Exploring the Energy Usage of PyTorch LLM Inference

Grant Wilkins
gfw27@cam.ac.uk
University of Cambridge
Cambridge, UK

ABSTRACT

As large language models (LLMs) continue proliferating, resolving efficiency bottlenecks is critical for unlocking their full potential while respecting sustainability constraints. This paper investigates the energy consumption tradeoffs between scale and efficiency for PyTorch-based LLM inference across different hardware configurations. We employ three open-source models - Llama-2, Phi-2, and Mistral - and profile their GPU/CPU power draw during inference on an M1 Pro laptop and multi-GPU cluster. Our results reveal that while distributed parallelism substantially improves throughput, translation into real-world energy savings is not guaranteed. Efficiency peaked at model-dependent "sweet spots" before declining due to factors like communication overhead. The findings quantify complex co-optimization needs across model architecture, software, and hardware to balance performance and carbon costs. By spotlighting leftover optimization opportunities, this initial study motivates future efforts at sustainable AI through system co-design.

KEYWORDS

Large-Scale Data, LLMs, Energy Consumption, Green Computing

ACM Reference Format:
Grant Wilkins. 2024. Exploring the Energy Usage of PyTorch LLM Inference. In *Proceedings of ACM Conference (HotCarbon'24).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With modern data-driven decision-making, large-scale data processing systems have become the backbone of modern computational infrastructure. Among these, large language models (LLMs), especially those operated on frameworks like PyTorch [17], represent a significant advancement in artificial intelligence. However, the growing sophistication of these models brings with it an often-overlooked challenge: energy consumption. The energy efficiency of LLMs during inference is not just an environmental concern but also a practical constraint for their scalability and applicability [19, 22].

The relevance of this problem extends beyond environmental sustainability. In an era where computational resources are increasingly democratized, the energy requirements of these models can be a limiting factor for their widespread adoption [4]. This issue is particularly acute in large-scale data processing systems, where the need for rapid and efficient processing of vast datasets is paramount. Interestingly, the architecture of these systems plays a pivotal role in their energy consumption patterns. Conventional wisdom suggests that massively parallel systems, designed to handle large-scale data processing tasks, would optimize energy efficiency and environmental impact [12, 19]. However, this hypothesis overlooks the potential inefficiencies and increased energy demands associated with scaling such systems. We suspect that a 'sweet spot' might exist - an optimal balance between scale and energy consumption, where the benefits of parallel processing are harnessed without disproportionately increasing energy demands.

This paper does a rudimentary exploration of this hypothesis, focusing on the energy consumption of large language models during inference in PyTorch. We aim to investigate whether massively parallel systems consume more energy than a singular, albeit powerful, computational node. Through this exploration, we seek to contribute to the broader discourse on sustainable AI development, providing insights that could guide future architectural choices in large-scale data processing systems.

2 BACKGROUND

2.1 Inference of Large Language Models

Introducing transformer-based neural network architectures has led to impressive gains in the performance of large language models (LLMs) on language understanding capabilities [3]. LLMs such as BERT [5], GPT-4 [16], and PaLM-2 [1] have demonstrated human-level proficiency on many language benchmarks while requiring billions of parameters and massive datasets for training.

2.2 Energy Consumption in AI Systems

Recent studies have found that the computational requirements for state-of-the-art AI entail massive energy consumption and carbon emissions that can question the benefits these systems provide [19]. The energy intensity of AI systems can be broadly divided into the energy required for training versus inference after models are deployed [9]. Training complex models on massive datasets is an exceptionally energy-intensive process, with estimates finding that a single training run of large language models like GPT-3 can emit as much carbon as five cars over their entire lifetimes and evaporate over 700,000L of groundwater [2, 12]. On the inference side, neural networks also lead to notable emissions depending on deployment scale and hardware efficiency [19]. For example, inference by natural language processing systems on cloud infrastructure can emit thousands of kilograms of carbon dioxide per year [4]. Optimizing software and hardware specifically for efficient AI workloads is thus essential even after models are trained.

2.3 Relating System Architecture and Energy Efficiency

The architecture of compute nodes can significantly impact large-scale computing systems' energy efficiency and processing capabilities. Conventional server architectures based on x86 multicore CPUs face energy proportionality and scalability limitations for modern data-intensive workloads [14]. However, finding the right balance between accelerators and CPUs remains an open challenge [10]. Several works have explored heterogeneous server configurations to improve efficiency. More extensive systems only sometimes translate to higher energy efficiency, as communication overhead can outweigh gains [6]. Still, specialized clusters like NVIDIA's DGX show 4x better performance per watt over conventional servers [20]. Modern systems demonstrate a complex interplay between scale, architecture, workload behavior and efficiency objectives. This paper investigates these factors for large language models running inference on PyTorch.

We aim to characterize the node configurations that offer the best tradeoff between processing power and energy consumption. The findings could guide the development of next-generation systems purpose-built for emerging AI services and their explosive growth.

3 METHODS

3.1 PyTorch Distributed Inference Platform

Our study leverages PyTorch [17] for inference tasks with large language models (LLMs), focusing on its capability for large-scale data processing. PyTorch uses a dynamic computation graph, which offers flexibility and intuitiveness in model development and testing. For inference with LLMs, PyTorch's memory usage efficiency and distributed computing capabilities are critical. PyTorch's distributed architecture allows for the scalable deployment of models across multiple GPUs or nodes, which is essential for handling the computational demands of LLMs. This scalability is achieved through its Distributed Data-Parallel (DDP) [13] framework, which efficiently manages model replication and gradient aggregation across multiple processing units.

Moreover, PyTorch provides optimized performance on NVIDIA GPUs using CUDA, further enhancing the speed and efficiency of the inference process. Performance on Apple Silicon provides similar efficiency, yet it is a newer feature due to the young age of the hardware platform. In our experiments, this combination of scalability, efficiency, and flexibility makes PyTorch a suitable framework for evaluating advanced LLMs' energy consumption and computational demands, providing insights into their operational implications in real-world applications.

We use Huggingface's Accelerate [7] to control the scaling across multiple nodes and GPUs in our implementation. This software uses a backend that references the available GPU resources across different architectures and takes full advantage of sharding models across hardware. The four major parts of this are (1) loading the tokenizer, (2) loading the model, (3) initializing the pipeline, and (4) performing the inference. These are shown below in Algorithm 1.

Algorithm 1 PyTorch Infrence Steps

```
Require: model_name: str, prompt: str
1: tokenizer = AutoTokenizer.from_pretrained(model_name)
2: model = AutoModelForCausalLM.from_pretrained(model_name, device_map="auto")
3: pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, device_map="auto")
4: sequences = pipe(prompt)
```

3.2 Model Selection

Our study employs three open-source LLMs for their capabilities and ability to efficiently run on diverse hardware: (1) Llama-2 (7B parameters) [21], (2) Phi-2 (2B parameters) [8], and Mistral (7B parameters) [11]. These models were selected to represent a spectrum of architectures, and below, we will go into more depth as to their specific features. We subject each model to a series of standardized NLP tasks to evaluate their energy consumption during inference.

3.2.1 Llama-2. We select Llama 2 for its optimization in dialogue tasks and its improvements in safety and helpfulness. The model's unique pretraining methodologies and advanced architectural features, such as grouped-query attention, make it an ideal candidate for analyzing energy efficiency in complex language tasks.

3.2.2 Phi-2. We use Phi-2 for its approach to teaching smaller models to reason, standing out for its ability to perform complex tasks. It was trained almost entirely on textbook material and is a landmark model for its capabilities at a small parameter count.

3.2.3 Mistral. We include Mistral for its grouped-query attention and sliding window attention mechanisms, contributing to fast and efficient inference. Its superior performance in various benchmarks, especially in reasoning, mathematics, and code generation, makes it an essential model for our analysis.

3.3 Energy Profiling

We monitor the energy usage of each model using a customized setup that captures real-time power metrics. This setup includes monitoring GPU power usage and other system-level indicators to provide a comprehensive view of each model's energy footprint. For this study, we focus on energy measurement for the GPU due to the AMD CPU's inability to provide energy readings.

3.3.1 NVIDIA GPUs. We use PyJoules [18], a Python-based energy measurement library, to quantify the energy consumption associated with inference on NVIDIA GPUs. PyJoules provides an interface to `nvidia-smi` [15], providing a granular and accurate energy usage assessment for targeted NVIDIA devices. This tool offers real-time energy consumption of GPUs for a given tracked process, which is a critical component of our analysis given the GPU-heavy computation involved in LLM inference.

3.3.2 Apple M1 GPU. Our study extends to evaluating the energy efficiency of language models on Apple's M1 GPU. There are no standard energy measurement tools available for this task. Therefore, we employ a Python-based approach integrating macOS's `powermetrics` utility, providing a detailed view of the energy usage during model inference. To capture the energy consumption of the M1 GPU, we execute the `powermetrics` command through a Python subprocess. We record the power draw of each process, including detailed metrics for CPU and GPU power consumption at 500ms intervals. The energy monitoring is conducted concurrently with the model inference. A separate thread is dedicated to running the `powermetrics` command, ensuring uninterrupted and real-time data collection. We launch a separate process for each step in Algorithm 1. Post-operation, the collected data is processed to extract the recorded power data and then find the energy consumption through integration over the runtime. Through testing, we confirmed we are the only process on the GPU during this operation; therefore the energy, $E_{GPUtotal}$, is easy to calculate for each recorded $P_{GPU,i}$ then can just be summed at each timestep Δt_i.

$$E_{GPUtotal} = \sum_i P_{GPU,i} \Delta t_i$$

However, the CPU power draw data is reported in total, where an "impact factor" for our PyTorch process is given as a fraction of the total energy. Therefore, the CPU energy, $E_{CPUtotal}$, must be inferred from a scaled $P_{CPU,i}$ with the impact factor α_i from each timestep

$$E_{CPUtotal} = \sum_i (\alpha_i P_{CPU,i}) \Delta t_i.$$

A significant difference between these two methods is that we receive energy directly from the NVIDIA system but must calculate it from the Apple system. Therefore, we can more finely visualize the Apple system and its different phases in our results.

3.4 Our Testing Strategy

We aim to compare how PyTorch LLM inference on an extensive scalable system with multiple GPUs compares energy and runtime efficiency to a commodity-level laptop with a GPU. Using the models identified in Section 3.2, we will profile their energy consumption and runtime for the different phases of the inference process as shown in Algorithm 1. From this, we can compare the two systems and observe how a scalable cluster handles LLMs from an energy perspective. For the Apple system, we run our tests 25 times per model, and on the Swing system, we run our tests 25 times per model per GPU count, scaling from 1 to 8 GPUs by powers of 2.

4 RESULTS

4.1 Hardware and Software Versions

In our study we employ the following hardware and software. The MacBook Pro uses an M1 Pro Chip with a 10-core CPU, 16-core GPU, and 32GB of RAM. The Argonne Swing cluster has six nodes with each node 8×NVIDIA A100 40GB GPUs and 2×AMD EPYC 7742 64-Core Processors with 128 cores. We utilize PyTorch v2.0.1, Torchvision v0.15.2, Numpy v1.26.0, Huggingface v0.20.2, and Accelerate v0.26.1.

4.2 Profiling of Inference Phases

Algorithm 1 shows the four primary steps of performing inference. In each of our tests we separate our profiling for these checkpoints. As mentioned at the end of Section 3.3.2, due to `powermetrics` providing raw power outputs, we can visualize the power profile of the inference for our system much better. One can see these different phases well for the red lines corresponding to these steps in Figure 1. One exception is Llama-2 in Figure 1(a), which only shows three lines, as the model loading and pipeline loading phase are the same due to the lack of transparency on the model architecture [21]. As can be seen, the significant amount of time spent on the model is in inference, with higher power consumption during this phase for both the GPU and CPU.

The inference phase profiling reveals several insightful patterns regarding the interplay between model architecture, hardware configuration, and computational efficiency.

Firstly, the significant inference time and energy usage align with expectations - this is the most computationally intensive part of the pipeline. However, the degree of intensity varies across models in ways that provide clues into their design. Mistral consumes over 2x more power than Phi-2 during inference on the MacBook Pro, likely due to its larger parameter size, which places greater demands. Comparing the GPU and CPU power profiles also reveals where the computational load is concentrated. The GPU dominates energy draw during inference for all models, confirming it handles most model calculations. Still, the CPU contributes non-trivial amounts, highlighting the importance of holistic system optimization.

4.3 Energy Comparison for Scalability

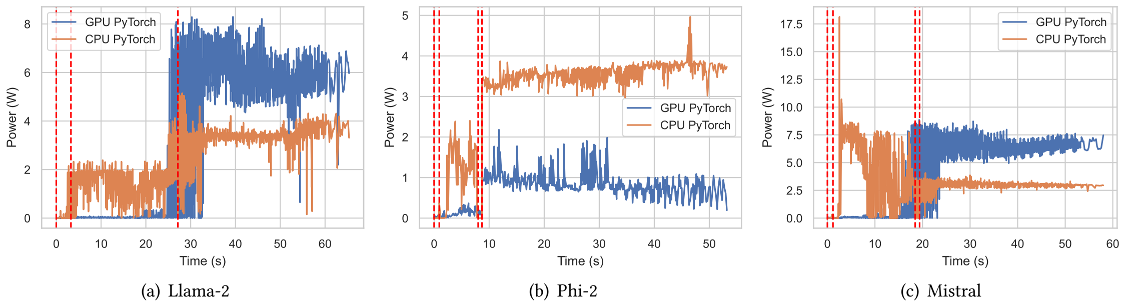
The results in Table 1 encapsulate the critical tradeoffs between response speed, throughput, and energy efficiency when scaling up inference hardware. Depending on the configuration, the 2-8× higher throughput on the Swing cluster comes at a 1.5-4× increase in energy consumption. However, the efficiency picture is more complex than simply a speed vs energy tradeoff. The superlinear rise in cluster energy usage, as shown in Figure 3, indicates additional constraints come into play. Communication overheads that grow with scale likely contribute, as do hardware utilization and load-balancing aspects. The fact that parallelism gains manifest differently across models supports this. More extensive models like Llama-2 better leverage added resources, achieving near-linear speedups of up to 4 GPUs before efficiency declines. Yet Mistral's downturn starts at 2 GPUs, while Phi-2 peaks at just 1. This reflects differences in how models map to hardware, with dimensionalities suiting some architectures better.

Overall, sweet spots show that while scale offers performance gains, whether it translates to real-world efficiency depends on balancing all aspects of the inference pipeline. This includes fitting model workloads, optimizing data movement, and mitigating duplicated computation. The fact that such inefficiencies emerge even on a state-of-the-art cluster indicates substantial room for tailoring hardware to inference tasks.

5 DISCUSSION AND FUTURE WORK

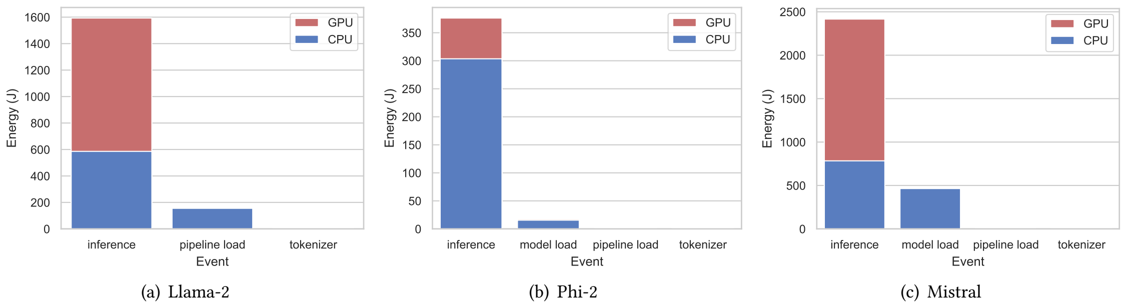
Our findings reveal an interesting story for LLM inference efficiency. While parallel distributed systems offer substantial performance gains, translation into real-world energy savings is not automatic. Without careful co-optimization of each component, inefficient resource usage can emerge to diminish or even reverse intended benefits.

The significant inference-phase energy draw highlights the need to look beyond model training when evaluating sustainability. The



(a) Llama-2 (b) Phi-2 (c) Mistral

Figure 1: Power Profile of LLMs on MacBook Pro



(a) Llama-2 (b) Phi-2 (c) Mistral

Figure 2: Per Phase Energy Consumption of LLMs on MacBook Pro

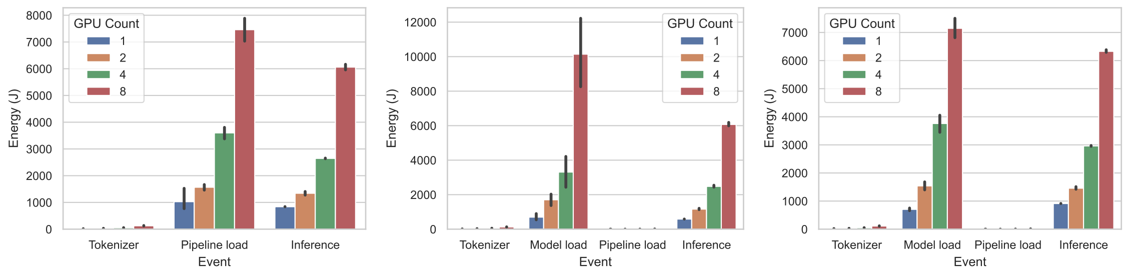
Table 1: Comparison Across Different Models for 200 Token Inference

Model	System	Tokenization Rate (tokens/s)	GPU Energy per Token (J/token)	GPU Energy (J)
Llama-2 (7B)	1×M1 Pro	5.265	5.0367	1007.382
	1×A100	32.891	3.138	627.555
	2×A100	25.045	4.899	979.705
	4×A100	20.574	10.499	2099.730
	8×A100	17.040	22.748	4549.635
Phi-2 (2B)	1×M1 Pro	4.497	0.363	72.699
	1×A100	34.581	1.621	324.270
	2×A100	25.046	3.619	723.785
	4×A100	20.639	7.304	1460.703
	8×A100	15.688	20.436	4087.186
Mistral (7B)	1×M1 Pro	5.111	8.162	1632.403
	1×A100	29.967	2.040	408.083
	2×A100	22.910	3.781	756.169
	4×A100	19.461	8.471	1694.125
	8×A100	16.267	16.998	3399.509

carbon impact of AI depends on total lifecycle costs across development, deployment, and operation [4]. Our results quantify efficiency tradeoffs in this last, oft-overlooked stage to demonstrate leftover low-hanging opportunities after models are already trained.

However, more research is needed to extend these initial conclusions towards actionable insights that can guide the development of next-generation architectures tailored for AI workloads. Further hyperparameter tuning and expanded sets of models, hardware, and inference tasks could reveal additional patterns around optimal configurations. Detailed breakdowns tracing resource usage would provide greater visibility into bottlenecks to address. Furthermore, future work should connect observed efficiency trends to concrete environmental impacts through lifecycle assessments and carbon accounting. Translating computational energy savings to emissions reductions is essential for quantifying sustainability gains.

This paper shows a complex interplay of factors influencing practical LLM efficiency. Significant opportunities exist to co-design solutions that balance inference performance and energy consumption. We hope these initial findings inspire future efforts toward



(a) Llama-2 (b) Phi-2 (c) Mistral

Figure 3: Per Phase Energy Consumption of LLMs on A100

responsible and scalable AI that pushes the boundaries of computational power and environmental stewardship.

ACKNOWLEDGMENT

We gratefully acknowledge the computing resources provided on Swing, a high-performance computing cluster operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

REFERENCES

[1] Anil, R., Dai, A. M., Firat, O., and et al. Palm 2 technical report, 2023.
[2] Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (New York, NY, USA, 2021), FAccT '21, Association for Computing Machinery, p. 610–623.
[3] Bommasani, R., Hudson, D. A., Adeli, E., and et al. On the opportunities and risks of foundation models, 2022.
[4] Chien, A. A., Lin, L., Nguyen, H., Rao, V., Sharma, T., and Wijayawardana, R. Reducing the carbon impact of generative ai inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (New York, NY, USA, 2023), HotCarbon '23, Association for Computing Machinery.
[5] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
[6] Georgiou, Y., Glesser, D., and Trystram, D. Adaptive resource and job management for limited power consumption. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop* (USA, 2015), IPDPSW '15, IEEE Computer Society, p. 863–870.
[7] Gugger, S., Debut, L., Wolf, T., Schmid, P., Mueller, Z., Mangrulkar, S., Sun, M., and Bossan, B. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.
[8] Gunasekar, S., Zhang, Y., Aneja, J., and et al. Textbooks are all you need, 2023.
[9] Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., and Pineau, J. Towards the systematic reporting of the energy and carbon footprints of machine learning. *J. Mach. Learn. Res. 21*, 1 (jan 2020).
[10] Hiremath, T., and Rekha, K. Energy efficient data migration concerning interoperability using optimized deep learning in container-based heterogeneous cloud computing. *Advances in Engineering Software 183* (2023), 103496.
[11] Jiang, A. Q., Sablayrolles, A., Mensch, A., and et al. Mistral 7b, 2023.
[12] Li, P., Yang, J., Islam, M. A., and Ren, S. Making ai less "thirsty": Uncovering and addressing the secret water footprint of ai models, 2023.
[13] Li, S., Zhao, Y., Varma, R., and et al. Pytorch distributed: Experiences on accelerating data parallel training, 2020.
[14] Lo, D., Cheng, L., Govindaraju, R., Ranganathan, P., and Kozyrakis, C. Heracles: Improving resource efficiency at scale. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)* (2015), pp. 450–462.
[15] NVIDIA. Nvidia-smi. http://developer.download.nvidia.com/compute/DCGM/docs/nvidia-smi-367.38.pdf, Accessed 2024. Available online.
[16] OpenAI, :, Achiam, J., Adler, S., Agarwal, S., and et al. Gpt-4 technical report, 2023.
[17] Paszke, A., Gross, S., Massa, F., Lerer, A., and et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Curran Associates Inc., Red Hook, NY, USA, 2019.
[18] powerapi ng. Pyjoules: Python-based energy measurement library for various domains including nvidia gpus. https://github.com/powerapi-ng/pyJoules, 2024. Accessed: 2024-01-10.
[19] Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference, 2023.
[20] Špeťko, M., Vysocký, O., Jansík, B., and Říha, L. Dgx-a100 face to face dgx-2—performance, power and thermal behavior evaluation. *Energies 14*, 2 (2021).
[21] Touvron, H., Martin, L., Stone, K., and et al. Llama 2: Open foundation and fine-tuned chat models, 2023.
[22] Wu, C.-J., Raghavendra, R., Gupta, U., and et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems 4* (2022), 795–813.