AI Greenferencing: Routing AI Inferencing to Green Modular Data Centers with Heron

Tella Rajashekhar Reddy, Palak, Rohan Gandhi, Anjaly Parayil, Chaojie Zhang, Mike Shepperd,
Liangcheng Yu, Jayashree Mohan, Srinivasan Iyengar, Shivkumar Kalyanaraman, Debopam Bhattacherjee

Microsoft

ABSTRACT

AI power demand is growing unprecedentedly thanks to the high power density of AI compute and the emerging inferencing workload. On the supply side, abundant wind power is waiting for grid access in interconnection queues.

In this light, this paper argues bringing AI workload to modular compute clusters co-located in wind farms. Our deployment right-sizing strategy makes it economically viable to deploy more than 6 million high-end GPUs today that could consume cheap, green power at its source. We built Heron[1], a cross-site software router, that could efficiently leverage the complementarity of power generation across wind farms by routing AI inferencing workload around power drops. Using 1-week of *coding* and *conversation* production traces from Azure and (real) variable wind power traces, we show how Heron improves aggregate goodput of AI compute by up to 80% compared to the state-of-the-art.

1 INTRODUCTION

With the rise of transformers [78] and large language models (LLMs) [36, 54, 55, 58, 59], and the rapid adoption of AI [16, 33, 63, 71] by the industry, governments, and individuals, it is now paramount to satiate the growing energy needs of AI. Reports [51, 66] indicate a GPT query consumes far more energy than a web search, and AI image generation rivals smartphone charging. GPUs running AI workloads are getting significantly more power dense [20, 37, 56, 57] and driving the data center (DC) power demand. BCG [48] projects the DC energy (likewise, power) demand in the US to rise at a CAGR (cumulative annual growth rate) of 15-20% [61] year-on-year, potentially reaching 1,000 TWh (100+ GW) by 2030—two-thirds of the US residential consumption. Similar trends are seen in the EU [53].

Sensing this surge in power demand, the risk of power-supply bottlenecks [32, 47], and mounting pressure to meet sustainability goals, hyperscalers like Microsoft, Google, and others have recently announced partnerships [21, 23, 43, 75] with many renewable[2] energy providers.

Unfortunately, simply adding more renewables to the utility grid may not suffice. In particular, as power demand surges, grids struggle to keep up. First, expanding power

---

[1] Heron of Alexandria [81] invented wind wheel that could power a machine.
[2] We discuss challenges with nuclear energy in §7.

infrastructure—via new transmission lines, distribution systems, or energy storage—is capital-intensive [29, 76], often faces regulatory and logistical delays [32], and is especially difficult when renewable sources are located far from consumption hubs [42, 77]. As a result, a Berkeley Lab report [47] highlights that by 2023, pending grid approvals for new power generation now exceed twice the installed capacity. Second, even approved projects frequently face curtailment due to grid congestion, leaving clean power underutilized [26, 42, 83]. Thirds, long-distance transmission and distribution (T&D) can incur significant power loss, inflating costs by over 50%[9, 29]. For example, the EIA reports [27] US enterprise power costs at 8.5 ¢/KWh (Sep' 24), while wind farms sell at 2.5 ¢/KWh[15, 72]. Finally, much of the grid is aging—approaching or exceeding its 50-year design lifespan [32, 66]—and may be forced to adopt short-term measures [25, 65] that compromise long-term sustainability goals of all stakeholders [80].

Observing the gap, we ask the questions: *Can we offer AI providers a useful lever for addressing power constraints and advancing sustainability goals?* In this light, we propose *AI Greenferencing* that shifts a significant share of AI inferencing to renewable energy farms, with a focus on wind farms due to their growing deployment [5, 7, 8]. While these sites currently support digital television streaming and cryptomining, LLM inferencing (90% of AI compute today [12, 62]) presents a major opportunity. Running AI at the power source eases grid stress, addressing interconnection queues, curtailment, T&D loss, and sustainability. AI Greenferencing is broadly a win-for-all: (1) users gain access to sustainable AI services; (2) AI providers unlock additional compute capacity, user reach, and revenue; (3) wind farms can monetize power locally; (4) power grids benefit from reduced load and more breathing room for expansion; and (5) co-located compute startups [5, 7, 8] gain a high-value workload.

While conceptually simple, realizing AI Greenferencing requires navigating several practical questions:

Enough wind power to make a dent? We find that there already lies a significant wind capacity (only 100+ MW deployments) of 640+ GW (150+ GW) globally (likewise, in the US), as per the Global Energy Monitor data [34], within 50 ms fiber round-trip time (RTT) of Azure DCs. 77% of this capacity is within 20 ms RTT (does not significantly inflate
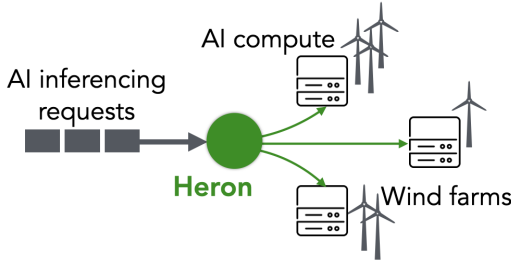
Figure 1: *AI Greenferencing with* Heron.

AI inferencing latency demands with time-to-first-token or TTFT SLOs of up to few seconds) of Azure DCs [1].

AI compute too costly? We show that it is economically feasible to deploy costly GPUs at wind farms with variable power. In particular, there exist multiple modular compute deployment choices [10, 24, 52, 56] which could be mass produced and shipped to wind farms. Our deployment right-sizing strategy recommends deploying AI compute at a lower-than-peak power generation. This approach guarantees peak compute power most of the time with residual uncertainty that is handled by the software router leveraging cross-site complementarity [9, 74] of generation. The hardware deployed on a wind farm incurs significantly lower power cost at the source 100% of the time, which translates to a comparable or lower total cost of ownership (TCO) than in traditional DCs. With this approach, we could deploy 6+ million NVIDIA H100s ($\geq 3\times$ NVIDIA's order book for 2024 [30, 68]) in wind farms 'today.'

Variations predictable? We find that wind power albeit variable is highly predictable at 15-min timescales, thus assisting the scheduler in taking informed decisions on routing online. Similarly, AI inferencing workload arrival is predictable allowing AI Greenferencing software to plan allocation ahead with power constraints.

Heron **for AI Greenferencing.** When co-locating GPUs with wind sites, another challenge is to servce AI inferencing requests under variability of power and workload (§3). To realize this vision, we design Heron (shown in fig. 1), a logically centralized cross-site router that intercepts inference requests near the AI provider's edge or datacenter, and routes them to wind-powered compute clusters. Heron accounts for power availability, hardware constraints, network latency, and workload patterns. It exploits both application-level knobs (e.g., tensor parallelism) and infrastructure-level controls (e.g., frequency scaling) while leveraging spatial complementarity across wind sites. Hierarchically structured for multi-timescale decisions, Heron improves AI compute goodput by up to 80% over a state-of-the-art energy-aware but variability-agnostic baseline scheduler [73].

This paper makes the following contributions:

- We show there is significant wind capacity close to demand.
- We show how right-sizing the AI compute per wind site could largely *mitigate the power uncertainty.*
- We show how low-cost power at its source could offset lost compute cycles due to the residual uncertainty.
- We show that both wind power generation and inferencing workload arrival are variable yet *predictable.*
- We build and demonstrate Heron, a cross-site router for AI Greenferencing that leverages predictions, workload insights, application and infrastructure-level knobs, and wind generation complementarity across sites to *address the residual uncertainty* (post right-sizing) in the system.

2 MOTIVATION & FEASIBILITY ANALYSES

AI compute is becoming increasingly power-dense [20, 37, 57], with datacenter demand in the U.S. alone growing at a CAGR of 10–15% [48], soon rivaling residential consumption. This accelerating demand risks overwhelm aging grid infrastructure and force short-term decisions [25, 65] that may undermine long-term sustainability goals.

Meanwhile, wind farms represent a massive, underutilized power opportunity. As of June 2024, there is 447 GW of operating global wind capacity and over 2.4 TW in total projected capacity (including under-construction, pre-construction, and announced 100+ MW sites) [34]. However, much of this capacity remains idle—stuck in interconnection queues [47] due to the high cost, complexity, and slow pace of grid expansion. Even approved wind farms often face curtailment [26, 42, 83] due to grid congestion. And when wind power is delivered via the grid, transmission and distribution (T&D) losses—often exceeding 50% [9, 29]—inflate costs significantly.

A case for AI greenferencing. This paper argues a disaggregated AI deployment framework across large wind farms that co-locates 'AI' compute clusters with power sources. Rather than bringing power to compute, AI greenferencing framework routes LLM inferencing requests across geographically dispersed sites, adapting to workload characteristics, power availability, generation complementarity, and hardware capacity. This model has the potential to unlock millions of GPUs for a sustainable AI infrastructure.

This section quantifies the opportunity and evaluates key feasibility dimensions: §2.1 assesses wind capacity near existing demand; §2.2 analyzes the cost trade-offs of sacrificing compute cycles for cheaper power; and §2.3 examines how to tame fluctuations in wind power and workload.

2.1 Significant Wind Capacity Reachable

Checking for wind farms within reasonable network distance of DCs (demand proxies) is crucial. AI inference latency service level objectives (SLOs) [73] include Time To First Token (TTFT) and Time Between Tokens (TBT). TTFT ranges from
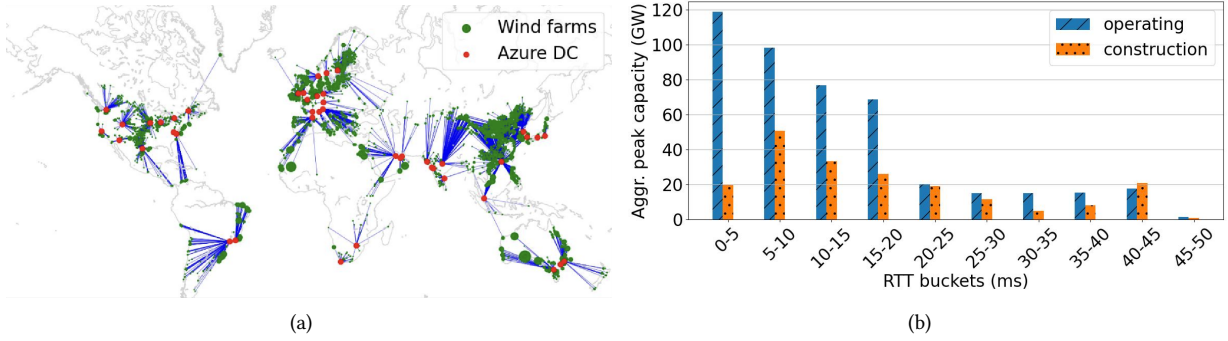
(a)   (b)

Figure 2: *(a) Wind capacity (100+ MW farms only) within a 50 ms (emulated) fiber RTT of Azure locations as of June '24. (b) A significant fraction of this peak capacity lies within 20 ms fiber RTT.*

a few 100 ms to seconds, while TBT is lower[87]. Network latency affects primarily TTFT, not TBT (streaming tokens).

Fig. 2(a) shows wind farms ($\geq$ 100 MW, as of June '24 [34]) within 50 ms RTT (we account for fiber latency inflation [17]) of Azure DCs (public, 2023 [1]). This map represents a hub-and-spoke model where AI requests reach Azure DCs before routing to wind farms. In real deployments, a cross-site router at the Azure network edge could further reduce this extra latency. Fig. 2(b) shows much of this capacity is within 20 ms RTT, inflating TTFT (Table. *IV* in [73]) by only 1-8%. Similar analysis with Google data centers [2] reveals comparable proximity (not shown). Not all capacity in Fig. 2 is available for AI deployment though. Wind power is uncertain, and §2.2 shows how right-sizing compute at wind sites ensures economic feasibility. Note that while operating capacity serves grids, consuming power at the source instead of via the grid should not, in principle, disrupt the power supply ecosystem.

> **Takeaway:** Significant wind power capacity exists near data center hubs (i.e., AI demand) without violating latency SLOs.

2.2 Hardware Provisioning

Hardware deployment choices. Microsoft has been deploying Modular Data Centers (MDCs) [10] for many years. MDCs could be mass-produced and assembled on site, hence incurring lower costs of production [22, 24, 41] than traditional DCs. Also, there are multiple startups [5, 8] that have been deploying and running compute in wind farms (and even within turbines). NVIDIA recently announced Super-PODs [56] that are modular clusters of 1,016 GPUs (like, H100) with integrated cabling, cooling, network, etc. There also exist past research (Parasol [35]) on running compute co-located with renewables. Given these capabilities, we rather focus on a more important question—does it make sense economically to deploy AI compute in wind farms?
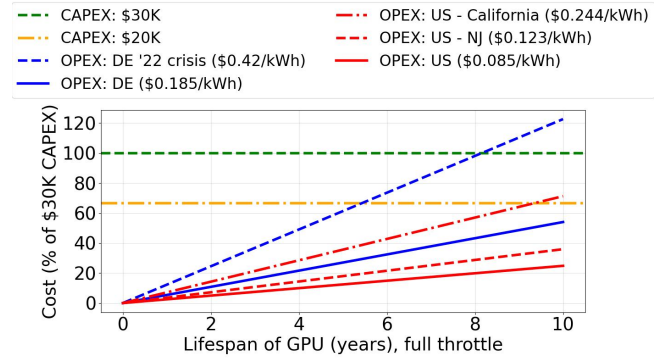

Figure 3: *Cost of power over the lifetime of an NVIDIA H100 GPU could be a significant fraction of its CAPEX.*

A major challenge in deploying GPUs at wind farms is power variability (§2.3). Given high GPU costs [69], we assess if variable but cheap power could justify the CAPEX (capital expenditure) investment. GPUs have a limited lifespan, and underutilization leads to wasted CAPEX. We compare OPEX (operational expenditure) vs. CAPEX to determine optimal provisioning, ensuring lower OPEX offsets lost CAPEX.

OPEX versus CAPEX. The total cost of ownership (TCO) of a DC is primarily driven by 4 components [13]: (1) DC CAPEX (as DC depreciation), (2) DC OPEX, (3) Server CAPEX (as server depreciation), and (4) Server OPEX. MDCs are known [22, 24, 41] to have a low cost of production due to economies of scale – they could be mass produced. Furthermore, when deployed on otherwise unused land in a wind farm, the cost of land acquisition or rent should also be low [40, 60] given the revenue opportunity this deployment brings to wind farms. Hence, we do not anticipate the DC CAPEX to be any higher for AI Greenferencing deployments than for traditional DCs. In addition, MDCs are easier to maintain given their standard design at scale. If wind farm
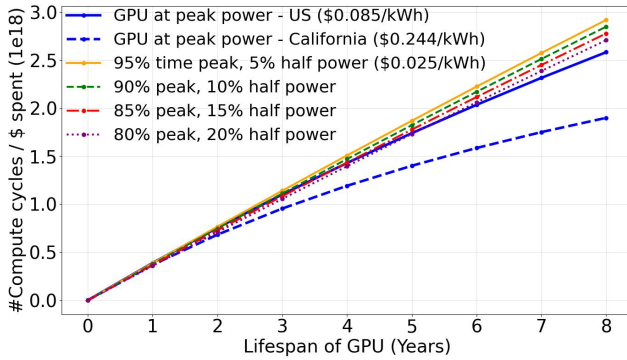
Figure 4: *Comparable C/P at wind sites.*

employees/operators are trained to perform basic DC maintenance activities [44, 86], this could rather bring down the DC OPEX cost than what a traditional DC incurs. Also, compute at wind farms could naturally leverage air-based cooling, effectively reducing reliance on liquid cooling and, hence, reduce OPEX further. Hence, here we focus more on only the server CAPEX and OPEX aspects—the former is primarily driven by GPU prices, and the latter, driven significantly by the cost of power, could be significantly lower with AI Greenferencing style deployments.

Given GPU costs have largely increased in the last few years [69] with NVIDIA H100s priced at around 30K USD per GPU, it could be easy to believe that CAPEX should drive server deployment choices, as the cost of power is traditionally low. Here we show otherwise – over the lifetime of a GPU, the OPEX (cost of power in our analysis) becomes a significant fraction of the TCO. Fig. 3 shows the CAPEX and OPEX components of an NVIDIA H100 GPU normalized by its publicly known price of 30K USD [69] (horizontal line at $y = 100$). A second horizontal line (CAPEX of 20K USD) represents a plausible lower bound of the economies of scale for bulk buyers. All other lines show the total OPEX spent in buying power in the US [27] and in Germany (DE) [18] anticipating peak GPU operations throughout. As we can see, in 5 years, the OPEX in the US is 12.4% (18.6%) assuming 30K USD (likewise, 20K USD) CAPEX. In Germany these numbers are much higher—27% and 40.5% respectively. In special cases, when compared against a 30K USD CAPEX, OPEX could be as high as 35.6% in the US (California), and 61% in Germany (2022 crisis [64]).

Takeaway: OPEX could be a significant fraction of the cost of ownership of even high-end GPUs.

Better C/P than in traditional DC. As the cost of power consumption is a significant fraction of the TCO of AI compute, we argue that it could be a better choice to lose a few GPU cycles during its lifetime if the cost of power is significantly cheaper throughout. Let us see here how a GPU
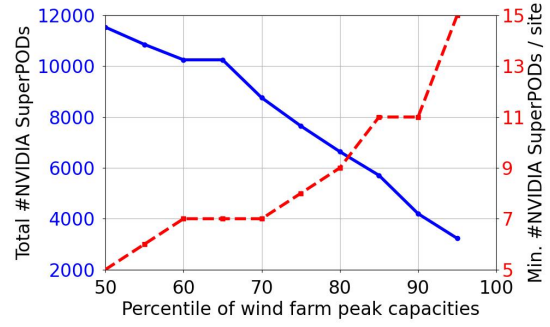

Figure 5: *Deploying less compute (20th ptile of generation) at the larger farms (x-axis) translates to less fragmentation (right-y) but a large number of GPUs in aggregate (left-y).*

deployment in a wind farm could fare against one in a traditional DC in terms of capability/price (C/P = total compute cycles / dollars spent) where the denominator is the sum of CAPEX and OPEX.

Fig. 4 plots C/P for a single GPU priced at 25K USD over time. Let's say that the GPU at its peak could sustain ~10^{22} flops per year [4]. The cost of power for enterprise customers in the US is 0.085 USD/kWh on average, with the cost in California being 0.244 USD/kWh as of Sep'24 (EIA Report [27]). On the other hand, the cost of wind power at the source, as in reported PPAs [15, 72] (power purchase agreements), is significantly lower[3]—0.025 USD/kWh in the major wind power producing regions in the US. This lower cost of power translates to significant OPEX reductions when AI compute is deployed at the source of power, that is the wind farms. Also, note that this lower cost is actually an upper bound for the cost of power at its source than via grid—curtailed power is anyway wasted and is virtually *free*.

Provisioning for peak wind power at a farm wastes compute cycles. Instead, we provision at the Xth percentile of generation, ensuring cheaper power 100% of the time while minimizing under-utilization. Fig. 4 shows that this strategy achieves higher C/P than traditional DCs within years. At the 5th percentile, parity is reached in 2 years; at the 15th & 20th (larger clusters), it takes 4 & 5 years respectively.

Takeaway: Right-sizing the AI compute provisioned in wind farms and consuming low cost power at source could effectively offset lost compute cycles.

High deployment appetite. The key is to provision AI compute at a lower percentile of wind power generation at a site, ensuring comparable or higher C/P than traditional DCs despite occasional lost cycles. ELIA data [28] shows that the Xth percentile power is much lower than X% of peak capacity—e.g., (5th ptile, 1%), (10, 2), (15, 4), (20, 5), etc.

---

[3]As per industry experts, often an order of magnitude cheaper.

This approach translates to a fixed number of GPU clusters, scaling with farm size.

We now evaluate how deployment at the 20th percentile of the peak farm capacity could look like globally, if we only pick the largest (in terms of peak generation capacity) $Y\%$ wind farms from the Global Energy Monitor dataset [34]. For this evaluation, we assume deployments will happen in multiples of NVIDIA's H100 SuperPODs, each hosting 1,016 H100 GPUs and consuming 1.3 MW peak power. As seen in Fig. 5, the aggregate number of NVIDIA SuperPODs that could be deployed at the largest 20% ($x = 80$, 250+ MW sites) of the operating wind farms across the globe is 6,636 – this translates to 6.7 million H100 GPUs (more than 3× the H100 order book of NVIDIA in 2024 [30, 68]). Given these are all large wind farms, even the 20th percentile (5%) of peak capacity in the smallest such farm is a reasonable sized deployment of 9 SuperPODs ($9K+$ H100 GPUs). In general as we pick a smaller percentile of peak power for compute right-sizing, larger the aggregate deployment but smaller the minimum deployment size (more fragmentation). This is a knob that AI inferencing providers should use based on factors like software router scalability, per-site maintenance overhead, physical security concerns, etc.

> **Takeaway:** Despite the right-sizing strategy, an estimated 6+ million H100 GPUs could still be deployed at the larger wind farms today.

2.3 Taming Power & Workload Uncertainty

AI Greenferencing, spanning multiple wind sites, needs to mitigate uncertainties at either end: (1) power generation at the wind sites and (2) inference workload arrival. Below we explore the specific characteristics of wind power and the AI inferencing workload (next) to better understand how predictable these variations are—a characteristic that could be leveraged by the software router.

2.3.1 Wind Power Characteristics

Predictability. Wind power is known to be variable due to temporal changes in wind patterns and weather conditions. We accommodate this variability while provisioning AI compute at wind farms, as discussed in §2.2. Albeit variable, wind power is predictable, and this characteristic could be leveraged by the AI Greenferencing software while routing workload across multiple wind RE sites. We observe that the mean autocorrelation score across 4 different combinations of wind regions (Wallonia, Flanders) and power grids (Elia, Dso) at 15 min granularity (Jan to Jul'24, ELIA dataset [28]) with a lag of 1 is 0.991. Across 235 wind farm sites in the EMHIRES dataset [85], the mean (likewise, median) autocorrelation score with a lag of 1 is 0.989 (0.99) for 1 year (2018-19) of generation data at a granularity of 1 hour. These observations demonstrate strong predictability
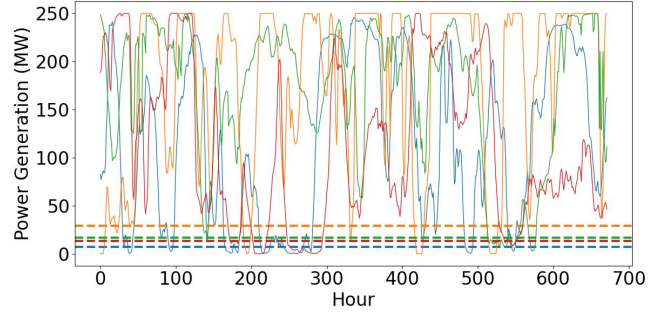

Figure 6: *Scaled wind power generation across 4 sites in Europe show complementarity. The dashed lines show their corresponding long-term (not just this window) 20th ptile generation.*

of wind power at different temporal granularities with time series or ML-based prediction models that could consume additional input features such as historical data, seasonality, local weather, turbine specification and orientation, farm layout, etc. Our custom-built AI-based energy prediction framework[4] consumes diverse onsite and offsite inputs and integrates various open source models like TFT [50] from Google and DeepMC [45] from Microsoft. The framework offers a relative improvement of 20% in prediction errors compared to state-of-the-art predictors like TFT (only onsite data). Such orthogonal work helps treat wind power generation as an oracle in systems design.

Complementarity. Wind power generation across sites could demonstrate complementarity, as already highlighted in past work, by leveraging different wind systems across geographies. While SkyBox [74] and Virtual Battery [9] leverage complementarity to proactively migrate virtual machine workloads, HERON could use this characteristic of wind power to route AI inference requests around low generation. Note that each site is still constrained by the provisioned hardware, as we discuss in §2.2. HERON could route to a set of sites that offer enough complementarity such that the generation at all sites do not go down together. As there are usually many site choices within a range of 50 ms fiber RTT (§2.1), we could identify sets of cross-country sites that minimize the coefficient of variation (CoV) of aggregate generation. In the EMHIRES dataset, we found that combinations of 4 wind sites (with assumed peak generation of 250 MW following our discussion in §2.2; normalized to 1.0 in EMHIRES) from multiple countries, all within a low network latency range, could reduce the CoV by 36% (likewise, 32%) compared to a single site (likewise, 4 different sites but within the same country). One such combination consists of 4 wind farms from Iceland, Norway, Switzerland, and the UK, as seen in Fig. 6 and offers a relatively low CV of 0.475 long-term (1 year of data, EMHIRES). The dashed lines show the peak demand

---

[4]Orthogonal to this work.

of the provisioned hardware (20th percentile of the power generation, long-term), as we discuss in §2.2. Note that the per-site generation is mostly higher than this peak demand and, also, very rarely the generation of all sites drop to the peak-demand thresholds at the same time.

> **Takeaway:** Wind power is predictable across time allowing software routing to plan-ahead. Spatial complementarity of wind power helps move around inference workload.
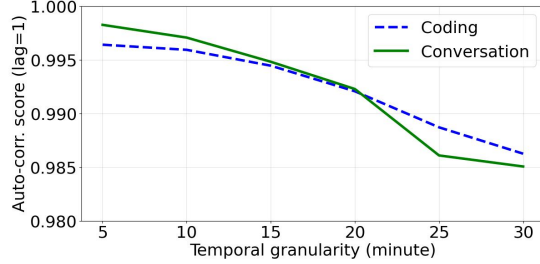
Figure 7: *AI inferencing request arrival has high autocorrelation scores with a lag of 1 for different temporal granularities.*

2.3.2 Predictable AI Inferencing Workload

We use real week-long *coding* and *conversation* production traces [11] from Azure and measure the variability and predictability. Both traces show clear diurnal and weekly patterns (Fig. 12, right). To understand the predictability of the workload we perform a simple time series analysis, as seen in Fig. 7, where we aggregate all request arrivals within a time window of X min (x-axis; multiples of 5) and compute the autocorrelation scores for a lag of 1. As we can see, the scores are very close to 1.0 in general. At 15 min granularity, the correlation scores are higher than 0.994 for both traces. This strong temporal correlation demonstrates that the workload arrival is highly predictable leveraging regression-based or machine learning based models with additional input features like time of the day, day of the week, etc. Workload predictability helps HERON make informed, intelligent decisions online.

> **Takeaway:** AI inferencing workload arrival is predictable thus allowing HERON intelligent planning ahead.

3 DESIGN OVERVIEW

Motivated by the availability of wind power capacity close to demand and the predictable variability of wind generation, we now detail our cross-site router, HERON, that intelligently routes the AI inferencing requests across sites for good performance. HERON takes into account latency SLOs, wind generation variability, workload variability, workload characteristics, and application and infrastructure-level capabilities while routing inferencing requests. This section
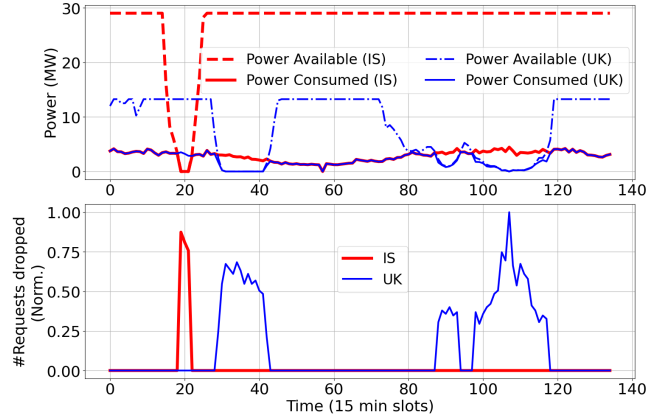
Figure 8: *DynamoLLM needs to drop requests in the absence of cross-site intelligence that leverage cross-site complementarity.*

outlines key design challenges and opportunities, while the next delves into HERON's design components.

3.1 Challenges

(C1) Residual power uncertainty. Right-sizing the compute deployment at each wind-site (see §2.2) significantly reduces the power uncertainty. Nevertheless, deploying compute, say, at the 20th percentile of peak generation at a site still translates to lower-than-peak power 20% of the time. A naive routing strategy could transiently bring more workload than a site can handle, given the power constraint, leading to request drops. Fig. 8 demonstrates this challenge. The AI inferencing workload (real trace, discussed later in §5) is routed across 2 wind farm sites in the UK (compute provisioned for peak demand of 13.25 MW) and Iceland (likewise, 29 MW) following a round-robin strategy. Each site runs a DynamoLLM scheduler [73] (our baseline; discussed further in §6) that aims to minimize power/energy consumption (assumes infinite hardware and power availability as in a traditional DC). We see that at times between 285-315 min for the Iceland site and 435-630, 1320-1395, and 1470-1755 min for the UK site, the available power drops below the provisioned peak and also the minimum ask of the assigned workload thus resulting in significant request drops (bottom plot, normalized by the peak drop).

(C2) Complex workload characteristics. In this work, we focus on large language model (LLM) inferencing. Different LLM use cases could have diverse input and output lengths, and latency SLOs, and could be run with different tensor parallelism (TP), GPU frequency, and load (requests per second or RPS) configurations. The configuration on which it runs decides the fate of the request—the power it consumes and the end-to-end (E2E) latency it incurs. In a cross-site dynamic-power and compute-constrained setting, this online matching problem poses a challenge.

(C3) Knobs with varying overheads. The TP and GPU frequency knobs discussed above have overheads to different degrees. While changing GPU frequencies take milliseconds via `nvidia-smi`, changing TP degree could take a few seconds to minutes due to the multiple steps involved—waiting for existing requests to complete, re-sharding the model weights, transferring them to the GPUs involved, and synchronizing the GPUs. Tuning these knobs online in a cross-site setting, where each site could also have power variability, keeping in mind their varying overheads, needs careful planning.

(C4) Latency versus power trade-off. Given the power and compute constraint, one could run the AI Greenferencing system to minimize either the aggregate request latency or the aggregate power consumption. We discuss in §5 how one could tradeoff latency for power and vice versa. While we leave it to the AI operator to decide on the operating point, AI Greenferencing design needs to be aware of both choices.

3.2 Key Ideas

HERON addresses the above challenges with the following key ideas:

(K1) Spatial power complementarity. *All* request drops in Fig. 8 (C1) could have been avoided by routing around 'bad' power conditions as the combination of sites offered full complementarity during the time window. HERON leverages this spatial complementarity to mitigate the residual temporal uncertainty, improving goodput over baselines (§5.2). It uses a site-wise power oracle (§2.3) while routing workloads.

(K2) Workload characteristics to lookup table. To address C2, we ran a deep power profiling benchmark building on top of past work [73] but for more recent Llama 3.1 70 B model. We classify inferencing requests based on their input and output lengths, quantify plausible latency SLOs, and quantify how different degrees of TP and frequencies could run the same request class (at different levels of load or RPS) at different power and latencies. The profiling exercise helps us obtain 2 useful functions (lookup tables): $e2e(c, f, t, l)$ that gives the end-to-end latency and $power(c, f, t, l)$ that gives the peak power consumed by the c-th class of requests at load l run with TP t and GPU frequency f. The lookup table contains approximately 2,000 rows. In a dynamic setting with available power being lower than the peak at times, it is important to leverage these tuning knobs well to remain within the resource constraints. We discuss profiling results in §5.1 for both traces albeit profiling could be agnostic to use cases (not models).

(K3) HERON's hierarchical design. To tackle the overhead of different configuration knobs (C3), HERON is hierarchically designed—one component runs at a coarser temporal granularity deciding the TP degree of GPU instances, frequencies and load across all sites, while another runs more frequently reconfiguring only the frequencies and load. Such
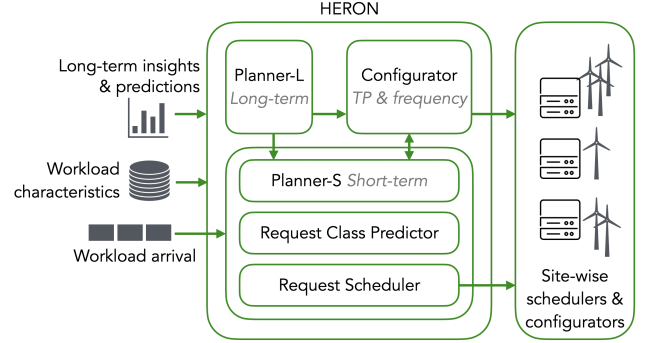

Figure 9: *AI Greenferencing architecture.*

an architecture helps HERON adapt to differences in changing such knobs as mentioned in challenge C3, while still optimizing for real-time conditions. HERON minimizes the number of TP and frequency reconfigurations. This goes well with the fact that both power and workload changes are gradual too (see §2.3). HERON also adopts DynamoLLM [73] optimizations (using inter-GPU NVLink and transferring weights between GPUs in the background) toward minimizing re-sharding overheads during TP reconfigurations. The hierarchical decision making effectively addresses the problem of mispredictions at coarser timescales.

(K4) HERON's different objectives. HERON components could run with either a minimum latency or a minimum power consumption objective (C4). The different components work in tandem with the same objective, except a heuristic that aims to pack requests better toward improving the response latencies. Our results compare both these objectives and quantifies the tradeoff.

4 HERON DESIGN

HERON cross-site request router consists of multiple components as shown in Fig. 9. The goal of HERON is to determine the number of GPUs, TP degree, GPU frequency, and load at each renewable site for different request classes. Note that inference workloads could be split into multiple classes [73] based on input and output length, with each class having different power demands and latency SLOs (discussed later in §5.1). We decompose HERON into three key components that run at different timescales: `Planner-L`, `Planner-S`, and `Request Scheduler` with a `packing heuristic`. `Planner-L` runs at a coarser timescale (e.g., 15 min) and assigns the TP degree, frequency, and load. `Planner-S` runs every few seconds and updates the frequency and load based on near-real-time power and workload conditions while sticking to the TP assignments done by `Planner-L`. A real-time `packing heuristic` runs with a `Request Scheduler` to further handle the changing conditions. We detail them below.

Table 1: Notations: ILP for Planner-L and Planner-S

Notation	Description
C, T	Set of workload classes and TP degrees
F, L	Set of all frequencies and loads (requests/s)
S	Set of wind farm sites
R_L	Limit on reconfigurations
N_s, P_s	Number of GPUs (right-sizing, §2.2) and power at site s (from real traces)
$Load_c$	Peak load for class c (from prod. trace)
$TP\_GPU(t)$	Number of GPUs for TP degree t
$e2e(c, f, t, l)$ $power(c, f, t, l)$	End-to-end latency and power for class c, frequency f, TP t, load l (from profiling exercise – lookup table)
$Old_{c,f,t,s,l}$	Number of GPUs in previous iteration for a $\{c, f, t, s, l\}$
$GPU_{s,c,t}$	Number of GPUs assigned to $\{s, c, t\}$
$X_{c,f,t,s,l}$	Number of instances for a $\{c, f, t, s, l\}$
$Y_{c,f,t,s,l}$	Binary flag set if at least one instance assigned for a $\{c, f, t, s, l\}$
$R_{c,f,t,s,l}$	Number of reconfigurations for a $\{c, f, t, s, l\}$
$Z_{c,f,t,s,l}$	Number of GPUs calculated by Planner-S

ILP Variable: $X_{c,f,t,s,l}$

Objective: Minimize $\displaystyle\sum_{c \in C, f \in F} \sum_{t \in T, s \in S, l \in L} X_{c,f,t,s,l} \cdot e2e(c, f, t, l)$

Constraints:

(1) GPU per site within limit:

$\forall s \in S, \displaystyle\sum_{c \in C, f \in F, t \in T, l \in L} X_{c,f,t,s,l} \cdot TP\_GPU(t) \leq N_s$

(2) Power per site within limit:

$\forall s \in S, \displaystyle\sum_{c \in C, f \in F, t \in T, l \in L} X_{c,f,t,s,l} \cdot power(c, f, t, l) \leq P_s$

(3) Enough serving capacity for each class:

$\forall c \in C, \displaystyle\sum_{s \in S, f \in F, t \in T, l \in L} X_{c,f,t,s,l} \cdot l \geq Load_c$

(4) One configuration of f and l:

$\forall s \in S, c \in C, t \in T, \displaystyle\sum_{f \in F, l \in L} Y_{c,f,t,s,l} \leq 1$

(5) Expressing Y in terms of X:

$\forall s \in S, c \in C, t \in T, f \in F, l \in L, 1 \geq Y_{c,f,t,s,l} \geq \dfrac{X_{c,f,t,s,l}}{sum(N_S)}$

(6) Number of reconfigurations are bounded:

$\displaystyle\sum_{c \in C, f \in F} \sum_{t \in T, s \in S, l \in L} R_{c,f,t,s,l} \leq R_L$

(7) Expressing $R_{c,f,t,s,l}$, $\forall s \in S, c \in C, t \in T, f \in F, l \in L$,

$R_{c,f,t,s,l} \geq Old_{c,f,t,s,l} - X_{c,f,t,s,l}, R_{c,f,t,s,l} \geq X_{c,f,t,s,l} - Old_{c,f,t,s,l}$

Figure 10: *Formulation: ILP for Planner-L*

Planner-L. Both workload arrival and wind power generation per site are highly predictable at 15 min granularity (see §2) and these predictions, alongside inferencing profiled data (a lookup table), are leveraged by a long-term (15 min) lookahead Planner-L ILP (integer linear program). Based on the

predictions, Planner-L comes up with a plan to assign the number of instances (of GPUs) for the next 15 min window for the peak load of each class of inferencing requests (based on input and output length) at each site. In addition, it also assigns a single frequency and load for a TP degree for an individual class at a specific site. We applied this constraint to scale the ILP without affecting optimality. The values for classes and frequencies are detailed in §5.1.

Planner-L bounds TP reconfigurations to further aid the strategies for hiding reconfiguration overhead (discussed in §3.2). Planner-L outputs TP and frequency reconfigurations needed, the load (RPS) per GPU instance, and also the fraction of inferencing requests per class that should be routed to individual instances across the multiple sites. Table. 1 and Fig. 10 detail the notation and formulation of the ILP – it takes as input the workload classes, TP degrees, GPU operating frequencies, load levels (RPS within latency SLO bounds) per class per instance, number of GPUs and power available per site in the next time slot, the peak load per class of requests, and previous configuration, and gives as output the optimal number of instances that minimizes end-to-end latency of inferencing. The ILP minimizes the aggregate end-to-end latency for the workload (thus not compromising on the user experience) making sure that the configuration respects the (1) hardware and (2) power availability constraints per site, (3) each class of requests get enough serving capacity that could satisfy the peak load, (4,5) a TP instance serving a class of requests get the same frequency assigned for the same load, and (6, 7) the number of reconfigurations needed per time slot is bounded.

Configurator. The Configurator is in charge of updating the site-specific configurators of the necessary changes in TP and operating frequencies of different clusters of GPUs. It restrains Planner-S from considering clusters with pending reconfigurations for workload placement.

ILP Variable: $Z_{c,f,t,s,l}$

Objective: Minimize

$\displaystyle\sum_{s \in S, c \in C, t \in T} \sum_{f \in F, l \in L} Z_{c,f,t,s,l} \cdot e2e(c, f, t, l)$

Constraints:

(1) Power per site within limit:

$\forall s \in S, \displaystyle\sum_{c,f,t,l} Z_{c,f,t,s,l} \cdot power(c, f, t, l) \leq P_s$

(2) Enough serving capacity for each class:

$\forall c \in C, \displaystyle\sum_{s \in S, f \in F, t \in T, l \in L} Z_{c,f,t,s,l} \cdot l \geq Load_c$

(3) Limiting number of GPUs per config:

$\forall s \in S, c \in C, t \in T, \displaystyle\sum_{l \in L, f \in F} Z_{c,f,t,s,l} \leq GPU_{s,c,t}$

Figure 11: *Formulation: ILP for Planner-S*

Planner-S. While `Planner-L` runs every 15 min and reconfigures TP, operating frequency, and load per GPU instance, we run a lightweight `Planner-S` every few seconds to tens of seconds (bounded by its scalability as the number of sites grow; see §5.3) that reconfigures only the frequencies and the load for a configuration, as TP changes are heavier. Running `Planner-S` at finer timescales has 2 major benefits: (1) it could tackle near-real-time power and workload arrival uncertainties and (2) it could unleash performance opportunity by running requests at higher than `Planner-L` assigned GPU frequencies as the available power goes up, thus improving TTFT and TBT latencies.

`Planner-S` is also an ILP (notations in Table. 1 and formulation in Fig. 11 that recalculates optimal frequency and load assignments for each configuration. Note that `Planner-S` has the same optimization goal as `Planner-L` (minimizes the end-to-end latency) but only reconfigures frequencies (significantly faster to do with minimal overhead) while working with TP calculated by `Planner-L`. The ILP variable $Z_{c,f,t,s,l}$ indicates number of instances assigned for each config (tuple of site, class and TP degree) and frequency and load. Note, only frequency and load are reassigned for individual configs. The constraints are: (1) power constraints are met, (2) the serving capacity is enough for the peak load, (3) fits in the same GPU budget assigned by Planner-L.

Note that `Planner-L` and `Planner-S` formulations could be trivially changed to minimize power consumption.

Request scheduler. This component is in charge of dispatching the inference requests, as they arrive in the arrival queue, across multiple sites. It consumes the output of `Planner-S` to learn the weights with which it performs a weighted round-robin (WRR) routing for each class (input/output length) of requests across clusters of GPUs assigned to individual classes across sites. To assign classes to requests, it uses a `Request Class Predictor` that takes as inputs the request and the model, predicts the output length and, hence, the class. Our predictor consists of an Albert/DistilBert/Longformer [14, 46, 67] layer for input embeddings followed by a regressor, and could predict the output length bucket (small/medium/long) with an accuracy of 99.95%+ (predicting buckets is easier than predicting absolute output lengths). Hence, following the footsteps of DynamoLLM [73], we treat output length as an oracle in our experiments. The `Request Scheduler` also has a `packing heuristic` that we describe next.

Packing heuristic. If the `Request Scheduler` observes online that the load for a request class for a GPU instance is lower than expected, it moves a fraction of the smaller requests (e.g., LS) to this class (e.g., LM) for better performance. It starts with the larger requests, reassigns, and move to the smaller ones, freeing up resources at each step.

Meta choices. (1) HERON can be deployed at the AI provider network ingress or near a DC to intercept workloads. (2) We exclude full-fledged DCs and on-site batteries/generators to stress-test HERON's ability to handle residual uncertainties in power and workload arrival. (3) While HERON supports any site-local scheduler, we use a FIFO round-robin scheduler for GPU assignment in this work.

5 HERON **EVALUATION**

We explore the following key questions (alongside others): **§5.1:** How do the power consumption and latency vary when workloads are run with different combinations of TP, frequency, and load? **§5.2:** How does having cross-site global view help improve the offered goodput under uncertainties? **§5.3:** Does a hierarchical HERON design unleash performance?

Data. For the experiments below, we use the following data: (1) 1-week of LLM inferencing production trace for 2 use cases, *coding* and *conversation*, on Azure [11]. Fig. 12 shows the characteristic differences between the *conversation* and *coding* production traces on Azure. The inputs (left figure) vary between 1 and $\sim 8K$ tokens, with *coding* inputs being $\sim 2\times$ larger than *conversation* at the median. The outputs (middle figure) lie within $\sim 1K$ tokens with *conversation* having $\sim 6\times$ larger outputs at the 95th percentile.

(2) Hourly power generation from 4 wind farms in Iceland, Norway, Switzerland, and the UK (with assumed peak generation of 250 MW [5] following our discussion in §2.2; normalized to 1.0 in EMHIRES) for 672 hours (see Fig. 6) in 2015 from EMHIRES [85]. For our experiments, we scale the time dimension by 0.25, thus translating it to 1 week of power generation data at 15 min granularity. We could not use the ELIA data (see §2.3.2) as it covers only 2 regions in Belgium.

(3) The 20th percentile of power generation for the 4 wind farms above for the entire year of 2015 at 1 hour granularity as recorded in EMHIRES. These long-term thresholds (Iceland: 29 MW, Norway: 16.5 MW, Switzerland: 7 MW, UK: 13.25 MW) are different across sites due to local conditions and decide the provisioned compute cluster size. We deploy compute (in simulation) in multiples of NVIDIA H100 Super-PODS (1,016 GPUs, 1.3 MW peak power draw).

5.1 **Profiling The Workload**

For each production trace, we categorize the requests into 9 buckets {SS, SM, SL, MS, MM, ML, LS, LM, LL}—3×3 combinations of **S**hort (33rd percentile over the entire week), **M**edium (66th ptile), and **L**ong (100th ptile) inputs and outputs. We profile the peak power, average power, TTFT (time to first token), TBT (time between tokens), and E2E (end-to-end) latency of each class of requests on H100 DGX and Llama 3.1 70B model for different combinations of load (requests/s),

---

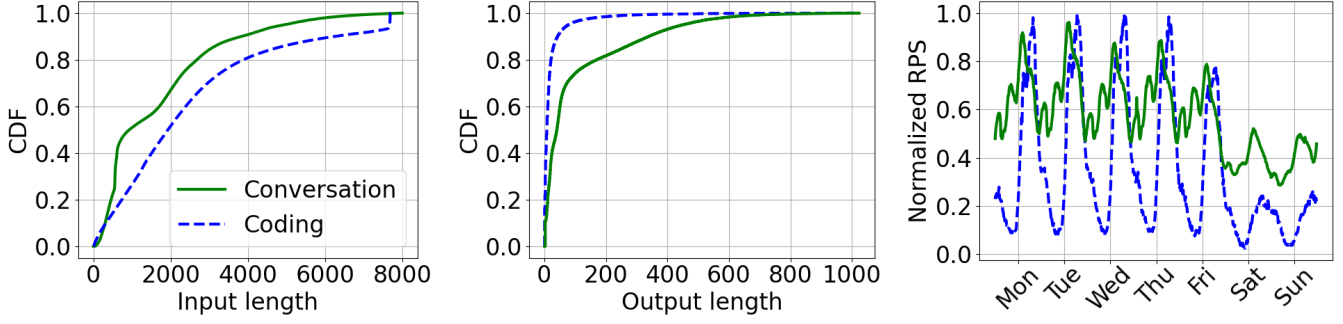[5] We deploy compute at these larger sites to avoid significant fragmentation.

Figure 12: *The input lengths (left), output length (middle), and arrival patterns (right) vary significantly across the 2 traces and even for the same trace.*
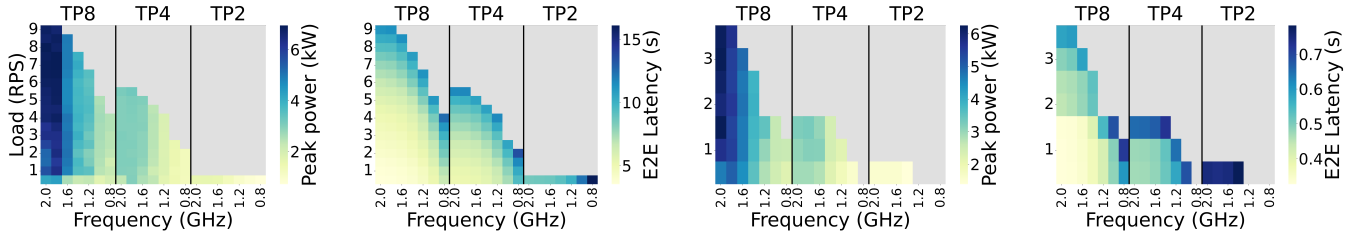

Figure 13: *Peak power and E2E latency for various load levels (requests per second or RPS) of MM requests across different TP and frequency combinations for conversation (left) and coding (right) workloads.*

tensor parallelism (TP2, TP4, and TP8), and GPU frequencies (0.8, 1, 1.2, 1.4, 1.6, 1.8, and 2 GHz). For each combination of class c, frequency f, TP t, and load l, the GPU instance (multiple GPUs; e.g. TP8 needs 8) caters to a Poisson arrival of requests with load l for 5 min, and we extract the power consumption statistics with NVIDIA's DCGM interface [3] and the latency statistics from the vLLM [6] serving engine. The TTFT and TBT latency SLOs are set to 5× the TTFT/TBT values when a single request from a particular class is run in isolation at TP8 and 2 GHz frequency.

Fig. 13 shows the peak power and E2E latency of MM *conversation* (left) and *coding* (right) requests under varying loads for different combinations of TP and frequencies. The grey cells (top right corners) denote TTFT/TBT violations—note that TP2 cannot sustain any load higher than 0.5 requests/s for this class of requests. For a specific TP and frequency, as the load increases, both peak power consumption and latency (E2E as well as TTFT) inflate. Keeping everything else constant, a higher TP degree (or a larger operating frequency) leads to better latencies at the cost of higher power consumption. We have verified that the trends are similar across different classes of the 2 traces, while the exact thresholds are different. *Coding* trace has larger input (context) in general and hence cannot sustain high load levels (SLO violations) as compared to *conversation*.

The profiling exercise generates lookup tables (functions) like $e2e(c, f, t, l)$ and $power(c, f, t, l)$ for all request classes for both *conversation* and *coding* traces that are leveraged by HERON. Note that HERON considers only those $\{c, f, t, s, l\}$ combinations that *do not lead to TTFT/TBT SLO violations*. Also, we used a fixed multiplier, 1.82, for all the power numbers from $power(c, f, t, l)$ in all the following experiments to accommodate other components (CPU, fans, networks, memory) in the mix. Each H100 DGX box (8 GPUs) consumes a peak power of 10.2 KW, which is 1.82× higher than the aggregate peak power of the 8 GPUs (8 × 700 W).

5.2 Planner-L Component Testing

Now we see `Planner-L` in action in isolation (without HERON component `Planner-S` & `Request Scheduler`'s packing `heuristic`). This exercise demonstrates how having a global view (akin to SDN's control plane [82]) across multiple sites and the power-variability awareness helps HERON unleash performance beyond the current baselines that use site-local views and naive round-robin routing. It also helps quantify the incremental benefits of the other HERON components in §5.3. For the following experiments, we assume that the `Planner-L` decisions are blindly followed by the `Request Scheduler` and the `Configurator`.

Setup. We run a week-long simulation for both production traces being served on Llama 3.1 70B parameter model across

the 4 wind sites in Europe. We compare 2 variants (*a & b* below) of `Planner-L` schedulers against 2 baselines as follows: (*a*) Planner-L-min-power: objective is to minimize total power consumed. (*b*) Planner-L-min-latency: objective is to minimize aggregate E2E latency per 15 min. (*c*) WRR (weighted round-robin) + DynamoLLM: DynamoLLM scheduler [73] tries to minimize power draw over time (energy). A WRR router forwards fractions of incoming requests assigned in proportion to the compute per site. (*d*) Greedy minimum latency baseline: a heuristic that attempts to achieve low latency by assigning TP8, highest operating frequency, and lowest load level to GPU instances. However, we observed that it fails to assign ~33% requests over time due to GPU capacity constraints in place. To address this limitation, we increase the assigned load per GPU instance, as needed, while still assigning TP8 and highest frequency. We observed that the latency vs. load curve results in a "knee point"—the latency increases before such a point is small. We calculate the knee point for each class of requests and limit the load on each GPU instance to these knee points.

`Planner-L` offers high goodput. We first run the 4 schedulers at 15 min granularity and compare, in Fig. 14 (left), how frequently the schedulers hit corner cases with not enough power for the entire incoming workload. As neither DynamoLLM nor the minimum latency baseline is power-variability aware with WRR router not able to leverage power complementarity across sites, they onboard more workload per site than could be served leading to request drops. Fig. 14 (left) shows how often (number of 15 min slots with at least one request drop) schedulers encounter request drops (due to local power unavailability) in a week for different volumes of the coding workload (*x*-times the arrival rate in the production traces). Both Planner-L variants (min. power and min. latency) could cater to significantly higher workload over time without any request drops. Essentially, planner-L min. latency converges to planner-L min. power in extreme resource-constrained cases – if `Planner-L` min. power already drops requests due to resource constraints, the best plan that `Planner-L` min. latency could offer is same as the min. power one; hence, the overlap in the figure. Also, note that our choice of not using batteries, generators, or full-fledged DCs in the mix is stress-testing planner-L. In a real deployment, Planner-L could essentially guarantee no request drops, given its power-variability awareness.

Next, for a fixed multiplier of 60× for the incoming *coding* workload, we plot the goodput (requests being actually served) improvement factor for the Planner-L (both versions coincide) with respect to the 2 baselines (*c* and *d*). This improvement factor is the ratio of the goodput with Planner-L and the goodput with the corresponding baseline. Fig. 14 (middle) shows that planner-L (hence, HERON) could offer up

to 1.8× improvement in goodput compared to the baselines. Note that none of the 4 choices have any difficulty serving the workload 80+% ($y = 0.8$) of the time, thanks to our compute right-sizing strategy in §2.2. *Even the baselines performing well up to a high-percentile in Fig. 14 (middle) is in fact one of our key contributions (deployment right-sizing discussed in §2.2).* The ratio stays at 1 for a while even beyond the 80th percentile mark in cases where the workload arrival rate at a site goes down as the power goes down. Also the mass at the top right (80% improvement in goodput) corresponds to the scenario when the Iceland site goes down completely at times—HERON (planner-L), in anticipation, intelligently assigns the workload to the other sites. Similarly, Fig 15 shows the goodput improvement with HERON (`Planner-L`) for the *conversation* trace to be 50× as compared to the baselines with WRR routing under the same setup as in §5.2. We see a 1.8× improvement for both variants of `Planner-L` none of which need to drop requests, thanks to HERON's capability to leverage spatial power generation complementarity.

`Planner-L` variants offer a trade-off. Fig. 16 demonstrates the trade-off offered by the 2 variants of Planner-L for *coding* trace (60×) over the entire week. We fit a cubic curve to represent the %-age improvement in E2E latency with Planner-L min. latency and the corresponding %-age power inflation with respect to the min. power version (mean values per 15 min slot over all requests). A 25% (likewise, 50%) E2E latency improvement comes at the cost of 42% (likewise, 198%) power inflation. For *conversation* (50×) trace, a 25% latency improvement corresponds to 49% power inflation.

`Planner-L` configuration stickiness. The planner-L ILP uses a reconfiguration threshold R_L (X%) that limits the number of TP changes that are allowed per 15 min slot (changes in generation are gradual in wind farms and span multiple such time-slots). We have run 1-week experiments with different values of R_L for both workloads, and have not seen any significant inflation in either the 95th-ptiles (across time-slots) of the mean (within a 15 min slot) E2E latency or the mean power consumed for R_L values as low as 3%. Nevertheless, the latency inflates below $R_L = 3$%.

`Planner-L` scalability. Fig. 14 (right) plots the execution time of `Planner-L` (alongside other HERON components that we discuss later) on a standard CPU for different number of wind-sites. Note that it increases linearly (*y*-axis is log-scale) in the range we tested for, always consuming ≤ 1% CPU and ≤ 1 GB RAM. Even for 64 wind farm sites (cumulative power 2+ GW), it executes in 6 min which is way below the 15 min budget. We used COIN-OR [19] solver for our experiments – the ILP could speed up, if needed, by running on commercial solvers like CPLEX [79] with multi-cores.
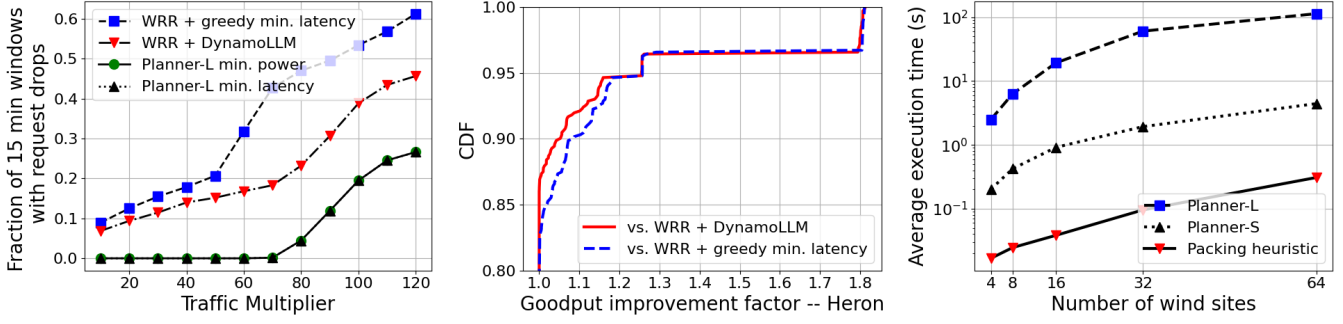
Figure 14: *(left) Across different volumes of coding workload, the baselines lead to more frequent request drops. (middle) For a given workload, the goodput factor is significantly higher with* HERON *(planner-L) at higher percentiles (residual uncertainty). (right) Execution times of* HERON *components.*
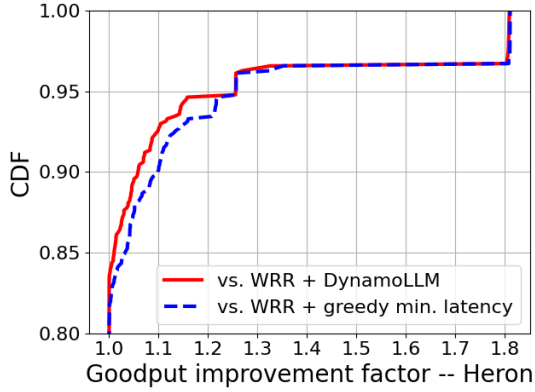

Figure 15: *For conversation (50×) workload, the goodput factor is significantly higher with* HERON *(planner-L) at higher percentiles (residual uncertainty).*
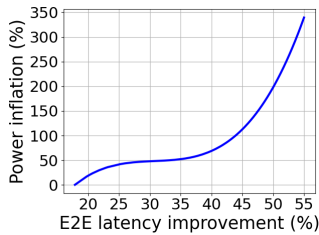

Figure 16: *Planner-L min. latency offers lower latency at the cost of higher power than Planner-L min. power.*

For all of the above results, the general observations and trends are similar across both *coding* and *conversation*. Hence, we focused on one of them reporting the other one as needed.

5.3 Unleashing More Performance

`Planner-S` runs every 5 seconds in this setup with 4 wind farm sites reconfiguring operating frequencies and load as needed, and the `packing heuristic` (with `Request Scheduler`) runs online greedily packing smaller requests to GPU instances configured for larger ones (e.g. LS requests could
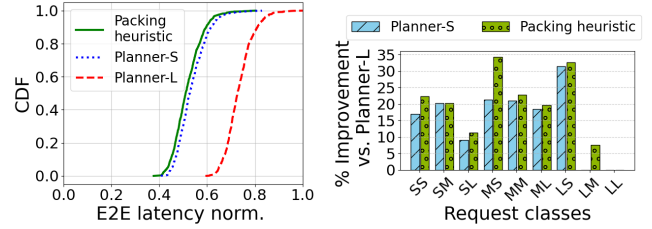

Figure 17: *(left) Planner-S and the greedy packing heuristic incrementally improve E2E latency over the planner-L (min. latency) assignments. (right) Request class-wise mean E2E latency improvements over Planner-L.*

be assigned to LM instance) leading to E2E latency improvements. In Fig. 17 we show the incremental latency benefits of running these components in tandem with `Planner-L` (min. latency) for a 15 min slot. The CDF plots (Fig. 17, left) show the E2E latencies (mean per second) when requests are routed based on `Planner-L`, `Planner-L+S`, and `Planner-L+S+packing heuristic` assignments. At the median, the *E2E* latency improvements in aggregate are 27% with `Planner-S` and an additional 3% with the `packing heuristic` compared to only `Planner-L`. For the *conversation* trace, `Planner-S` offers 12% improvement (at the median) in latency over `Planner-L` min. latency and the `packing heuristic` offers a further 2% improvement. Fig. 17 (right) shows the request class-wise mean reductions in latency over `Planner-L`—the incremental improvements vary between 7.5% and 34.4% with LL requests seeing no improvements as the `packing heuristic` does not have a better class to assign these requests to. Note that both `Planner-L` and `Planner-S` are run with the same optimization goal – in this case, minimizing the E2E latency.

Planner-S power elasticity. We stress-tested `Planner-L` by reducing the available power by 20% and still routing as per `Planner-L` assignments. We picked a 5 s window at

random and observed that, only for one 1 s window, ~7% requests had to be dropped to be within the reduced power budget (than anticipated). `Planner-S` though could accommodate all requests for this 1 s window.

Scalability. As seen in Fig. 14 (right), `Planner-S` is ~30× faster than `Planner-L`. For smaller setups with 4 wind sites, `Planner-S` could run multiple times in a second, and for larger 64 site deployments it could run every 5 seconds (180 times in 15 min). Hence, `Planner-S` could be multiple orders of magnitude more fine-grained than `Planner-L`, also because it does not touch the TP configurations. The `packing heuristic` is, again, significantly quicker (sub-second even for larger deployments) than the other components and could run with the online `Request Scheduler`.

6 RELATED WORK

DC running on renewables. Previous works [35, 39, 49, 70, 84] explored renewable-powered DCs leveraging solar power, battery, grid draw, and scheduling optimization. Key differences with AI Greenferencing are: (1) *Driver:* We aim to unlock more AI compute motivated by the growing power demand. (2) *Power source:* We focus on wind energy as solar generation needs to rely on batteries or generators heavily. (3) *Right-sizing:* We recommend smaller compute in larger wind farms, a strategy that significantly reduces power uncertainty. (4) *Cross-site:* Our work leverages power complementarity across sites. (5) *Workload:* AI inferencing is more amenable to power fluctuations than traditional workloads.

Moving compute to power. Virtual Battery & SkyBox [9, 74] use renewable power complementarity across sites to run compute. Differences with AI Greenferencing: (1) *Provisioning:* We focus on compute right-sizing and intelligent routing software alongside power complementarity. (2) *Scalability:* While SkyBox does not leverage sites with large variations in generation, we could still include them. (3) *Workload:* SkyBox focuses on heavier VM placement and migration. We rather focus on intelligent routing of AI inferencing requests.

Power-aware AI Inferencing. DynamoLLM [73] minimizes energy/power consumption in AI inferencing by tuning TP degree and GPU frequencies. While we build HERON on top of this work, the key differences are: (1) *Cross-site:* Our work focuses on cross-site routing and not site-specific scheduling. (2) *Assumptions:* DynamoLLM assumes no power or compute limitations, while we are constrained on both at wind sites. (3) *Approach:* We are power-variability aware.

7 DISCUSSIONS

Batteries/generators. They help alleviate some of the wind power uncertainty at the cost of round-trip loss, hybrid energy mix, and higher CAPEX. Here, we stress-tested HERON without using any of these additional modalities and rather

leverage right provisioning, power complementarity, and application/infrastructure knobs to tackle power uncertainty.

Beyond wind. We leave exploring other sustainable energy sources for AI workloads as future work. Solar farms face ~50% unavailability, requiring costly battery/generator backup, increasing CAPEX and OPEX. While nuclear energy is on the radar of big AI players [21, 75] for its stable base load, it has its own challenges – (1) high CAPEX, (2) nuclear waste management, (3) public perception and regulations, (4) restricted zones, and (5) long setup times.

GPU phase-out. With GPU leaders like NVIDIA announcing a new generation almost every year, hyperscalars need concrete plans to phase-out the quickly aging GPUs. AI Greenferencing could help ship these GPUs to wind farms and run a fraction of the relevant AI workload at low CAPEX.

Autonomous maintenance. Alongside training farm-local labor on basic cluster maintenance, one could also engage cheap robots in these MDCs for basic repairing tasks [44].

Elephants versus mice. In AI Greenferencing, a large wind farm paired with smaller deployments might cause imbalances, requiring reroutes beyond the smaller sites' capacity. Fig. 2(a) shows DCs have diverse wind farms within low network latency, enabling sizeable deployments to handle power uncertainty effectively.

Security of the models. An AI inferencing provider can either deploy their own compute or leverage 3rd party compute (startups [5, 7, 8] co-locating compute in wind farms). With the latter approach, the AI provider needs to resort to privacy-preserving model-splitting techniques [31, 38].

8 CONCLUSION

AI Greenferencing brings AI compute to wind energy sources, bypassing power delivery challenges and meeting rising AI demand sustainably. We present a feasibility study and a right-sizing strategy to mitigate wind power uncertainties. Our cross-site router efficiently manages residual uncertainties, delivering up to 1.8× the goodput of state-of-the-art solutions. *This work does not raise any ethical issues.*

REFERENCES

[1] 2023. Azure locations (public). https://gist.github.com/lpellegr/8ed20 4b10c2589a1fb925a160191b974. (2023).

[2] 2024. GCP locations (public). https://github.com/GoogleCloudPlatfo rm/region-picker/blob/main/data/regions.json. (2024).

[3] 2025. DCGMI. https://microsoft.github.io/VirtualClient/docs/workloa ds/dcgmi/. (2025).

[4] 2025. NVIDIA H100 Tensor Core GPU. https://resources.nvidia.com/e n-us-tensor-core/nvidia-tensor-core-gpu-datasheet. (2025).

[5] 2025. Soluna. https://www.solunacomputing.com/. (2025).

[6] 2025. vLLM. https://docs.vllm.ai/en/latest/. (2025).

[7] 2025. WestfalenWIND. https://www.westfalenwind.de/. (2025).

[8] 2025. windCORES. https://www.windcores.de/en/. (2025).

[9] Anup Agarwal, Jinghan Sun, Shadi Noghabi, Srinivasan Iyengar, Anirudh Badam, Ranveer Chandra, Srinivasan Seshan, and Shivkumar

Kalyanaraman. 2021. Redesigning data centers for renewable energy. In *ACM HotNets*.

[10] Azure. 2025. Azure Modular Data Center (MDC) Operator and User Documentation. https://learn.microsoft.com/en-us/azure-stack/mdc/. (2025).

[11] AzurePublicDataset. 2025. Azure LLM inference trace 2024. https://github.com/Azure/AzurePublicDataset/blob/master/AzureLLMInferenceDataset2024.md. (2025).

[12] Jeff Barr. 2019. Amazon EC2 Update – Inf1 Instances with AWS Inferentia Chips for High Performance Cost-Effective Inferencing. https://aws.amazon.com/blogs/aws/amazon-ec2-update-inf1-instances-with-aws-inferentia-chips-for-high-performance-cost-effective-inferencing/. (2019).

[13] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. 2019. *The datacenter as a computer: Designing warehouse-scale machines*. Springer Nature.

[14] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150* (2020).

[15] Berkeley Lab. 2024. Land-based wind market report. https://emp.lbl.gov/sites/default/files/2024-08/Land-Based%20Wind%20Market%20Report_2024%20Edition.pdf. (2024).

[16] Alexander Bick, Adam Blandin, and David J Deming. 2024. *The rapid adoption of generative AI*. Technical Report. National Bureau of Economic Research.

[17] Ilker Nadi Bozkurt, Waqar Aqeel, Debopam Bhattacherjee, Balakrishnan Chandrasekaran, Philip Brighten Godfrey, Gregory Laughlin, Bruce M Maggs, and Ankit Singla. 2018. Dissecting Latency in the Internet's Fiber Infrastructure. *arXiv:1811.10737* (2018).

[18] Clean Energy Wire. 2024. Industry electricity prices for German companies drop almost one quarter in early 2024. https://www.cleanenergywire.org/news/industry-electricity-prices-german-companies-drop-almost-one-quarter-early-2024. (2024).

[19] COIN-OR Foundation. 2025. COIN-OR. https://www.coin-or.org/. (2025).

[20] CoreSite. 2024. AI and the Data Center: Driving Greater Power Density. https://www.coresite.com/blog/ai-and-the-data-center-driving-greater-power-density. (2024).

[21] Casey Crownhart. 2024. Why Microsoft made a deal to help restart Three Mile Island. https://www.technologyreview.com/2024/09/26/1104516/three-mile-island-microsoft/. (2024).

[22] Databank. 2024. Exploring Modular Data Centers: Benefits, Design, And Deployment. https://www.databank.com/resources/blogs/exploring-modular-data-centers. (2024).

[23] Tim De Chant. 2024. Google kicks off $20B renewable energy building spree to power AI. https://techcrunch.com/2024/12/10/google-kicks-off-20b-renewable-energy-building-spree-to-power-ai/. (2024).

[24] Delta Power Solutions. 2024. Modular Data Centers: The Rise and the Advantages. https://www.deltapowersolutions.com/en-in/mcis/technical-article-modular-data-centers-the-rise-and-the-advantages.php. (2024).

[25] Diana DiGangi. 2023. Dominion Energy projects adding up to 9 GW of gas-fired capacity in Virginia to bolster reliability. https://tinyurl.com/4fbcdx24. (2023).

[26] EIA. 2024. Why are Midwest grid operators turning away wind power? https://www.eia.gov/todayinenergy/detail.php?id=62406. (2024).

[27] EIA. 2025. Electric Power Monthly. https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a. (2025).

[28] Elia. 2025. Wind power generation. https://www.elia.be/en/grid-data/generation-data/wind-power-generation. (2025).

[29] Robert L Fares and Carey W King. 2017. Trends in transmission, distribution, and administration costs for US investor-owned electric utilities. *Energy Policy* (2017).

[30] Karl Freund. 2024. Welcome To 2024. It Promises To Be Quite The AI Ride! https://www.forbes.com/sites/karlfreund/2024/01/04/welcome-to-2024-it-promises-to-be-quite-the-ai-ride/. (2024).

[31] Shucun Fu, Fang Dong, Dian Shen, and Tianyang Lu. 2024. Privacy-preserving model splitting and quality-aware device association for federated edge learning. *Software: Practice and Experience* (2024).

[32] Bill Gates. 2023. The surprising key to a clean energy future. https://www.gatesnotes.com/Transmission. (2023).

[33] GitHub. 2025. GitHub Copilot. https://github.com/features/copilot. (2025).

[34] Global Energy Monitor. 2024. Global Wind Power Tracker. https://globalenergymonitor.org/projects/global-wind-power-tracker/. (2024).

[35] Íñigo Goiri, William Katsak, Kien Le, Thu D Nguyen, and Ricardo Bianchini. 2013. Parasol and greenswitch: Managing datacenters powered by renewable energy. *ACM SIGPLAN Notices* (2013).

[36] Google. 2023. Gemini. https://blog.google/technology/ai/google-gemini-ai/. (2023).

[37] Diana Goovaerts and Matt Hamblen. 2024. Could GPU power levels break the data center ecosystem? https://www.fierce-network.com/cloud/could-gpu-power-levels-break-data-center-ecosystem. (2024).

[38] Jialin Guo, Jie Wu, Anfeng Liu, and Neal N Xiong. 2021. LightFed: An efficient and secure federated edge learning system on model splitting. *IEEE TPDS* (2021).

[39] Md E Haque, IŽigo Goiri, Ricardo Bianchini, and Thu D Nguyen. 2015. Greenpar: Scheduling parallel high performance applications in green datacenters. In *ACM ICS*.

[40] Dylan Harrison-Atlas, Anthony Lopez, and Eric Lantz. 2022. Dynamic land use implications of rapidly expanding and evolving wind power deployment. *Environmental Research Letters* (2022).

[41] Astrid Hennevogl-Kaulhausen and Ulrike Ostler. 2024. Modular data centers: Faster, more flexible and more energy-efficient in the data centers. https://www.deltapowersolutions.com/en-in/mcis/technical-article-modular-data-centers-faster-more-flexible-and-more-energy-efficient-in-the-data-centers.php. (2024).

[42] Javier C. Hernandez. 2017. It Can Power a Small Nation. But This Wind Farm in China Is Mostly Idle. https://www.nytimes.com/2017/01/15/world/asia/china-gansu-wind-farm.html. (2017).

[43] Bobby Hollis. 2024. Accelerating the addition of carbon-free energy: An update on progress. https://www.microsoft.com/en-us/microsoft-cloud/blog/2024/09/20/accelerating-the-addition-of-carbon-free-energy-an-update-on-progress/. (2024).

[44] Freddie Hong, Iason Sarantopoulos, Elliott Hogg, David Richardson, Yizhong Zhang, Hugh Williams, David Sweeney, Andromachi Chatzieleftheriou, and Antony Rowstron. 2024. Self-maintaining [networked] systems: The rise of datacenter robotics!. In *ACM HotNets*.

[45] Peeyush Kumar, Ranveer Chandra, Chetan Bansal, Shivkumar Kalyanaraman, Tanuja Ganu, and Michael Grant. 2021. Micro-climate prediction-multi scale encoder-decoder based deep learning framework. In *ACM SIGKDD*.

[46] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (2020). arXiv:cs.CL/1909.11942 https://arxiv.org/abs/1909.11942

[47] Lawrence Berkeley National Laboratory. 2024. Queued Up: 2024 Edition. https://emp.lbl.gov/sites/default/files/2024-04/Queued%20Up%202024%20Edition_R2.pdf. (2024).

[48] Vivian Lee. 2024. U.S. Data Center Power Outlook: Balancing competing power consumption needs. https://www.linkedin.com/pulse/us-data-center-power-outlook-balancing-competing-consumption-lee-iz4pe/. (2024).

[49] Chao Li, Amer Qouneh, and Tao Li. 2012. iSwitch: Coordinating and optimizing renewable energy powered server clusters. *ACM SIGARCH Computer Architecture News* (2012).

[50] Bryan Lim, Nicolas Loeff, Sercan Arik, and Tomas Pfister. 2021. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting.

[51] Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power hungry processing: Watts driving the cost of AI deployment?. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*.

[52] Shreya Mandi, Srinivas Varadarajan, Kalyan Mangalapalli, Tanvi Lall, Mohan Srinivas, Sasank Chilamkurthy, and Swaroop Rajagopalan. 2024. Micro is the new mega. https://tinyurl.com/47rk6ptb. (2024).

[53] McKinsey & Company. 2024. The role of power in unlocking the European AI revolution. https://tinyurl.com/bdf952sr. (2024).

[54] Meta. 2023. Llama 2. https://www.llama.com/llama2/. (2023).

[55] Meta. 2024. Llama 3.1. https://ai.meta.com/blog/meta-llama-3-1/. (2024).

[56] NVIDIA. 2023. NVIDIA DGX SuperPOD Data Center Design. https://docs.nvidia.com/nvidia-dgx-superpod-data-center-design-dgx-h100.pdf. (2023).

[57] NVIDIA. 2025. NVIDIA Data Center GPUs. https://www.nvidia.com/en-in/data-center/data-center-gpus/. (2025).

[58] OpenAI. 2023. GPT-3.5 Turbo. https://platform.openai.com/docs/models#gpt-3-5-turbo. (2023).

[59] OpenAI. 2023. GPT-4. https://openai.com/index/gpt-4/. (2023).

[60] Our World of Energy. 2023. How much land does a wind farm require? https://www.ourworldofenergy.com/vignettes.php?type=wind-power&id=9. (2023).

[61] Dylan Patel, Daniel Nishball, and Jeremie Eliahou Ontiveros. 2024. AI Datacenter Energy Dilemma – Race for AI Datacenter Space. https://semianalysis.com/2024/03/13/ai-datacenter-energy-dilemma-race/#datacenter-math. (2024).

[62] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *ACM ASPLOS*.

[63] Cheng Peng, Xi Yang, Aokun Chen, Kaleb E Smith, Nima PourNejatian, Anthony B Costa, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, et al. 2023. A study of generative large language model for medical research and healthcare. *NPJ digital medicine* (2023).

[64] Reuters. 2022. Germany takes new steps to tackle the energy crisis. https://www.weforum.org/stories/2022/08/energy-crisis-germany-europe/. (2022).

[65] Martin Rosenberg. 2024. Evergy Struggles to Cut Carbon as Energy Demand Soars. https://tinyurl.com/mtw7rmj6. (2024).

[66] Goldman Sachs. 2024. AI is poised to drive 160% increase in data center power demand. https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand. (2024).

[67] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. (2020). arXiv:cs.CL/1910.01108 https://arxiv.org/abs/1910.01108

[68] Anton Shilov. 2023. Nvidia to Reportedly Triple Output of Compute GPUs in 2024: Up to 2 Million H100s. https://www.tomshardware.com/news/nvidia-to-reportedly-triple-output-of-compute-gpus-in-2024-up-to-2-million-h100s. (2023).

[69] Anton Shilov. 2023. Nvidia to Sell 550,000 H100 GPUs for AI in 2023. https://www.tomshardware.com/news/nvidia-to-sell-550000-h100-compute-gpus-in-2023-report. (2023).

[70] Rahul Singh, David Irwin, Prashant Shenoy, and Kadangode K Ramakrishnan. 2013. Yank: Enabling green data centers to pull the plug. In *USENIX NSDI*.

[71] Jared Spataro. 2023. Microsoft 365 Copilot. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/. (2023).

[72] statista. 2024. Average levelized PPA wind power prices in the U.S. from 2008 to 2022. https://www.statista.com/statistics/217841/us-cumulative-capacity-weighted-average-wind-power-price/. (2024).

[73] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2025. Dynamollm: Designing llm inference clusters for performance and energy efficiency. In *IEEE HPCA*.

[74] Jinghan Sun, Zibo Gong, Anup Agarwal, Shadi Noghabi, Ranveer Chandra, Marc Snir, and Jian Huang. 2024. Exploring the Efficiency of Renewable Energy-based Modular Data Centers at Scale. In *ACM SoCC*.

[75] Michael Terrell. 2024. New nuclear clean energy agreement with Kairos Power. https://blog.google/outreach-initiatives/sustainability/google-kairos-power-nuclear-energy-agreement/. (2024).

[76] Thunder Said Energy. 2024. US electric utilities: transmission and distribution costs? https://thundersaidenergy.com/downloads/us-electric-utilities-transmission-and-distribution-costs/. (2024).

[77] US Dept. of Energy. 2022. Advantages and Challenges of Wind Energy. https://www.energy.gov/eere/wind/advantages-and-challenges-wind-energy. (2022).

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). http://arxiv.org/abs/1706.03762

[79] Wikipedia. 2025. CPLEX. https://en.wikipedia.org/wiki/CPLEX. (2025).

[80] Wikipedia. 2025. Energy mix. https://en.wikipedia.org/wiki/Energy_mix. (2025).

[81] Wikipedia. 2025. Hero of Alexandria. https://en.wikipedia.org/wiki/Hero_of_Alexandria. (2025).

[82] Wikipedia. 2025. Software-defined networking. https://en.wikipedia.org/wiki/Software-defined_networking. (2025).

[83] Wikipedia. 2025. Wind power in the United Kingdom: Constraint payments. https://en.wikipedia.org/wiki/Wind_power_in_the_United_Kingdom?#Constraint_payments. (2025).

[84] Fan Yang and Andrew A Chien. 2016. ZCCloud: Exploring wasted green power for high-performance computing. In *IEEE IPDPS*.

[85] Zenodo. 2021. EMHIRES dataset: wind and solar power generation. https://zenodo.org/records/8340501. (2021).

[86] Mary Zhang. 2023. Data Center Maintenance: A Comprehensive Guide. https://dgtlinfra.com/data-center-maintenance/. (2023).

[87] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. *arXiv:2401.09670* (2024).