complex scenarios like multi-turn chat conversations, shared prefix caching and autoscaling.

- Comprehensive metrics collection from both the benchmarking client and the model server to aid in debugging performance issues and discrepancies.

- Observability into the load generated by the benchmarking client. This is important because benchmarking clients can be artificially constrained by external factors like resource contention on client machines, underlying python library limitations, etc. which can lead to performance differences. Being able to observe these limitations is essential.

- Ability to replay traces to mimic production traffic using traces recorded from production and to reproduce the same load pattern in different runs.

Standardized Metrics

`inference-perf` defines the key metrics required to measure inference performance and aims to standardize these metrics and their definitions. The set of metrics measured by `inference-perf` as listed below, provides a comprehensive view of the performance of the inference server in terms of throughput, latency and price-performance. Detailed definitions of the below metrics can be found in (Inference Perf Contributors, 2024b).

DRAFT

Throughput

- Output tokens / second
- Input tokens / second
- Requests / second

Latency

- Time per request (e2e request latency)
- Time to first token (TTFT)
- Time per output token (TPOT)
- Normalized time per output token (NTPOT)

Price-Performance

- Price per million output tokens
- Price per million input tokens
- Throughput per dollar

The above metrics can also be plotted into charts using the analyze command in the tool at various request rates (QPS) to understand how the latency and throughput scales with the load as shown in the below charts.

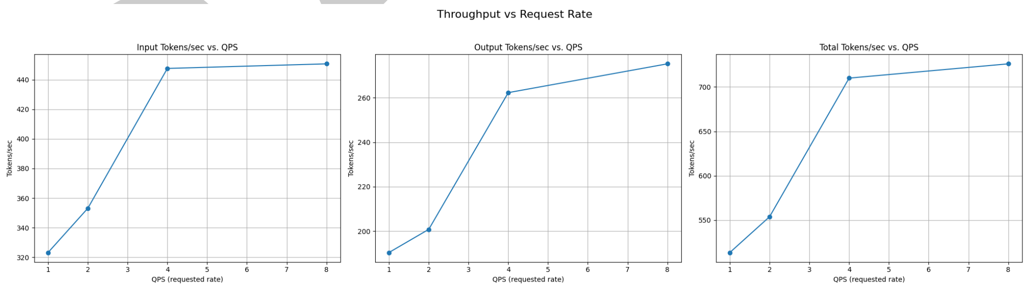

Figure 2: Throughput vs QPS

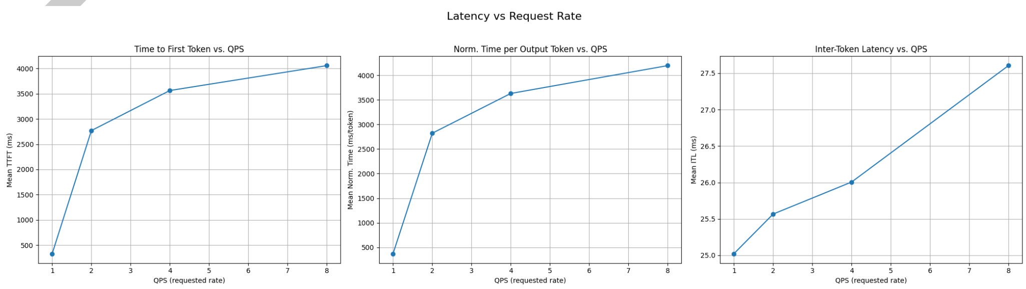

Figure 3: Latency vs QPS

DRAFT

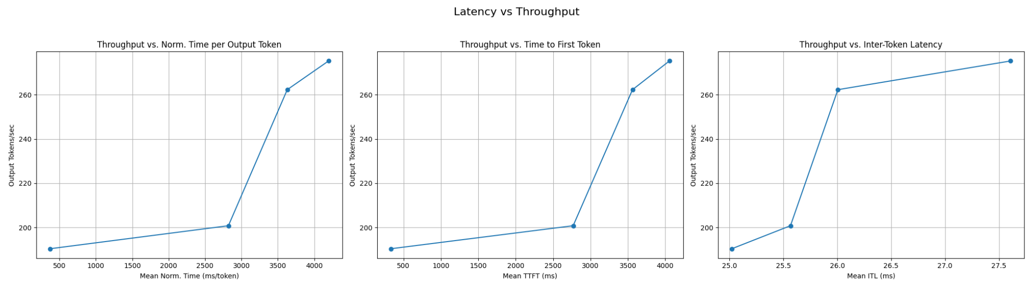

Figure 4: Throughput vs Latency

Research Impact Statement

`inference-perf` is the primary benchmarking tool used by state of the art open source distributed inference frameworks like llm-d which implements novel model serving optimizations for LLM orchestration like multi-host serving, efficient load balancing and autoscaling as seen in (LLM-D Team, 2025b). An example usage of `inference-perf` to benchmark and optimize can be found in (LLM-D Team, 2025a). It is also used by the Kubernetes community and different organizations for various evaluation and development purposes as seen from the contributors and issue creators in Github (Inference Perf Contributors, 2024a).

AI Usage Disclosure

AI usage follows the Linux Foundation's guidance on AI usage (The Linux Foundation, 2023) where contributors are allowed to use code assist tools. But all of the pull requests are manually reviewed and approved by at least 2 reviewers or maintainers of the project and the contributor is responsible for the code quality and addressing any comments from the reviews. Since there are many contributors in this project, not all specific tools used by the contributors could be called out here.

Acknowledgements

We acknowledge the contributions from the Kubernetes `wg-serving` community and the contributors of the inference-perf project (Inference Perf Contributors, 2024a) and the supported model servers.

References

Grafana Labs. (2021). *k6: Open-source load testing tool*. https://k6.io/

Heyman, J., Byström, C., Hamrén, J., Heyman, H., & Holmberg, L. (2011). *Locust: An open source load testing tool*. https://locust.io/

Hugging Face. (2023a). Text generation inference. In *GitHub repository*. GitHub. https://github.com/huggingface/text-generation-inference

Hugging Face. (2023b). Text generation inference benchmark tool. In *GitHub repository*. GitHub. https://github.com/huggingface/text-generation-inference/tree/main/benchmark

Inference Perf Contributors. (2024a). Inference perf contributors. In *GitHub repository*. GitHub. https://github.com/kubernetes-sigs/inference-perf/graphs/contributors

DRAFT

Inference Perf Contributors. (2024b). Inference perf metrics. In *GitHub repository*. GitHub. https://github.com/kubernetes-sigs/inference-perf/blob/release-v0.3.0/docs/metrics.md

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Rostaing, J. E., Zhang, H., Hendry, G., & Stoica, I. (2023). vLLM: Easy, fast, and cheap LLM serving with PagedAttention. *Proceedings of the 29th ACM Symposium on Operating Systems Principles*, 611–626. https://doi.org/10.1145/3600006.3613165

LLM-D Team. (2025a). Llm-d KV cache aware routing. In *GitHub repository*. GitHub. https://github.com/llm-d/llm-d-kv-cache/blob/main/benchmarking/37-capacity/README.md

LLM-D Team. (2025b). *Llm-d: A kubernetes-native high-performance distributed LLM inference framework*. https://llm-d.ai/

NVIDIA. (2024a). *GenAI-perf*. https://github.com/triton-inference-server/perf_analyzer/tree/main/genai-perf

NVIDIA. (2024b). NVIDIA dynamo. In *GitHub repository*. GitHub. https://github.com/ai-dynamo/dynamo

The Kubernetes Authors. (2024). Kubernetes wg-serving. In *GitHub repository*. GitHub. https://github.com/kubernetes-sigs/wg-serving

The Linux Foundation. (2023). *Generative AI policy*. https://linuxfoundation.org/legal/generative-ai-policy

vLLM Team. (2023). vLLM benchmarks. In *GitHub repository*. GitHub. https://github.com/vllm-project/vllm/tree/main/benchmarks

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., & Sheng, Y. (2024). SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems 37*, 62557–62583. https://doi.org/10.52202/079017-2000

DRAFT