GPU SMITH
COMPUTE ENGINEERING

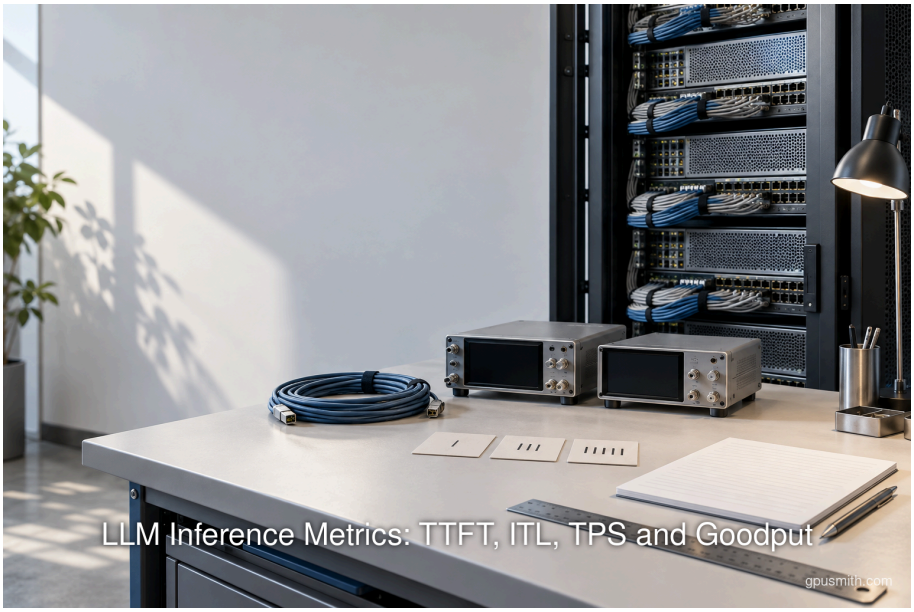
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

LLM Inference Metrics: TTFT, ITL, TPS and Goodput

> Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-09-28 · llm inference metrics · ttft · inter-token latency · tokens per second · goodput · inference benchmarking · latency · throughput · vllm

Original article: https://gpusmith.com/articles/en/llm-inference-metrics-ttft-itl-tps-goodput



In this article

Executive Summary

LLM inference metrics are comparable only when the timing boundary, token count, workload, and aggregation rule match. Time to first token (TTFT) measures the wait for the first streamed token; inter-token latency (ITL) describes spacing during generation; output tokens per second can describe one request or a whole service; and goodput counts requests that finish within specified service objectives. NVIDIA's NIM guide includes queueing, prefill, and network delay in client-observed TTFT (docs.nvidia.com), while Google Cloud also publishes a server-side first-token interval from request receipt to token send (docs.cloud.google.com). Those measures answer different questions. A buyer should reject a side-by-side latency claim until the measurement point is stated.

The largest ambiguity is **tokens per second (TPS)**. The NIM benchmarking guide defines its per-user value as output sequence length divided by full request latency (docs.nvidia.com). NVIDIA AIPerf's `output_token_throughput_per_user` instead excludes TTFT and uses the inverse of ITL (docs.nvidia.com). Artificial Analysis calls output speed the rate after the first received token (artificialanalysis.ai), while its system load test reports aggregate tokens per second across concurrent requests (artificialanalysis.ai). All can be correctly calculated yet yield different numbers for the same run. This report names each numerator and denominator rather than treating a TPS label as a unit.

A usable comparison records input and output token distributions, model and tokenizer, request arrival process, concurrency, streaming mode, batch policy, cache state, generation controls, software versions, topology, network vantage point, and the statistic's time window. These are real experimental variables: vLLM exposes separate request arrival, burstiness, and concurrency controls (docs.vllm.ai), and Artificial Analysis warns that longer prompts can increase TTFT and reduce output speed (artificialanalysis.ai). The two-minute gate below marks a claim **comparable**, **recomputable from raw samples**, or **cannot normalize because a required descriptor is missing**. It never converts a median to a p95 or divides fleet throughput by a guessed user count.

The quantitative evidence illustrates why workload controls matter. MLCommons describes a summarization workload averaging **778 input and 73 output tokens** (mlcommons.org), and its server and interactive scenarios use different TTFT and time-per-output-token limits (mlcommons.org). Those thresholds are benchmark rules, not universal user requirements. Goodput should therefore be reported as compliant completed requests per second under a named objective; AIPerf also tracks the fraction of attempted requests that comply, counting errors in its denominator (docs.nvidia.com). The worked arithmetic here is explicitly hypothetical, not a GPU or provider result. Procurement should request per-request samples, error counts, and the full workload manifest before using any headline benchmark in a purchasing decision.

Introduction and Background

(Hypothetical Example) In a procurement packet, one serving claim might read 120 output tokens per second, another 8,000 output tokens per second, and a third 0.8 seconds TTFT. Those labels alone do not establish which system serves an interactive workload better. The first rate might be a single response after the first token, the second an entire fleet with many users, and the TTFT might end when a server sends a chunk rather than when a client receives a usable token. Google Cloud's service metric is defined from request receipt to first token sent (docs.cloud.google.com); Amazon Bedrock describes its streaming first-token metric from request send to first token received (docs.aws.amazon.com). The endpoints of the clock must travel with the number.

The purpose of a benchmark comparison is to hold the service objective and workload sufficiently constant to interpret a change in the result. MLCommons uses a common load generator to schedule queries, track latency, and compute benchmark metrics (docs.mlcommons.org). Artificial Analysis publishes a separate API performance method that can be cited when interpreting a vendor claim, as a product post by Inception Labs demonstrates (www.inceptionlabs.ai). Neither design is a generic substitute for a buyer's actual request mix. Its value is the explicit method: reviewers can identify what was measured and what must be changed to reproduce the comparison.

For an independent engineering advisor, metric definitions are procurement inputs. GPU Smith's public description says it derives topology, node count, and serving stack from workload models and throughput targets (gpusmith.com). This report applies that workload-first discipline to evidence supplied by vendors or internal teams. It does not rank hardware or prescribe a node count. It asks whether two numbers represent the same user-visible operation, whether a claimed gain survives an unchanged traffic pattern, and which raw fields would permit recalculation.

The working rule is simple: **a faster figure is evidence only for the workload and boundary that produced it**. A benchmark may be valuable even if it cannot be compared with another. Its status should be recorded honestly, then the missing measurements requested. The following dictionary fixes the units before the worksheet evaluates whether any conversion is defensible.

Metric Dictionary and Measurement Boundaries

Let request *i* be submitted at `s_i`, its first nonempty output token arrive at `f_i`, and its final token arrive at `e_i`. Let `O_i` be the number of generated output tokens and `I_i` the number of input tokens under a declared tokenizer. A streaming client's `f_i` is an arrival time; a server timestamp for sending the token is a different event. NVIDIA describes client-side TTFT as including request queueing, prefill, and network delay (docs.nvidia.com). Hugging Face's Text Generation Inference exposes separate request-duration and queue-duration histograms, illustrating why a server-side series need not equal the client's end-to-end observation (huggingface.co).

Table 1 is the metric dictionary. Its formulas are **operational definitions for this worksheet**. A source's similarly named metric should be mapped to the row only after its own formula is checked.

Metric	Unit and worksheet formula	Boundary or necessary qualification
TTFT	seconds per request; `f_i - s_i`	First **nonempty** streamed token received by the client; state whether reasoning or answer token.
ITL	seconds per token interval; `t_(j+1) - t_j`	Individual adjacent-token gaps, summarized over a stated set of intervals. Google Cloud distinguishes these from request-level time per output token. (docs.cloud.google.com)
TPOT	seconds per output interval; `(e_i - f_i)/(O_i - 1)`	Request-level average after the first token; undefined for fewer than two output tokens. (docs.cloud.google.com)
Request latency	seconds per request; `e_i - s_i`	Completion to last token, including the selected clock boundary. (docs.aws.amazon.com)
Full-request output rate	output tokens/second per request; `O_i/(e_i - s_i)`	Includes TTFT and queue delay.
Decode-only output rate	output tokens/second per request; `(O_i - 1)/(e_i - f_i)`	Equals `1/TPOT` when the same output count and timing points are used.
Request throughput	completed requests/second; `N_success / T_run`	Run starts and ends at declared events; report attempts and errors too. (docs.nvidia.com)
Aggregate output throughput	output tokens/second for a declared system; `sum(O_i)/T_run`	Name the denominator: GPU, node, replica, cluster, or fleet. (artificialanalysis.ai)
SLO-qualified goodput	compliant requests/second; `N_good/T_run`	State every service level objective (SLO), token mix, and error rule. (docs.nvidia.com)

The table's rows are not interchangeable. (Hypothetical Example) If `O_i=1`, full-request output rate exists but TPOT does not. If tokens are delivered in bursts, a request-level TPOT can hide a long pause followed by rapid delivery. A client benchmark can store every interval and later calculate both the mean and a high percentile. SGLang's detailed benchmark output can include per-request ITL arrays (github.com). Reporting only the single mean loses that timing pattern.

A second boundary is the **identity of the first token**. For a reasoning model, reviewers should record whether TTFT ends on a reasoning token or the first answer token. A person waiting for the answer may care about the latter. Azure Monitor describes time to first response chunk and maps it to TTFT (learn.microsoft.com). A chunk may not have the same semantics as an answer token. The report should state whether the clock stops at bytes, a nonempty token, a reasoning token, or visible answer text.

First-Token and Decode Metrics

TTFT and the prefill path

TTFT begins before decoding becomes visible. It can include client dispatch, transport, queue time, prompt processing, and the first decode step. Client-observed TTFT is therefore the correct headline for interactive user perception, provided the client location and streaming mode are fixed. A client-observed benchmark should report tester location because its network path contributes to the observed interval (docs.anyscale.com). A server's internal prefill timer can still be useful for diagnosis, but it cannot be silently substituted for client TTFT.

Prompt length and cache reuse change the work before the first token. vLLM documents reuse of previously computed key-value blocks for requests with the same prefix (docs.vllm.ai); Cloudflare says prefix caching is enabled by default for selected Workers AI models (developers.cloudflare.com). A cache-hot benchmark and a cache-cold benchmark represent different workload states. Record prefix overlap, cache enablement, hit rate, and the reset or warmup procedure. If a public result omits these, a shorter TTFT is not necessarily a faster uncached prefill path.

ITL, TPOT, and output speed

ITL is a series of gaps, one between each pair of emitted tokens. TPOT is one request's average gap after the first token. NVIDIA describes the latter as `(end-to-end latency - TTFT)/(output tokens - 1)` (docs.nvidia.com), and Google Cloud writes the same form for TPOT while distinguishing individual ITL (docs.cloud.google.com). The equivalence is valid only for the same timestamps and token count. A p95 of all gaps, an average of each request's TPOT, and a p95 of request-level TPOT are different statistics.

Output speed usually takes a reciprocal, but the numerator is easy to change. A decode-only rate uses $O_i - 1$ token intervals; a full-request rate uses O_i tokens and includes TTFT. The worksheet uses a decode-only rate after first-token receipt. Google's Gemini provisioned-throughput service objective instead specifies an interval from the first returned non-thinking token through the last (cloud.google.com). These are labeled methods, not a universal convention. A claim of “tokens per second” should be read as incomplete until the output-token counting method, interval, and aggregation level are disclosed.

Throughput and Goodput Under Load

Concurrent throughput and the latency tradeoff

Aggregate output throughput rises when more requests share a system, until the workload and scheduling policy impose a limit. A concurrent result can therefore coexist with slower per-request output and a longer p95 wait. vLLM lists continuous batching and chunked prefill among its serving features (docs.vllm.ai); Hugging Face notes that admitting a waiting request for prefill can pause an active decode batch (huggingface.co). The precise tradeoff is an empirical property of a declared runtime configuration, not a fixed conversion factor. Plot request throughput or aggregate output throughput against p50 and p95 TTFT and TPOT as offered load rises.

The **arrival process** matters as much as a concurrency cap. vLLM separately controls arrival rate, burstiness, and maximum concurrency (docs.vllm.ai). Its immediate-send mode places all requests into the run at once (docs.vllm.ai). MLPerf's Server workload models Poisson arrivals (mlcommons.org), whereas its Single Stream rules send the next query after the prior response completes (github.com). The former tests arrival-driven queueing; the latter cannot reproduce the same saturation pattern. A batch size without traffic shape does not characterize a serving benchmark.

Goodput and error handling

Goodput answers a stronger procurement question: how many completed requests per second met every stated limit? A serving research paper defines useful capacity as the maximum request rate within TTFT and TPOT constraints (www.usenix.org). Separately, the fraction of attempts that comply should count errors as noncompliant attempts. Those are different measures: a high compliant fraction at light load can have low request goodput, and high raw throughput at heavy load can violate the latency objective. Research on disaggregated serving likewise defines performance by the maximum rate satisfying both TTFT and TPOT constraints (www.usenix.org).

The SLO must specify which percentile or per-request rule applies, the allowed input/output mix, any error budget, and whether first-token and token-gap thresholds both apply. vLLM's serving benchmark can accept request-level TTFT, TPOT, and end-to-end latency objectives for goodput calculations (docs.vllm.ai). No generic “goodput” number can be carried to a different SLO without request-level measurements.

Workload Descriptor and Two-Minute Comparability Gate

A benchmark claim is a tuple, not a scalar. The **workload descriptor** below should accompany every point on a chart. It is intentionally operational: a reviewer can request these fields without knowing a vendor's internal scheduler. The first six establish whether the same work arrived; the rest establish where time was measured and which run was summarized.

- **Model identity:** model name, exact weights or revision, adapter state, context limit, and tokenizer.
- **Token mix:** input and output token distributions, not only means; include stop reasons and any reasoning-token treatment.
- **Prompt content:** prompt source, prefix overlap, multilingual or structured content, and whether prompts repeat.

- **Generation policy:** maximum output length, sampling parameters, tool calls, speculative decoding, and rejection or retry handling.
- **Arrival pattern:** open-loop offered rate or closed-loop concurrency, burstiness, duration, and ramp schedule.
- **Batch policy:** continuous or static batching, batch token limits, prefill scheduling, and queue limits.
- **Cache state:** prefix cache, key-value cache, warmup, flushing, cache hit rates, and cold-start inclusion.
- **Serving stack:** runtime and version, quantization, kernel or attention path, parallelism, and routing policy.
- **Topology:** GPUs per replica, replicas, nodes, network path, and resource-sharing or tenancy conditions.
- **Clock boundary:** client or server timestamps, client region, streaming mode, first byte or first token, and final-response event.
- **Statistic:** p50, p95, p99, mean, sample count, time window, aggregation scope, and histogram method.
- **Accounting:** attempts, completions, errors, retries, timed-out requests, measured duration, and units.

These fields are not decorative. Artificial Analysis uses a prompt workload of approximately **10,000 input tokens with at least 1,500 answer tokens** (artificialanalysis.ai) and also has a parallel workload of **10 simultaneous prompts** (artificialanalysis.ai). Its API performance results are a **p50 over the preceding 72 hours** (artificialanalysis.ai). A test with short prompts, one request at a time, and a one-hour p95 is asking a different question even if the model name matches.

The **two-minute gate** starts with four checks:

- **Identity check:** same model version, tokenizer, generation settings, and task content, or a stated scientific reason for the difference.
- **Work check:** comparable input/output distributions and cache state, with matched streaming and warmup.
- **Load check:** same arrival process, offered load or concurrency sweep, batch policy, and measurement duration.
- **Clock check:** same client/server vantage point, token event, aggregation level, percentile, and error accounting.

If all four pass, compare the full curves and uncertainty, not one cherry-picked point. If a field differs but raw request records and a common workload can be reconstructed, mark **recomputable** and calculate both results under a shared definition. If a field is absent and cannot be recovered, mark **cannot normalize because ...** followed by the missing field. Anyscale's benchmark guidance likewise calls out concurrency, traffic shape, and client-observed latency together (docs.anyscale.com). The gate is a rejection test for a purchasing claim, not a judgment that an experiment has no value.

Normalization Worksheet and Rejection Test

Table 2 is a claim-completeness scorecard and a worksheet. Enter each source's values in columns A and B. “Required” means that an absent value blocks the named comparison; “raw rescue” means request-level records can sometimes repair a difference. A pass count is only a completeness count, **never a weighted**

performance score.

Field to enter for A and B	Why it changes interpretation	Worksheet disposition if different or absent
Model, weights, tokenizer, decoding settings	Changes tokens and work per token.	Required for direct speed claims; different models are separate evaluations.
Input and output distributions	Prefill, decode span, and output-rate denominator vary.	Match buckets or reweight raw requests; means alone are insufficient.
Arrival rate, burstiness, concurrency	Changes queueing and batching. (docs.vllm.ai)	Compare equal offered-load points or regenerate traffic.
Streaming and first-token event	Non-streaming TTFT may equal completion latency. (github.com)	Required for TTFT; rerun if only an incompatible event exists.
Cache state and warmup	Cache reuse changes prefill work. (docs.vllm.ai)	Separate cold and warm cohorts; do not estimate an unknown hit mix.
Runtime, version, batch policy	Scheduler and admitted work can differ. (docs.vllm.ai)	Disclose and compare as tested; rerun to isolate a single change.
Measurement point and client region	Transport affects client-observed timing. (docs.cloud.google.com)	Align vantage point or obtain both client and server records.
Statistic, window, sample count	A mean, p50, and p95 answer different questions. (sre.google)	Recompute from raw samples or compatible histograms.
Denominator scope	Per-request, per-GPU, and fleet TPS differ.	Report scope; divide only if allocation and time basis are known.
Errors, timeouts, retries, SLO	Raw throughput can include unsuccessful work.	Recompute goodput from attempts and per-request outcomes.

The worksheet makes several legitimate conversions explicit. Seconds and milliseconds can be changed by a unit factor. For each request with at least two output tokens, **decode-only rate = `1/TPOT`** when the same timestamps and tokenizer are used. Aggregate output throughput can be recomputed as `sum(output_tokens)/run_seconds` from a complete trace. A full-request output rate can be computed from `O_i/(e_i-s_i)`; it cannot be inferred from decode-only speed alone because TTFT is missing. The NIM guide uses the full-request rate for “TPS per user,” while AIPerf excludes TTFT in a differently named per-user metric (docs.nvidia.com). That documented discrepancy is enough reason to demand the formula in any vendor chart.

Valid Conversions and Rejection Rules

Other conversions require raw data. A p95 cannot be derived from a mean, a p50, or a handful of plotted percentiles. Prometheus states that precomputed quantiles cannot be aggregated across replicas (prometheus.io); compatible histogram buckets or observations are needed. Histogram resolution also bounds percentile accuracy (prometheus.io). The same rule applies to pooling request-level TPOT values across unequal cohorts: first decide whether the estimand is a percentile of requests, a percentile of individual intervals, or a token-weighted mean.

Several transformations must be rejected rather than approximated:

- **No client/server shortcut:** server prefill or first-token-send timing cannot become client TTFT without network and queue timestamps.
- **No guessed user share:** fleet output TPS divided by concurrency is not a measured per-user rate when requests overlap unevenly.
- **No percentile algebra:** p95 TTFT plus p95 decode time is not p95 end-to-end latency.
- **No cache correction by assumption:** missing prefix overlap or hit rate cannot be normalized using a generic discount.
- **No hidden token conversion:** tokenizer or reasoning-token differences invalidate a simple rate ratio.
- **No SLO transfer:** goodput under one threshold is not goodput under another unless per-request outcomes exist.

A spreadsheet-ready record should have one row per attempt: `request_id`, `arrival_s`, `first_token_s`, `finish_s`, `input_tokens`, `output_tokens`, `status`, `cache_state`, `replica_id`, and `model_revision`. Add `TTFT_s = first_token_s - arrival_s`, `E2E_s = finish_s - arrival_s`, and `TPOT_s = (finish_s - first_token_s)/(output_tokens - 1)` only where the denominator is positive. Then calculate a Boolean `good = status_success AND TTFT_s <= target_ttft AND TPOT_s <= target_tpot`; sum it over measured run time. vLLM can save per-request TTFT and TPOT details (docs.vllm.ai). A telemetry export that lacks arrival times or request IDs may prevent reconstruction even when a dashboard displays attractive summary values. The worksheet should preserve the original claim alongside the normalized result, identify every excluded request, and record who approved the common workload and SLO before a procurement comparison is circulated.

Data Analysis and Evidence

The numbers in this section separate **published benchmark design** from **hypothetical arithmetic**. They are evidence about measurement choices, not cross-vendor performance results. MLCommons' Llama 2 70B workload selected **24,576 samples** (mlcommons.org) and controlled generation length around **294.45 tokens per sample with a 10% tolerance** (mlcommons.org). Its authors chose a token-based throughput measure because queries have different input and output lengths (mlcommons.org). Another MLCommons summarization workload reports average lengths of **778 input and 73 output tokens** (mlcommons.org). Neither profile can be silently substituted for a short chat exchange or a long document workload.

For its Llama 3.1 8B benchmark, MLCommons specifies a Server scenario limit of **TTFT at most 2 seconds and TPOT at most 100 milliseconds** (mlcommons.org). The Interactive scenario uses **0.5 seconds and 30 milliseconds** respectively (mlcommons.org). The same raw completed requests would have different compliance counts under those two rules. A benchmark's service-level objective is therefore part of the result, not a note beneath it. Goodput uses compliant requests per second under a stated objective (www.usenix.org); the separate compliant fraction divides by all attempts.

Table 3 gives synthetic, spreadsheet-ready arithmetic (Hypothetical Example). Each row is an invented request or run, not an observed GPU measurement. The per-request values use the formulas in Table 1; aggregate rows use an invented **60-second** measurement window. The purpose is to show how a headline rate and user

wait can move in opposite directions.

Hypothetical row	Declared inputs	Calculated output	Interpretation
A, interactive request	`O=101`, `TTFT=0.20 s`, `E2E=2.20 s`	`TPOT=(2.20-0.20)/100=0.020 s`; decode rate `50 tok/s`; full-request rate `45.9 tok/s`	One request, short first-token wait.
B, same output length	`O=101`, `TTFT=1.20 s`, `E2E=3.20 s`	`TPOT=0.020 s`; decode rate `50 tok/s`; full-request rate `31.6 tok/s`	Identical decode rate, much slower visible start.
C, aggregate run	`600 successes`, `60 s`, `60,600 output tokens`	`10 requests/s`; `1,010 output tok/s`	Service throughput says nothing about p95 TTFT.
D, same run under SLO	480 of 600 successes meet both limits	`8 good requests/s`; compliant fraction `480/600=80%`	Goodput falls below raw request throughput.

The arithmetic in rows A and B shows a direct denominator trap: both requests decode at **50 tokens per second after the first token**, yet the full-request rate differs because TTFT differs. Rows C and D show why a throughput chart should be paired with a compliance curve. Nothing in the aggregate token total gives the distribution of first-token waits. Google SRE guidance warns that simple averages can hide tail latency (sre.google). Request samples or appropriate histograms are needed to calculate the p95 that an interactive purchaser may specify.

Published methodology also shows why test scheduling matters. MLPerf's Server rules use Poisson arrivals (github.com), while a closed-loop Single Stream sends work after the previous completion (github.com). Artificial Analysis's load test enables streaming (artificialanalysis.ai) and separately reports aggregate output throughput. An analysis that pools results from these designs would combine different queueing experiments. A neutral outside example is Inception Labs' product post, which explicitly cites Artificial Analysis's methodology when describing end-to-end latency (www.inceptionlabs.ai). That citation supports the value of a stated method, not the transfer of one workload's number into another workload.

Implications and Future Directions

Procurement evidence should be a trace, not a screenshot. A report should publish a workload manifest, exact metric formulas, client and server vantage points, per-request samples or mergeable histograms, run duration, errors, and the SLO definition. Anyscale recommends warmup before measurement to avoid effects from initial model loading or cache misses (docs.anyscale.com). SGLang exposes separate warmup and cache-flush controls in its benchmark harness (github.com). These controls should be logged as settings, not inferred from the tool name. Prefix caching, speculative decoding, and streaming can all change which work occurs before and after the first visible token; Hugging Face describes speculative decoding as generating candidates before large-model verification (huggingface.co), and its streaming documentation describes progressive token delivery (huggingface.co).

At the telemetry layer, NVIDIA NIM documents `/v1/metrics` as a pass-through of backend-native Prometheus metrics (docs.nvidia.com). The vLLM reference lists `vllm:time_to_first_token_seconds`, `vllm:inter_token_latency_seconds`, and `vllm:e2e_request_latency_seconds` as histograms, alongside

generation-token and request-success counters (docs.vllm.ai). A benchmark record should retain the histogram buckets, scrape interval, and raw request samples; a dashboard name alone does not identify a client vantage point.

The acceptance packet should therefore include:

- **Scope:** one model revision, tokenizer, task mix, dataset provenance, and privacy-safe prompt examples.
- **Traffic:** offered rate, concurrency sweep, burst distribution, run duration, ramp, warmup, and cooldown.
- **Serving:** runtime version, quantization, batching, cache policy, speculative settings, replica count, and routing.
- **Clocks:** client region, server timestamps, streaming events, and network path.
- **Outcomes:** attempt and success counts, timeouts, retries, TTFT, interval ITL, TPOT, end-to-end latency, and output lengths.
- **Statistics:** p50, p95, p99, sample count, confidence or run-to-run spread, and histogram buckets.
- **Objective:** exact TTFT, TPOT, end-to-end, and error limits used to calculate goodput.
- **Reproduction:** command lines, workload seed, configuration, and raw trace or shareable aggregated observations.

For GPU Smith's stated assessment model, workload characterization includes models, context length, and tokens per day (gpusmith.com). Those are natural inputs to a buyer's test manifest, but they are not substitutes for request distributions or tail latency. The decision is whether the vendor's result matches the buyer's required service envelope. If it does not, the next step is a controlled rerun, not a multiplier applied to a headline TPS figure.

Future benchmark reports may make this easier by publishing both client and server spans, preserving per-token event times, and standardizing an export schema for attempts and outcomes. Google Research describes Inference Perf as designed to work across model servers (research.google), and existing runtime benchmarks already expose raw-request options (docs.vllm.ai) (github.com). The remaining work for evaluators is governance: specify a common workload before results arrive, lock the scoring rule, and preserve enough observations to audit the tail.

Frequently Asked Questions (FAQs)

What is the difference between TTFT and ITL?

TTFT is the wait to the first token; ITL is the spacing between later tokens. The former can reflect queueing, prefill, and transport, while the latter measures the decode stream at the chosen boundary. Google Cloud explicitly separates individual ITL from request-level TPOT (docs.cloud.google.com). Report both p95 TTFT and an interval or request-level TPOT statistic for an interactive service, with the sampling unit named.

Is TPS per user the same as aggregate token throughput?

No. A per-user figure is attached to a request or a user session. Aggregate token throughput sums output across concurrent requests and divides by a shared run duration (artificialanalysis.ai). Even per-user labels differ: a full-request rate includes TTFT, while a decode-only rate excludes it. Ask for numerator, denominator, concurrency, and whether prompt tokens are counted.

Can p95 be estimated from average TPS or p50 latency?

No defensible conversion exists without observations or a sufficiently detailed, compatible histogram. Prometheus cautions that precomputed quantiles cannot be aggregated (prometheus.io). A mean token rate also does not encode how waits are distributed among requests. The worksheet should return **“cannot normalize because raw per-request timing and output lengths are unavailable.”**

How should concurrency and batch size be reported?

Give the request arrival process, offered rate, maximum in-flight requests, batch policy, and tokens admitted per batch. vLLM documents separate arrival, burstiness, and concurrency parameters (docs.vllm.ai). A fixed concurrency test and a Poisson-arrival test are distinct load models, even if their average completed requests per second happens to match.

What is an acceptable goodput claim?

It states **compliant requests per second**, the complete SLO, the measured interval, token-length mix, and how errors and retries were counted. Goodput is an SLO-qualified rate (www.usenix.org); a compliant fraction has a different denominator. Buyers should ask for the goodput versus offered-load curve, together with p95 latency and error rate at each point.

Conclusion

The safest comparison begins by translating every claim into a formula and a workload descriptor. TTFT describes the first visible response at a stated clock boundary. ITL and TPOT describe the decode stream, with different sampling units. Tokens per second may describe a request after first token, a full request, or an entire serving system. Goodput adds an explicit service objective and error treatment. Those distinctions turn a marketing-sized number into testable evidence.

A buyer can compare results directly only when the model, token accounting, prompt and output distributions, traffic, serving configuration, vantage point, statistic, and SLO align. When raw observations exist, a common definition may be recomputed. When they do not, the appropriate worksheet result is **“cannot normalize because ...”**, followed by the missing field. The synthetic calculations show why equal decode rate need not mean equal user experience, and why high aggregate throughput need not imply high SLO-qualified capacity.

A complete benchmark report should include the manifest, per-request outcomes, latency distributions, and enough timing detail to reproduce its formulas. That evidence lets platform engineers and procurement teams choose a serving option for their own workload, while keeping uncertainty visible where a public claim omits the data needed for comparison.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html
2. https://docs.cloud.google.com/monitoring/api/metrics_gcp_a_b
3. https://docs.nvidia.com/aiperf/reference/ai-perf-metrics-reference
4. https://artificialanalysis.ai/methodology/performance-benchmarking
5. https://artificialanalysis.ai/methodology/system-load-test
6. https://docs.vllm.ai/en/stable/benchmarking/cli/
7. https://mlcommons.org/2025/09/small-llm-inference-5-1/
8. https://docs.aws.amazon.com/bedrock/latest/userguide/monitoring-runtime-metrics.html
9. https://docs.mlcommons.org/inference/submission/
10. https://www.inceptionlabs.ai/blog/mercury-refreshed
11. https://gpusmith.com/
12. https://huggingface.co/docs/text-generation-inference/reference/metrics
13. https://docs.cloud.google.com/kubernetes-engine/docs/concepts/machine-learning/inference
14. https://github.com/sgl-project/sglang/blob/main/docs/docs/developer_guide/bench_serving.mdx
15. https://learn.microsoft.com/en-us/azure/foundry/openai/how-to/latency
16. https://docs.anyscale.com/llm/serving/benchmarking/benchmarking-guide
17. https://docs.vllm.ai/en/latest/design/prefix_caching/
18. https://developers.cloudflare.com/workers-ai/features/prompt-caching/
19. https://cloud.google.com/vertex-ai/generative-ai/sla
20. https://docs.vllm.ai/en/stable/
21. https://huggingface.co/docs/text-generation-inference/basic_tutorials/launcher
22. https://mlcommons.org/2024/03/mlperf-llama2-70b/
23. https://github.com/mlcommons/inference_policies/blob/master/inference_rules.adoc
24. https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin
25. https://docs.vllm.ai/en/stable/cli/bench/serve/
26. https://sre.google/sre-book/service-level-objectives/
27. https://prometheus.io/docs/practices/histograms/
28. https://huggingface.co/docs/text-generation-inference/conceptual/speculation
29. https://huggingface.co/docs/text-generation-inference/en/conceptual/streaming
30. https://docs.nvidia.com/nim/large-language-models/latest/reference/logging-and-observability.html
31. https://docs.vllm.ai/en/stable/usage/metrics/
32. https://research.google/pubs/inference-perf-a-benchmarking-tool-for-genai-inference/

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the hardware library, GPU references, networking references and vendor research. Contact GPU Smith with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: https://gpusmith.com