Hybrid Heterogeneous Clusters Can Lower the Energy Consumption of LLM Inference Workloads

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

Srinivasan Keshav
sk818@cam.ac.uk
University of Cambridge
Cambridge, UK

Richard Mortier
rmm1002@cam.ac.uk
University of Cambridge
Cambridge, UK

ABSTRACT

Both the training and use of Large Language Models (LLMs) require large amounts of energy. Their increasing popularity, therefore, raises critical concerns regarding the energy efficiency and sustainability of data centers that host them. This paper addresses the challenge of reducing energy consumption in data centers running LLMs. We propose a *hybrid* data center model that uses a cost-based scheduling framework to dynamically allocate LLM tasks across hardware accelerators that differ in their energy efficiencies and computational capabilities. Specifically, our workload-aware strategy determines whether tasks are processed on energy-efficient processors or high-performance GPUs based on the number of input and output tokens in a query. Our analysis of a representative LLM dataset, finds that this hybrid strategy can reduce CPU+GPU energy consumption by 7.5% compared to a workload-unaware baseline.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Hardware** → *Impact on the environment*.

KEYWORDS

sustainable computing, heterogeneous computing, large language models, artificial intelligence

ACM Reference Format:
Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Hybrid Heterogeneous Clusters Can Lower the Energy Consumption of LLM Inference Workloads. In *The 15th ACM International Conference on Future and Sustainable Energy Systems (E-Energy '24), June 04–07, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3632775.3662830

1 INTRODUCTION

Large Language Models (LLMs) such as OpenAI's GPT-4 [24] and Google's PaLM [4] have become emblematic of the AI revolution, driving significant advancements not only in natural language understanding, generation, and translation but also in summarizing and contextualizing large volumes of textual data. Characterized by their extensive scale and depth, their deployment demands substantial computational resources and hence poses significant challenges in terms of energy consumption and operational efficiency [38]. The increasing application of LLMs across diverse sectors further compounds these challenges, because datacenters, which are responsible for a considerable portion of global electricity consumption, must balance performance targets for LLM tasks running on heterogeneous hardware with the need for energy efficiency [7, 21]. Increasing the energy efficiency of LLMs thus emerges as both a technical challenge and an environmental imperative [22].

Traditional data center designs often struggle to best exploit the capabilities of heterogeneous hardware-based LLMs, particularly when trying to minimize energy consumption without sacrificing output quality and latency [6]. However, this challenge also presents an opportunity to innovate in datacenter architecture and management. We show that by rethinking how GPU resources are allocated and managed, there is potential to significantly reduce the energy footprint of LLM deployments while maintaining or even enhancing computational performance.

We find that a dynamic task-scheduling model that assigns LLM tasks to GPUs based on the resulting energy efficiency can reduce overall energy. Moreover, implementing a workload-aware system for input and output token processing can further reduce energy usage. Thus, a hybrid datacenter task allocation model, which allocates different tasks to different hardware accelerators based on their system demands, can reduce the overall energy consumption of LLM inference compared to a workload-unaware baseline.

Our contributions are as follows:

(1) We analyze the energy consumption and runtime of several 7B-parameter LLMs' across various hardware configurations.
(2) We propose and evaluate a workload-aware scheduler for LLMs that optimizes energy efficiency based on the size of input and output token loads, demonstrating a 7.5% decrease in energy consumption over non-workload-aware baselines.
(3) We release a comprehensive dataset and benchmark suite for evaluating the energy efficiency of LLM inference, enabling researchers and practitioners to assess the impact of their design choices.

Through these contributions, we hope to support more sustainable and cost-effective AI inference deployments.

The remainder of this paper is as follows: Section 2 provides background information on LLM inference and energy consumption in AI systems. Section 3 formulates the problem and introduces our cost function. Section 4 details the methods used for benchmarking LLM inference on diverse systems. Section 5 presents the performance results of LLM inference across multiple hardware configurations. Section 6 proposes and evaluates our energy-optimal hybrid data center design. Finally, Section 7 discusses related works, and Section 8 summarizes the conclusions of the paper.

E-Energy '24, June 04–07, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0480-2/24/06
https://doi.org/10.1145/3632775.3662830

2 BACKGROUND

2.1 Inference Using Large Language Models

Transformer-based neural network architectures have led to impressive gains in the performance of LLMs for language understanding and generation [5]. LLMs such as OpenAI's GPT-4 [24] and Google's Gemini [32] have demonstrated human-level proficiency on many language benchmarks while requiring billions of parameters and massive datasets for training. The inference phase of LLMs involves utilizing a trained model to make predictions based on new, unseen data. Unlike the training phase, which is typically a one-time, compute-intensive process that occurs offline, inference is an ongoing, real-time process that directly impacts end-user experiences [7]. This phase is critical as it represents the point at which AI capabilities become accessible to users.

Inference in LLMs can be computationally expensive due to several factors: **(1) Model Size:** The sheer size of these models, often billions of parameters, necessitates significant computational power to process each query [38]. **(2) Latency Expectations:** Many applications based on LLMs, such as digital assistants, automated writing aids, and real-time translators, require low-latency responses [35]. **(3) Scalability:** The ability to scale inference operations to accommodate varying user demands without degradation in response times is crucial.

2.2 Energy Consumption in AI Systems

Recent reports have found that the computational requirements for state-of-the-art AI entail massive energy consumption and carbon emissions [7, 21, 26, 29, 38]. The energy intensity of AI systems can be broadly divided into the energy required for training versus inference after models are deployed [13]. Training complex models on massive datasets is an energy-intensive process, with estimates finding that training GPT-3 required 1,287 megawatt-hours of energy [26]. LLMs can also have huge emissions depending on deployment scale and hardware efficiency [29]. For example, over a year of use, inference by LLMs on cloud infrastructure can consume over 25× more energy than training a model [7]. Optimizing software and hardware specifically for AI workloads is thus essential [3].

2.3 Heterogeneous Systems for Efficient Computing

Modern systems demonstrate a complex interplay between scale, architecture, workload behavior and efficiency objectives. The architecture of compute nodes can significantly impact the energy efficiency and processing capabilities of large-scale computing systems [18]. Conventional server architectures based on multicore CPUs face energy proportionality and scalability limitations for modern data-intensive workloads [20]. Several researchers have explored heterogeneous server configurations to improve energy efficiency [12, 15, 16, 19]. Distributed solutions can translate to lower energy efficiency, as communication overheads dominate [9]. Still, specialized clusters like NVIDIA's DGX show 4x better performance per watt over conventional servers [30].

3 PROBLEM FORMULATION

To model the operational demands of a hybrid, heterogeneous datacenter hosting LLMs, we define a cost function to reflect the workload distribution across different systems. We define a cost function $U(m, n, s)$ that accounts for both energy consumption and runtime:

$$U(m, n, s) = \lambda E(m, n, s) + (1 - \lambda)R(m, n, s),$$

where m and n denote the number of input and output tokens, respectively. $\lambda \in [0, 1]$ is a tunable parameter that balances the weight of energy efficiency versus speed. $E(m, n, s)$ is the energy consumed by system s to process m input tokens and generate n output tokens, measured in joules. $R(m, n, s)$ is the time required to process these tokens on system s, measured in seconds.

Our objective is to minimize the total cost across all tasks and systems:

$$\min_{\{Q_s\}_{s\in S}} \sum_{s\in S} \sum_{(m,n)\in Q_s} U(m, n, s) \tag{1}$$

$$\text{s.t.} \quad \bigcup_{s\in S} Q_s = Q \tag{2}$$

$$\forall s : Q_s \cap Q_{s'} = \emptyset \text{ for } s \neq s' \tag{3}$$

where S is the set of all systems, Q is the total set of queries, Q_s is the subset of queries assigned to system s.

This model ensures that each query is processed exactly once, optimizing for energy efficiency or quick response times, depending on the operational needs, as parameterized by λ. We note, however, that certain systems may be better suited to specific tasks, based on the workload characteristics, such as the need for rapid response times. Adjustments in λ allow the datacenter to shift its focus between minimizing energy consumption and reducing runtime as operational priorities change.

4 METHODS

Here, we describe the methods and tools we use to benchmark LLM inference. In all cases, we use Huggingface's Accelerate [11] to standardize hardware optimization for inference across all platforms. T his library takes advantage of the available accelerator resources and shards models accordingly to minimize intermediate communication and maximize the distributed capabilities for computation across the devices.

4.1 Model Selection

Our study employs three 7B-parameter, open-source LLMs for their capabilities and ability to run on diverse hardware efficiently: (1) Falcon [2], (2) Llama-2 [33], and (3) Mistral [17]. These models were selected to represent a spectrum of architectures and training corpora. We subject each model to a series of standardized NLP tasks to evaluate their energy consumption during inference.

4.1.1 Falcon. The Falcon (7B) [2] model utilizes multi-query attention, significantly reducing memory requirements and increasing processing speed. The model's training on the bilingual RefinedWeb dataset enhances its applicability across diverse linguistic contexts.

4.1.2 Llama-2. We select Llama-2 (7B) 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.

4.1.3 Mistral. We include Mistral (7B) [17] 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.

4.2 Energy Profiling of Diverse Systems

Depending on the platform, we profile each system's energy consumption during inference using customized setups that capture runtime and energy or power metrics. Here, we describe how we monitor the energy usage of NVIDIA GPUs, Apple Silicon CPU/GPU, Intel CPUs, and AMD CPUs.

4.2.1 NVIDIA GPUs. We use PyJoules [27], a Python-based energy measurement library, to quantify the energy consumption associated with inference on NVIDIA GPUs. PyJoules provides an interface to `NVML` [23], providing a software-defined 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.

4.2.2 Apple Silicon CPU/GPU. No standard energy measurement tools are available for profiling energy and power usage for Apple Silicon through an API like PyJoules or RAPL. Therefore, we employ a daemon-based approach to poll macOS' `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. This command returns the percentage of the CPU power each CPU `top` process uses and the total CPU and GPU power consumption in 200ms intervals. This interval was chosen after testing to find the finest granularity measurement without incurring a significant CPU overhead for the I/O of buffering the large `powermetrics` output into memory.

The energy monitoring is conducted concurrently with the LLM inference. A separate thread is dedicated to running the `powermetrics` command, ensuring real-time data collection. Post-inference, the collected data is processed to extract the recorded power data and then find the energy consumption through integration over the runtime. The GPU energy consumption, $E_{Total,GPU}$, is straightforward to calculate for each recorded power value, $P_{GPU,i}$, at each timestep Δt_i.

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

The CPU power draw data is less clear, as many processes run on the CPU. However, an "energy impact factor" through `powermetrics` allows us to infer how much power our Python inference process uses. Therefore, we calculate the CPU energy, $E_{Total,CPU}$, by multiplying $P_{CPU,i}$ by the "energy impact factor," which we denote as α_i, at each timestep:

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

4.2.3 Intel CPUs. For Intel CPUs, we leverage PyJoules, a Python-based energy measurement library similar to our approach for NVIDIA GPUs. This tool supports RAPL (Running Average Power Limit) interfaces, enabling us to obtain fine-grained energy consumption data [36]. We focus on two primary RAPL domains: Package 0 and Package 1, which correspond to the entire CPU package's energy consumption, including all cores in the package.

PyJoules allows us to capture the energy usage of these domains in real time, enabling us to profile the energy consumption specifically during model inference tasks. To account for base energy consumption unrelated to our inference process, we conduct a pre-analysis phase to measure the CPU's average idle power draw. This idle measurement is then subtracted from the total energy consumption during inference to accurately determine the net energy expenditure attributable to the inference process.

We instrument our code to query the RAPL readings at the start and end of the inference task, calculating the energy consumption as follows:

$$E_{Total,CPU} = \sum_i \Bigg(\Big(P_{Package-0,i} - P_{Package-0,Idle} \Big) + \Big(P_{Package-1,i} - P_{Package-1,Idle} \Big) \Bigg) \Delta t_i,$$

where $P_{Package-0,i}$ and $P_{Package-1,i}$, represent the power draw from Package 0 and Package 1, respectively, and $P_{Package-0,Idle}$ and $P_{Package-1,Idle}$ represent the average idle power draw of the CPU packages, respectively.

4.2.4 AMD CPUs. We adopt a different strategy for AMD CPUs due to the absence of a Python API. Instead, we utilize AMDμProf's `timechart` feature, which provides detailed power draw metrics for every core on the chip at fine-grained intervals. By polling AMDμProf at 100ms intervals, we can capture the power draw of each physical core throughout the model inference process.

To ensure we accurately attribute the energy consumption to our inference task, we monitor the CPU core residency through `psutil`. This information allows us to identify and record the specific cores actively engaged in the inference process at each time step. The total energy consumption for the inference task is then calculated by summing the power usage across all active cores and summing over the product of the power usage and time of inference, as follows:

$$E_{Total,CPU} = \sum_{core} \left(\sum_i P_{core,i} \Delta t_i \right)$$

where $P_{core,i}$ represents the power draw of an individual core at each time step, i.

5 LLM INFERENCE PERFORMANCE ON DIVERSE CLUSTERS

5.1 Hardware and Software Versions

The systems we profile are shown in Table 1. We consider these systems as they demonstrate three prominent CPU manufactures and different generations of GPUs. We utilize PyTorch v2.0.1, Torchvision v0.15.2, Numpy v1.26.0, Huggingface v0.20.2, and Accelerate v0.26.1.

System Name	CPU	GPU(s) per Node	DRAM per Node	VRAM per GPU
Macbook Pro	10-core M1 Pro	14-core M1 Pro	32GB	-
Swing AMD+A100	2×64-core AMD EPYC 7742	8×NVIDIA A100	1TB	40GB
Palmetto Intel+V100	40-Core Intel Xeon 6148G	2×NVIDIA V100	376GB	16GB

Table 1: Our System Configurations

We note that the M1-Pro results only include the Llama-2 (7B) and Mistral (7B) results, as Falcon (7B) generally did not complete tasks in less than two orders of magnitude greater runtime.

5.2 Experimental Strategy

To comprehensively evaluate the performance of different system configurations across various models, we conducted a series of controlled experiments. We systematically varied the number of input and output tokens to measure their effects on runtime and energy consumption under two main experimental conditions. In each experiment we do not allow for key-value caches to be re-used to ensure our testing environment is standardized.

5.2.1 Vary Input Tokens. For the first experimental condition, we executed inference requests with increasing input token sizes, ranging from 8 to 2048 tokens, while maintaining a fixed output token size of 32. This setup allowed us to isolate the impact of input size on the system's performance and energy efficiency.

5.2.2 Vary Output Tokens. In the second set of experiments, we varied the output token limit from 8 to 4096 tokens, keeping the input token size constant at 32. This approach helped us understand how increasing output demands affect the runtime and energy consumption of the systems tested.

5.2.3 Randomization and Stopping Criteria. Each experiment was conducted in a randomized order to mitigate any potential bias introduced by the sequence of tests. To ensure the reliability of our results, we adhered to strict criteria for statistical confidence. Each configuration was tested repeatedly until either of two conditions was met: (1) The measured runtime had to be within 0.5 seconds of the actual mean runtime with 95% confidence. (2) A maximum of 25 trials were conducted for each setting if the first condition could not be met.

5.3 Input Token Analysis

Here, we present the impacts on runtime, energy consumption per token, and throughput for LLMs across different hardware configurations while varying the number of input tokens. We perform these experiments using the suite of systems outlined in Table 1 with the models outlined in Section 4.1. In our experiments on the Palmetto Intel+V100 system, the V100 GPU had an out-of-memory error beyond 1024 output tokens for Falcon (7B).

Our runtime measurements show a significant increase as input tokens grow. As depicted in Figure 1(a), all systems exhibit a nonlinear escalation in runtime with increasing token counts, with the M1-Pro system showing the most significant magnitude. This trend highlights the computational burden imposed by larger input sizes, particularly on smaller systems that are not as well designed to handle extensive workloads.

For all systems, we notice that throughput follows a "roofline model" with increasing input tokens [37]. Figure 1(b) illustrates these dynamics, indicating an increase in throughput for all systems until a certain point where inference becomes bound by compute and not by the overhead of the software, as described by roofline performance models [37].

Energy efficiency varies markedly across different systems. The M1-Pro demonstrates consistently low energy consumption per token, particularly for smaller input sizes, as shown in Figure 1(c). This efficiency reflects the M1-Pro's design optimization for low-power operations. In contrast, the Swing AMD+A100, while capable of handling more significant token inputs more efficiently, consumed more energy per token for small workloads yet became more energy efficient at larger input token sizes, underscoring a trade-off between workload size and energy efficiency.

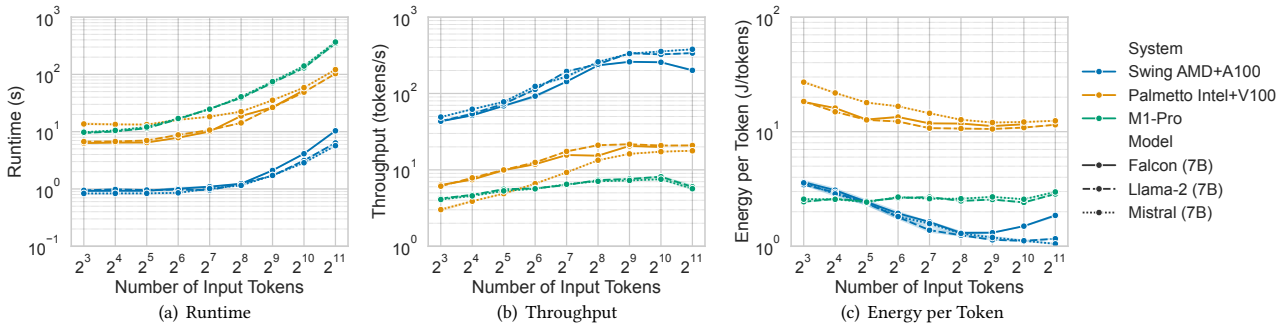
5.4 Output Token Analysis

Here we examine the performance trends associated with increasing the number of output tokens for our LLMs and systems of interest, specifically focusing on runtime, energy consumption per token, and throughput. In our experiments, the M1-Pro also could not generate more than 512 output tokens without significant runtime penalties. For the Palmetto Intel+V100 system, the V100 GPU had an OOM error beyond 1024 output tokens for Falcon (7B) and for all models beyond 2048 tokens.

Runtime significantly increases with the number of output tokens across all systems. As illustrated in Figure 2(a), the escalation in runtime is pronounced, particularly as the output token count reaches higher magnitudes. This increase is indicative of the substantial computational effort required by LLMs to generate successive tokens.

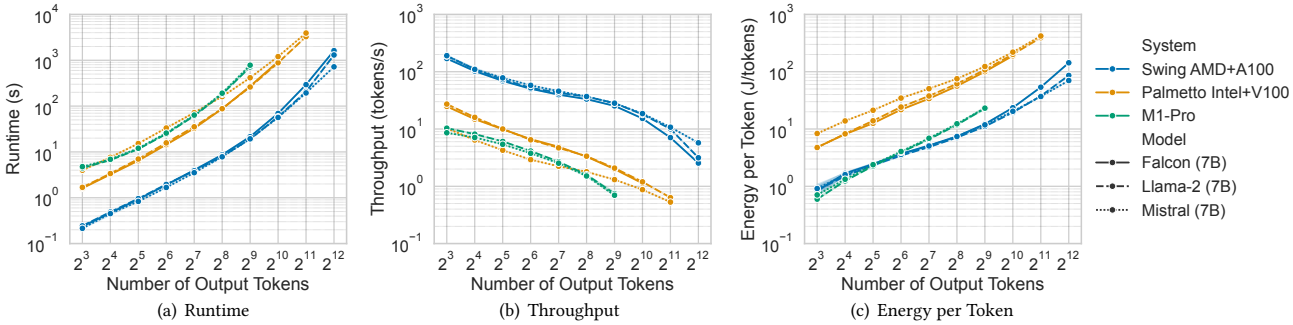
In Figure 2(b), we observe a decrease in throughput across all systems as the number of output tokens increases. This trend highlights the inherent computational complexity involved in generating larger sequences of tokens in LLM tasks. As the output token count grows, the system must process each additional token, recalculating the context and updating internal model states [34]. This not only increases the total computation per query but also leads to a greater accumulation of processing time per token, which consequently lowers the overall throughput.

Energy consumption per token also shows an increasing trend as the number of output tokens grows. Displayed in Figure 2(c), this trend underscores the energy-intensive nature of producing larger outputs. Systems such as the M1-Pro, while generally more energy-efficient, begin to consume more energy per token as output demands increase, reflecting the intensive processing involved in output generation.



(a) Runtime (b) Throughput (c) Energy per Token

Figure 1: Performance of Various Systems and Models for Processing Variable Input Tokens–Due to the low variance in the data, error bars are too small to be visible.



(a) Runtime (b) Throughput (c) Energy per Token

Figure 2: Performance of Various Systems and Models for Processing Variable Output Tokens–Missing data points in M1-Pro and Palmetto Intel+V100 are due to CUDA out of memory errors. Due to the low variance in the data, error bars are too small to be visible.

5.5 Comparing the Input and Output Analyses

When comparing Figure 1(a) and Figure 2(a), we observe that increases in the number of output tokens result in a more considerable increase in runtime than increases in input tokens. The computational complexity of processing input tokens primarily involves encoding the input context, which occurs once per input sequence and follows a more linear computational trajectory. In contrast, generating output tokens is inherently more complex and iterative. Each new output token requires the model to run through all its layers to predict the next token based on an ever-expanding context, which includes both the initial input and all previously generated tokens [34]. This ongoing computation involves recalculating attention across an increasing number of tokens, updating hidden states, and generating a probability distribution over the vocabulary for each new token. Consequently, as the number of output tokens grows, the computational load increases significantly, leading to more significant runtime increases than processing input tokens.

The impacts on runtime also translate to the throughput, depicted in Figure 1(b) and Figure 2(b). There is a noticeable decline in throughput as output tokens increase, more so than input tokens. The decrease in throughput for output tokens is primarily due to the heightened computational requirements for generating subsequent tokens, where each token's generation slows down as the sequence lengthens. Furthermore, the energy per token also increases as output tokens grow, as shown in our analysis. The energy required to generate each output token becomes significant due to longer passes through the transformer network. We contrast this with the energy consumption when processing input tokens, which, despite increasing, does so at a less steep rate.

6 ENERGY-OPTIMAL HYBRID DATACENTER FOR LLM INFERENCE

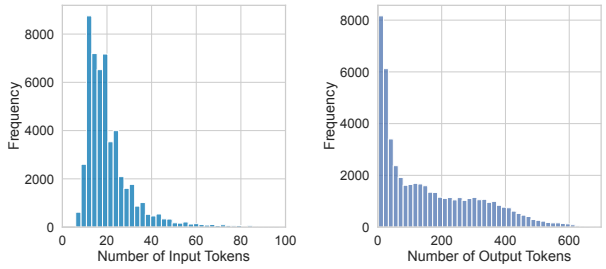
Considering the performance results we collect from LLM inference across multiple systems, we notice that there is an energy-optimal way to construct a hybrid datacenter with a combination of M1 Pro's and A100s. The intuition behind this is that the energy expended per token for the M1 Pro is lower than that of the A100 up to a certain point in the number of input and output tokens as seen in Figures 1(c) and 2(c). However, the energy efficiency characteristics are different when varying the number of input and output tokens, and therefore, we will proceed with separate analyses.

6.1 Number of Input Tokens Analysis

Suppose we have a hybrid data center with M1-Pros and A100s. Then, we have some workload for an LLM, a set of queries with some outputs. In such a configuration, we implement a scheduling heuristic based on a cutoff threshold, T_{in}, for input token length.

This heuristic dictates that queries with $n \leq T_{in}$ tokens are processed on M1 Pro systems, which we have shown have good energy efficiency with handling smaller computational loads. Conversely, queries with $n > T_{in}$ tokens leverage the greater computational ability of A100 GPUs, which offer greater energy-per-token advantages for larger tasks despite their higher power usage. We point out that this is the same method mentioned in the problem formulation in Eqn. 1, where our queries Q are partitioned into Q_{M1} and Q_{A100} strictly on input and output size.

To find an optimal threshold T_{in} empirically, we analyze the token distribution in prompts from the Alpaca [31] dataset, a benchmark dataset frequently used in model fine-tuning. This dataset comprises 52K prompts, offering a diverse range of lengths akin to a typical workload in systems like GPT-4 [24]. The distribution of input tokens, visualized in our analysis (see Fig. 3(a)), serves as a proxy for understanding the variegated nature of LLM workloads.



(a) Input Tokens (b) Output Tokens

Figure 3: Distribution of Token Counts for Alpaca [31]

The energy component of our cost function, split over the token threshold, is as follows:

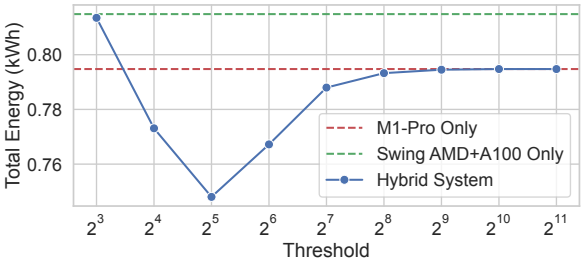
$$E_{Total,in} = \sum_{m=1}^{T_{in}} m f_{in}(m) E_{M1,in}(m) + \sum_{m=T_{in}+1}^{M} m f_{in}(m) E_{A100,in}(m),$$

where $E_{Total,in}$ represents the total energy consumption for a given dataset of input lengths m with corresponding frequencies $f_i n(m)$, and $E_{M1,in}(m)$ and $E_{A100,in}(m)$ denote the mean energy per token for varying the input token size for the M1-Pro and A100 systems, respectively. Utilizing this model with our dataset enables the approximation of total energy consumption for various threshold settings, offering insights into the energy dynamics of hybrid datacenter operation. In Figure 4, we show the energy and runtime simulation results of performing inference for the input token sizes from the Alpaca dataset.

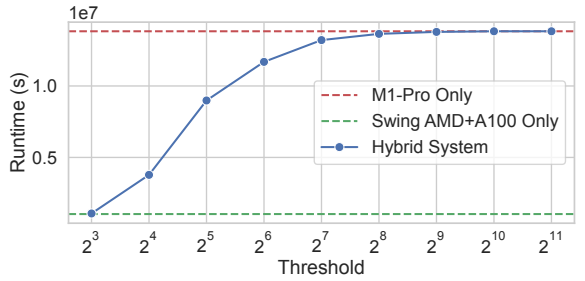
Our findings indicate that a threshold of 32 tokens strikes an optimal balance, significantly reducing energy consumption by relegating the inference of shorter queries to the more energy-efficient M1 Pro systems. This policy not only capitalizes on the inherent energy efficiency of the M1 Pro for smaller tasks but also reserves the computational might of the A100 for queries that necessitate its robust capabilities. However, it's important to note that this energy optimization comes at the cost of increased runtime.

6.2 Number of Output Tokens Analysis

We want to use the same scheduling heuristic and performance model to determine a threshold T_{out} for the number of output



(a) Energy Consumption for Changing T_{in}



(b) Runtime for Changing T_{in}

Figure 4: Performance of Hybrid Datacenter for Input Tokens Processing Alpaca–Dashed line shows the value for using only one kind of hardware for inference

tokens. Except this time, we have different frequencies $f_{out}(n)$ for the n output tokens and different mean energy per token for varying the output token size, $E_{M1,out}(n)$ and $E_{A100,out}(n)$. We also utilize the distribution of the number of output tokens in the Alpaca dataset (see Fig. 3(b)). We revise our performance model as follows:

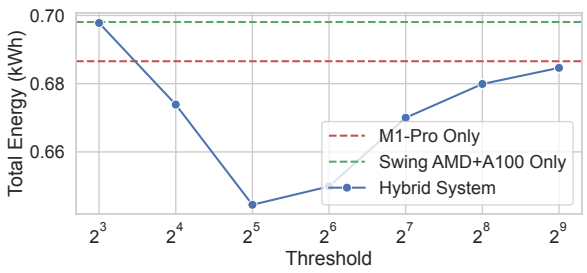
$$E_{Total,out} = \sum_{n=1}^{T_{out}} n f_{out}(n) E_{M1,out}(n) + \sum_{n=T_{out}+1}^{N} n f_{out}(n) E_{A100,out}(n).$$

As the M1 Pro could only generate up to 512 tokens of a response, we only test T_{out} up until this point. In Figure 5, we show the energy and runtime simulation results of performing inference for the input token sizes from the Alpaca dataset.

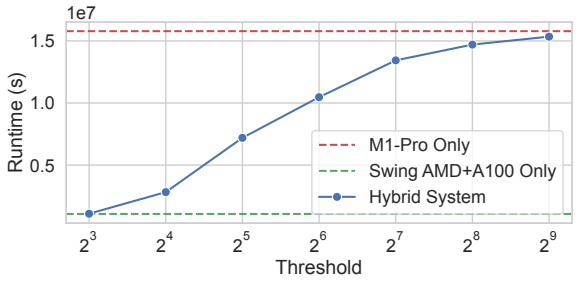
Fig. 5(b) and Fig. 2(c) assess the energy consumption and runtime implications of various threshold settings for output generation. Our findings suggest that although higher thresholds may leverage the M1 Pro's energy efficiency for smaller outputs, there is an optimal point at 32 output tokens that minimizes energy consumption.

6.3 Balancing Energy Efficiency and Runtime Performance

Our analysis of both input and output token processing within a hybrid, heterogeneous datacenter framework has led to the identification that with certain thresholds at $T_{input} = 32$ and $T_{output} = 32$,



(a) Energy Consumption for Changing T_{out}



(b) Runtime for Changing T_{out}

Figure 5: Performance of Hybrid Datacenter for Output Tokens Processing Alpaca – Dashed line shows the value for using only one kind of hardware for inference

we can strategically allocate tasks to M1 Pro systems or A100 GPUs based on token count, optimizing for energy efficiency.

Shifting the token distribution leverages the M1 Pro's superior energy efficiency for input and output tasks up to the threshold, beyond which we utilize the A100's computational power. This policy saves energy as smaller-token tasks are handled by the more efficient M1 Pro for outputs up to the threshold. However, this energy optimization comes at the expense of increased runtime, which is particularly noticeable in output token generation where the M1 Pro, despite its efficiency, does not match the A100's speed.

The energy-runtime trade-off presents a favorable scenario for applications that have low runtime sensitivity. For instance, batch processing of LLM tasks, such as overnight data analyses or non-time-critical computations, can benefit significantly from this energy-efficient configuration. Similarly, free or not directly monetized services, where the cost of computation impacts operational sustainability, stand to gain from minimizing energy expenditures even at the cost of longer processing times.

This approach also opens discussions on Quality of Service (QoS) for LLMs, an area that still needs to be explored [1, 35]. Traditional QoS metrics often prioritize speed and reliability, but energy efficiency may also become a critical QoS dimension for LLM applications, particularly in energy-constrained or cost-sensitive scenarios.

7 RELATED WORK

7.1 Hybrid and Energy Efficient Heterogeneous Data Centers

Recent studies in optimizing data center architectures for deep learning have highlighted the necessity of energy-efficient scheduling and task allocation across diverse hardware. Gu et al. [10] explore GPU clusters' energy-efficient scheduling, revealing substantial improvements in power utilization without considering diverse GPU types for different task requirements. This work highlights a gap in understanding how various GPU configurations could enhance energy efficiency further. Similarly, Patel et al. [25] demonstrate the benefits of hybrid computing environments, emphasizing FPGA over GPU diversity. This focus leaves room to explore the specific impacts of different GPU classes in such settings.

In the realm of LLMs, Zhao et al. [39] introduce strategies like phase-aware partitioning and adaptive quantization in heterogeneous clusters but do not integrate energy considerations into their analysis, which is crucial for understanding the real-world applicability of these models in power-sensitive environments. On the other hand, Radovanović et al. [28] and Chien et al. [7] discuss broader aspects of carbon-aware computing and reducing the carbon impact of AI inference, respectively. These works emphasize the importance of node/device-level energy metrics, often overlooked in typical LLM deployment strategies, thus underscoring the need for detailed energy consumption profiling across different models and hardware types.

7.2 LLM Inference as a Service

Further focusing on energy consumption, Hu et al. [14] analyze deep learning workloads in GPU datacenters, offering insights into energy conservation strategies through workload scheduling. This research aligns with our objectives by confirming the critical role of scheduling in reducing energy footprints. Anderson et al. [3] propose carbon-aware datacenter software that could complement physical hardware adjustments by making energy and carbon metrics visible to application developers, encouraging more energy-efficient coding practices.

Addressing service quality, Wang et al. [35] study the efficiency and reliability of LLM serving, highlighting the challenges of maintaining high-quality service while managing computational loads effectively. This perspective is pertinent as it underscores the trade-off between performance and energy efficiency, which is central to our study. Lastly, Desislavov et al. [8] provide a timely examination of trends in AI inference energy consumption, arguing that while performance has increased dramatically, energy consumption has not escalated at the same pace, thanks to hardware optimizations and algorithmic innovations. This outlook is necessary as it suggests the potential for further optimizations in LLM inference tasks, which are typically energy-intensive.

8 CONCLUSIONS AND FUTURE WORK

Future work will explore minimizing the energy and runtime and maximizing the accuracy of serving differently-sized LLMs. Larger models are generally more accurate but come at the expense of requiring more hardware accelerators and often greater runtime; therefore, exploring this trade-off is highly relevant. Also, we plan to make our solution for energy-optimal routing of incoming queries an online decision-making heuristic to increase its efficacy. Similarly, we aim to extend our energy model to reflect carbon awareness and water consumption to decrease the environmental impact of LLM inference further.

By carefully analyzing the energy and runtime of heterogeneous compute hardware to host LLMs, we show that a hybrid, heterogeneous datacenter and a cost-based scheduling framework can allocate LLM tasks to accelerators that are best suited to run them in terms of energy efficiency and computational performance. This decision is based simply on the size of input and output tokens, making the decision process easy to integrate into existing workloads.

ACKNOWLEDGMENTS

We gratefully acknowledge the computing resources provided on Swing and Palmetto, both high-performance computing clusters operated by the Laboratory Computing Resource Center at Argonne National Laboratory and Clemson University, respectively. During this work GW was supported by a Churchill Scholarship.

REFERENCES

[1] Megha Agarwal, Asfandyar Qureshi, Linden Li Nikhil Sardana, Julian Quevedo, and Daya Khudia. 2023. LLM Inference Performance Engineering: Best Practices. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices

[2] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, et al. 2023. The Falcon Series of Language Models: Towards Open Frontier Models. (2023).

[3] Thomas Anderson, Adam Belay, Mosharaf Chowdhury, Asaf Cidon, and Irene Zhang. 2023. Treehouse: A Case For Carbon-Aware Datacenter Software. *SIGENERGY Energy Inform. Rev.* 3, 3 (oct 2023), 64–70. https://doi.org/10.1145/3630614.3630626

[4] Rohan Anil, Andrew M. Dai, Orhan Firat, et al. 2023. PaLM 2 Technical Report. arXiv:2305.10403 [cs.CL]

[5] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, and et al. 2022. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258 [cs.LG]

[6] Le Chen, Nesreen K. Ahmed, Akash Dutta, et al. 2024. The Landscape and Challenges of HPC Research and LLMs. arXiv:2402.02018 [cs.LG]

[7] Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. 2023. Reducing the Carbon Impact of Generative AI Inference (Today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 11, 7 pages. https://doi.org/10.1145/3604930.3605705

[8] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. 2023. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems* 38 (2023), 100857. https://doi.org/10.1016/j.suscom.2023.100857

[9] Yiannis Georgiou, David Glesser, and Denis Trystram. 2015. Adaptive Resource and Job Management for Limited Power Consumption. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW '15)*. IEEE Computer Society, USA, 863–870. https://doi.org/10.1109/IPDPSW.2015.118

[10] Diandian Gu, Xintong Xie, Gang Huang, Xin Jin, and Xuanzhe Liu. 2023. Energy-Efficient GPU Clusters Scheduling for Deep Learning. arXiv:2304.06381 [cs.DC]

[11] Sylvain Gugger, Lysandre Debut, Thomas Wolf, et al. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

[12] Yuxiong He and Sameh Elnikety. 2011. Position paper: embracing heterogeneity-improving energy efficiency for interactive services. In *Proceedings of the 8th AAAI Conference on AI for Data Center Management and Cloud Computing (AAAIWS'11-08)*. AAAI Press, New York, NY, 11–14.

[13] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *J. Mach. Learn. Res.* 21, 1, Article 248 (jan 2020), 43 pages.

[14] Qinghao Hu, Peng Sun, Shengen Yan, Yonggang Wen, and Tianwei Zhang. 2021. Characterization and prediction of deep learning workloads in large-scale GPU datacenters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (<conf-loc>, <city>St. Louis</city>, <state>Missouri</state>, </conf-loc>) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 104, 15 pages. https://doi.org/10.1145/3458817.3476223

[15] Xiaoxuan Hu, Peng Li, and Yanfei Sun. 2021. Minimizing energy cost for green data center by exploring heterogeneous energy resource. *Journal of Modern Power Systems and Clean Energy* 9, 1 (2021), 148–159.

[16] Mehboob Hussain, Lian-Fu Wei, Abdullah Lakhan, Samad Wali, Soragga Ali, and Abid Hussain. 2021. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems* 30 (2021), 100517.

[17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and et al. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

[18] Willis Lang, Jignesh M. Patel, and Srinath Shankar. 2010. Wimpy node clusters: what about non-wimpy workloads?. In *Proceedings of the Sixth International Workshop on Data Management on New Hardware* (Indianapolis, Indiana) *(DaMoN '10)*. Association for Computing Machinery, New York, NY, USA, 47–55. https://doi.org/10.1145/1869389.1869396

[19] Wenyu Liu, Yuejun Yan, Yimeng Sun, Hongju Mao, Ming Cheng, Peng Wang, and Zhaohao Ding. 2023. Online job scheduling scheme for low-carbon data center operation: An information and energy nexus perspective. *Applied Energy* 338 (2023), 120918.

[20] David Lo, Liqun Cheng, Rama Govindaraju, et al. 2015. Heracles: Improving resource efficiency at scale. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, 450–462. https://doi.org/10.1145/2749469.2749475

[21] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2022. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. arXiv:2211.02001 [cs.LG]

[22] David Mytton and Masaō Ashtine. 2022. Sources of data center energy estimates: A comprehensive review. *Joule* 6, 9 (2022), 2032–2056. https://doi.org/10.1016/j.joule.2022.07.011

[23] NVIDIA. Accessed 2024. NVIDIA-NVML. https://docs.nvidia.com/deploy/nvml-api/index.html. Available online.

[24] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, et al. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[25] Pratyush Patel, Katie Lim, Kushal Jhunjhunwalla, Ashlie Martinez, Max Demoulin, Jacob Nelson, Irene Zhang, and Thomas Anderson. 2023. Hybrid Computing for Interactive Datacenter Applications. arXiv:2304.04488 [cs.DC]

[26] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs.LG]

[27] powerapi ng. 2024. PyJoules: Python-based energy measurement library for various domains including NVIDIA GPUs. https://github.com/powerapi-ng/pyJoules. Accessed: 2024-01-10.

[28] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. 2022. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems* 38, 2 (2022), 1270–1280.

[29] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. arXiv:2310.03003 [cs.CL]

[30] Matej Špeťko, Ondřej Vysocký, Branislav Jansík, and Lubomír Říha. 2021. DGX-A100 Face to Face DGX-2—Performance, Power and Thermal Behavior Evaluation. *Energies* 14, 2 (2021). https://doi.org/10.3390/en14020376

[31] R. Taori, I. Gulrajani, T. Zhang, and et al. 2024. Stanford alpaca: An instruction following llama model. https://github.com/tatsu-lab/stanford_alpaca. Accessed: 2024-01-15.

[32] Google Gemini Team. 2024. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL]

[33] Hugo Touvron, Louis Martin, Kevin Stone, and et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[35] Yuxin Wang, Yuhan Chen, Zeyu Li, Zhenheng Tang, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, and Xiaowen Chu. 2024. Towards Efficient and Reliable LLM Serving: A Real-World Workload Study. arXiv:2401.17644 [cs.DC]

[36] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. 2012. Measuring Energy and Power with PAPI. In *2012 41st International Conference on Parallel Processing Workshops*. 262–268. https://doi.org/10.1109/ICPPW.2012.39

[37] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (apr 2009), 65–76. https://doi.org/10.1145/1498765.1498785

[38] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, and et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

[39] Juntao Zhao, Borui Wan, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. LLM-PQ: Serving LLM on Heterogeneous Clusters with Phase-Aware Partition and Adaptive Quantization. *arXiv preprint arXiv:2403.01136* (2024).