5 Energy-Aware Model Architecture and Inference Systems

Ishita Verma*
Senior AI Research Engineer, Netflix, Inc.
*Corresponding Author: ishita.verma178@gmail.com

Abstract

For any model that reaches production, the energy spent answering questions overtakes the energy spent learning to answer them within weeks. This chapter treats inference as the primary sustainability problem in applied machine learning. It sets out one accounting identity for the carbon cost of a query, shows which engineering levers move which term of that identity, and argues that the order in which you pull those levers matters more than any single technique. The chapter closes with the multi-objective view, in which quality, latency and watts define a frontier to choose from, and with the measurement discipline needed to make any of these claims credible.

Keywords: Energy-Aware Model, Machine Learning, Carbon Cost, Primary Sustainability, Multi-Objective View.

Introduction

The Inference-Dominant Era

For most of the last decade, the public conversation about the environmental cost of machine learning has been a conversation about training. Strubell, Ganesh and McCallum's estimate of the emissions from training a large transformer became one of the most cited figures in the field largely because it was a number you could put in a headline.

For a deployed system it is the wrong number. The comparison that matters is not one training run against one query, but one training run against the integral of all queries over the model's service life. That integral grows without bound while the training cost stays fixed, so the only interesting question is how fast the crossing arrives.

It arrives fast. Take a 70-billion-parameter model trained on roughly 1.7 million accelerator-hours, which is in line with published figures for models of that class (Patterson and colleagues) and which puts the training run near 800 MWh once facility overheads are counted. Suppose each user query costs on the order of

1 Wh to serve. Figure 1 plots cumulative serving energy against that fixed training footprint at three deployment scales.

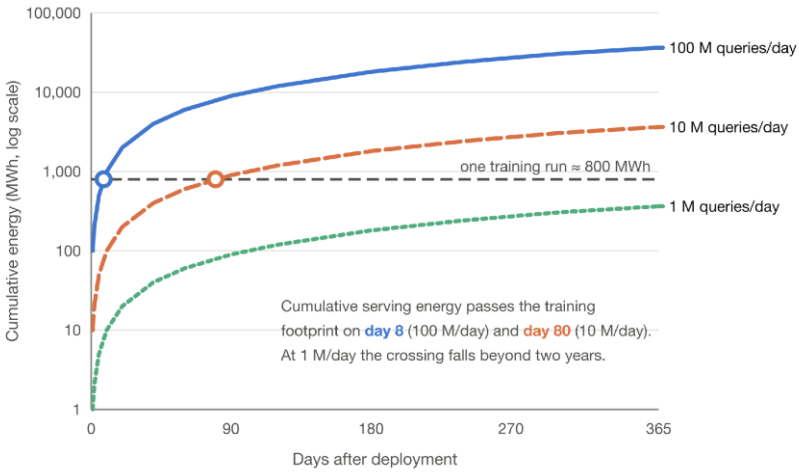


Fig. 1: Cumulative serving energy against days after deployment at three traffic volumes, on a logarithmic scale

In Fig. 1 the dashed line marks the roughly 800 MWh consumed by one training run of a 70-B-parameter model. At 100 million queries per day the crossing happens on day 8; at 10 million per day, on day 80. Published estimates for LLM queries (Luccioni, Jernite and Strubell) span roughly 0.3–3 Wh depending on response length, model size and batching efficiency, which moves the crossing by a factor of three either way without changing its character. Three consequences follow from this picture, and they organise the rest of the chapter.

The first is that the crossing arrives too early to treat as a later problem. Under a week at hyperscale.

The second is that slope matters more than intercept. Once past the crossing, lifetime footprint is determined entirely by joules per query. A 30 % improvement in training efficiency saves 240 MWh once. A 30 % improvement in serving efficiency, at 10 million queries per day, saves that much every 80 days for as long as the model runs.

The third is less comfortable. Efficiency gains get spent. Every technique in this chapter lowers the cost per query, and lower cost per query has reliably produced more queries: longer contexts, more reasoning tokens, agentic loops that issue dozens of calls where a user once issued one. This is the Jevons paradox.

What a Query Actually Costs

ISBN: 978-93-49468-03-0

Sustainability discussions usually go wrong at the level of units. “More efficient” can mean fewer parameters, fewer FLOPs, fewer joules, fewer grams of CO_2-equivalent, or lower dollars, and the conversion factors between these vary by an order of magnitude across deployments. Before optimising anything, fix the identity.

- **The Emissions Identity**

For a single served request:

$$\mathrm{gCO_2e\ per\ query} = \underbrace{E_{\mathrm{device}}}_{\text{kWh at the accelerator}} \times \underbrace{\mathrm{PUE}}_{\text{facility overhead}} \times \underbrace{I_{\mathrm{grid}}}_{\mathrm{gCO_2e\ per\ kWh}}$$

Three terms, and different engineering disciplines move different ones. Model compression and serving-runtime work reduce. Facility engineering reduces PUE. Carbon-aware scheduling changes and nothing else; it does not save a single joule.

Fig. 2 maps the four engineering layers onto the terms they move. It also serves as the map for sections 1.3 to 1.6.

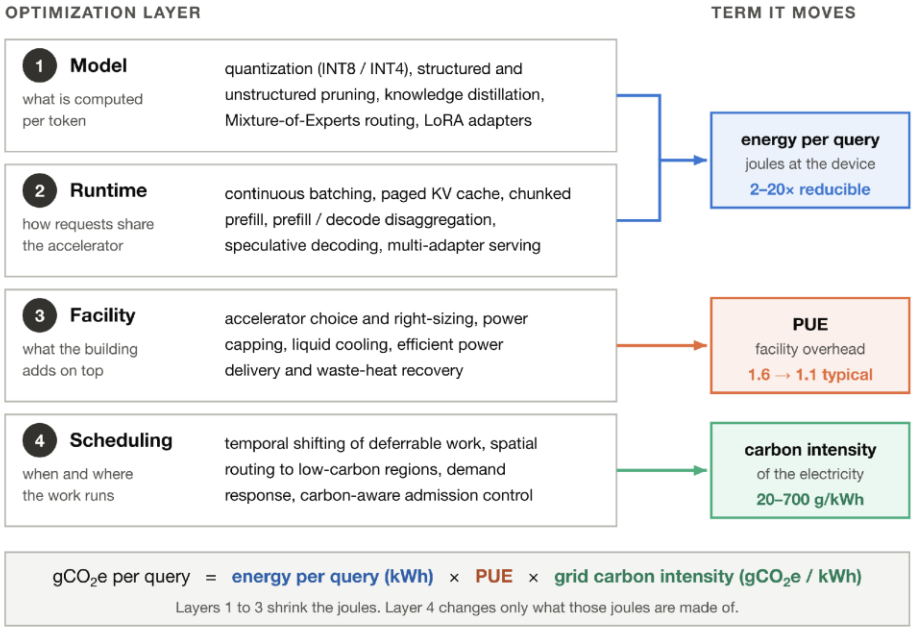


Fig. 2: The four optimization layers and the term of the emissions identity each one changes. Layers 1 to 3 reduce the joules consumed. Layer 4 changes only what those joules are made of. Both are legitimate. Reporting them as the same thing is not

ISBN: 978-93-49468-03-0

- **A Worked Example**

Abstract identities are easy to nod along to and hard to act on, so Table 1 carries a single query from accelerator draw through to grams of CO_2-equivalent. The scenario is a 70-B-parameter model quantized to INT8, served on one 8×H100 node under continuous batching, producing a 500-token response.

Table 1: From Accelerator Watts to Grams of CO_2e, for one 500-token Response

#	Quantity	Value
1	Accelerator draw under sustained load	4.96 kW
3	Node draw	5.86 kW
5	Device energy per 1,000 output tokens	3,781 J = 1.05 Wh
7	× PUE 1.12 (efficient hyperscale facility)	0.588 Wh
8	× 380 gCO_2e/kWh (US average grid)	0.223 gCO_2e
9	Same query on a 25 gCO_2e/kWh grid	0.015 gCO_2e
10	Annualised at 10 M queries/day, average grid	about 815 tCO_2e/yr

Two things stand out. Rows 8 and 9 differ by a factor of fifteen for identical computation, so the grid term dominates the arithmetic, which is what section 1.6 is about. And row 10 is a real number: 815 tonnes a year is roughly the annual footprint of 180 average US passenger cars, for one model, at a traffic level many mid-sized products exceed.

- **Why Utilization Dominates**

Accelerators are not energy-proportional. A GPU at 10 % utilization does not draw 10 % of its peak power. Depending on the part and the workload it may draw 30 to 50 %, because of the static power floor from HBM refresh, clock distribution and leakage. Barroso and Hölzle made this argument for servers in 2007 and it has only become sharper with accelerators.

Layer 1: The Model, or what gets Computed per Token

- **Quantization**

Quantization reduces the numerical precision of weights, activations, or both. It is the highest-return technique in the catalogue, for a reason worth understanding rather than just memorising: large-model inference is usually bound by memory bandwidth, not by compute. Halving the bytes per weight therefore roughly halves the time and energy per token, almost independently of how fast the arithmetic units are.

INT8 with 8-bit activations (W8A8) is close to free. With per-channel scaling and attention to activation outliers, an insight behind LLM.int8() (Dettmers and colleagues), both of which observed that a handful of channels carry disproportionately large activations and wreck naive per-tensor scaling, INT8 post-training quantization typically costs well under one point on standard benchmarks while halving weight memory relative to FP16.

ISBN: 978-93-49468-03-0

INT4 weight-only schemes (GPTQ, Frantar and Alistarh ; NF4, Dettmers and colleagues) push to roughly a quarter of FP16 weight memory. The quality cost is real but modest, commonly one to three points, concentrated in tasks that require precise recall or arithmetic. The throughput benefit is largest exactly where it matters most, in the memory-bound low-batch decode regime.

- **Pruning, and Why the Structured Distinction Matters**

Pruning removes parameters. The distinction that matters operationally is not how many you remove but whether the resulting sparsity is one your hardware can exploit.

Unstructured pruning zeroes individual weights wherever they are least important. It achieves the highest sparsity for a given quality budget; SparseGPT (Frantar and Alistarh) demonstrated one-shot sparsity above 50 % on very large models with modest degradation. On conventional dense matrix kernels it delivers no speedup at all.

N:M semi-structured sparsity, typically 2:4, meaning two zeros in every group of four, is the compromise that modern accelerators support directly in hardware. It gives roughly 2× on weight storage and meaningful throughput gains on supported layers, usually at the cost of a retraining pass.

- **Knowledge Distillation**

Distillation trains a small student model to reproduce the behaviour of a large teacher, matching not just its answers but its output distribution, which carries far more information per example than a hard label. Hinton and colleagues framed this in 2015 as transferring dark knowledge [8]. DistilBERT later became the canonical demonstration, retaining roughly 97 % of BERT's benchmark performance at 40 % fewer parameters and 60 % faster inference.

Distillation differs from quantization and pruning in a way that is easy to underestimate.

- **Mixture-of-Experts and Conditional Computation**

MoE architectures (Shazeer and colleagues ; Fedus, Zoph and Shazeer) replace a dense feed-forward block with many parallel expert blocks plus a learned router that activates only a small subset per token. A model with 47 billion total parameters might activate 13 billion for any given token, paying roughly the compute of a 13-B model while retaining much of the capacity of a far larger one.

The energy argument is sound and the accounting has a catch. FLOPs per token fall substantially. Memory does not. Every expert must be resident in accelerator memory in case the router selects it, so an MoE model's memory footprint tracks its total parameter count while its compute tracks its active count. On

ISBN: 978-93-49468-03-0

a decode workload bound by memory bandwidth, that is an awkward combination: you have optimised the term that was not the bottleneck.

- **LoRA and Serving Many Behaviours from One Base**

Low-rank adaptation (Hu and colleagues) freezes the base model and learns a small low-rank update per task, typically a fraction of a percent of the base parameters. For serving, the significant property is not the training efficiency that made LoRA famous but a deployment consequence: you can serve many specialised behaviours from one resident copy of the weights.

The alternative is a fully fine-tuned model per customer, per task or per language, which multiplies your memory footprint by the number of variants and fragments your batches, since requests for different models cannot batch together. Systems such as S-LoRA (Sheng and colleagues) have demonstrated thousands of concurrent adapters against a single base.

- **Speculative Decoding**

Speculative decoding deserves separate treatment because it breaks the pattern of everything above. A small, fast draft model proposes several tokens (Leviathan, Kalman and Matias). The large model verifies them all in a single forward pass. A rejection-sampling step accepts the longest correct prefix.

Implemented correctly, the output distribution is mathematically identical to that of the large model alone. It is the only technique in this chapter that is free in quality terms. Reported speedups commonly fall in the 1.5–3× range, depending on how well the draft model's distribution matches the target's.

Table 2: Efficiency levers compared. Ranges are representative of published results and production practice. They vary widely with model, hardware and workload, and should be read as orders of magnitude rather than specifications.

Lever	Memory Effect	Throughput / Latency Effect	Typical Quality Cost
INT8 PTQ (W8A8)	about 2× smaller than FP16	1.5–2×	under 1 point
INT4 weight-only	about 4× smaller than FP16	1.5–3× at low batch, less at high batch	1–3 points
Unstructured pruning	50–90 % zeros on disk	about 1× on dense kernels	small at moderate sparsity
2:4 semi-structured sparsity	about 2× on weights	1.2–1.6× on supported layers	0–2 points
Knowledge distillation	2–10× smaller	proportional to size	3–12 points, task-dependent

ISBN: 978-93-49468-03-0

Mixture-of-Experts	larger total, smaller active	2–5× fewer FLOPs per token	near zero against a dense model of equal active size
LoRA adapters (multi-tenant)	0.1–1 % of base per task	about 1× per request, large win in aggregate	near zero on the adapted task
Speculative decoding	plus a draft model	1.5–3× wall clock	none, with rejection sampling
Continuous batching	negligible	2–10× throughput at fixed hardware	none
Paged KV cache	2–4× more concurrent requests	follows from occupancy	none
Carbon-aware scheduling	n/a	n/a	none

Layer 2: The Runtime, or how Requests Share the Accelerator

- **The Naive Server Wastes Most of the GPU**

A straightforward inference server accepts a request, runs it to completion, returns the result. At batch size one, a large model's accelerator utilization during decoding is poor, often in the single-digit percentages of peak FLOPs,

- **Static, Dynamic and Continuous Batching**

Static batching collects N requests, runs them together, returns them together. The problem appears the moment you consider that generation lengths vary widely. A request that finishes in 20 tokens occupies its slot until the request generating 800 tokens is done.

Continuous batching, introduced as iteration-level scheduling in the Orca system (Yu and colleagues) and now standard in vLLM, TensorRT-LLM and TGI, dissolves the batch as a unit. Panel (a) of Fig. 3 shows the difference on identical work.

- **The KV Cache is the Real Memory Bottleneck**

Every token attends to all previous tokens, so a serving system caches the key and value tensors for the entire context. This cache grows linearly with sequence length and batch size, and for long contexts it can exceed the size of the model weights.

The naive implementation reserves, per request, a contiguous buffer sized for the maximum sequence length the system supports, Measured internal fragmentation in pre-paging systems commonly ran 60 to 80 %.

PagedAttention, introduced with vLLM (Kwon and colleagues), applies the insight that operating systems solved this problem in the 1960s. Divide the cache into

ISBN: 978-93-49468-03-0

fixed-size blocks. Allocate blocks on demand as the sequence grows. Fragmentation collapses to a few percent, and the memory recovered translates directly into more concurrent requests, which translates into utilization, which translates into joules per query. Panel (b) of Figure 1.3 shows the same memory pool under both schemes.

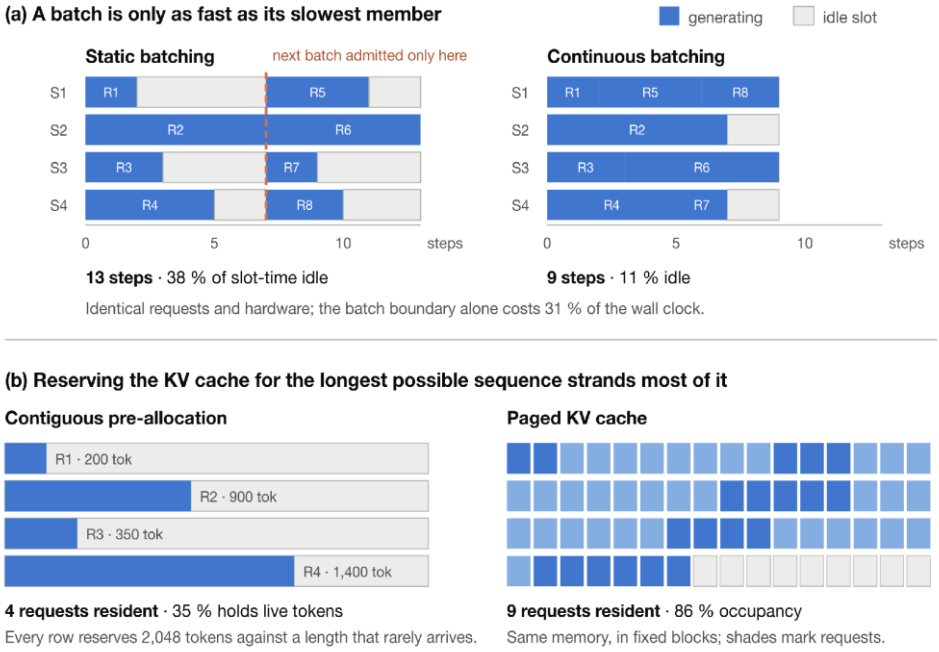


Fig. 3: Two runtime mechanisms, each recovering capacity that the naive implementation throws away.

In Fig. 3 panel (a), static and continuous batching are compared on eight requests across four slots. Under static batching, slots that finish early wait for the batch boundary, leaving 38 % of slot-time idle and stretching the work across 13 steps. Under continuous batching a waiting request enters the moment a slot frees, finishing in 9 steps with 11 % idle. Panel (b) compares KV-cache allocation. Reserving the maximum sequence length per request fills the pool with four requests while only 35 % of it holds live tokens. Paged allocation in fixed blocks holds nine requests in the same memory at 86 % occupancy.

- **Chunked Prefill and Prefill/Decode Disaggregation**

Transformer inference has two phases with opposite hardware characteristics. Prefill processes the whole input prompt in parallel and is compute-bound, saturating the arithmetic units. Decode generates tokens one at a time, is bound by memory bandwidth, and leaves arithmetic units idle.

ISBN: 978-93-49468-03-0

Two families of answer have emerged. Chunked prefill, as in Sarathi-Serve (Agrawal and colleagues) and related work, splits long prefills into pieces and interleaves them with ongoing decode steps, smoothing the latency spikes that a long prompt otherwise inflicts on every other request in the batch.

Layer 3: The Facility, or what the Building Adds

The facility layer is largely outside the control of a model team,

Power Usage Effectiveness is total facility energy divided by IT equipment energy. A PUE of 1.5 means every watt delivered to a server costs another half-watt in cooling, power conversion and lighting. Industry-wide surveys (Uptime Institute) have put the average large data centre near 1.5 to 1.6 for years, with limited movement. Well-run hyperscale facilities operate near 1.1, and the best report lower.

Within the facility, three levers matter for AI workloads specifically. Liquid cooling is increasingly a requirement rather than an optimization, as accelerator board power passes 700 W and air cooling reaches its physical limits. Power capping exploits the fact that accelerators can often be held at 70 to 80 % of board power for a much smaller proportional throughput loss.

Layer 4: Scheduling, or When and Where the Work Runs

Everything so far reduced joules. This layer does not. Carbon-aware scheduling exploits the fact that grid carbon intensity, the third term of the identity, varies by more than an order of magnitude across regions and by a factor of two or three within a single region across a day. Identical computation can emit fifteen times more carbon in one place than another, as rows 8 and 9 of Table 1.1 showed.

- **Temporal and Spatial Shifting**

Temporal shifting defers flexible work to hours when the local grid is cleaner, typically midday in solar-heavy grids and overnight in wind-heavy ones. Google's carbon-intelligent computing platform (Radovanović and colleagues) is the best-documented production example, shifting deferrable batch workloads within a day against a forecast of grid carbon intensity.

- **What is actually shiftable**

This is where honest engineering diverges sharply from the marketing version. Interactive inference, with a user waiting for a response, is not shiftable. In a typical AI platform, the work that is shiftable includes:

- **Marginal versus Average Intensity, and Additionality**

Two accounting subtleties routinely inflate reported savings. The first is average versus marginal intensity. The average carbon intensity of a grid describes the mix currently generating. The marginal intensity describes what generator responds to one additional unit of demand, which is often a fossil peaker even on a

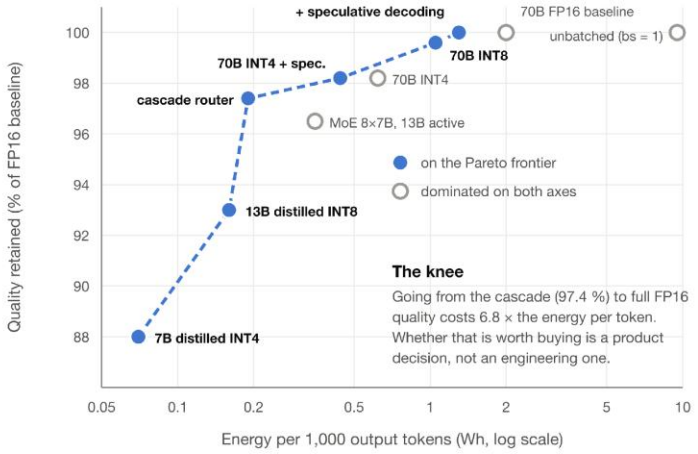
ISBN: 978-93-49468-03-0

grid with a clean average. Marginal intensity is the decision-relevant signal and the harder one to obtain.

The second is additionality. Purchasing renewable energy certificates to offset consumption is not the same as consuming clean electricity, and neither is the same as causing additional clean generation to exist. The strongest claim available to most operators is 24/7 carbon-free energy matching, where consumption is matched to clean generation hour by hour on the same grid.

The Pareto Frontier and Choosing an Operating Point

Fig.4 plots ten serving configurations for a hypothetical deployment. Six lie on the Pareto frontier. Four are dominated and should never be chosen, whatever your preferences.



Illustrative composite values assembled from published ranges; not measurements of a single deployed system.

Fig. 4: Quality retained against energy per 1,000 output tokens for ten serving configurations

In Fig.4 the filled points are Pareto-optimal. Open points are dominated on both axes by some other configuration. Note the FP16 baseline, which is dominated by the same model with speculative decoding: identical output for 35 % less energy, at no cost. Note also the unbatched configuration at 9.5 Wh, an order of magnitude off the frontier, and recall that this is the position a naive serving stack occupies regardless of how well the model is compressed. Three lessons follow from the figure.

Dominated configurations are common and they are pure waste. The FP16 baseline is dominated by the same model with speculative decoding, which produces

ISBN: 978-93-49468-03-0

mathematically identical output for substantially less energy. Eliminating them requires no product judgement at all, so do it before tuning any preference weights.

The frontier has a knee, and the knee is where the product conversation lives. Going from the cascade at 97.4 % quality to full FP16 quality costs 6.8× the energy per token. Whether the last 2.6 points are worth seven times the energy is not an engineering question. An engineer's job here is to present the frontier accurately, and a product owner's job is to choose a point on it.

Model cascades and routing. Send every request to a small model first and escalate to the large model only when a confidence signal, a verifier or a learned router says the small model's answer is inadequate. If the small model handles 80 % of traffic adequately, you pay large-model energy on 20 % of requests.

Measuring Honestly

Most published efficiency claims are not comparable with one another, and many are not reproducible. If the techniques in this chapter are to be adopted on evidence rather than fashion, the measurement discipline needs to match the engineering.

- **Report the denominator.** "40 % more efficient" means nothing without a unit. Joules per token, per request, or per successfully completed task? The last is the only one that resists gaming, because a model that is cheap per token but needs three attempts is not cheap.

Report the operating point, not the peak. Throughput benchmarks are routinely run at batch sizes no production system uses, with no latency constraint, on warm caches, with input and output lengths chosen to flatter the system under test. Report throughput at a stated latency SLO or do not report it.

Measure power rather than estimating it from TDP. Thermal design power is a cooling specification, not a consumption measurement. MLPerf Inference's power measurement methodology (Reddi and colleagues) is the most mature public standard and is worth adopting even if you never submit results.

- **State your carbon conventions.** Location-based or market-based? Average or marginal grid intensity? Does the figure include embodied carbon, PUE, networking? Two organisations can report numbers differing by 5× while both remain technically defensible.

What Remains Unsolved

Rebound effects. Per-query efficiency has improved by orders of magnitude over the past few years. Aggregate consumption has risen throughout. Cheaper inference produces longer contexts, more reasoning tokens, and agentic architectures issuing dozens of model calls where a user once issued one. The

ISBN: 978-93-49468-03-0

binding constraint is ultimately a question about what we choose to compute, which is not an engineering question.

- **Embodied carbon is becoming the larger share.** As operational electricity gets cleaner and accelerator refresh cycles stay short, the manufacturing footprint of the hardware grows as a fraction of the total. The data needed to account for it properly is largely held by manufacturers and largely not published.
- **Quality measurement is the weak link.** Every claim of the form "X % energy reduction for Y % quality loss" is only as good as the measurement of Y, and our ability to measure the quality of generative systems remains poor. The efficiency literature has raced ahead of the evaluation literature it depends on.

Results

- **Inference dominates.** For any deployed model, cumulative serving energy passes the training footprint within days to months, and then keeps going. Optimise the slope, not the intercept.
- **Use one identity.** gCO_2e per query = energy × PUE × grid intensity. Know which term each technique moves, and never report a change in the third as though it were a change in the first.
- **Utilization first, compression second.** Accelerators are not energy-proportional, and an idle GPU still burns a third of its peak power. Continuous batching and paged KV caches typically recover more energy than any compression technique, and they cost nothing in quality.
- **Know what each model-level lever actually buys.** Quantization wins because decoding is memory-bound. Unstructured pruning buys nothing without sparse kernels. MoE reduces FLOPs but not memory. Distillation trades generality for efficiency. Speculative decoding is the only one that is free.
- **Eliminate dominated configurations before tuning preferences.** A large fraction of deployed systems sit strictly inside the frontier, most often because of the serving stack rather than the model.
- **The knee of the frontier is a product decision.** Present it honestly, let the people who own the user experience choose the point, then make the point dynamic.
- **Measure like it matters.** Report the denominator, the latency SLO, measured power rather than TDP, quality on real traffic, and your carbon accounting conventions.

ISBN: 978-93-49468-03-0

References

1. Agrawal, A. *et al.* (2024) 'Sarathi-Serve: Taming prefill-decode disaggregation in large language model serving', *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2024)*.
2. Barroso, L.A. and Hölzle, U. (2007) 'The case for energy-proportional computing', *IEEE Computer*, 40(12), pp. 33–37.
3. Dettmers, T. *et al.* (2022) 'LLM.int8(): 8-bit matrix multiplication for transformers at scale', *Advances in Neural Information Processing Systems (NeurIPS 2022)*.
4. Dettmers, T. *et al.* (2023) 'QLoRA: Efficient finetuning of quantized LLMs', *Advances in Neural Information Processing Systems (NeurIPS 2023)*.
5. Fedus, W., Zoph, B. and Shazeer, N. (2022) 'Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity', *Journal of Machine Learning Research*, 23, pp. 1–39.
6. Frantar, E. and Alistarh, D. (2023a) 'GPTQ: Accurate post-training quantization for generative pre-trained transformers', *International Conference on Learning Representations (ICLR 2023)*.
7. Frantar, E. and Alistarh, D. (2023b) 'SparseGPT: Massive language models can be accurately pruned in one-shot', *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*.
8. Hinton, G., Vinyals, O. and Dean, J. (2015) 'Distilling the knowledge in a neural network', *arXiv preprint* arXiv:1503.02531.
9. Hu, E.J. *et al.* (2022) 'LoRA: Low-rank adaptation of large language models', *International Conference on Learning Representations (ICLR 2022)*.
10. Kwon, W. *et al.* (2023) 'Efficient memory management for large language model serving with PagedAttention', *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP 2023)*.
11. Leviathan, Y., Kalman, M. and Matias, E. (2023) 'Fast inference from transformers via speculative decoding', *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*.
12. Luccioni, A.S., Jernite, Y. and Strubell, E. (2024) 'Power hungry processing', ACM FAccT.
13. Patterson, D. *et al.* (2021) 'Carbon emissions and large neural network training', *arXiv preprint* arXiv:2104.10350.
14. Radovanović, A. *et al.* (2023) 'Carbon-aware computing for datacenters', *IEEE Transactions on Power Systems*.

ISBN: 978-93-49468-03-0

15. Reddi, V.J. *et al.* (2020) ‘MLPerf inference benchmark’, *Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA 2020)*.
16. Shazeer, N. *et al.* (2017) ‘Outrageously large neural networks: The sparsely-gated mixture-of-experts layer’, *International Conference on Learning Representations (ICLR 2017)*.
17. Sheng, Y. et al. (2024) ‘S-LoRA’, MLSys.
18. Strubell, E., Ganesh, A. and McCallum, A. (2019) ‘Energy and policy considerations for deep learning in NLP’, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, pp. 3645–3650.
19. Uptime Institute (annual) *Global Data Center Survey*. Uptime Institute.
20. Yu, G. *et al.* (2022) ‘Orca: A distributed serving system for transformer-based generative models’, *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2022)*.

☯☯☯

ISBN: 978-93-49468-03-0