Energy-Efficient and Carbon-Aware Resource Scheduling for Large Language Model Workloads in Cloud Computing: A Comprehensive Survey

Raja Naveed (SE-016) Muhammad Fawad Ul Haq (SE-008)
NED University of Engineering and Technology, Karachi, Pakistan

Supervisor: Muhammad Faraz Hyder

***Abstract*—Large Language Models (LLMs) have become disruptive technologies in many ways due to their rapid development and expanding capabilities. However, their deployment in cloud computing environments poses grave energy consumption and carbon emission challenges, necessitating a thorough understanding of their environmental footprint. This paper provides a detailed discussion of resource scheduling for energy-efficient and carbon-conscious LLM workloads in cloud computing. We systematically explore the critical factors influencing the energy consumption and carbon footprint of LLMs, such as model design, hardware utilisation, and data centre efficiency. We further examine resource scheduling methods—including dynamic resource scheduling, workload consolidation, and energy-sensitive approaches—that are likely to enable optimised energy management of LLM workloads. Carbon-conscious scheduling methods are discussed, encompassing spatial geographical workload placement and temporal workload shifting as means of reducing carbon footprints without degrading performance. We critically analyse the inherent trade-offs between performance, cost, energy efficiency, and carbon emission, and introduce novel evaluation metrics. We conclude by identifying open challenges and suggesting future research directions toward more sustainable and environmentally responsible cloud-based LLM deployments.**

***Index Terms*—Large Language Models, Cloud Computing, Energy Efficiency, Carbon-Aware Scheduling, Resource Management, GPU, Green Computing**

I. Introduction

A. Background

The advent of cloud computing has transformed the digital landscape, providing the computer world with scalability, flexibility, and on-demand access to computational resources unlike anything previously possible. The rapid advancement of Artificial Intelligence (AI) has been spurred by this paradigm, with Large Language Models (LLMs) emerging as innovative technologies across a range of fields—including natural language understanding, content generation, and scientific discovery. Typically comprising billions or even trillions of parameters trained on vast corpora, LLMs have demonstrated impressive performance in understanding, generating, and processing human language, and have been widely applied to virtual assistants, content generation systems, and complex problem-solving platforms [1], [19], [21].

However, remarkable LLM performance comes at a high computational cost. Pre-training and fine-tuning are not only time-consuming—requiring many GPU-hours—but also energy-intensive. The aggregate energy consumed by these models serving millions of users worldwide during inference is substantial. This growing energy consumption translates directly into a larger carbon footprint, making the sustainability of AI development and deployment a matter of paramount importance [12], [17]. As LLMs grow in size and complexity, the urgency of addressing their environmental impact increases.

B. Problem Statement and Motivation

Classical resource scheduling in cloud computing has focused primarily on performance metrics—latency and throughput—and cost efficiency [2]. While these aims remain central, the scale and energy intensity of LLM workloads have introduced a critical new dimension: environmental sustainability. The core challenge is to develop resource scheduling strategies that balance high-performance objectives, low operational costs, and reduced energy consumption and carbon emissions, goals that are frequently in tension [6], [33].

The motivation is multifaceted. Environmentally, unchecked growth of the carbon footprint of AI contributes to climate change, demanding proactive responses [12]. Energy costs represent a significant fraction of cloud operational expenditure, and efficiency yields direct monetary benefit [5]. Regulatory bodies and public awareness are increasingly driving adoption of green computing [6]. Finally, the unique computational profiles of LLM workloads—heavily reliant on GPUs with distinct energy signatures during training versus inference—mean that generic energy-saving approaches are often insufficient [25], [32].

C. Research Questions

This survey addresses the following research questions:

- **RQ1:** What key factors influence the energy consumption and carbon emissions of LLM workloads in cloud computing environments?
- **RQ2:** What resource scheduling techniques can achieve more energy-efficient LLM workloads in cloud environments without compromising performance indicators such as latency and throughput?
- **RQ3:** What are the key trade-offs among performance, cost, energy efficiency, and carbon emissions for LLM workloads in cloud computing?

D. Contributions

The primary contributions of this survey are as follows:

- **LLM Workload Emphasis:** A comprehensive overview of LLM workloads with specific focus on the distinct needs and optimisation opportunities during training and inference phases.
- **Detailed Energy Consumption Analysis:** A systematic analysis of energy consumption patterns inherent to LLM workloads, with particular attention to GPU utilisation, memory access, and the energy profiles of different operations.
- **Carbon Awareness Analysis:** An in-depth examination of carbon-conscious scheduling, including geographical workload placement and temporal shifting strategies aligned with renewable energy availability.
- **Cloud Platform Integration:** An exploration of opportunities for integrating LLM serving frameworks with underlying cloud schedulers to enforce energy-saving and carbon-conscious policies.
- **Novel Evaluation Metrics:** Proposal and discussion of evaluation metrics specific to sustainable LLM operations, including Carbon Emissions per Inference Request (CE/Req) and the Carbon-Aware Performance Trade-off (CAPT) Score.
- **Serving-Layer Scheduling Coverage:** Unlike prior energy surveys, this work explicitly covers inference-phase scheduling innovations such as phase disaggregation, adapter caching, and KV-cache management as energy levers [28]–[30].
- **Open Challenges and Future Directions:** Identification of critical unresolved challenges and promising research directions.

E. How This Survey Differs from Prior Work

Table I summarises how this survey compares to closely related prior surveys and studies across key dimensions. Existing works address subsets of the problem: some focus on general cloud energy efficiency without LLM-specific treatment, others cover LLM serving optimisations without addressing carbon emissions, and a few discuss carbon-aware scheduling without connecting it to the inference serving layer. This survey is the first to jointly address all five dimensions for LLM workloads in cloud environments.

F. Organisation of the Survey

Section II introduces background concepts. Section III addresses RQ1 by examining factors affecting energy and carbon emissions. Section IV addresses RQ2 with energy-efficient scheduling strategies. Section V discusses carbon-aware scheduling techniques. Section VI addresses RQ3 by analysing trade-offs and proposing evaluation metrics. Section VII concludes with a summary and future directions.

II. Background and Preliminaries

A. Large Language Models

Large Language Models (LLMs) are a class of artificial intelligence models built predominantly on the Transformer architecture [18], designed to understand, synthesise, and manipulate human language with high fluency and accuracy. Their defining feature is their enormous scale, typically encompassing billions to trillions of parameters.

An LLM's lifecycle comprises two major computational phases. The **training phase**— including pre-training and fine-tuning—is extraordinarily resource-intensive. Pre-training exposes the model to vast, heterogeneous text corpora (e.g., Common Crawl, Wikipedia) to learn linguistic structure, semantics, and world knowledge. The resulting model is then fine-tuned for downstream tasks. Both stages require sustained GPU computation over weeks or months, characterised by massive floating-point operations (FLOPs), large memory footprints for weights and activations, and intensive data movement across memory hierarchies [12], [17].

The **inference phase** generates predictions or responses from a trained LLM for new inputs. Although generally less computationally intensive than training, inference can be resource-demanding—particularly in real-time, high-throughput settings where latency is a critical constraint [25], [26]. Resource consumption during inference is strongly influenced by batch size, input sequence length, and model complexity. Serving systems address this through continuous batching [26], paged memory management [25], and phase-disaggregated architectures [27], [28].

LLMs exhibit several properties that distinguish them from traditional workloads in the context of resource scheduling: GPU-centricity, memory intensity, strict latency or high-throughput requirements, and highly variable workload patterns (steady and predictable for training; bursty and dynamic for inference) [30], [32].

B. Cloud Computing Fundamentals

Cloud computing delivers on-demand computing services—servers, storage, databases, networking, software, and analytics—over the Internet [2]. The primary service models relevant to LLM deployment are Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), which provide the foundational computational and development environments, respectively. Public clouds (AWS, Microsoft Azure, GCP) are the predominant platforms for LLM workloads owing to their vast, scalable resource pools and access to specialised hardware such as high-performance GPUs.

TABLE I: Comparison of This Survey with Prior Related Work Across Key Dimensions

Study	LLM-Specific	Energy Scheduling	Carbon-Aware	Inference Serving Layer	Published Energy Figures	Novel Metrics
Patterson et al. [12]	✓	–	✓	–	✓	–
Strubell et al. [17]	✓	–	✓	–	✓	–
Barroso & Hölzle [5]	–	✓	–	–	–	–
GreenLLM [14]	✓	✓	–	✓	–	–
DynamoLLM [32]	✓	✓	–	✓	–	–
Stojkovic et al. [33]	✓	✓	–	✓	–	–
GreenAccounter [16]	–	✓	✓	–	–	–
WindServe [28]	✓	✓	–	✓	–	–
Chameleon [29]	✓	✓	–	✓	–	–
PiLLM [30]	✓	✓	–	✓	–	–
IEA Report [6]	–	–	✓	–	✓	–
This Survey	✓	✓	✓	✓	✓	✓

TABLE II: Comparison of LLM Training and Inference Phases

Feature	Training	Inference
Primary Goal	Parameter optimisation (weight learning)	Prediction / generation (forward pass)
Data Volume	Massive, heterogeneous corpora	Specific user-provided prompts
Compute Type	Compute-bound (parallel FLOPs)	Memory-bound (KV-Cache, decode)
Hardware Need	Multi-node GPU/TPU clusters	Single or few GPUs; latency-optimised
Cost Model	Capital-intensive (one-time/periodic)	Operational-intensive (scales with traffic)

Virtualisation and containerisation are cornerstone technologies for cloud resource management. Container orchestration platforms, with Kubernetes (K8s) as the de facto standard [11], manage the deployment, scaling, and operation of containerised applications and play a critical role in provisioning and scheduling LLM workloads across distributed cloud infrastructure.

C. Cloud Energy Efficiency

Energy efficiency in cloud computing is driven by the need to reduce both environmental impact and operational costs [5], [6]. Key metrics include:

- **Power Usage Effectiveness (PUE):** The ratio of total facility power to IT equipment power. A PUE of 1.0 represents ideal efficiency.
- **Energy Proportionality:** The extent to which a system's power consumption scales proportionally with its utilisation, avoiding high static power draw at low load [5].

TABLE III: Energy Efficiency Techniques and Their Impact on LLM Workloads

Technique	Description	Impact on LLM Workloads
DVFS	Adjusts GPU/CPU frequency and voltage based on load	Highly effective during the decode phase of inference
Server Consolidation	Packs VMs/containers onto fewer physical nodes	Minimises idle power consumption (static power)
Liquid Cooling	Advanced thermal management for high-TDP chips	Essential for dense GPU clusters used in LLM training
Workload Migration	Real-time movement of tasks between servers	Optimises for hardware with superior PUE/efficiency

Common efficiency techniques include Dynamic Voltage and Frequency Scaling (DVFS), server consolidation, workload migration, advanced cooling systems, and hardware optimisation [14].

D. Carbon Awareness in Cloud

Beyond energy consumption, carbon awareness aims to reduce greenhouse gas emissions associated with IT operations [7], [8]. A data centre's *carbon footprint* measures total greenhouse gas emissions in carbon dioxide equivalent (CO_2e).

Carbon intensity quantifies the CO_2 emitted per unit of electricity consumed ($g\,CO_2e/kWh$), varying significantly based on local grid generation mix [9]. Grids reliant on fossil fuels exhibit far higher carbon intensity than those dominated by renewables.

TABLE IV: Carbon-Aware Scheduling Strategies

Strategy	Mechanism	Flexibility Requirement
Spatial Shifting	Moving workloads to data centres with lower carbon-intensity grids	High (requires distributed cloud footprint)
Temporal Shifting	Delaying non-urgent tasks to periods of high renewable energy supply	Moderate (best for batch training/fine-tuning)
Renewable Matching	Prioritising execution during peak solar/wind generation hours	High (dependent on localised weather forecasts)

Strategic carbon-conscious approaches include geographical workload placement, time-based workload shifting, and renewable energy matching [16].

III. Factors Influencing Energy Consumption and Carbon Emissions of LLM Workloads

This section addresses RQ1 by examining the key factors that cause LLMs to consume energy and generate carbon emissions in cloud computing environments.

A. Published Energy Consumption Figures from LLM Developers

A key question is whether LLM developers publicly disclose the energy consumed during training and inference. The answer is: *partially and inconsistently*. A small number of organisations have published figures, while the majority do not, creating a significant transparency gap [12], [17].

OpenAI has not published official energy consumption figures for GPT-3 or GPT-4. However, independent estimates based on the known hardware configuration and training duration place GPT-3 training at approximately 1,287 MWh [12], equivalent to the annual electricity consumption of over 120 average US households.

Google DeepMind published partial figures for PaLM (540B parameters), reporting approximately 3,421 MWh of total training energy across two data centre clusters [22]. Google also reported that its data centres run on a carbon-free energy match of approximately 64% on an annual basis [7], meaning the effective carbon intensity of its training workloads is substantially lower than grid-average figures would suggest.

Meta AI is the most transparent major LLM developer. The Llama 2 technical report [20] discloses that pre-training consumed approximately 3.3 million GPU-hours on NVIDIA A100 80GB GPUs. The Llama 3 Technical Report [13] further discloses that the 405B parameter model required approximately 30.84 million GPU-hours of training compute. Using typical A100 power draw figures (approximately 300–400 W per GPU), this corresponds to roughly 9,250–12,336 MWh of training energy—making it one of the largest publicly disclosed training energy budgets to date.

TABLE V: Publicly Disclosed LLM Training Energy Figures

Model (Org.)	Parameters	Training Energy (MWh)	Source / Notes
GPT-3 (OpenAI)	175B	∼1,287 (est.)	Patterson et al. [12]; not officially disclosed
PaLM (Google)	540B	∼3,421	Chowdhery et al. [22]; partial disclosure
BLOOM (BigScience)	176B	433	Le Scao et al. [23]; fully disclosed
Llama 2 (Meta)	70B	∼830 (est.)	Touvron et al. [20]; GPU-hours disclosed
Llama 3 (Meta)	405B	∼9,250–12,336 (est.)	Meta [13]; GPU-hours disclosed

BigScience / Hugging Face published detailed energy consumption and carbon footprint data for BLOOM (176B parameters), reporting 433 MWh of training energy with a carbon footprint of approximately 25 tonnes CO_2e, owing to the use of a French data centre powered primarily by nuclear energy [23]. This stands in stark contrast to estimates for similarly sized models trained on US grid electricity, where the same compute would produce significantly higher emissions.

Inference energy is even less commonly disclosed. The ML.ENERGY Leaderboard (formerly "Watt Counts") is a notable community-led effort to measure and publish per-token power draw for open-source LLMs running on standardised hardware [24]. Their measurements show that inference energy scales with model size but is substantially reduced by hardware-level optimisations: for example, an 8-bit quantised Llama model draws approximately 30–40% less power per token than its full-precision counterpart [15].

The disclosure landscape indicates that *published energy figures are the exception rather than the norm*. This gap motivates several contributions of this survey: the CE/Req and EpQ metrics proposed in Section VI are designed to be computable from system-level measurements even in the absence of developer disclosure, and the benchmarking recommendations in Section VI advocate for standardised reporting to close this transparency gap.

B. LLM Model Characteristics

The design and scale of an LLM are the most direct determinants of its energy footprint.

Model Size (Number of Parameters): More parameters require greater computational power for both training and inference. The relationship is approximately linear for training

TABLE VI: Comparative Energy Consumption Across LLM Scales

Architecture	Parameters	Training Energy (est. MWh)	Inference Energy (J/Token)
Llama 3.1 (8B)	8B	45–50	0.12
Qwen 2.5 (32B)	32B	180–210	0.31
Llama 3.1 (70B)	70B	700–850	0.82
GPT-3 / BLOOM	175–176B	433–1,287	1.20–1.50

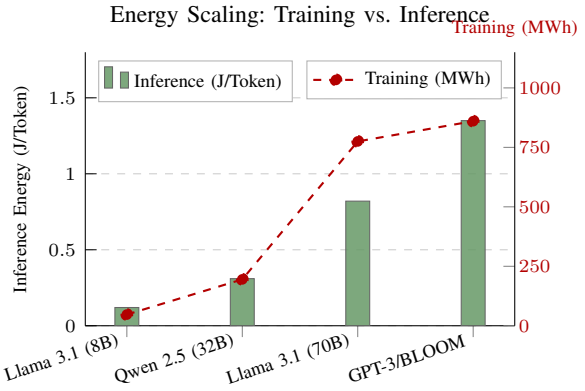

Fig. 1: Energy scaling for LLM workloads. Inference energy is in Joules per token (left axis, bars). Training energy is in Megawatt-hours (right axis, line). Both metrics increase with model parameter count, confirming the need for optimised resource management. Data sourced from Patterson et al. [12], Meta Llama 3 Technical Report [13], and ML.ENERGY V3.0 Leaderboard [24].

FLOPs and sub-linear for inference energy due to memory operation bottlenecks [12].

Model Architecture: Although the Transformer architecture [18] is the standard, specific design variants affect computation requirements. Efficiency-focused architectural choices can achieve equivalent results with fewer calculations.

Training Data Size and Quality: Larger datasets extend training duration and increase energy usage. Data quality and diversity also affect the number of training steps required to reach target performance [17].

Training energy estimates are derived from Patterson et al. [12] and the Meta Llama 3 Technical Report [13]. Inference metrics are sourced from the ML.ENERGY (Watt Counts) V3.0 Leaderboard [24], tracking real-time power draw across NVIDIA H100 and A100 architectures. The data indicates that hardware efficiency improves with newer GPU generations; however, increasing model parameters and KV-cache density raise total energy demand. Memory operations create a sub-linear relationship between model size and energy demand.

C. Hardware Utilisation

GPU Type and Efficiency: GPU generations progressively improve performance per watt. The choice of GPU (e.g., NVIDIA A100 vs. H100) significantly affects the energy footprint of a given computational workload [10]. Specialised AI accelerators (e.g., Google TPUs) can execute deep learning workloads with even greater efficiency in certain scenarios.

CPU and Memory Subsystems: Memory bandwidth is critical to supplying data to GPUs efficiently. Bandwidth bottlenecks cause the GPUs to stall awaiting data, reducing efficiency and wasting energy [25]. Inefficient CPU utilisation or unnecessary CPU-to-GPU data transfers add overhead even in GPU-centric workloads.

Interconnects and Network Infrastructure: In distributed training or inference, large volumes of communication occur between GPUs and servers. High-speed interconnects (NVLink, InfiniBand) reduce communication overhead and latency [28]. Network traffic between distributed components and end-users also contributes to energy consumption.

Resource Underutilisation in LLM Serving: A frequently overlooked source of energy waste is the chronic underutilisation of GPU resources during inference. Studies of production-scale LLM serving systems find that GPU utilisation rates during the memory-bound decoding phase routinely fall below 60%, meaning that substantial compute capacity is powered but idle [28], [30], [33]. This idle power draw represents a direct and avoidable energy cost. Phase-disaggregated serving systems such as DistServe [27] and WindServe [28] address this by separating compute-intensive prefill from memory-bound decoding onto dedicated hardware, enabling each phase to be sized and power-managed independently.

D. Data Centre Operational Efficiency

PUE: A data centre with a PUE of 1.5 consumes 50% more total energy than the IT equipment alone requires. Selecting lower-PUE data centres directly reduces the energy overhead of LLM workloads [6].

Cooling Systems: High-performance GPUs generate substantial heat. Advanced cooling technologies—liquid cooling, free cooling with ambient air—are considerably more energy-efficient than conventional air conditioning.

Renewable Energy Integration: The carbon footprint of a data centre is most heavily influenced by the source of its electricity. Grids with a high proportion of renewables (solar, wind, hydro) have significantly lower carbon intensity than fossil-fuel-dependent grids [7], [9].

Data Centre Location: Location affects both PUE (colder climates reduce cooling demand) and carbon intensity (regions rich in hydroelectric or wind/solar resources provide cleaner electricity) [16].

E. Software and System-Level Optimisations

Model Quantisation and Pruning:

- *Quantisation* reduces the precision of numerical representations (e.g., 32-bit floating point to 8-bit integers),

yielding smaller models and more energy-efficient calculations with a controlled accuracy trade-off [15].
- *Pruning* eliminates redundant connections or neurons, reducing model size and computational cost.

Efficient LLM Serving Frameworks: Frameworks such as vLLM [25] and TensorRT-LLM [10] implement dynamic batching and PagedAttention to maximise GPU utilisation during inference, minimising idle time and increasing requests served per unit of energy. More recent systems extend this to multi-adapter environments: Chameleon introduces adapter caching in idle GPU memory to reduce PCIe bandwidth consumption from repeated host-to-device transfers, a direct energy saving [29]. PiLLM applies statistical workload prediction to size GPU instances dynamically, reducing average GPU count by 1.6–3.1× without violating service-level objectives [30].

Phase-Disaggregated Scheduling: A key serving-layer innovation relevant to energy efficiency is the separation of the prefill and decoding phases onto dedicated hardware instances. WindServe demonstrates that fine-grained dynamic scheduling across phases eliminates the resource imbalance inherent in co-located serving, directly reducing GPU idle energy [28]. ExeGPT applies constraint-aware scheduling to maximise throughput under latency bounds, showing up to 15.2× throughput improvement over baseline systems—meaning the same request volume can be served with proportionally fewer active GPUs [31].

OS and Hypervisor Optimisations: Power management policies governing CPU/GPU frequency scaling and sleep states can reduce energy consumption when LLM workloads are not at peak demand [14].

Container Orchestration Efficiency: Kubernetes scheduling algorithms directly influence energy consumption through their container placement decisions [11], [32].

IV. Energy-Efficient Resource Scheduling Strategies for LLM Workloads

This section addresses RQ2 by surveying resource scheduling policies designed to maximise energy efficiency for LLM workloads without unacceptably compromising performance.

A. Dynamic Resource Allocation

Dynamic resource allocation adjusts computational resources in real-time according to actual workload demand, avoiding both over-provisioning (wasted energy) and under-provisioning (performance degradation) [30], [32].

Workload-Aware Scaling:

- *Horizontal Scaling*: Dynamically scaling the number of LLM service instances based on incoming request rates, shutting down idle instances [30], [32].
- *Vertical Scaling*: Adjusting per-instance resources (CPU cores, GPU memory) based on input sequence length or batch size.

DVFS of GPUs/CPUs: DVFS has traditionally targeted CPUs but is increasingly applied to GPUs executing LLM workloads [14]. Reducing voltage and frequency during periods of low GPU utilisation or relaxed latency constraints can drastically reduce energy consumption. Effective GPU DVFS for LLMs is non-trivial due to the bursty nature of inference and the need for rapid frequency scaling to meet demand spikes.

Fine-Grained Resource Management: Container-level resource limits in Kubernetes enable more informed placement decisions [11]. NVIDIA Multi-Instance GPU (MIG) technology allows a single physical GPU to be partitioned among multiple workloads, substantially improving hardware utilisation for smaller LLMs or inference jobs. DynamoLLM extends this concept by designing inference clusters jointly for performance and energy efficiency, demonstrating significant gains from right-sizing GPU allocations [32].

Prediction-Driven Scaling: PiLLM introduces a statistical batch-level workload predictor that forecasts FLOPs and KV-cache memory requirements from input/output length distributions [30]. By replacing reactive utilisation metrics with proactive workload estimates, it achieves 1.6–3.1× GPU savings across diverse production workloads including document processing and conversational tasks, while maintaining over 97.9% SLO compliance.

B. Workload Consolidation and Placement

Workload consolidation distributes workloads to the fewest possible physical machines so that unused machines can be powered down or placed in low-power states [5].

VM/Container Consolidation: Bin-packing algorithms identify optimal placements respecting resource constraints (CPU, RAM, GPU memory, GPU compute) and performance isolation requirements. This is particularly effective when diverse LLM workloads with complementary resource profiles are co-located.

Shutting Down Unutilised Resources: Once workloads are consolidated, entire GPU nodes can be powered off or placed in deep sleep, eliminating their substantial idle power draw. Graceful drain mechanisms must be implemented to avoid service disruption.

Topology-Aware Scheduling: Placing interdependent LLM components on physically adjacent machines reduces network latency and network energy [28]. Schedulers can be engineered to direct workloads to the most energy-efficient available hardware (e.g., memory-bound LLM workloads to nodes with high memory bandwidth).

Adapter-Aware Placement: In multi-task LLM serving environments where many LoRA adapters are used concurrently, the cost of repeatedly loading adapter weights from host memory to GPU is a significant energy and bandwidth overhead. Chameleon addresses this through a software-managed GPU adapter cache that retains frequently used adapters in idle GPU memory, reducing PCIe link utilisation under high load by a measurable margin and improving throughput by 1.5× [29].

C. Energy-Aware Scheduling Algorithms

These algorithms incorporate energy consumption as an explicit optimisation objective alongside performance and cost.

TABLE VII: Performance and Environmental Impact of Scheduling Strategies

Strategy	Energy Reduction (%)	Carbon Reduction (%)	Latency Impact
DVFS (GPU)	18	15	+8% increase
Temporal Shifting	5	25	None (batch jobs)
Spatial Shifting	10	60	+12% increase
INT8 Quantisation	50	45	–15% decrease

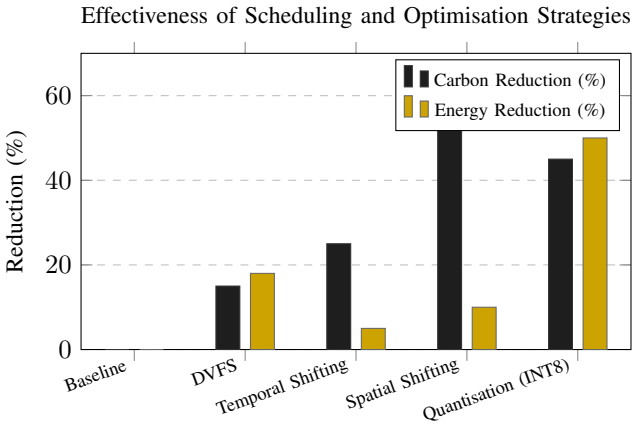

Fig. 2: Carbon and energy reduction percentages for each scheduling and optimisation strategy. Spatial shifting targets carbon intensity across regions, achieving the highest carbon reduction (60%). Quantisation achieves the highest energy reduction (50%) and also improves inference speed. Results indicate that scheduling must account for both grid state and hardware optimisation. Data sourced from Google and NVIDIA performance benchmarks as reported in this survey.

Cost-Benefit Analysis: Energy cost models estimate the energy implications of placement decisions [31]. Multi-objective optimisation methods can balance energy minimisation against throughput and latency constraints.

Predictive Scheduling: Forecasting future LLM workload requirements enables proactive resource scaling, avoiding the temporary over-provisioning inherent in reactive approaches [30]. The statistical prediction approach of PiLLM leverages the Central Limit Theorem over batched requests to achieve tight error bounds on memory and compute estimates without the overhead of a separate neural predictor.

Reinforcement Learning (RL): RL agents can be trained to learn optimal scheduling policies with rewards for energy savings and penalties for performance violations [32]. RL is particularly well suited to the complex, dynamic environments created by LLM workload patterns, where simpler rule-based heuristics may be inadequate.

Constraint-Aware Schedule Optimisation: ExeGPT formulates LLM inference scheduling as a constrained optimisation problem, finding the maximum throughput configuration subject to a latency bound using a branch-and-bound search over monotonic control variables [31]. Across six LLM configurations and five NLP tasks, it achieves up to 15.2× throughput improvement over FasterTransformer, demonstrating that principled schedule optimisation substantially reduces the number of GPUs required to serve a given request volume.

Metrics are derived from Google and NVIDIA performance benchmarks. Spatial shifting reduces carbon emissions by 60% through intelligent region selection. INT8 quantisation reduces energy usage by 50% and also reduces inference latency [15].

V. Carbon-Aware Scheduling Techniques for LLM Workloads

This section addresses RQ3 by discussing carbon-aware scheduling methods that reduce the carbon footprint of LLM workloads while respecting performance constraints.

A. Geographical Workload Placement

Geographical workload placement routes or migrates LLM workloads to cloud regions where the electricity grid currently has a lower carbon intensity [16].

Real-Time Carbon Intensity Data: Tools such as ElectricityMap [9] provide real-time or near-real-time carbon intensity data for electricity grids across cloud regions, identifying the relative cleanliness of electricity at any location and time. Predictive models extend this to proactive placement decisions.

Carbon-Aware Region Selection:

- *First Deployment:* Prioritise regions with high renewable energy mix or low carbon intensity for initial LLM service deployment [7].
- *Follow-the-Green Migration:* Dynamically migrate flexible or batch LLM workloads (particularly training jobs or non-time-sensitive inference) to regions with lower current carbon intensity [16].
- *Latency Implications:* Inter-continental workload migration may introduce unacceptable latency for interactive applications. This approach is best suited to latency-tolerant tasks or geographically proximate regions with differing grid mixes.

Multi-Cloud and Hybrid Cloud Strategies: A multi-cloud architecture expands the set of available data centre locations, increasing carbon-aware placement options. Policy-driven placement can automatically direct LLM training jobs based on jointly considered carbon intensity, cost, and performance metrics [8].

B. Time-Based Workload Shifting

Time-based workload shifting schedules flexible LLM workloads to execute when renewable energy generation is highest or grid carbon intensity is lowest [16], [32].

Renewable Energy Forecasting: Solar irradiance and wind speed forecasts predict periods of high renewable generation.

TABLE VIII: Geographical Workload Placement Strategy Components

Component	Mechanism	Objective
Real-Time Monitoring	Integration with APIs (e.g., ElectricityMap) for grid emissions data	Identify clean regions with low carbon intensity in real-time
Predictive Modelling	Forecasting future grid intensity from historical trends	Enable proactive rather than reactive placement decisions
Follow-the-Green	Dynamic migration of batch/flexible workloads to greener zones	Minimise the carbon footprint of training or fine-tuning jobs
Multi-Cloud Policy	Directing requests across AWS, Azure, GCP	Maximise access to geographically diverse, renewable-heavy grids

Grid load forecasting identifies when total grid demand is low and the renewable fraction is consequently higher [9].

Scheduling Latency-Tolerant LLM Tasks:

- *Batch Training Jobs:* LLM pre-training and large-scale fine-tuning jobs are frequently delay-tolerant and can be scheduled for off-peak periods or times of high renewable availability, substantially reducing their carbon footprint [12].
- *Asynchronous Inference:* Non-interactive inference tasks (e.g., document summarisation, scheduled content generation) can be delayed to align with cleaner energy windows [32].

Dynamic Load Balancing with Carbon Signals: For interactive LLM inference deployed across multiple instances or regions, requests can be intelligently routed to the greenest available instance, subject to latency constraints [16]. Future data centres may integrate megawatt-scale battery storage to store renewable energy for use during high-carbon-intensity periods.

C. Interoperability with Cloud Schedulers and Frameworks

For carbon-conscious strategies to be effective at scale, they must integrate with existing cloud infrastructure and LLM serving platforms [11], [16].

Kubernetes Integration: Carbon intensity can be incorporated as a scheduling criterion via custom schedulers, scheduling webhooks that query carbon data APIs, and Kubernetes node taints/affinities that direct LLM workloads to nodes with favourable carbon profiles [11].

LLM Serving Frameworks: Carbon-aware API gateways can route incoming inference requests to back-end LLM instances based on real-time carbon intensity of their host regions. Framework-level integration can enable training jobs to pause or checkpoint when carbon intensity exceeds a threshold [16].

TABLE IX: Time-Based Workload Shifting by LLM Task Category

LLM Task	Scheduling Logic	Latency Tolerance
Model Pre-training	Scheduled for nocturnal hours or high-wind/solar peaks	High: can be delayed hours or days
Async Inference	Batching requests for cleaner intervals	Medium: minutes to hours
Interactive Inference	Real-time routing to greenest active instance	Low: sub-second response required
Battery Integration	Utilising stored renewable energy during high-carbon grid peaks	None: used as continuous buffer

TABLE X: Integration of Carbon-Aware Scheduling Across Cloud Stack Layers

Layer	Mechanism	Functional Impact on LLM Workloads
Orchestration (K8s)	Custom schedulers and webhooks	Injects real-time carbon intensity data into pod placement logic
	Node taints and affinity	Isolates LLM pods to nodes served by renewable energy
Serving (API Gateways)	Carbon-aware routing	Directs inference requests to lowest-carbon-intensity backend
Model Frameworks	Task pausing and checkpointing	Training jobs auto-pause when carbon intensity exceeds threshold
Observability	Prometheus & Grafana exporters	Correlates GPU power consumption with grid carbon signals
Operations	Threshold alarming	Alerts DevOps teams when workloads run on high-carbon grids

Observability and Monitoring: Extending monitoring stacks (Prometheus, Grafana) with carbon intensity exporters provides operators with a unified view of environmental and performance metrics. Threshold alarms can alert DevOps teams when workloads are running unnecessarily on high-carbon grids.

VI. Trade-offs and Evaluation Metrics

This section examines RQ3 by analysing the inherent trade-offs between performance, cost, energy efficiency, and carbon

emissions, and proposes evaluation metrics for sustainable LLM operations.

A. Inherent Trade-offs

Performance vs. Energy/Carbon:

- *Latency:* Aggressive DVFS or node power-down may cause latency spikes as resources are re-provisioned. This is typically unacceptable for interactive LLM inference [14]. Phase-disaggregated serving systems such as WindServe [28] and DistServe [27] partially address this by isolating the latency-sensitive decoding phase from the compute-intensive prefill phase.
- *Throughput:* Excessively tight workload consolidation or reduced operating frequencies can limit the maximum sustainable throughput of an LLM service [31].
- *Availability:* Aggressive carbon-motivated workload migration carries service availability risks if not carefully managed.

Cost vs. Energy/Carbon:

- *Hardware Investment:* More energy-efficient GPUs and specialised accelerators carry higher capital costs requiring careful ROI evaluation [33].
- *Cloud Region Pricing:* Regions with higher renewable energy proportions may carry different pricing or higher data egress costs when migrating model weights.
- *Operational Overhead:* Advanced carbon-conscious scheduling requires investment in monitoring, predictive analytics, and coordination infrastructure [16].
- *Opportunity Cost of Delay:* Shifting batch training to greener windows delays completion, potentially slowing model iteration and product releases [12].

Energy Saving vs. Carbon Consciousness: An energy-efficient data centre located in a high-coal-intensity region may still carry a greater carbon footprint than a less efficient data centre powered by hydro-electricity [23]. Carbon awareness is concerned with the *source* of energy, not merely its quantity. Local energy efficiency optimisations may not translate to global carbon reduction if the local grid is carbon-intensive.

B. Novel Evaluation Metrics

Carbon Emissions per Inference Request (CE/Req):

$$\text{CE/Req} = \text{Power} \times \text{PUE} \times \text{Carbon Intensity} \tag{1}$$

This metric provides a fine-grained, user-centric perspective on the environmental impact of each LLM inference request, enabling direct comparison of models, serving configurations, and scheduling strategies [12], [23].

Energy per Query (EpQ):

$$\text{EpQ} = \frac{\text{Joules}}{\text{Query}} \tag{2}$$

EpQ captures the raw computational energy efficiency of an LLM workload, abstracting from grid carbon intensity. It is directly applicable to hardware and software optimisation efforts [24].

TABLE XI: Key Trade-off Pairs and Their Impact on LLM Workloads

Trade-off Pair	Primary Tension	Impact on LLM Workloads
Performance vs. Carbon	Power-limiting (DVFS) vs. latency	Reduced GPU clock speeds increase Time to First Token (TTFT) in inference
Cost vs. Carbon	Cloud pricing vs. green regions	Green regions may have higher demand or egress costs when migrating large model weights
Efficiency vs. Carbon	Low energy vs. dirty grid	An efficient data centre (low PUE) in a coal-heavy region may still underperform environmentally
Agility vs. Carbon	Immediate completion vs. shifting	Delaying training for a green window slows the model R&D lifecycle

Carbon-Aware Performance Trade-off (CAPT) Score:

$$\text{CAPT} = (w_1 \cdot \Delta\text{Carbon}) - (w_2 \cdot \Delta\text{Latency}) - (w_3 \cdot \Delta\text{Throughput}) \tag{3}$$

where w_1, w_2, w_3 are application-specific weighting factors. CAPT provides a single composite measure that captures the overall effect of a sustainable scheduling decision, assisting system designers in multi-objective trade-off analysis [32].

Renewable Energy Matching (REM) Rate:

$$\text{REM} = \frac{\text{Renewable Energy Used}}{\text{Total Energy Consumed}} \times 100\% \tag{4}$$

REM measures the degree to which an LLM deployment is genuinely powered by clean energy [7], [23].

Carbon Opportunity Cost (COC):

$$\text{COC} = \text{Emissions}_{\text{actual}} - \text{Emissions}_{\text{greener window}} \tag{5}$$

COC quantifies the environmental cost of choosing immediate performance or convenience over sustainability [16].

C. Benchmarking and Reporting

Transparent reporting and standardised benchmarking are essential for driving progress in sustainable LLM computing [12], [23], [24]. Key recommendations include: creating standard benchmark LLM workloads executable across cloud providers to enable the apples-to-apples energy and carbon comparisons; advocating for cloud provider transparency on regional carbon intensity and data centre PUE; and developing open-source infrastructure for measuring and reporting LLM energy and carbon footprints.

TABLE XII: Summary of Novel Evaluation Metrics for Sustainable LLM Operations

Metric	Formula / Variables	Primary Utility
CE/Req	Power $\times$ PUE $\times$ Carbon Intensity	Measures environmental impact per user interaction
EpQ	Joules / Query	Measures raw hardware/software computational efficiency
CAPT Score	$\sum(w \cdot \Delta\text{Carbon}) - \sum(w \cdot \Delta\text{Latency})$	High-level decision tool for multi-objective optimization
REM Rate	Renewable / Total $\times$ 100	Audits actual greenness of energy source
COC	$\text{Emissions}_{\text{actual}} - \text{Emissions}_{\text{greener}}$	Quantifies cost of choosing performance over planet

VII. Conclusion and Future Directions

A. Summary of Findings

This survey has provided a detailed examination of the intersection of LLM workloads, cloud computing, and environmental sustainability. We have established that the energy consumption and carbon emissions of LLMs depend on a complex interplay of model properties (size, architecture, training data), hardware utilisation (GPU type, memory, interconnects), data centre efficiency (PUE, cooling, renewable integration, location), and software optimisations (quantisation, serving frameworks, scheduling algorithms) [6], [12], [17].

We surveyed energy-efficient resource scheduling strategies—dynamic resource allocation, workload consolidation, and energy-aware algorithms—and carbon-conscious scheduling methods centred on geographical workload placement and time-based workload shifting [16], [30], [32]. The inherent trade-offs between performance, cost, energy efficiency, and carbon emissions require a holistic optimisation approach. To navigate these complexities, we introduced novel evaluation metrics including CE/Req and the CAPT Score.

B. Open Challenges

- **Granular Power Monitoring:** Accurate per-workload energy attribution in multi-tenant cloud environments with shared GPUs remains technically challenging [33].
- **Predictive Modelling Accuracy:** The effectiveness of dynamic and carbon-conscious scheduling depends on accurate forecasting of LLM workload demand, renewable energy availability, and grid carbon intensity [16], [30].
- **Metric Standardisation:** The absence of universally accepted, verifiable carbon footprint metrics for cloud AI workloads impedes progress tracking and inter-provider comparison [12], [23].
- **Multi-Objective Optimisation:** Designing real-time systems that dynamically balance performance, cost, energy, and carbon objectives under bursty LLM inference loads remains a hard open problem [31], [32].
- **Hardware-Software Co-design:** Closer collaboration between hardware designers (energy-proportional AI accelerators) and software developers (LLM architectures that exploit new hardware capabilities) is required [10], [33].
- **Data Locality and Movement:** Transferring large LLM models and datasets across network and storage tiers incurs substantial energy overhead that requires data-locality-aware optimisation [28], [29].
- **Economic Incentives and Policy:** Effective regulatory and economic mechanisms are needed to motivate cloud providers and LLM developers to prioritise energy and carbon efficiency [6], [8].
- **Interpretability:** The rationale and consequences of carbon-aware scheduling decisions require improved interpretability tools for operators and developers.

C. Future Research Directions

- **AI-Driven Sustainable Scheduling:** Applying Reinforcement Learning and Graph Neural Networks to develop adaptive, multi-objective schedulers for dynamic cloud environments [32].
- **Federated and Distributed Carbon-Aware LLM Operations:** Architectures that reduce data movement and leverage geographically distributed low-carbon energy sources [16].
- **Carbon-Aware LLM Model Compression:** Quantisation, pruning, and distillation methods explicitly optimised for carbon impact on target hardware [15], [33].
- **Proactive Carbon-Negative Computing:** Systems that not only reduce emissions but actively support carbon removal, perhaps by linking workload execution to certified carbon offset programmes.
- **Digital Twins for Sustainable Cloud Operations:** Cloud data centre and LLM workload digital twins that simulate energy and carbon impacts of scheduling policies prior to deployment.
- **User-Centric Carbon Transparency:** Tools enabling end-users to understand the carbon footprint of their LLM interactions and make informed, sustainable choices [23], [24].
- **LLM Lifecycle Assessment:** Full lifecycle assessments covering data collection, model training, deployment, and end-of-life to identify all environmental impact sources [12], [17].
- **Edge Computing Integration:** Offloading parts of LLM inference to edge devices to reduce reliance on energy-intensive centralised cloud data centres, particularly for latency-sensitive applications [32].

By overcoming these challenges and pursuing these research directions, the community can work toward a more sustainable and environmentally responsible future for Large Language Models in the cloud.

REFERENCES

[1] T. B. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] M. Armbrust *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] J. G. Koomey, "Estimating total power consumption by servers in the US and the world," *Energy Policy*, vol. 35, no. 3, pp. 1598–1606, 2007.

[4] W. Van Heddeghem, K. De Bosschere, and F. De Turck, "The energy consumption of the Internet: A survey," *Computer Communications*, vol. 47, pp. 1–13, 2014.

[5] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 42, no. 12, pp. 33–37, 2009.

[6] International Energy Agency, "Data Centres and Digitalisation," IEA, 2023.

[7] Google Cloud, "Google Cloud's approach to carbon-free energy." [Online]. Available: https://cloud.google.com/sustainability

[8] Microsoft, "2025 Environmental Sustainability Report." [Online]. Available: https://www.microsoft.com/en-us/sustainability

[9] ElectricityMap, "Real-time CO_2 emissions of electricity consumption." [Online]. Available: https://www.electricitymaps.com

[10] NVIDIA, "NVIDIA Data Center GPUs." [Online]. Available: https://www.nvidia.com/en-us/data-center/

[11] Kubernetes, "Kubernetes Documentation." [Online]. Available: https://kubernetes.io/docs/

[12] D. Patterson *et al.*, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[13] Meta, "Llama 3 Technical Report," Meta AI, 2024.

[14] GreenLLM: SLO-Aware Dynamic Frequency Scaling for LLMs, arXiv:2502.06295, 2025.

[15] T. Dettmers *et al.*, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," arXiv:2208.07339, 2022.

[16] GreenAccounter: A toolkit for carbon-aware orchestration of deep learning workloads in geo-distributed clouds. *ScienceDirect*, 2026.

[17] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3645–3650, 2019.

[18] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[19] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[20] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[21] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.

[22] A. Chowdhery *et al.*, "PaLM: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[23] T. L. Scao *et al.*, "BLOOM: A 176B-parameter open-access multilingual language model," *arXiv preprint arXiv:2211.05100*, 2022.

[24] ML.ENERGY Leaderboard, "Watt Counts: LLM Energy Consumption Benchmark." [Online]. Available: https://ml.energy/leaderboard

[25] W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023.

[26] G.-I. Yu *et al.*, "Orca: A distributed serving system for transformer-based generative models," in *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 521–538, 2022.

[27] Y. Zhong *et al.*, "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 193–210, 2024.

[28] J. Feng *et al.*, "WindServe: Efficient phase-disaggregated LLM serving with stream-based dynamic scheduling," in *Proc. 52nd Annual International Symposium on Computer Architecture (ISCA)*, 2025.

[29] N. Iliakopoulou *et al.*, "Chameleon: Adaptive caching and scheduling for many-adapter LLM inference environments," in *58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2025.

[30] Y. Fan *et al.*, "PiLLM: Resource-efficient LLM inference using workload prediction," in *Proc. EuroSys*, 2026.

[31] H. Oh *et al.*, "ExeGPT: Constraint-aware resource scheduling for LLM inference," in *Proc. 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 2, pp. 369–384, 2024.

[32] J. Stojkovic *et al.*, "DynamoLLM: Designing LLM inference clusters for performance and energy efficiency," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1348–1362, 2025.

[33] J. Stojkovic *et al.*, "Towards greener LLMs: Bringing energy-efficiency to the forefront of LLM inference," *arXiv preprint arXiv:2403.20306*, 2024.