TokenPowerBench: Benchmarking the Power Consumption of LLM Inference

Chenxu Niu[1], Wei Zhang[2], Jie Li[1], Yongjian Zhao[1], Tongyang Wang[1], Xi Wang*[3], Yong Chen[1]

[1]Texas Tech University
[2]Texas Advanced Computing Center
[3]Southeast University
chenxu.niu@ttu.edu, wei.zhang@tacc.utexas.edu, {jie.li, yongjian.zhao, tongyang.wang}@ttu.edu,
xi.wang@seu.edu.cn, yong.chen@ttu.edu

Abstract

Large language model (LLM) services now answer billions of queries per day, and industry reports show that inference, not training, accounts for more than 90% of total power consumption. However, existing benchmarks focus on either training/fine-tuning or performance of inference and provide little support for power consumption measurement and analysis of inference. We introduce TokenPowerBench, the first lightweight and extensible benchmark designed for LLM-inference power consumption studies. The benchmark combines (i) a declarative configuration interface covering model choice, prompt set, and inference engine, (ii) a measurement layer that captures GPU-, node-, and system-level power without specialized power meters, and (iii) a phase-aligned metrics pipeline that attributes energy to the prefill and decode stages of every request. These elements make it straightforward to explore the power consumed by an LLM inference run; furthermore, by varying batch size, context length, parallelism strategy and quantization, users can quickly assess how each setting affects joules per token and other energy-efficiency metrics. We evaluate TokenPowerBench on four of the most widely used model series (Llama, Falcon, Qwen, and Mistral). Our experiments cover from 1 billion parameters up to the frontier-scale Llama3-405B model. Furthermore, we release TokenPowerBench as open source to help users to measure power consumption, forecast operating expenses, and meet sustainability targets when deploying LLM services.

Introduction

Large Language Models (LLMs) such as LLaMA-series (Touvron et al. 2023), Mixtral (Jiang et al. 2024), Falcon (Almazrouei et al. 2023), Qwen (Bai et al. 2023a), GPT-series (Radford et al. 2018, 2019; Brown et al. 2020), OPT (Zhang et al. 2022), and BERT (Devlin et al. 2019) have rapidly become foundational infrastructure for modern AI applications. With remarkable capabilities in reasoning, summarization, and text generation (Zhang et al. 2019; Niu et al. 2022; Xu et al. 2025; Niu et al. 2023; Wang et al. 2024; Niu et al. 2025b; Wan et al. 2024; Niu et al. 2025a; Liu et al. 2024; Niu et al. 2025c; Li et al. 2025; Ye et al. 2025), LLMs are now central to large-scale production systems that translate, recommend, and converse across billions of user interactions. This rapid adoption has led universities, national laboratories, and cloud providers to set up dedicated GPU-backed inference services and AI testbeds. For instance, Indiana University (Jetstream2 2025) now dedicate on-premise GPU clusters exclusively to LLM inference, while many national laboratories (Argonne Leadership Computing Facility 2025; Li et al. 2023) operate AI "testbeds" that expose thousands of accelerators to external users. As these services scale, a fundamental systems question remains largely underexplored: "**What is the power consumption of serving an LLM prompt?**"

While prior work (Mattson et al. 2020; Banbury et al. 2021; Reddi et al. 2020) primarily focused on the energy demands of training LLMs, recent evidence shows that the inference process now dominates the energy footprint in large-scale deployments. As LLMs are integrated into countless applications serving billions of users, the cumulative cost of inference is expected to significantly exceed the cost of training. According to a report on the operational lifecycle of LLMs from Amazon Web Services (AWS) (Hutt, Viswanathan, and Nadolski 2019), inference consumes more than 90% of the energy consumption. While training a frontier model represents a significant one-time capital expense, inference is a continuous operational expenditure that occurs with every use of the model. This constant demand makes inference the primary driver of computational expense, latency, and energy use. The global LLM market, valued at approximately $5.6 billion in 2024, is projected to exceed $35 billion by 2030, with a compound annual growth rate (CAGR) of 36.9% (MarketsandMarkets 2024). Meanwhile, the AI inference market is forecast to grow from $106 billion in 2025 to over $250 billion by 2030, with a compound annual growth rate of 19.2% (MarketsandMarkets 2025). Furthermore, Gartner predicts that by 2028, over 80% of data center workload accelerators will be dedicated to inference, making a significant shift from the historically training-centric deployments (Gartner, Inc. 2025). As shown in the sustainability report of Microsoft (Microsoft Corporation 2025), for a LLM service, electricity expenditure is the single largest component of ongoing operating cost. In other words, **Every Watt Counts as Cost**! Therefore, measuring and optimizing inference power consumption have become

*Corresponding author

essential for researchers and companies to reduce both costs and the energy footprint of AI.

Despite its importance, the community still lacks a comprehensive, reproducible, and scalable benchmarking methodology for measuring and analyzing the power and energy cost of LLM inference across model scales, hardware generations, software stacks, and deployment modes (from single-node to multi-node distributed inference). Existing efforts only partially address this need. For instance, MLPerf Inference benchmark (Reddi et al. 2020) standardizes performance benchmarking across a range of machine learning (ML) tasks but does not systematically capture end-to-end power or normalize energy to LLM-specific service units (e.g., Joules per generated token, per prompt). MLPerf Power benchmark (Tschand et al. 2025) extends inscope measurements, but its current workflows typically (i) emphasize single-node setups, (ii) focus on modest model sizes relative to frontier state-of-the-art (SOTA) LLMs (e.g., Llama 3 405B-class deployments remain largely uncharacterized), and (iii) often depend on external, high-precision metering equipment that is costly and difficult to replicate across institutional testbeds. Moreover, existing benchmark suites rarely explore the configuration space that practitioners routinely tune in production, such as batch sizing strategies, tensor/pipeline parallelism, context length and quantization levels. All of these configuration settings significantly impact both instantaneous power draw and energy per token in average.

To address this gap, we present **TokenPowerBench**, the first lightweight and extensible benchmark specifically designed to quantify the power consumption and energy cost of LLM inference. TokenPowerBench features a modular instrumentation layer that integrates vendor telemetry APIs (e.g., GPU/CPU/Memory power sensors), node-level energy sampling, and optional rack- or facility-level measurements when available. A single metrics pipeline lines up every power sample with the two main inference phases (prefill and decode) to allow us to pinpoint exactly where energy is spent. We conduct extensive experiments to systematically analyze the relationship between energy cost and inference-relevant parameters, including batch size, maximum context length, parallelization strategy, and quantization. Our major contributions of this work are:

- **First comprehensive benchmark**: TokenPowerBench is the **first** open-source framework that couples phase-aware power telemetry with token-level normalization, filling a critical gap left by MLPerf and prior profiling tools.
- **Broad model coverage**: Our initial release profiles **15+ popular open-source LLMs** (1B405B parameters), including LLaMA series, Mixtral series, Falcon series, and Qwen series, providing the community with the most comprehensive energy dataset to date.
- **First Parameter-sensitivity Analysis**: Provides fine-grained comparisons of how inference parameters (batch size, context length and quantization) impact energy consumption across decode phases.
- **Ready for frontier SOTA models**: Features a detailed analysis based on a case study of Llama 3.1 405B, a representative example of SOTA LLMs.

TokenPowerBench enables systematic comparisons across models, deployment scales, and optimization strategies. It offers actionable insights for researchers, developers, and data center operators seeking to reduce the carbon and operational cost of intelligent services.

Related Work

Machine Learning and System-Level Power Benchmarks

MLPerf Power (Tschand et al. 2025) is one of the most comprehensive benchmarks for measuring energy efficiency of machine learning systems. It integrates performance and power measurements using high-precision equipment across edge, datacenter, and cloud systems. However, it treats LLM inference as a generic machine learning inference task and does not account for LLM-specific characteristics such as parallelism techniques and memory optimization. Furthermore, its workflows often assume a static model and dataset configuration, and require expensive power instrumentation setups, limiting accessibility and extensibility for broader LLM deployments.

Green500 (Feng and Cameron 2007) evaluates the energy efficiency of supercomputers using the HPL benchmark. Although it ranks large-scale systems by FLOPS/Watt, it is not designed for ML or LLM-specific workloads. Other efforts propose power-aware profiling tools for general AI workloads, but they do not capture end-to-end inference flows in token-based generation models.

In contrast, TokenPowerBench focuses specifically on LLM inference and provides phase-aware, token-level measurements aligned with model execution patterns. Our framework enables energy-normalized benchmarking (e.g., Joules per token) and supports reproducible multi-node configurations without requiring external metering hardware.

LLM Inference Profiling and Power Analysis

Recent studies begin to highlight the performance and energy cost of LLM inference. For example, LLM-Inference-bench (Chitty-Venkata et al. 2024) is a inference benchmark across diverse hardware but limits its power analysis to the accelerators themselves. Poddar et al. (Poddar et al. 2025) analyze the energy consumption of large transformer models during inference, often using cloud GPUs (e.g., A100, H100) and focusing on distillation or quantization strategies. However, these studies are typically limited to fixed setups, lack systematic benchmarking frameworks, and do not explore parameter space (e.g., batch size, context length) or cluster-wide energy variation. Samsi (Samsi et al. 2023) propose a more focused academic approach using direct hardware telemetry (nvidia-smi/DCGM (Corporation 2023)) to measure only the accelerator's power consumption. Some works (Jegham et al. 2025) investigate the carbon footprint of foundation models, yet mostly focus on training-phase emissions rather than inference.

Category	Method	Node-Level Power	System-Level Power	LLM-Specific	Hardware Flexibility	Cost Analysis	Real-World Scenarios	Standardized	Open Source
Benchmark	MLPerf Power@HPCA '25	✓	✓	✗	✗	✗	✓	✓	✗
Benchmark	LLM-Inference-Bench@SC '24	✓	✗	✓	✓	✗	✓	✗	✓
Measurement	Sustainable NLP@arxiv '25	✓	✗	✓	✗	✗	✓	✗	✓
Measurement	Samsi et al.@HPEC '23	✓	✗	✓	✓	✗	✓	✗	✓
Estimation	Jegham et al.@arxiv '25	✗	✓	✓	✗	✓	✗	✗	✓
Benchmark	**TokenPowerBench**	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of LLM Power Benchmarking Approaches

TokenPowerBench addresses this gap by offering a lightweight, extensible framework to benchmark and analyze energy efficiency across LLM sizes, hardware generations, and deployment modes. It introduces a normalized metric suite (Joules/token, power imbalance, energy-delay product) and automates parameter sweeps over key inference configuration dimensions.

System-Level Power Instrumentation

A number of tools and frameworks exist for system-level power monitoring (Li et al. 2020; Stefanov et al. 2021). NVIDIAs NVIDIA Management Library (NVML), Data Center GPU Manager (DCGM) and "nvidia-smi" provide GPU-level power telemetry, while Intels Running Average Power Limit (RAPL) (Intel Corporation 2023) interface supports package-level CPU/DRAM energy readings. Cluster-wide monitoring systems like Intelligent Platform Management Interface (IPMI) (Corporation 2006), Redfish (DMTF 2023), and rack-level PDUs offer coarse-grained wall power data.

TokenPowerBench integrates vendor-native telemetry where available and aligns measurements with LLM inference phases. Our framework supports real-time sampling at multiple granularity levels (GPU, node, rack) and performs time-correlated breakdown across prefill, decode, and idle phases to enable fine-grained attribution of power consumption.

Positioning Against Standardized Benchmarks

Table **??** contrasts TokenPowerBench with the main benchmark and measurement efforts proposed to date for LLM power analysis. TokenPowerBench is the first framework that combines LLM-specific coverage (dense and MoE models up to LLaMA-3-405B), component-level and power measurement without specialized meters, and a fully open-source implementationthereby filling every gap highlighted in the comparison.

TokenPowerBenchis not a replacement for MLPerf Power or any other benchmark but rather a complementary, agile tool. While MLPerf answers the question "Which hardware is more efficient on a standard task?", TokenPowerBench answers the question "**What is the real-world energy cost of running my massive, distributed model with my specific configuration on my available hardware?**". Its lightweight nature makes it the ideal solution for this practical, operator-centric measurement problem.

TokenPowerBench Architecture

Benchmarking LLM inference power consumption is fundamentally different from benchmarking throughput or latency. Unlike traditional performance metrics, energy consumption is shaped by hardware heterogeneity, software stack complexity, and workload dynamics. All of the above settings vary significantly across deployments. Existing LLM benchmarks offer limited support for this variability, often assuming fixed configurations, static model sizes, and expensive instrumentation.

As shown in the Figure 1, **TokenPowerBench** overcomes these limitations through a three-layer architecture: Configuration, Execution & Measurement, and Report Generation. It offers a modular, configurable, fully reproducible, and scalable benchmarking methodology tailored to modern LLM inference.

Configuration

The first task of TokenPowerBench is to help users set up the test environment by selecting the model to run, the inference engine to use, and prompts to feed into it. To keep this step lightweight, TokenPowerBench exposes three plug-and-play modules: a **model pool** (choose any supported LLM), a **prompt-dataset menu** (select from Alpaca, LongBench, or custom prompts), and an **inference-engine selector** (vLLM, TensorRT-LLM, Transformers, or DeepSpeed).

- **Model Pool**: We include models with different underlying architectures to capture how design choices impact energy profiles. This includes standard decoder-only transformers (e.g., the LLaMA series) and Mixture-of-Experts (MoE) models (e.g., Mixtral), which exhibit distinct computational patterns, particularly in parameter activation and memory access. The benchmark covers a broad spectrum of model sizes, from smaller models with fewer than 1 billion parameters–suitable for a single consumer-grade GPU–to frontier-scale models like Llama 3-405B, which require multi-GPU and/or multi-node distributed inference. This allows us to study how energy consumption sacles with increasing model size and hardware complexity.
- **Prompt-dataset memu**: Prompt length and style can significantly affect energy consumption during inference. To capture this variability, TokenPowerBench provides built-in support for two representative datasets: Alpaca, featuring short, chat-style prompts, and LongBench, which includes extended contexts of up to 10k tokens. Users can also supply custom data in CSV or

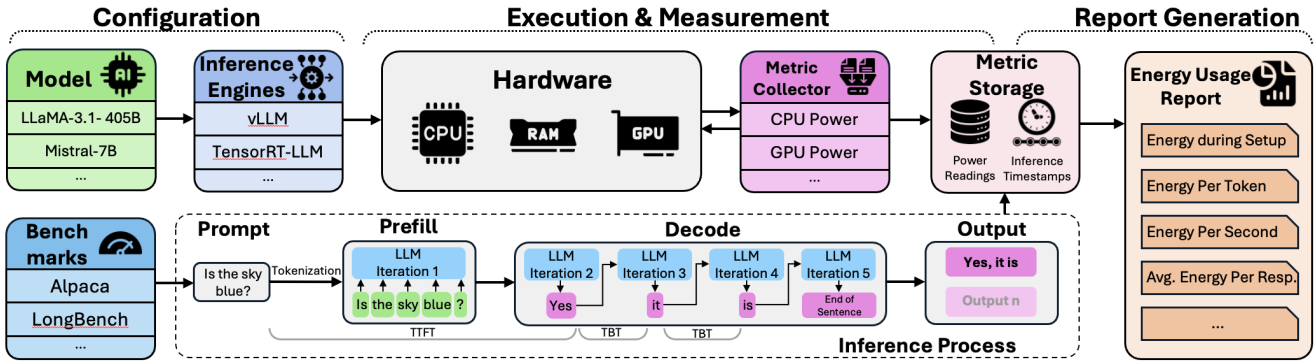

Figure 1: Overview of the TokenPowerBench Architecture

JSON format, including arbitrary lists of prompts, code snippets, or full-text articles.

- **Inference Engines**: TokenPowerBench supports four widely used engines on a single node: vLLM (Kwon et al. 2023), TensorRT-LLM (Vaidya, Oh, and Comly 2023), DeepSpeed (Aminabadi et al. 2022), and Transformers (Face 2024). To accommodate frontier-scale models such as Llama3-405B–which requires at least 780 GB of FP16 memory–we integrate support for distributed inference using the Ray framework. TokenPowerBench automatically launches Ray services across nodes and partitions model weights accordingly. Once initialized, inference engines are executed on Ray, and TokenPowerBench continues to collect power consumption data using a consistent, unified methodology.

Beyond the choice of model, inference engine, and dataset, the specific configuration of the inference service is a primary determinant of power consumption. TokenPowerBench is designed to explore the multi-dimensional configuration space that practitioners need to navigate. Instead of reporting a single performance number, it systematically varies key parameters to build a detailed power profile across different operational conditions. The primary axes of our benchmark scenarios are:

- **Hardware Configuration:** We define scenarios for single-GPU workstations, single-node multi-GPU servers (e.g., 8x H100s), and multi-node clusters, enabling direct comparisons of power draw in scale-up versus scale-out deployments.
- **Parallelism Strategy:** For large models, we evaluate the energy implications of different distribution strategies–primarily **Tensor Parallelism (TP)** and **Pipeline Parallelism (PP)**. TP partitions model layers across GPUs, increasing inter-GPU communication, while PP stages layers sequentially, introducing potential pipeline bubbles. These trade-offs manifest in distinct patterns of GPU utilization, idle time, and network power draw.
- **Workload Parameters:** We investigate the impact of dynamic request-level variables:
  - **Batch Size:** We sweep from batch size 1 (latency-critical interactive use) to large batches (throughput-oriented offline processing) to evaluate power consumption and energy costs.
  - **Context Length:** We vary the number of input prompt tokens to analyze the energy cost of the computationally-intensive prefill stage versus the memory-bandwidth-bound decode stage.
  - **Quantization:** We benchmark various numerical formats (e.g., FP16, FP8). While quantization reduces memory footprint and can accelerate computation, its overall impact on system-wide energy consumption remains a key question.

By sweeping through these scenarios, TokenPowerBench provides a holistic view of how operational choices, from hardware provisioning to software optimization, affect the energy efficiency of LLM inference.

Execution and Measurement

1) Spatial view: where the power is spent. During each run, TokenPowerBench samples power consumption from all major node components: *GPU*, *CPU*, *DRAM*, and–when sensors are available–the network interface and fan tray. GPU power is collected via NVML/DCGM; CPU and DRAM power via Intel RAPL; and full-node power via IPMI or a rack-mounted PDU. Storing these readings side-by-side enables insights such as GPUs typically account for over 60% of total energy use, while fans contribute only a few percent. Because all telemetry streams share the same timestamp, we can also sum them precisely to compute the total wall energy for each request.

2) Temporal view: when the power is spent. The same logger records two key stages of every inference: *prefill*, when the model reads the input tokens, and *decode*, when it produces new tokens. Each power sample is tagged with the stage that is active at that moment. After the run we integrate these tagged samples to obtain two clear numbers: energy consumed during prefill and energy consumed during decode. This separation reveals, for instance, that long prompts raise the prefill share, while large batch sizes increase the decode share. By combining spatial and temporal perspectives, we can determine both *which component* and

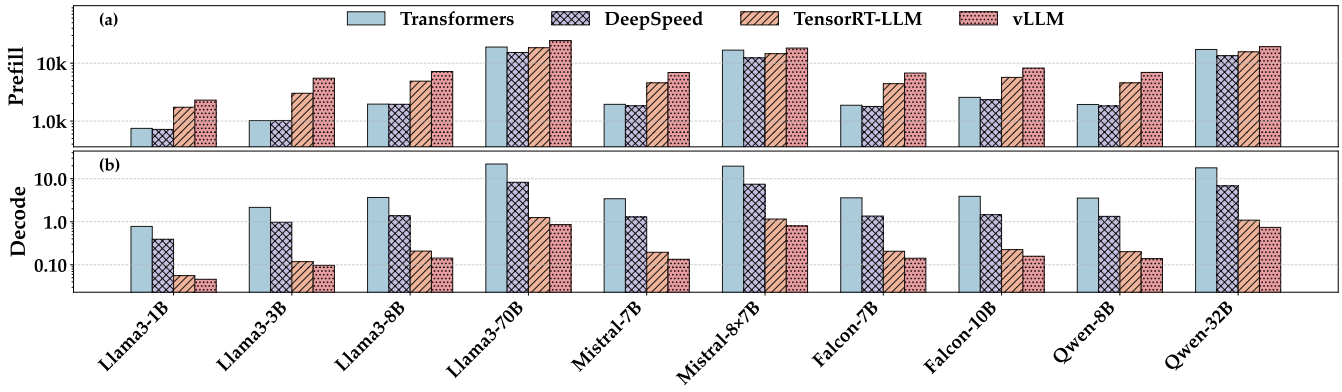

Figure 2: Prefill Energy (a) and Decode Energy per Token (b) Across Models and Inference Engines

which stage accounts for each joule consumed during LLM inference.

$$E_{\text{total}} = E_{\text{Prefill}} + E_{\text{Decode}} \quad (1)$$
$$= E_{\text{GPU}} + E_{\text{CPU}} + E_{\text{DRAM}} + E_{\text{Others}} \quad (2)$$

Report Generation

When an experiment ends, TokenPowerBench collects the log data and turns the raw time-stamped samples into a compact summary. First, it integrates the GPU, CPU, DRAM, and wall-plug traces over the two stages of inference including prefill and decode stages, so the user can see at a glance how many joules each component consumed in each part of the process. Users can get the data such as energy per token, energy per response, energy per second, peak powe, and energy consumed during prefill stage. Because every value is computed directly from the aligned samples, there is no need for post-hoc scaling or hand calculations.

The tool then writes the results in three complementary formats. A CSV file holds the numbers most people plot with Pandas or Excel; a matching JSON file fits easily into automated dashboards; If the user supplies electricity price and a regional carbon factor, the same pass also converts kilowatt-hours into dollar cost and CO_2 equivalents, so budget and sustainability discussions start from the same sheet as the technical metrics. Because every run follows the same pipeline, reports produced on different days or clusters line up without extra scripting, making it straightforward to track regressions or show savings after an optimization.

Evaluation

To demonstrate the utility and effectiveness of TokenPowerBench, we conduct an in-depth case study on a representative Nvidia H100 GPU cluster. Our evaluation is designed to showcase how the benchmark can be used to derive actionable insights into the performance, energy consumption, and cost of deploying large-scale LLMs.

Experimental Setup

Hardware The experiments were performed on a 8-node GPU cluster, each node is equipment with 4 NVIDIA H100 GPUs (94 GB memory each), paired with two Intel Xeon Gold 6426Y CPUs (16 cores, 32 threads each) and 512 GB of RAM.

Prompt Datasets We evaluate our benchmark on two datasets: Alpaca (Taori et al. 2023) and LongBench (Bai et al. 2023b). Alpaca contains 52,002 prompts generated by OpenAI's text-davinci-003 engine (OpenAI 2022). LongBench is an open-source benchmark and has longer prompts in average.

Cross-Model Power Consumption Benchmarking of LLMs

We tested all popular LLM models, including Llama3 - 1B, 3B, 8B, 70B and 405B; Mistral 7B, 24B, 8×7B, 8×22B; Qwen 8B, 32B and 480B; Falcon 7B, 10B and 180B. The representative results as shown in the following table. Please check the total results in the supplementary material.

Figure 2 compares energy consumed by prefill stage and total energy per token for 10 open-source models (dense and MoE) served by four mainstream engines: Transformers, DeepSpeed-Inference, TensorRT-LLM, and vLLM on a single node with 4 H100 GPUs. Within each LLM family, energy rises faster than the parameter count. For LLaMA-3, moving from 1 B to 70 B parameters increases energy per token by 7.3×, even though parameter count grows 70×. This super-linear trend confirms that larger models pay an extra cache-bandwidth and memory-traffic penalty beyond pure FLOPs.

Dense versus MoE. Mixtral-8×7B consumes roughly the same energy per token as a dense 8B model while delivering quality closer to a 56 B dense model. The sparse routing that activates only two experts per token cuts token energy by 23× compared with dense models of similar emergent accuracy.

Engine impact. Across *all* models, TensorRT-LLM and vLLM consume 3 times more than DeepSpeed and Transformers in the prefill stage. However, TensorRT-LLM and vLLM reduce energy per token by 2540% relative to Transformers engine due to the optimization techniques

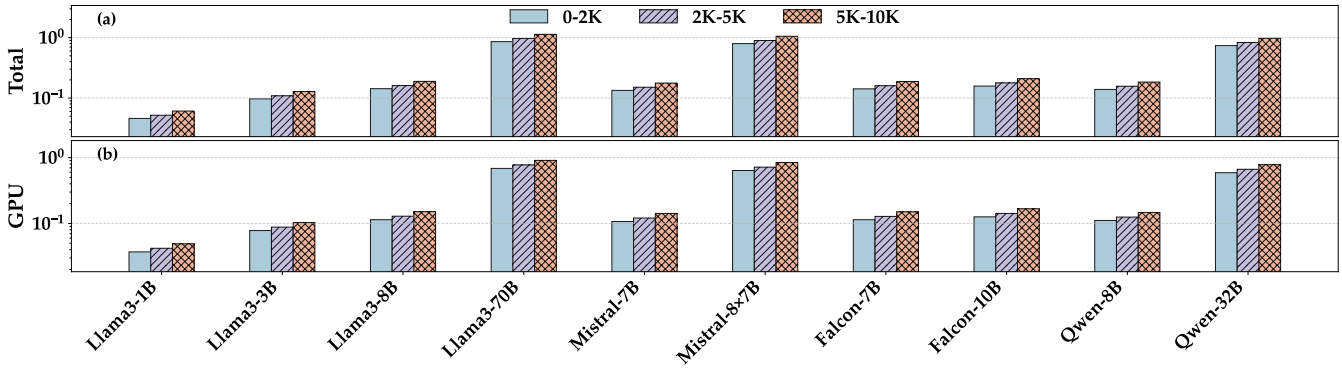

Figure 3: Comparison of Total (a) and GPU (b) Energy per Token Across Models at Varying Context Lengths

they adopt. DeepSpeed-Inference sits between the two extremes, showing that software optimization alone can rival architecture-level gains.

Parameter-sensitivity Analysis

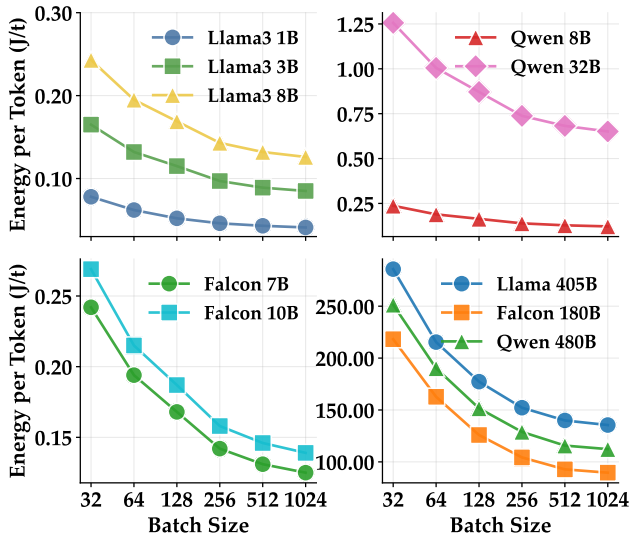

Figure 4: Comparison of Total Energy per Token Across Models at Varying Batch Sizes

Batch Size Figure 4 plots energy per token for batch sizes from 32 to 1024 on six model families, including Llama, Falcon, Mistral Qwen families. For every family, enlarging the batch lowers energy per token at first, because the fixed set-up and kernel-launch overheads are spread over more tokens. The steepest drop appears between 32 and 256, where most GPUs move from $< 50\%$ to nearly full utilization; the 70 B model, for example, cuts per-token energy by about 25% in this range.

Beyond batch 256 the curve flattens: power draw stays roughly constant while additional tokens add proportionally less work, so further efficiency gains are modest. Here batch sizes up to 1024 still fit in memory, so energy per token continues to fall, though at a slower rate, giving an overall two-to-three-fold spread between the smallest and largest batches.

Context Length Figure 3 reports energy (total and gpu) per token for ten models at three prompt length range : 0-2K, 2K to 5K, and 5K to 10K tokens. Across all models, energy grows steadily as the prompt gets longer because the prefill stage must process every input token while the compute cost of each new output token stays the same.

For the largest dense model we tested (Llama3 70B), the jump from 2K to 10 K tokens raises energy per token by roughly a factor of three; medium models (e.g., Llama 3 8B, Mistral 7B) see a smaller but still clear rise. The pattern is nearly identical in the GPU-only trace and in the node-level trace, confirming that most of the extra power is drawn by the accelerators rather than by the host CPU or memory.

Longer prompts therefore hurt energy efficiency in two ways: they increase the joules spent before the first output token appears, and they lower overall throughput because GPUs stay busy on attention over a larger context window.

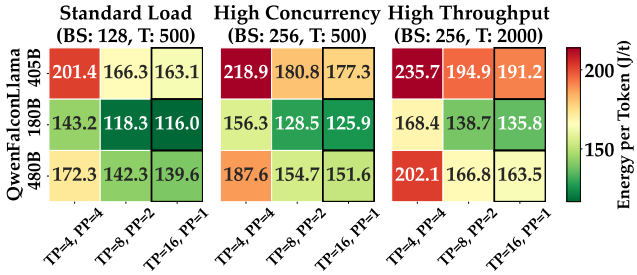

Figure 5: Heatmap of Energy per Token of SOTA models

Parallelism Strategy Study (TP / PP) **SOTA LLMs** Three workload configurations were used during the token generation stage to simulate the workloads of real-world scenarios: Standard Load, High Concurrency, and High Throughput.

The heatmap figure presents a heatmap for three SOTA models: Llama 3 405B, Falcon180B, and Qwen 480B on 16 H100 GPUs under three workload profiles. For each model

we tested three ways to split work across the GPUs: a balanced mix of tensor and pipeline parallelism with TP 4 and PP 4, a tensor-heavy mix with TP 8 and PP 2, and pure tensor parallelism with TP 16 and PP 1. Green cells in the figure mark lower energy use.

Across every workload the **pure tensor parallelism** setting delivers the best energy efficiency because long pipelines leave some GPUs idle. The gap between the best and worst split widens as the workload grows from about 40 J/token under Standard Load to more than 60 J/token under High Throughput. The results show that parallelism tuning matters most in heavy, batch-oriented jobs.

Quantization - A Case Study of Llama 3 405B (FP16 vs FP8) Figure 6 compares full-precision FP16 (16-bit Floating Point) inference with FP8 (8-bit Floating Point) weight quantization for Llama 3 405B. Across the three workload profiles: Standard Load, High Concurrency, and High Throughput, quantization cuts energy per token by roughly 30%. The saving is consistent regardless of the traffic pattern because the reduced-precision weights lower both memory traffic and arithmetic cost in every phase of generation. Measured per batch, total energy falls from about 45 kJ to 32 kJ under the heaviest load, mirroring the token-level reduction.

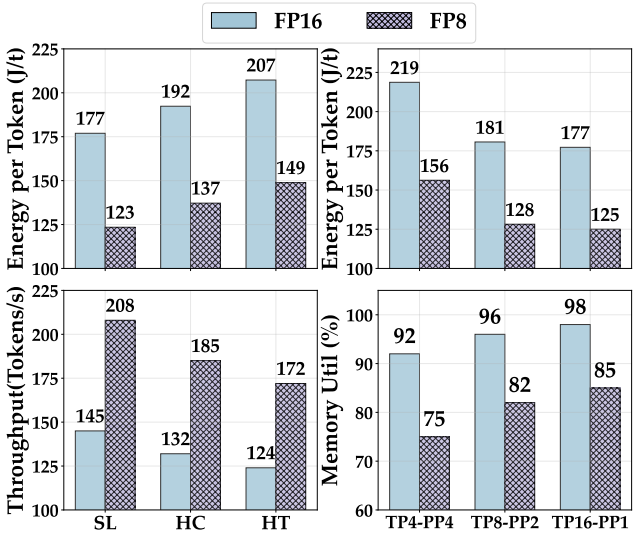

Figure 6: Comparison of FP16 and FP8 Performance for the LLaMA 3 405B Model Across Varying Workloads and Parallelism Levels. (SL: Standard Load, HC: High Concurrency, HT: High Throughput)

Performance also benefits by the quantization. FP16 increases effective memory-bandwidth utilization by 1317 percentage, but FP8 raises end-to-end throughput from about 48-63 tokens/s in the largest batch setting, without noticeable accuracy loss in our prompt set. Together, these results confirm that low-precision formats can deliver a double dividend lower power and higherspeed, when the underlying hardware and kernels fully support them.

Conclusion and Future Work

TokenPowerBenchs core contribution is a lightweight, extensible, and reproducible framework for benchmarking the power and energy consumption of LLM inference across diverse hardware configurations, model families, and workload patterns. We briefly summarize the core aspects of our benchmark design here.

Energy-normalized inference metrics. To enable meaningful comparison of inference systems, TokenPowerBench introduces a set of energy-centric metrics including Joules per token, Joules per response and instantaneous power draw. These metrics are aligned with the internal execution phases of transformer-based LLMs (prefill and decode), enabling fine-grained attribution of energy costs. Unlike traditional performance metrics like latency or throughput, these energy metrics directly reflect sustainability and cost-efficiency, which are critical concerns in modern AI infrastructure.

Reproducible measurement methodology. Power measurement in LLM inference is highly sensitive to hardware access level and instrumentation fidelity. TokenPowerBench defines a three-level measurement modelfrom GPU-only telemetry to full-system power monitoring using IPMI and rack-level PDUsallowing users to adopt the benchmark in environments ranging from user-space workstations to institutional testbeds. We pair this with a declarative configuration harness that ensures repeatability across experiments.

Scalable scenario design. LLM inference workloads vary widely based on system scale, model size, and deployment constraints. TokenPowerBench systematically explores key configuration dimensions, including batch size, context length, quantization format, parallelism strategy to expose how each factor impacts energy usage. We support scaling from single-GPU inference to multi-node distributed serving, capturing both component-level breakdowns and cluster-wide energy imbalances.

The landscape of LLM inference is rapidly evolving, and TokenPowerBench is designed to evolve with it. We will extend coverage beyond our current H100 testbed to other GPU architectures, including the next NVIDIA generations and AMD accelerators, as well as emerging AI chips and DPUs. We will also polish the benchmark and extend the functionality to quantify the trade-off between inference accuracy and energy efficiency, providing users with guidance on where energy savings begin to erode model quality.

Acknowledgments

We are thankful to the anonymous reviewers for their valuable feedback. This research is supported in part by the National Science Foundation under grant OAC-2404438 and CNS-1939140 (A U.S. National Science Foundation Industry-University Cooperative Research Center on Cloud and Autonomic Computing). We are also very grateful to the High Performance Computing Center of Texas Tech University for providing HPC resources for this project.

References

Almazrouei, E.; Alobeidli, H.; Alshamsi, A.; Cappelli, A.; Cojocaru, R.; Debbah, M.; Goffinet, É.; Hesslow, D.; Launay, J.; Malartic, Q.; et al. 2023. The Falcon Series of Open Language Models.

Aminabadi, R. Y.; Rajbhandari, S.; Awan, A. A.; Li, C.; Li, D.; Zheng, E.; Ruwase, O.; Smith, S.; Zhang, M.; Rasley, J.; et al. 2022. Deepspeed-Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–15. IEEE.

Argonne Leadership Computing Facility. 2025. ALCF AI Testbed. https://www.alcf.anl.gov/alcf-ai-testbed. Accessed: 2025-07-22.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023a. Qwen Technical Report. *arXiv preprint arXiv:2309.16609*.

Bai, Y.; Lv, X.; Zhang, J.; Lyu, H.; Tang, J.; Huang, Z.; Du, Z.; Liu, X.; Zeng, A.; Hou, L.; et al. 2023b. Longbench: A Bilingual, Multitask Benchmark for Long Context Understanding.

Banbury, C.; Reddi, V. J.; Torelli, P.; Holleman, J.; Jeffries, N.; Kiraly, C.; Montino, P.; Kanter, D.; Ahmed, S.; Pau, D.; et al. 2021. MLPerf Tiny Benchmark. *arXiv preprint arXiv:2106.07597*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-shot Learners.

Chitty-Venkata, K. T.; Raskar, S.; Kale, B.; Ferdaus, F.; Tanikanti, A.; Raffenetti, K.; Taylor, V.; Emani, M.; and Vishwanath, V. 2024. LLM-Inference-Bench: Inference Benchmarking of Large Language Models on AI Accelerators. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1362–1379. IEEE.

Corporation, I. 2006. Intelligent Platform Management Interface Specification, v2.0. https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-specifications.html.

Corporation, N. 2023. PyNVML: Python Bindings for the NVIDIA Management Library (NVML). https://github.com/NVIDIA/pynvml.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

DMTF. 2023. Redfish API Specification, v1.18.0. Online. DSP0266.

Face, H. 2024. Transformers: State-of-the-art Machine Learning for PyTorch TensorFlow and JAX.

Feng, W.-c.; and Cameron, K. 2007. The Green500 List: Encouraging Sustainable Supercomputing. *Computer*, 40(12): 50–55.

Gartner, Inc. 2025. Gartner Predicts by 2028, 80% of GenAI Business Apps Will Be Developed on Existing Data Management Platforms. Gartner Newsroom. Available at: https://www.gartner.com/en/newsroom/press-releases/2025-06-02-gartner-predicts-by-2028-80-percent-of-genai-business-apps-will-be-developed-on-existing-data-management-platforms, Accessed: 2025-07-29.

Hutt, G.; Viswanathan, V.; and Nadolski, A. 2019. Deliver High Performance ML Inference with AWS Inferentia.

Intel Corporation. 2023. Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide, Part 2. Online. Chapter 14.9: Running Average Power Limit (RAPL).

Jegham, N.; Abdelatti, M.; Elmoubarki, L.; and Hendawi, A. 2025. How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference. *arXiv preprint arXiv:2505.09598*.

Jetstream2. 2025. Jetstream2 Offers LLM Inference Service for Research. https://jetstream-cloud.org/news-events/news/4-11-25_llm-inference-service.html. Accessed: 2025-07-22.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of Experts.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with Pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.

Li, J.; Ali, G.; Nguyen, N.; Hass, J.; Sill, A.; Dang, T.; and Chen, Y. 2020. Monster: an Out-of-the-box Monitoring Tool for High Performance Computing Systems. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, 119–129. IEEE.

Li, J.; Michelogiannakis, G.; Cook, B.; Cooray, D.; and Chen, Y. 2023. Analyzing Resource Utilization in an HPC System: A Case Study of NERSCs Perlmutter. In *International Conference on High Performance Computing*, 297–316. Springer.

Li, J.; Wong, S.-Z.; Wan, G.-W.; Wang, X.; and Yang, J. 2025. EDA-Debugger: An LLM-Based Framework for Automated EDA Runtime Issue Resolution. In *2025 26th International Symposium on Quality Electronic Design (ISQED)*, 1–7. IEEE.

Liu, T.; Tian, Q.; Ye, J.; Fu, L.; Su, S.; Li, J.; Wan, G.-W.; Zhang, L.; Wong, S.-Z.; Wang, X.; et al. 2024. ChatChisel: Enabling Agile Hardware Design with Large Language Models. In *2024 2nd International Symposium of Electronics Design Automation (ISEDA)*, 710–716. IEEE.

MarketsandMarkets. 2024. Large Language Model (LLM) Market by Type (Domain-specific (Zero-shot, One-shot, Few-shot), General-purpose, Multilingual), Model Architecture (Autoregressive Language Models, Autoencoding Language Models, Hybrid Language Models) - Global Forecast to 2030. MarketsandMarkets. Report Code: TC 8977, Available at: https://www.marketsandmarkets.com/Market-Reports/large-

language-model-llm-market-102137956.html, Accessed: 2025-07-29.

MarketsandMarkets. 2025. AI Inference Market by Compute (GPU, CPU, FPGA), Memory (DDR, HBM), Network (NIC/Network Adapters, Interconnect), Deployment (On-premises, Cloud, Edge), Application (Generative AI, Machine Learning, NLP, Computer Vision) - Global Forecast to 2030. MarketsandMarkets. Report Code: SE 9299, Available at: https://www.marketsandmarkets.com/Market-Reports/ai-inference-market-189921964.html, Accessed: 2025-07-29.

Mattson, P.; Cheng, C.; Diamos, G.; Coleman, C.; Micikevicius, P.; Patterson, D.; Tang, H.; Wei, G.-Y.; Bailis, P.; Bittorf, V.; et al. 2020. MLPerf Training Benchmark. *Proceedings of Machine Learning and Systems*, 2: 336–349.

Microsoft Corporation. 2025. 2025 Environmental Sustainability Report. Microsoft Corporate Responsibility. Available at: https://www.microsoft.com/en-us/corporate-responsibility/sustainability/report/, Accessed: 2025-07-29.

Niu, C.; Zhang, W.; Byna, S.; and Chen, Y. 2022. Kv2vec: A Distributed Representation Method for Key-value Pairs from Metadata Attributes. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–7. IEEE.

Niu, C.; Zhang, W.; Byna, S.; and Chen, Y. 2023. PSQS: Parallel Semantic Querying Service for Self-describing File Formats. In *2023 IEEE International Conference on Big Data (BigData)*, 536–541. IEEE.

Niu, C.; Zhang, W.; Side, M.; and Chen, Y. 2025a. ICEAGE: Intelligent Contextual Exploration and Answer Generation Engine for Scientific Data Discovery. In *Proceedings of the 37th International Conference on Scalable Scientific Data Management*, 1–10.

Niu, C.; Zhang, W.; Zhao, Y.; and Chen, Y. 2025b. Energy Efficient or Exhaustive? Benchmarking Power Consumption of LLM Inference Engines. *ACM SIGENERGY Energy Informatics Review*, 5(2): 56–62.

Niu, J.; Liu, X.; Niu, D.; Wang, X.; Jiang, Z.; and Guan, N. 2025c. Rechisel: Effective Automatic Chisel Code Generation by LLM with Reflection. *arXiv preprint arXiv:2505.19734*.

OpenAI. 2022. text-davinci-003 [Large language model]. https://platform.openai.com/docs/models/gpt-3.

Poddar, S.; Koley, P.; Misra, J.; Podder, S.; Ganguly, N.; and Ghosh, S. 2025. Towards Sustainable NLP: Insights from Benchmarking Inference Energy in Large Language Models. *arXiv preprint arXiv:2502.05610*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving Language Understanding by Generative Pre-Training. arXiv:2311.00176.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language Models are Unsupervised Multitask Learners.

Reddi, V. J.; Cheng, C.; Kanter, D.; Mattson, P.; Schmuelling, G.; Wu, C.-J.; Anderson, B.; Breughe, M.; Charlebois, M.; Chou, W.; et al. 2020. MLPerf Inference Benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 446–459. IEEE.

Samsi, S.; Zhao, D.; McDonald, J.; Li, B.; Michaleas, A.; Jones, M.; Bergeron, W.; Kepner, J.; Tiwari, D.; and Gadepally, V. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–9. IEEE.

Stefanov, K. S.; Pawar, S.; Ranjan, A.; Wandhekar, S.; and Voevodin, V. V. 2021. A Review of Supercomputer Performance Monitoring Systems. *Supercomputing Frontiers and Innovations*, 8(3): 62–81.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. LLaMA: Open and Efficient Foundation Language Models.

Tschand, A.; Rajan, A. T. R.; Idgunji, S.; Ghosh, A.; Holleman, J.; Kiraly, C.; Ambalkar, P.; Borkar, R.; Chukka, R.; Cockrell, T.; et al. 2025. MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Systems from μWatts to MWatts for Sustainable AI. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 1201–1216. IEEE.

Vaidya, N.; Oh, F.; and Comly, N. 2023. Optimizing Inference on Large Language Models with Nvidia Tensorrt-LLM, now Publicly Available.

Wan, G.-W.; Wong, S.-Z.; Liu, D.; and Wang, X. 2024. Assessment of Pre-Trained Large Language Models for Hardware Trojan Detection in RTL Designs. In *2024 IEEE LLM Aided Design Workshop (LAD)*, 1–1. IEEE.

Wang, X.; Wan, G.-W.; Wong, S.-Z.; Zhang, L.; Liu, T.; Tian, Q.; and Ye, J. 2024. ChatCPU: An Agile CPU Design and Verification Platform with LLM. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 1–6.

Xu, Q.; Stok, L.; Drechsler, R.; Wang, X.; Zhang, G. L.; and Markov, I. L. 2025. Revolution or Hype? Seeking the Limits of Large Models in Hardware Design. *arXiv preprint arXiv:2509.04905*.

Ye, J.; Liu, T.; Tian, Q.; Su, S.; Jiang, Z.; and Wang, X. 2025. ChatModel: Automating Reference Model Design and Verification with LLMs. *arXiv preprint arXiv:2506.15066*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. OPT: Open Pre-trained Transformer Language Models.

Zhang, W.; Byna, S.; Niu, C.; and Chen, Y. 2019. Exploring Metadata Search Essentials for Scientific Data Management. In *2019 IEEE 26th international conference on high performance computing, data, and analytics (HiPC)*, 83–92. IEEE.