



Energy per Token: Measuring LLM Inference Efficiency

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 · 2026-10-29 · energy per token llm inference · joules per token gpu · mlperf power methodology · dcgm total energy consumption · gpu power measurement wall vs board power · llm inference energy efficiency benchmark

Original article: <https://gpusmith.com/articles/en/energy-per-token-measurement-mlperf-power>



In this article

- 1. Why energy-per-token figures rarely compare
- 2. Defining the metric
- 3. What MLPerf Power measures, and what it does not
- 4. GPU-level counters: NVML and DCGM
- 5. Sampling versus integrating a counter
- 6. Wall power versus GPU power
- 7. Designing the test
- 8. Computing and reporting: a worked example
- 9. Interpreting results
- 10. Pitfalls
- 11. Sources

Why energy-per-token figures rarely compare

A figure such as "0.3 joules per token" means little until the reader knows which watts were counted, over which interval, and which tokens were divided into them. Two operators running the same model on the same accelerator can publish substantially different numbers without either being wrong: one integrated GPU board energy, the other metered AC power at the server inlet; one divided by output tokens, the other by input plus output tokens. Four boundaries are in common use, each legitimate for a specific question and each systematically different for the same workload.

Table 1: Measurement boundaries for inference energy

Boundary	What it includes	Typical instrument	What it omits
GPU board	GPU die, HBM, board-level regulators	NVML / DCGM / nvidia-smi	Host CPUs, DRAM, NICs, switches, fans, PSU losses
GPU module (Grace-based parts)	GPU plus supported NVIDIA CPU and other module components	nvidia-smi module power readings	Rest of the chassis, PSU losses
Full system (AC)	Everything behind the server or system power inlet	External power analyzer, metered PDU, BMC	Cooling plant, facility distribution losses
Facility	System energy scaled by facility overhead	Utility metering, PUE	Nothing, but mixes in site efficiency

Academic studies, such as the "From Words to Watts" benchmark of LLaMA inference energy on V100 and A100 GPUs sharded across up to 32 GPUs (Samsi et al., [arXiv:2310.03003](#)), are useful for trends but sit at whatever boundary their authors instrumented.

Defining the metric

The core definition is simple: **energy per token = energy consumed over a measurement window ÷ tokens produced in that same window**. The difficulty lies in the three choices hidden inside it.

Which tokens. Output tokens are the conventional denominator because they map to user-visible work, but prefill cost scales with input length, so a J/output-token figure from 2,000-token prompts cannot be compared with one from 200-token prompts. The report should state the denominator and publish the input and output sequence length (ISL/OSL) distribution.

Which window. Per-request attribution is hard because batched requests share GPU cycles. A steady-state figure divides the energy of a fixed interval at constant load by all tokens completed in it; it is reproducible and is what this guide recommends.

Which tokens count. Tokens from requests that missed the latency objective or failed consumed energy without delivering useful work, so reporting J per SLO-qualified token alongside the gross figure is worthwhile; the companion guide on [LLM inference metrics such as TTFT, ITL and goodput](#) covers how to define the qualified set.

Tokens per joule is the reciprocal of J/token, and kWh per million tokens is $J/\text{token} \times 1,000,000 \div 3,600,000$, so 1 J/token equals about **0.278 kWh per million tokens**.

What MLPerf Power measures, and what it does not

MLPerf Power is the most rigorous public methodology for inference energy, and its scope is narrow by design. MLCommons states that the power metrics are computed from "the measured average AC power (energy) consumed by the entire system", that the AC power "is measured at the wall", and that "MLPerf Power is only capable of measuring and validating the full system power" ([MLCommons: Inference Datacenter](#)). The page lists System Power as the metric for the Server and Offline scenarios; TDP settings or power supply ratings quoted in a submission are not measured or validated.

The published power rules require all submitters to use the MLPerf Power working group's software, which relies on SPEC PTDaemon (the code lives in the [MLCommons power-dev repository](#)), and evaluate power only during the performance phase using LoadGen's start and end timestamps. The system under test includes components "sensitized by LoadGen", such as the host processor, accelerators, memory and fans ([MLCommons: Power Measurement Rules](#)). That file is still the canonical power rules document in the repository (last edited 2023-06-26, checked 2026-10-28), but it is headed "MLPerf Inference v2.0" and dated 5 January 2022; its statements that only Yokogawa analyzers and one meter per SUT were supported, and that disaggregated systems were out of scope, are written for the v1.0 to v2.0 rounds, so confirm them with MLCommons before planning a submission.

The general inference rules require all necessary networking to be populated when power is measured, count switches and load balancers outside the LoadGen node as part of the SUT "for counting power and latency" in network submissions, and set a 600-second duration for Server and Offline ([MLCommons: Inference Rules](#)). The LLM latency constraints live in those rules, not on the power page: for example, Llama2-70B is listed with 2,000 ms / 200 ms for the conversational category and 450 ms / 40 ms for the interactive category, and

DeepSeek-R1 with 2,000 ms / 80 ms (Server) and 1,500 ms / 15 ms (Interactive) ([MLCommons: Inference Rules](#)). A power result is therefore energy at a stated latency envelope, the property private measurements usually lack.

The methodology paper reports 1,841 reproducible measurements from 60 systems, developed by a consortium of more than 20 organizations, covering power levels from microwatts to megawatts ([Tschand et al., MLPerf Power, arXiv:2410.12032](#)). For larger inference systems a separate Director host synchronizes time with the SUT over NTP and logs a SPEC-approved analyzer through PTDaemon API calls; throughput benchmarks are scored as (samples/s)/W, which yields samples per joule; and the paper rejects PUE as an ML-system metric because "PUE reflects datacenter efficiency, not ML system efficiency" ([arXiv:2410.12032v2, HTML](#)). It also rejects TDP and PSU ratings as proxies because they "grossly overestimate actual power consumption".

The paper gives one LLM data point that can be turned into a per-token figure with care: Llama2-70B at 111.4 J/sample, converted from tokens using an estimated median of 292 tokens per sample "with an error margin" ([arXiv:2410.12032v2, HTML](#)). Dividing back gives about 0.38 J/token at the full-system AC boundary. That number is **derived here, not published**, inherits the token-length estimate and predates current hardware; it is only an order-of-magnitude anchor. For using current-round results in purchasing decisions, see the [MLPerf Inference 6.1 procurement guide](#).

GPU-level counters: NVML and DCGM

Most operators cannot put a calibrated analyzer on every server, so GPU telemetry is the practical starting point.

Table 2: NVIDIA power and energy telemetry (docs fetched 2026-10-28)

Interface	Name	Unit	Documented semantics
NVML	<code>nvmlDeviceGetTotalEnergyConsumption</code>	mJ	Total energy since the driver was last reloaded; Volta or newer
NVML	<code>nvmlDeviceGetPowerUsage</code>	mW	GPU and associated circuitry; 1-second average on Ampere (except GA100) or newer, instantaneous on GA100 and older
DCGM	<code>DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION (156)</code>	mJ	Total GPU energy since the driver was last reloaded
DCGM	<code>DCGM_FI_DEV_BOARD_POWER_WATTS (155)</code>	W	Power usage for the device
DCGM	<code>DCGM_FI_DEV_BOARD_POWER_RAW_WATTS (157)</code>	W	Current instantaneous power usage
DCGM	<code>DCGM_FI_DEV_BOARD_POWER_LIMIT_ENFORCED_WATTS (164)</code>	W	Effective limit after all limiters
nvidia-smi	Average / Instantaneous Power Draw	W	Board average over the last second / last measured value

The NVML entries come from the [NVML API Reference: Device Queries](#); the DCGM field definitions from the [DCGM Field Identifiers](#) page; and the nvidia-smi definitions from the [nvidia-smi documentation](#), which also documents Module Power Readings on Hopper and newer datacenter products, where a module is "GPU + supported NVIDIA CPU + other components which consume power".

Three version-sensitive details matter. First, dcgm-exporter's default counter file exports the legacy alias DCGM_FI_DEV_POWER_USAGE (a gauge in W) and DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION (a counter in mJ) ([NVIDIA dcgm-exporter default-counters.csv](#)). Second, the DCGM 4.7.0 source header maps that alias to field 155 and marks field 156 as deprecated in favour of DCGM_FI_DEV_GPU_ENERGY_JOULES_TOTAL (field 1611, joules since driver reload) ([NVIDIA DCGM v4.7.0 dcgm_fields.h](#)); on 2026-10-28 the field-ID page (last updated 2026-06-17) did not list field 1611, and dcgm-exporter 4.8.4 (2026-09-18) still exports field 156 by default, so dashboards need checking at upgrade time. Third, the only accuracy statement in the NVML power documentation is "+/- 5%" for Fermi and Kepler GPUs; no accuracy or update-rate figure is published for current GPUs or for the energy counter, so an operator who needs one must establish it against an external meter.

DCGM job statistics wrap the energy counter in a job abstraction: `dcgmi stats -g <group> --enable`, then `-s <jobid>` to start and `-x <jobid>` to stop, with `dcgmi stats -j <jobid> -v` reporting "Energy Consumed (Joules)" alongside average, maximum and minimum power. The documentation notes that stats recording must be enabled before the workload starts ([DCGM Feature Overview](#)). Collection practice is covered in the [DCGM exporter metrics and alerting guide](#).

Sampling versus integrating a counter

Polling a power gauge and multiplying by elapsed time is the most common error in home-grown measurement. A reading is at best a one-second average, a 15-second scrape sees one second in fifteen, and LLM serving is bursty. An illustrative case: assume a GPU repeats a 15-second cycle of 4 seconds at 700 W and 11 seconds at 300 W. True energy per cycle is $4 \times 700 + 11 \times 300 = 2,800 + 3,300 = 6,100$ J, an average of $6,100 \div 15 \approx 406.7$ W. A scraper whose timing happens to land in the burst each cycle reports 700 W, overstating energy by about 72%; one that lands in the lull reports 300 W, understating it by about 26%. Real traces are less regular, so the average error is smaller but any single run can be badly off.

A cumulative energy counter avoids the problem: read it at the start and end of the window and subtract, so the scrape interval affects only where the edges fall. In Prometheus this means `increase()` on the energy counter, converted from mJ to J, rather than averaging the power gauge. The counter restarts when the driver reloads, so a window spanning a reload must be rejected, and the edges must align with the load generator's timestamps, which is why MLPerf uses NTP synchronization and LoadGen timestamps.

Wall power versus GPU power

In an eight-GPU server, board readings exclude host CPUs and DRAM, NICs and DPUs, NVSwitch silicon (which DCGM reports separately through fields such as DCGM_FI_DEV_NVSITCH_POWER_VDD_WATTS), storage, fans, and the conversion losses of the power supplies themselves. On Grace-based modules the nvidia-smi module reading includes the CPU, which changes the boundary again.

Three instruments bridge to the wall: a calibrated external analyzer (the reference, and what MLPerf uses), a metered PDU outlet, or the BMC, which can expose node power and energy through Redfish; the DMTF EnvironmentMetrics schema (release 2026.2) defines PowerWatts, EnergykWh and EnergyJoules properties ([DMTF Redfish EnvironmentMetrics v1.7.0](#)). Implementations vary in which properties they populate and how often they update, and a kWh-resolution counter is too coarse for a 10-minute single-server run. The MLPerf Power paper accepts IPMI or Redfish telemetry for datacenter-scale training and recommends out-of-band collection ([arXiv:2410.12032v2](#), [HTML](#)).

A practical method is to measure the wall-to-GPU energy ratio on one representative server at several load levels and apply that curve, not a single constant, to fleet telemetry. The ratio typically falls as load rises because part of host and fan power is fixed.

Facility energy is the last step, applied once. PUE, standardized in [ISO/IEC 30134-2:2026](#), relates total facility energy to IT equipment energy; multiplying wall-boundary J/token by the site's PUE gives a facility figure. That product belongs in sustainability reporting, labelled with the PUE value and period, never in a hardware comparison.

Designing the test

Table 3: Test design checklist

Item	Recommendation	Reason
Workload	Fixed ISL/OSL distribution or a recorded production trace, published with the result	Prefill share changes J/token
Warm-up	Discard an initial period until power and throughput stabilize	Clocks, caches and temperatures drift early
Window	At least 600 s of steady load	Matches MLPerf Server/Offline duration; dilutes edge effects
Load	Concurrency or request-rate sweep, not one point	Efficiency varies strongly with batch size
Latency	Record TTFT and TPOT percentiles in the same window	Energy without latency is not comparable
Configuration	Driver, CUDA, engine version, precision, power limit, clocks	Each changes the result
Idle policy	Report gross energy; report idle-subtracted energy only as a second, labelled figure	Idle subtraction is a policy choice
Repeats	At least three independent runs	Needed for an interval estimate
Time base	NTP-synchronized hosts; window edges from the load generator's log	Avoids misaligned counters

The enforced power limit should be read during the run (DCGM field 164) rather than assumed, because a tighter platform policy can override the requested cap. The tradeoffs of deliberately lowering it are covered in [GPU power cap economics for inference SLOs](#).

Computing and reporting: a worked example

The following figures are **illustrative**, not measurements of any product. Assumptions: one eight-GPU server, a 600-second steady-state window, 12,000,000 output tokens completed (20,000 tokens/s), 30,000,000 input tokens processed.

1. **GPU boundary.** The difference in each GPU's energy counter averages 390,000,000 mJ = 390,000 J (equivalent to $650\text{ W} \times 600\text{ s}$). Summed across eight GPUs: $8 \times 390,000 = 3,120,000\text{ J}$. Energy per output token = $3,120,000 \div 12,000,000 = \mathbf{0.26\text{ J/token}}$.
2. **Wall boundary.** A PDU meter on the same server records 4,680,000 J (an average of 7,800 W). Energy per output token = $4,680,000 \div 12,000,000 = \mathbf{0.39\text{ J/token}}$. The wall-to-GPU ratio at this load is $4,680,000 \div 3,120,000 = 1.5$.
3. **Including input tokens.** $4,680,000 \div (12,000,000 + 30,000,000) \approx 0.111\text{ J}$ per total token, which looks 3.5 times better while describing the same run.
4. **Idle-subtracted (second figure only).** The server idles at 2,400 W, or 1,440,000 J over 600 s. Net energy = $4,680,000 - 1,440,000 = 3,240,000\text{ J}$, giving 0.27 J/token . Idle power is still paid for in production, so this must not replace the gross figure.
5. **SLO-qualified tokens.** If requests violating the TTFT or TPOT objective account for 6% of output tokens, qualified tokens = 11,280,000, and $4,680,000 \div 11,280,000 \approx 0.415\text{ J}$ per qualified token.
6. **Facility.** With an assumed PUE of 1.3, $0.39 \times 1.3 \approx 0.507\text{ J/token}$ at the facility boundary.
7. **Energy per million tokens.** $0.39\text{ J/token} \times 1,000,000 = 390,000\text{ J} = 390,000 \div 3,600,000 \approx 0.108\text{ kWh}$ per million output tokens at the wall.

For repeat runs, report a mean and an interval rather than the best run. With three illustrative wall results of 0.385, 0.392 and 0.396 J/token, the mean is 0.391 and the sample standard deviation is about 0.0056. A 95% t-interval with two degrees of freedom uses $t = 4.303$, so the half-width is $4.303 \times 0.0056 \div \sqrt{3} \approx 0.014$, giving $0.391 \pm 0.014\text{ J/token}$. Additional runs shrink both the t multiplier and the standard error.

A reproducible disclosure states the boundary and instrument, token denominator, ISL/OSL distribution, model, precision, engine, driver and DCGM/NVML versions, power limit and clocks, load level, TTFT/TPOT percentiles, window length, run count, idle policy and, for facility figures, the PUE value and period.

Interpreting results

Energy per token usually falls as concurrency rises, because fixed power is spread across more tokens, while TTFT and TPOT rise. The minimum-J/token point of a sweep is therefore often outside the latency objective; the relevant number is the lowest J/token among configurations that meet the SLO. Plotting J/token against p99 TPOT exposes vendor claims quoted at an unconstrained batch size.

Power caps shift the same frontier. Lower limits often reduce J/token at moderate load while cutting peak throughput, so a capped configuration can win on energy and lose on the capacity needed to hold the SLO at peak traffic. Speculative decoding, quantization and prefill/decode disaggregation change both axes at once and each needs its own sweep. Hardware comparisons need the same boundary on both sides; a board figure against a wall figure differs by the full host overhead, a factor of 1.5 in the illustrative example.

Pitfalls

- **Counter resets.** NVML and DCGM energy counters restart from zero when the driver reloads; a negative or implausibly small delta signals a reset, and the run should be repeated.
- **Unit changes.** Field 156 reports mJ; the newer field 1611 reports J. Mixing them produces errors of a factor of 1,000.
- **MIG.** Power is a whole-device quantity. DCGM 2.3.2 release notes describe a fix where "whole device metrics such as temperature, power etc." were reported as 0 for GPU instances ([DCGM 2.3 Release Notes](#)), which reflects that power belongs to the physical GPU; the current MIG user guide (updated 2026-10-13) documents no per-instance power reading. Per-instance energy is an allocation whose rule must be disclosed.
- **Double counting.** Summing GPU counters and then adding a wall reading counts GPU energy twice. On Grace-based modules, adding separately measured CPU power to module power does the same.
- **Tensor and pipeline parallelism.** Tokens are counted once per model replica, while energy is summed over every GPU in that replica.
- **Thermal drift.** Power at a fixed clock can rise as the system warms, so short runs on a cold system may flatter results.
- **Mixed windows.** Energy from a dashboard interval divided by tokens from a benchmark summary produces arbitrary results.

GPU Smith can review an energy-per-token measurement plan or a vendor efficiency claim against the boundary and disclosure points above.

Sources

1. [MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Systems from Microwatts to Megawatts for Sustainable AI](#) - Tschand et al., arXiv; scale (1,841 measurements, 60 systems, more than 20 organizations), microwatt-to-megawatt scope (accessed 2026-10-28)
2. [MLPerf Power \(arXiv HTML, v2\)](#) - Tschand et al., arXiv; Director/PTDaemon flow, samples per joule, PUE and TDP critique, Llama2-70B 111.4 J/sample and 292 tokens/sample, IPMI/Redfish telemetry (accessed 2026-10-28)
3. [Inference: Datacenter](#) - MLCommons; full-system AC power at the wall, System Power metric for Server and Offline, TDP not validated (accessed 2026-10-28)
4. [MLPerf Inference v2.0 Power Measurement Rules](#) - MLCommons, GitHub; SPEC PTDaemon, performance-phase measurement, SUT components, historical analyzer and disaggregation constraints (accessed 2026-10-28)
5. [MLPerf Inference Rules](#) - MLCommons, GitHub; 600 s scenario duration, networking and switches in the SUT for power, LLM TTFT/TPOT constraints (accessed 2026-10-28)
6. [MLCommons power-dev repository](#) - MLCommons, GitHub; MLPerf power measurement code uses SPEC PTDaemon (accessed 2026-10-28)
7. [NVML API Reference: Device Queries](#) - NVIDIA; total energy in mJ since driver reload, power usage averaging behaviour, Fermi/Kepler accuracy statement (accessed 2026-10-28)

8. [DCGM Field Identifiers](#) - NVIDIA; field IDs 155, 156, 157, 164 and NVSwitch power fields (accessed 2026-10-28)
9. [DCGM v4.7.0 dcmg_fields.h](#) - NVIDIA, GitHub; deprecation of field 156 in favour of field 1611 in joules, legacy POWER_USAGE alias (accessed 2026-10-28)
10. [dcmg-exporter default-counters.csv](#) - NVIDIA, GitHub; default exported power gauge and energy counter (accessed 2026-10-28)
11. [DCGM Feature Overview](#) - NVIDIA; job statistics commands and Energy Consumed (Joules) output (accessed 2026-10-28)
12. [nvidia-smi documentation](#) - NVIDIA; average and instantaneous board power, module power definition (accessed 2026-10-28)
13. [DCGM 2.3 Release Notes](#) - NVIDIA; whole-device power metrics on MIG GPU instances (accessed 2026-10-28)
14. [Redfish EnvironmentMetrics v1.7.0 schema](#) - DMTF; PowerWatts, EnergykWh and EnergyJoules properties (accessed 2026-10-28)
15. [ISO/IEC 30134-2:2026 Power usage effectiveness \(PUE\)](#) - ISO; standard defining data centre PUE, replacing the 2016 edition (accessed 2026-10-28)
16. [From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference](#) - Samsi et al., arXiv; LLaMA inference energy on V100/A100 up to 32 GPUs (accessed 2026-10-28)

Source links

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

1. <https://arxiv.org/abs/2310.03003>
2. <https://mlcommons.org/benchmarks/inference-datacenter>
3. <https://github.com/mlcommons/power-dev>
4. https://github.com/mlcommons/inference_policies/blob/master/power_measurement.adoc
5. https://github.com/mlcommons/inference_policies/blob/master/inference_rules.adoc
6. <https://arxiv.org/abs/2410.12032>
7. <https://arxiv.org/html/2410.12032v2>
8. https://docs.nvidia.com/deploy/nvml-api/api/group__nvmlDeviceQueries.html
9. <https://docs.nvidia.com/datacenter/dcmg/latest/dcmg-api/dcmg-api-field-ids.html>
10. <https://docs.nvidia.com/deploy/nvidia-smi/index.html>
11. <https://github.com/NVIDIA/dcmg-exporter/blob/main/etc/default-counters.csv>
12. https://github.com/NVIDIA/DCGM/blob/v4.7.0/dcmglib/dcmg_fields.h
13. <https://docs.nvidia.com/datacenter/dcmg/latest/user-guide/feature-overview.html>
14. https://redfish.dmtf.org/schemas/v1/EnvironmentMetrics.v1_7_0.json
15. <https://www.iso.org/standard/30134-2>
16. <https://archive.docs.nvidia.com/datacenter/dcmg/2.3/dcmg-release-notes/index.html>

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>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.