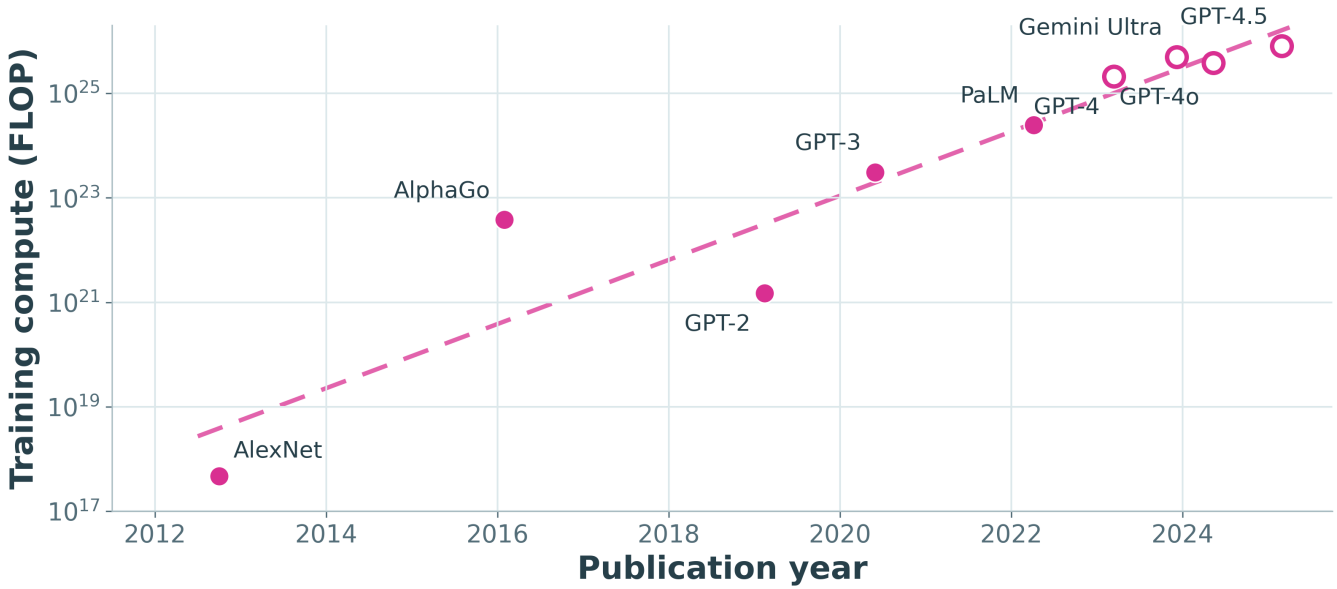
Carbon-Aware Dynamic Parallelism for Distributed Training

Wafik Aboualim Stephen Lee

University of Pittsburgh

Pretraining is Carbon Intensive



Open circles denote estimated compute for closed models. Source: Epoch AI Frontier Models.

- GPT-3 training is estimated to emit approximately 552 tCO_2e (Patterson et al. 2021).
- Meta reports 11,390 tCO_2e in location-based emissions for the Llama 3.1 family (Grattafiori et al. 2024).

As training scales, its carbon impact becomes increasingly significant.

Related Work

- **Temporal shifting**: run during lower-carbon periods, but pausing may require checkpointing and releasing accelerators (Wiesner et al. 2021).
- **Spatial shifting**: move training to a lower-carbon region, requiring multi-region infrastructure (Dodge et al. 2022).
- **Elastic scaling**: vary accelerator count, assuming resources can be released and reacquired (Hanafy et al. 2023).

Our setting

Time, location, and accelerator allocation remain fixed. We instead change how the workload is distributed.

Distributed Training 101

Why distribute training?

Training large models requires multiple accelerators working in parallel (Narayanan et al. 2021).

- **Data parallelism**: replicate the model and split the batch (Narayanan et al. 2021).
- **Tensor parallelism**: split computation within each layer (Shoeybi et al. 2019).
- **Pipeline parallelism**: split layers across stages (Huang et al. 2019).

The parallelism strategy affects both power and throughput.

Different Parallelism Dimensions

- **Tensor parallelism**: splits each layer across accelerators and communicates during forward and backward passes (Shoeybi et al. 2019).
- **Pipeline parallelism**: splits layers into stages and passes activations and gradients between stages for each microbatch (Huang et al. 2019).
- **Data parallelism**: processes different batch shards, then averages gradients during the backward pass (Narayanan et al. 2021).

Key idea

Each strategy creates a different compute and communication pattern.

Example

Example: 8 accelerators

$$TP = 2, \qquad PP = 2, \qquad DP = 2$$

- $TP = 2$: each layer is split across 2 accelerators.
- $PP = 2$: the model is split into 2 stages.
- $DP = 2$: two copies process different batch shards.

Constraint: $TP \times PP \times DP = 8$

What is Dynamic Parallelism?

Dynamic parallelism allows a training job to change its parallelism layout while it is running (Kang et al. 2025).

Intuition

Instead of choosing one fixed setup at launch, the system can switch between different ways of splitting work across accelerators.

Different layouts create different throughput and power profiles:

- **More data parallelism**: generally more compute-bound, but gradient synchronization grows at large scale.
- **More tensor parallelism**: generally communication-bound, because accelerators communicate within each layer.
- **More pipeline parallelism**: lies in between; too many stages increase pipeline bubbles, where accelerators remain idle.

Hardware Setup

We profile distributed training on fixed accelerator allocations.

- **Single node:** 4 NVIDIA L40 accelerators
- **Multi-node:** NVIDIA A100 nodes, 4 accelerators per node
- **Training stack:** Megatron Core / Megatron-LM (Narayanan et al. 2021)
- **Measurements:** throughput from training logs; power from NVML

For each setup, we sweep valid (TP, PP, DP) configurations.

Power Profiles Across Models

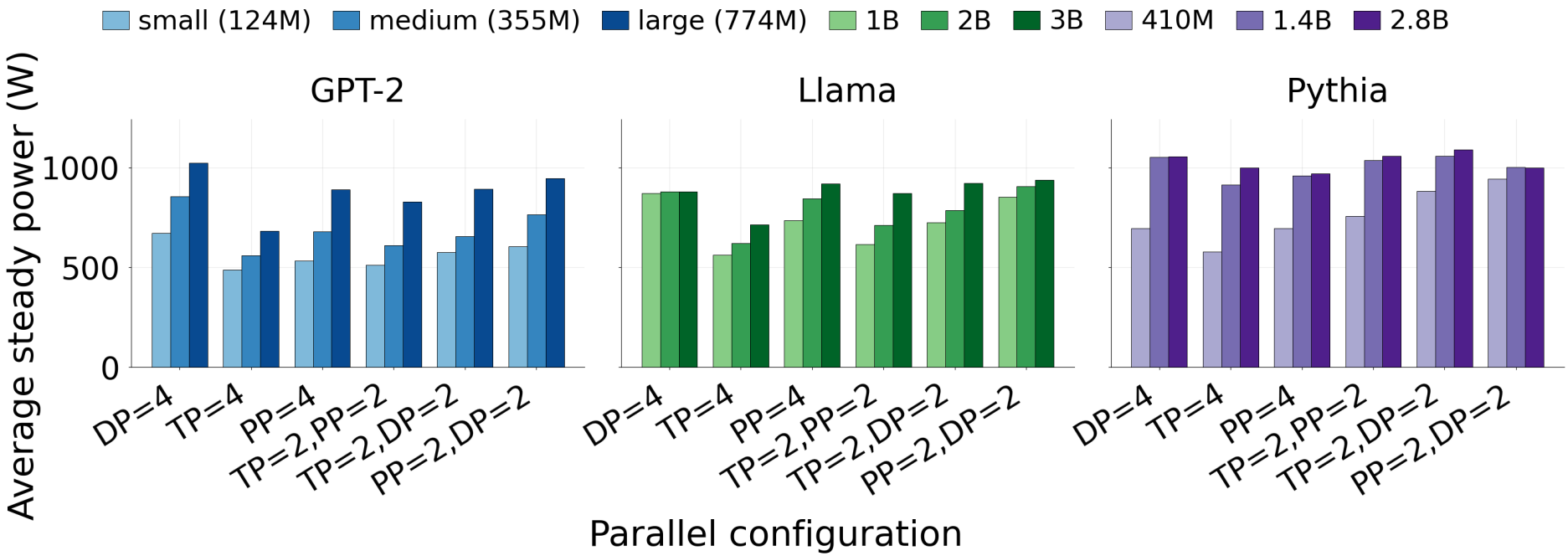


Figure 1: Average power draw across feasible parallel configurations.

Takeaway: Power draw depends not only on the parallel configuration, but also on the model family and architecture.

Power Profiles Across Models

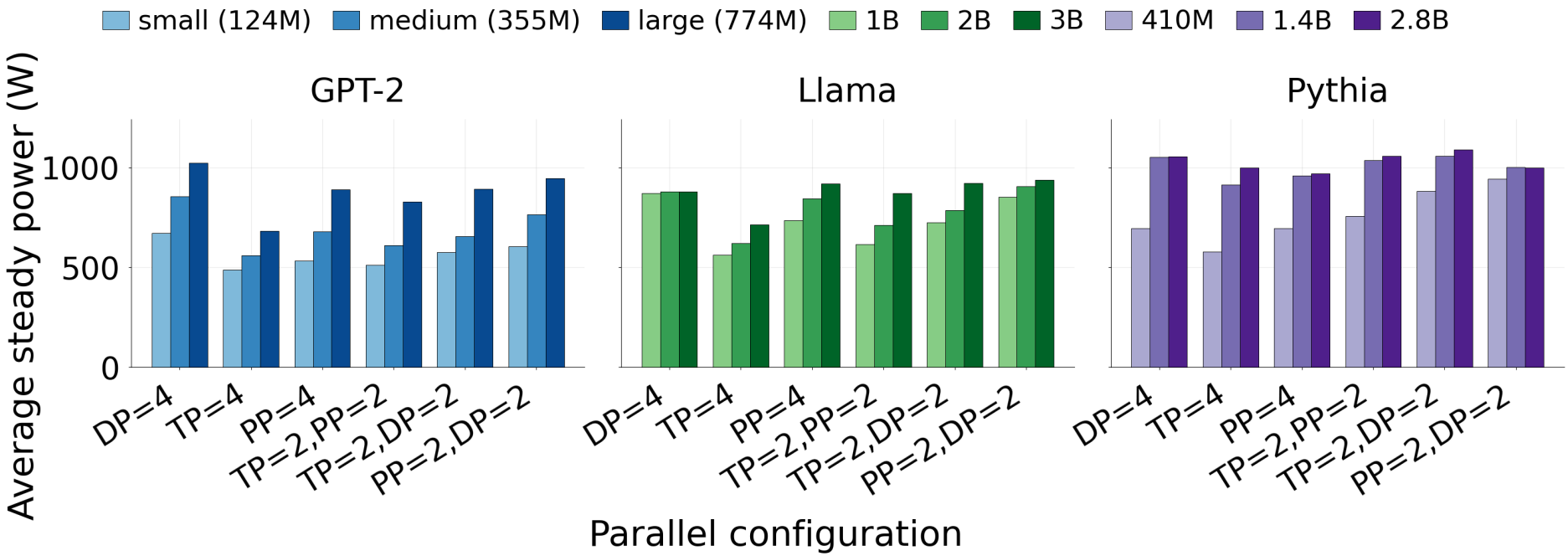


Figure 2: Average power draw across feasible parallel configurations.

Takeaway: Power draw depends not only on the parallel configuration, but also on the model family and architecture.

Power Profiles Across Models

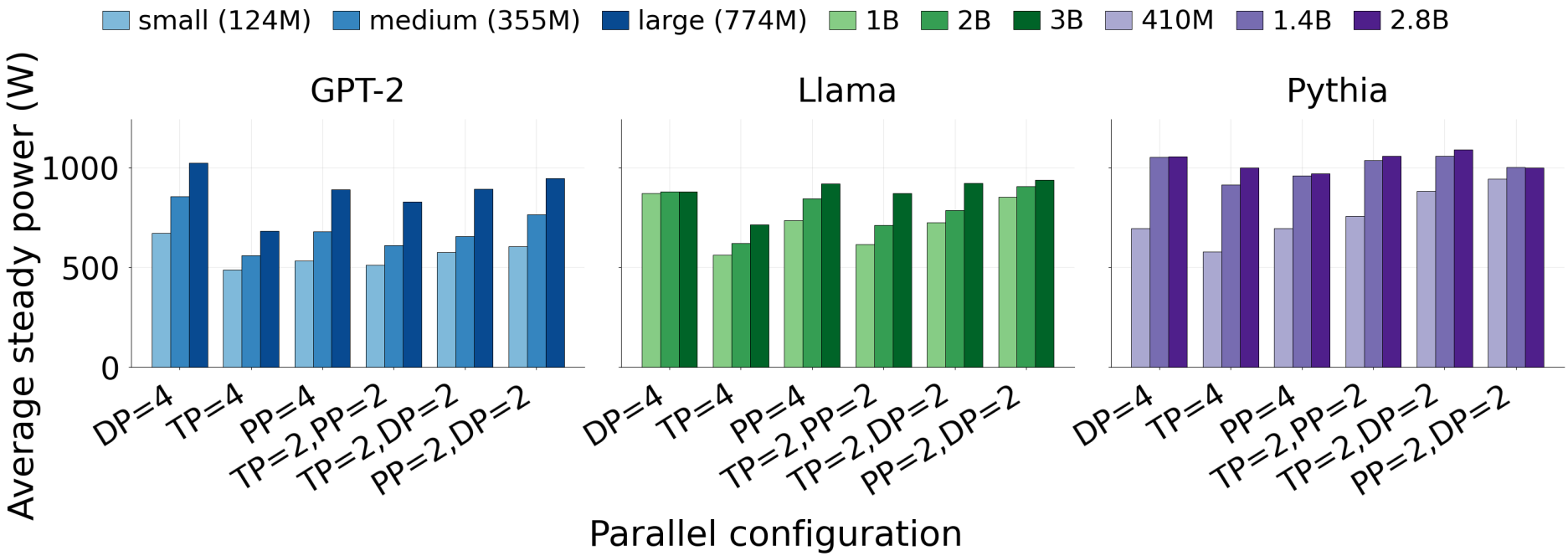


Figure 3: Average power draw across feasible parallel configurations.

Takeaway: Power draw depends not only on the parallel configuration, but also on the model family and architecture.

Throughput Profiles Across Models

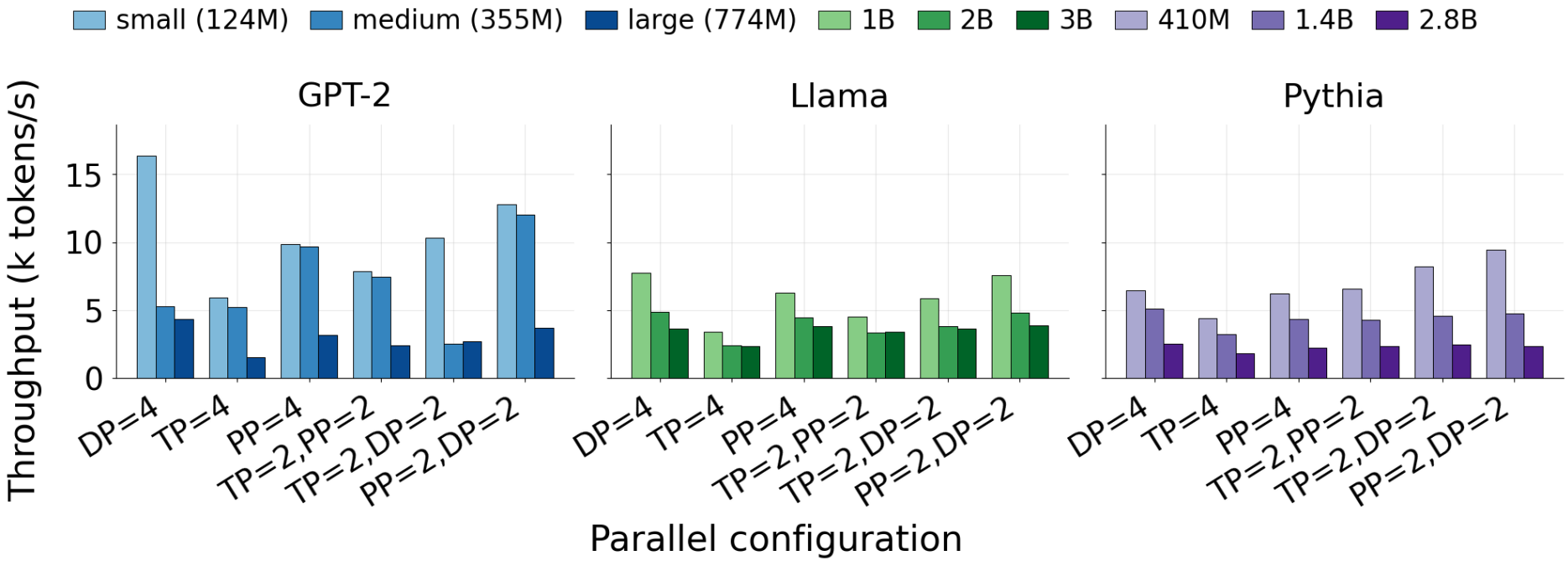


Figure 4: Throughput across feasible parallel configurations.

Takeaway: Throughput depends not only on the parallel configuration, but also on the model family and architecture.

Throughput Profiles Across Models

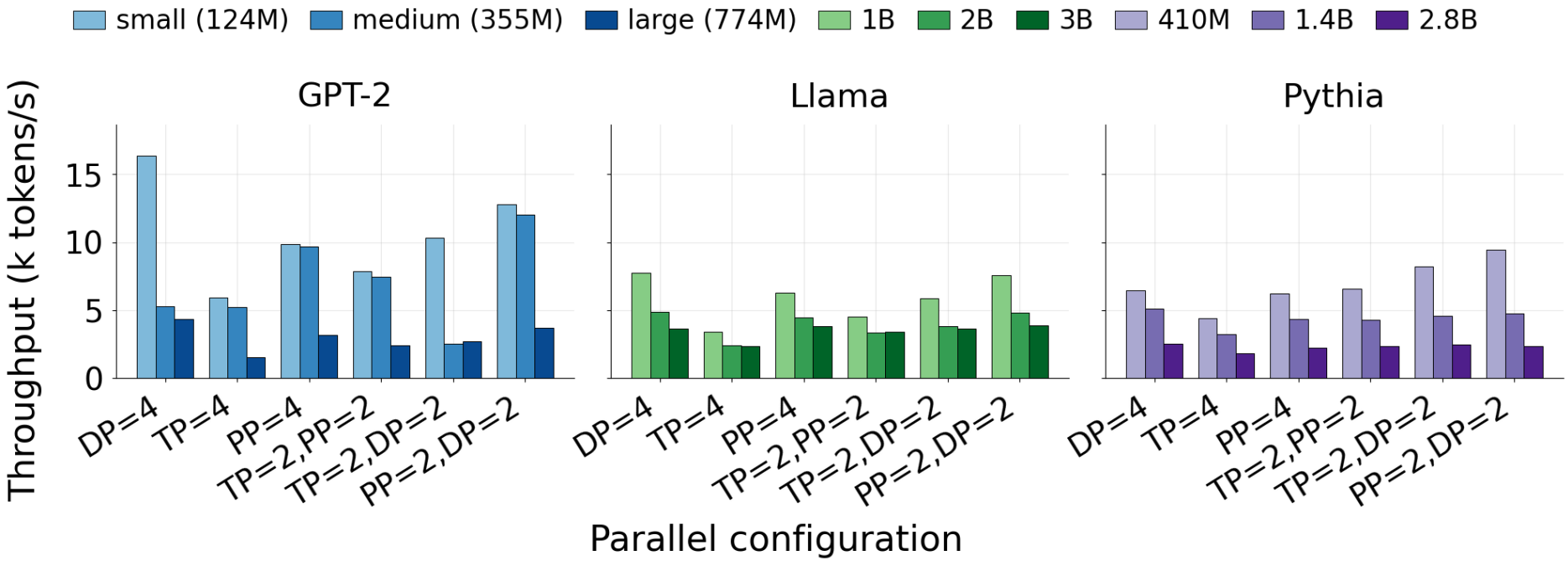


Figure 5: Throughput across feasible parallel configurations.

Takeaway: Throughput depends not only on the parallel configuration, but also on the model family and architecture.

Throughput Profiles Across Models

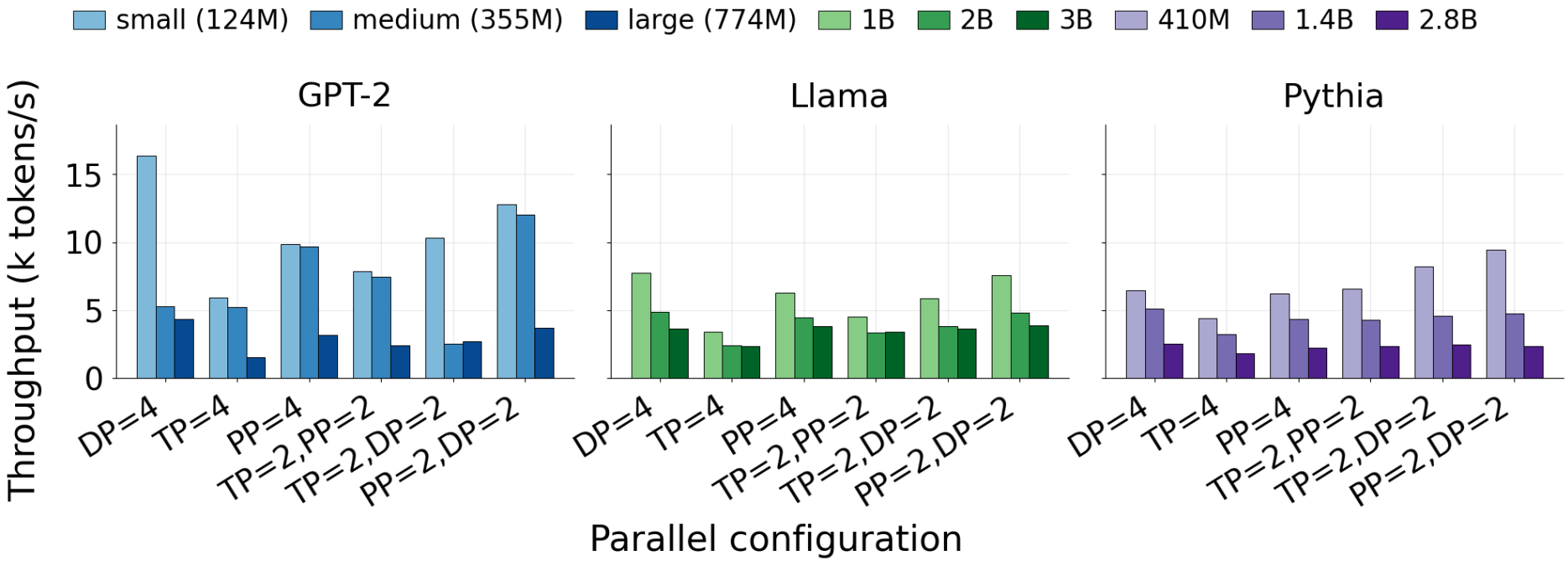


Figure 6: Throughput across feasible parallel configurations.

Takeaway: Throughput depends not only on the parallel configuration, but also on the model family and architecture.

Power Profiles Across Cluster Sizes

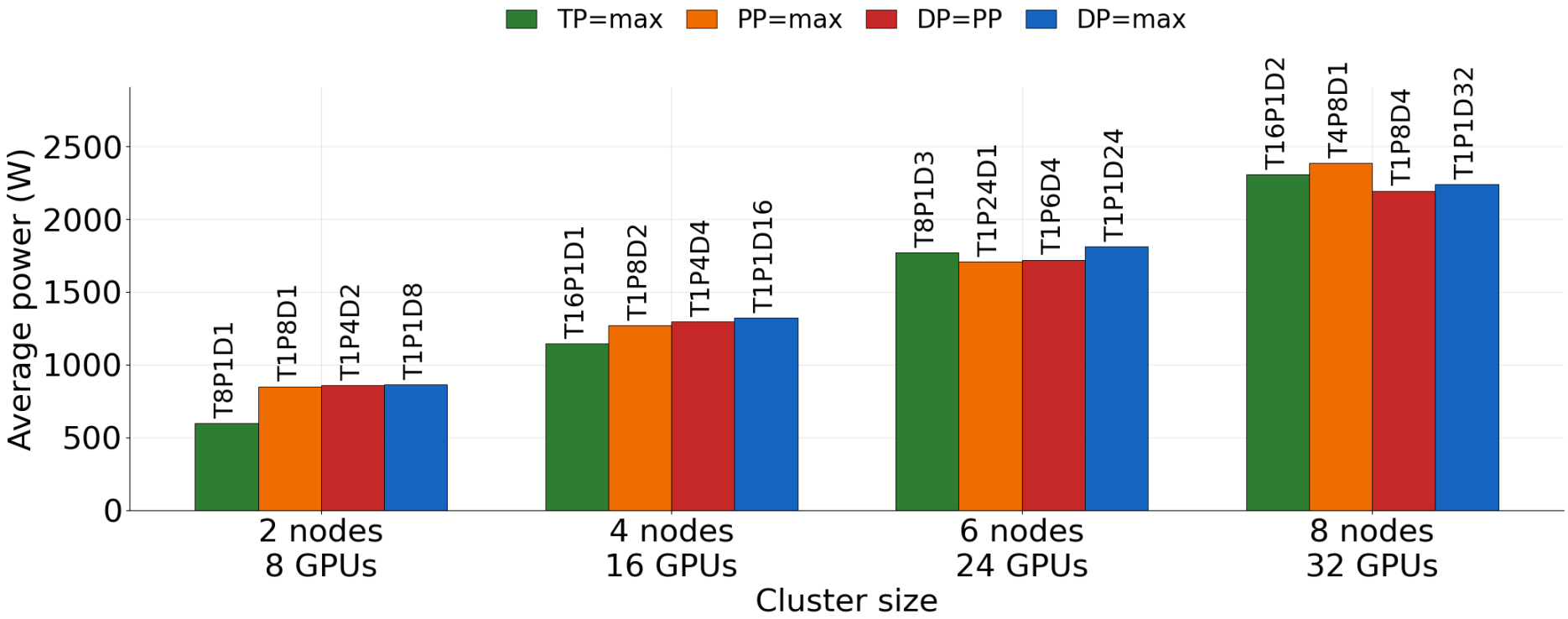


Figure 7: Power draw changes as the same strategy scales across cluster sizes.

Takeaway: Things become less predictable at larger scales due to factors such as pipeline bubbles, accelerator placement, and thermal effects.

Power Profiles Across Cluster Sizes

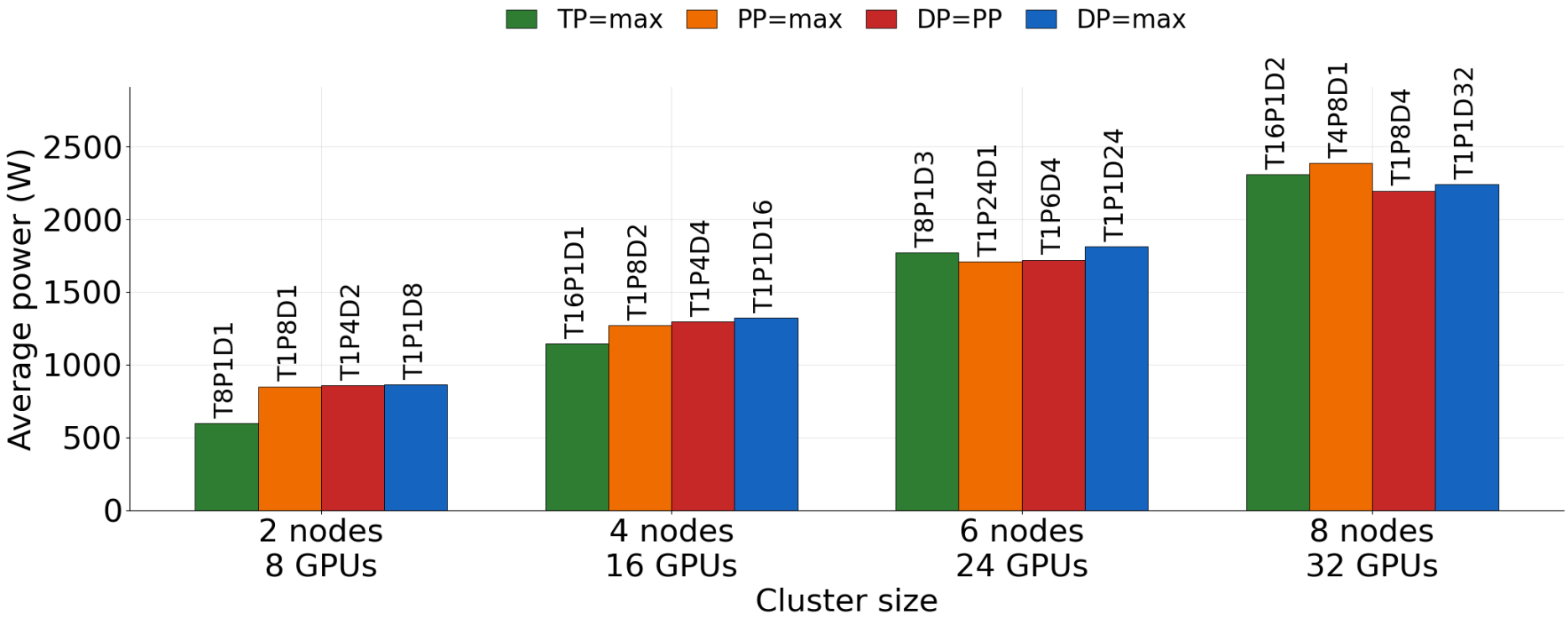


Figure 8: Power draw changes as the same strategy scales across cluster sizes.

Takeaway: Things become less predictable at larger scales due to factors such as pipeline bubbles, accelerator placement, and thermal effects.

Throughput Profiles Across Cluster Sizes

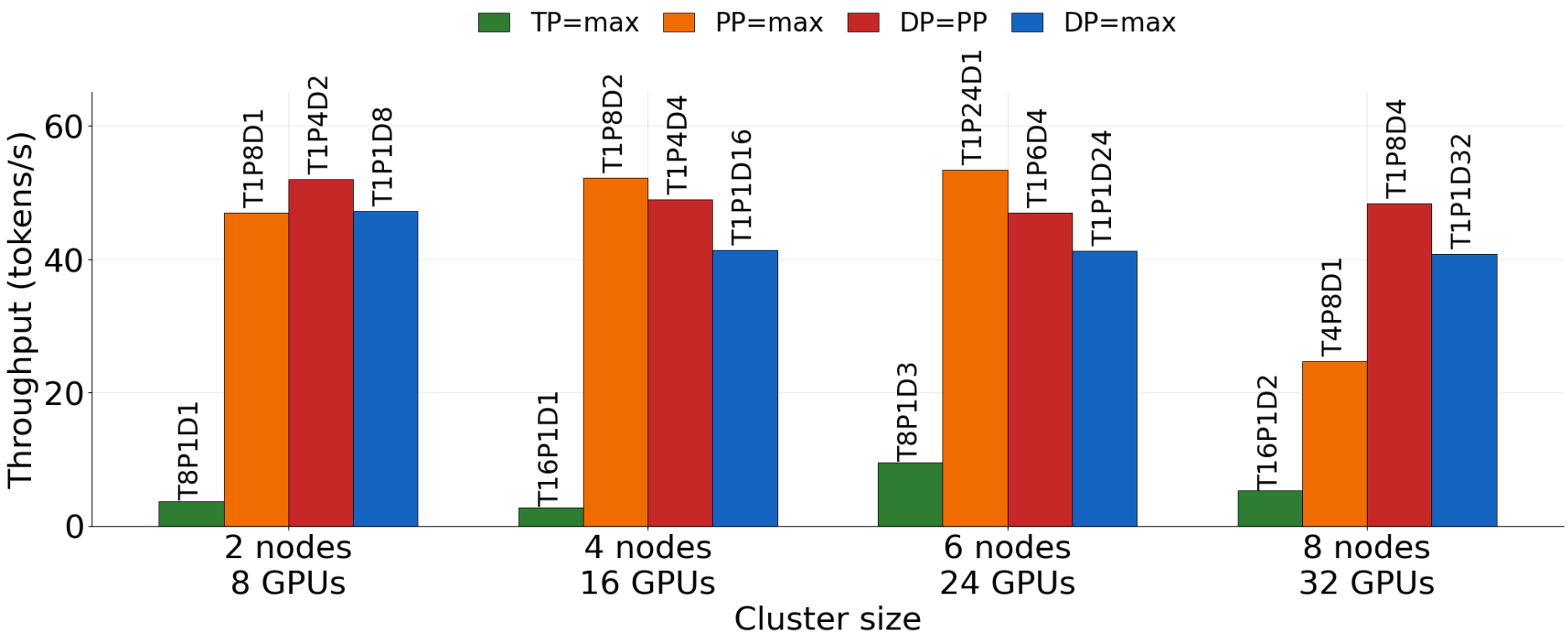


Figure 9: Throughput changes as the same strategy scales across cluster sizes.

Takeaway: Things become less predictable at larger scales due to factors such as pipeline bubbles, accelerator placement, and thermal effects.

Throughput Profiles Across Cluster Sizes

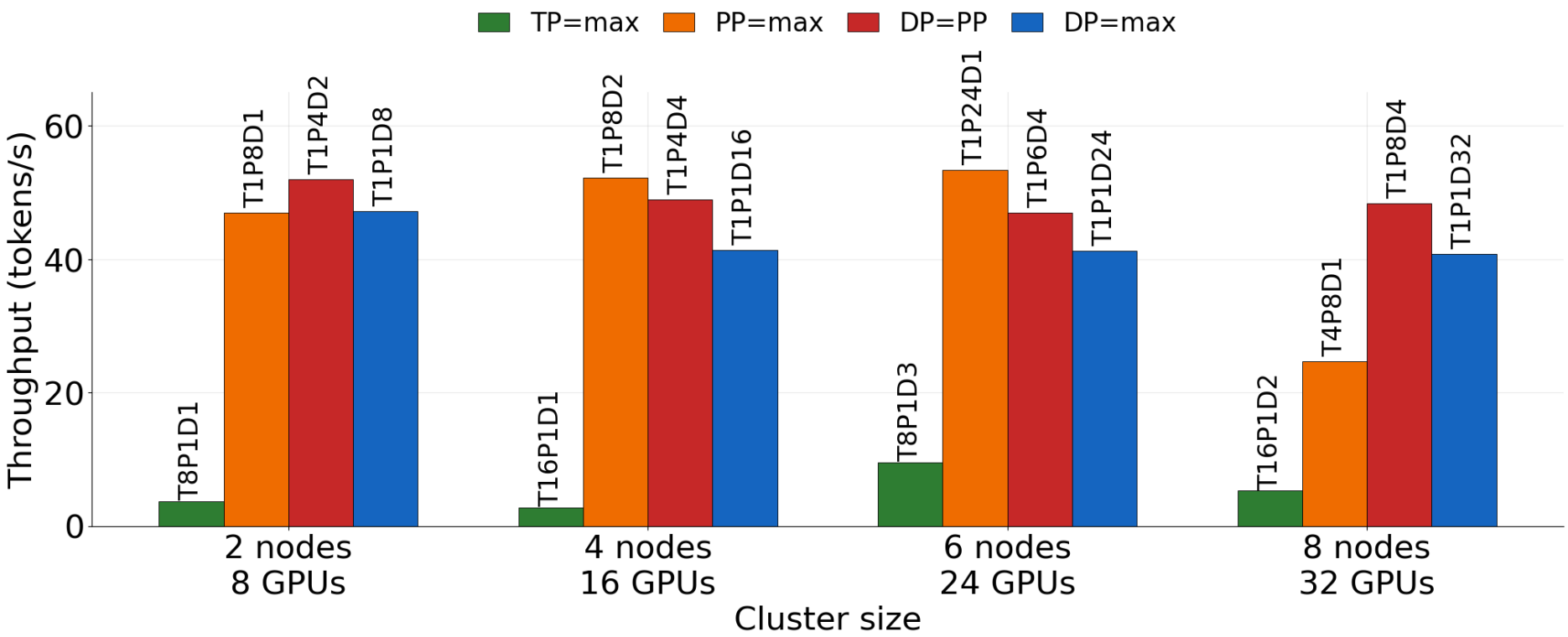


Figure 10: Throughput changes as the same strategy scales across cluster sizes.

Takeaway: Things become less predictable at larger scales due to factors such as pipeline bubbles, accelerator placement, and thermal effects.

Key Insights

Multi-node insight

- The single-node result that higher tensor parallelism minimizes power does not necessarily hold at larger scales.
- The single-node result that higher data parallelism maximizes power and throughput also does not necessarily hold at larger scales.

The best parallelism strategy depends on factors such as the model, cluster size, and network topology.
Some strategies offer lower power but lower throughput, while others offer higher throughput at higher power.

Our idea: exploit these tradeoffs by switching strategies over time to reduce carbon emissions.

Our Idea: Parallelism as a Carbon-Control Knob

Standard practice

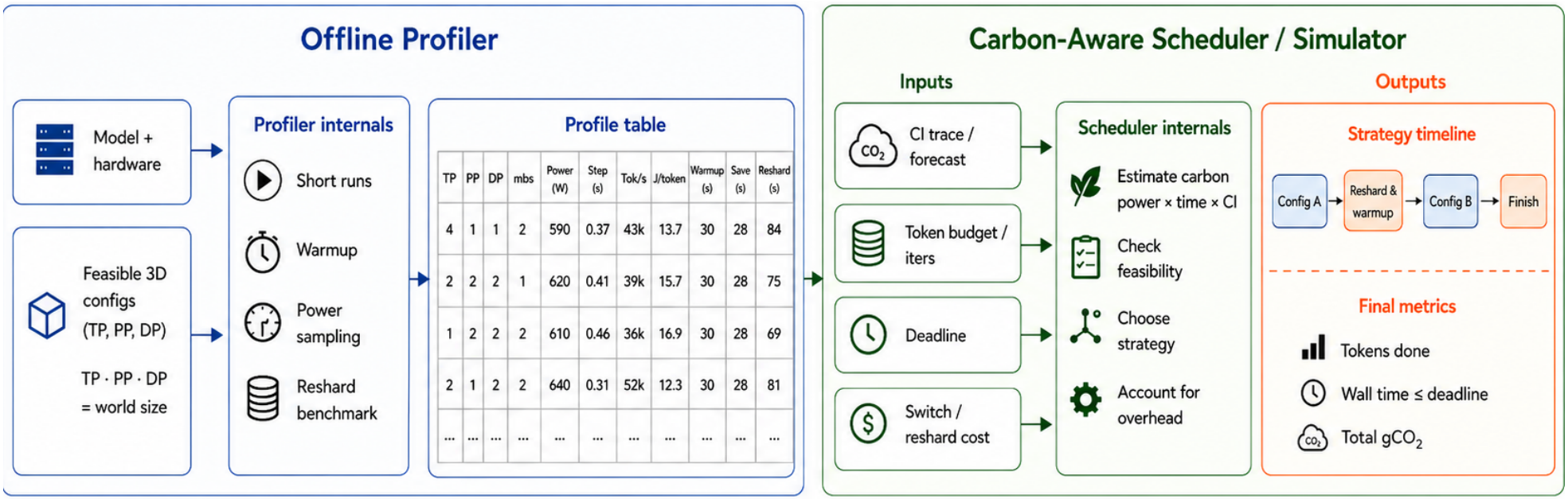
Choose one (TP, PP, DP) configuration at launch and keep it fixed, often the configuration with the highest throughput (Li et al. 2024).

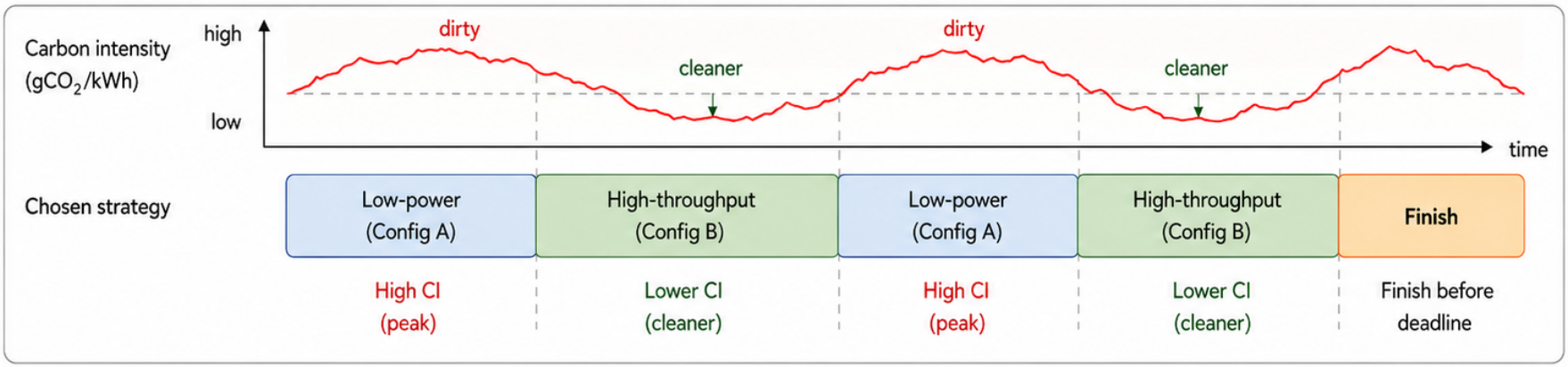
Our idea

Treat (TP, PP, DP) as a tunable operating point along a power–throughput tradeoff curve.

- Dirty grid hours $\rightarrow$ leaner / lower-power configurations.
- Clean grid hours $\rightarrow$ higher-throughput configurations.
- Preserve token budget and deadline.

System Design





Scheduler Formulation

Decision variable

$$x_{t,\pi} \in \{0,1\}, \qquad \sum_{\pi \in \Pi} x_{t,\pi} = 1$$

t: decision interval; $\pi = (TP, PP, DP) \in \Pi$: feasible configuration; $x_{t,\pi} = 1$ if π is used during interval t.

Objective

$$\min_{x} \sum_{t} \sum_{\pi \in \Pi} x_{t,\pi} \, P_\pi \, \Delta t \, CI_t$$

P_π: profiled power in kW; Δt: interval length in hours; CI_t: carbon intensity in gCO_2/kWh.

Fixed start: Training begins at the start of the carbon trace.
Choose one configuration per interval to minimize total emissions.

Scheduler Constraints

$$\sum_{t} \sum_{\pi \in \Pi} x_{t,\pi} r_{\pi} \Delta t \geq N$$

Reach the token budget.

$$s + \sum_{t} \Delta t \, a_t + m\bar{K} \leq D$$

Finish before the deadline.

$$\sum_{\pi \in \Pi} x_{t,\pi} \leq 1 \quad \forall t$$

At most one configuration per window.

Evaluation Setup

- Carbon traces: May 2024, hourly, Electricity Maps.
- Regions: `us-west-1`, `us-west-2`, `us-east-2`.
- Workloads: GPT-2 on 8×A100 and Llama 1B on 4×L40.
- Token budgets: 1B–8B tokens.
- Arrival offsets: 64 sampled offsets per model/region/token budget.
- Deadline: 25% slack over fastest measured static strategy.

Static vs Dynamic Parallelism

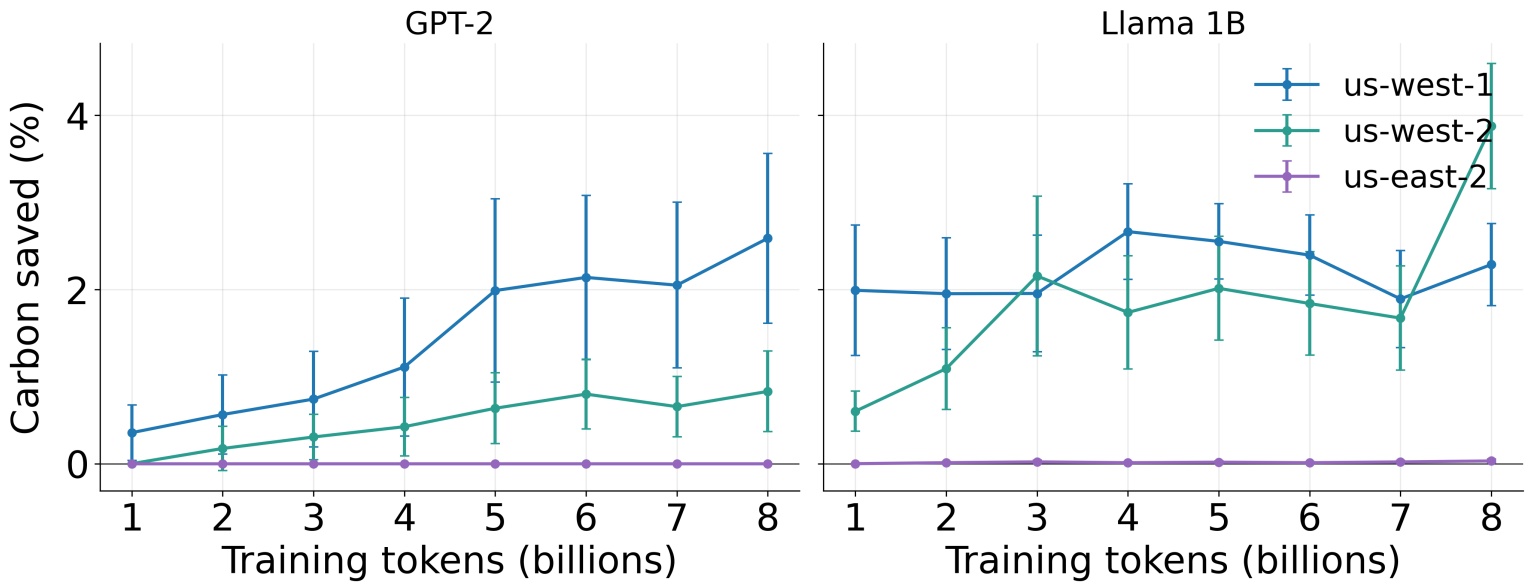
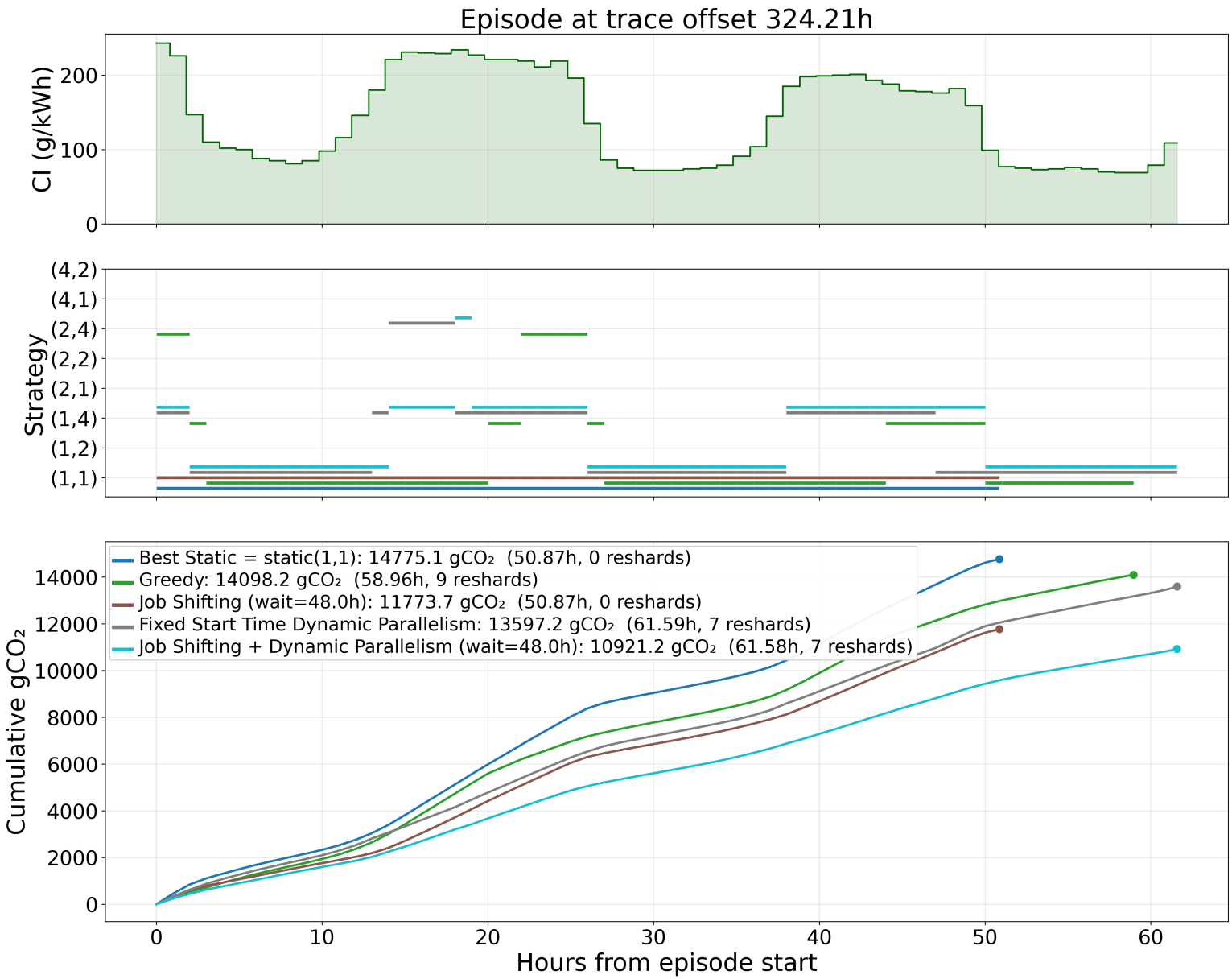


Figure 11: Dynamic parallelism reduces emissions by up to **3.87%**.

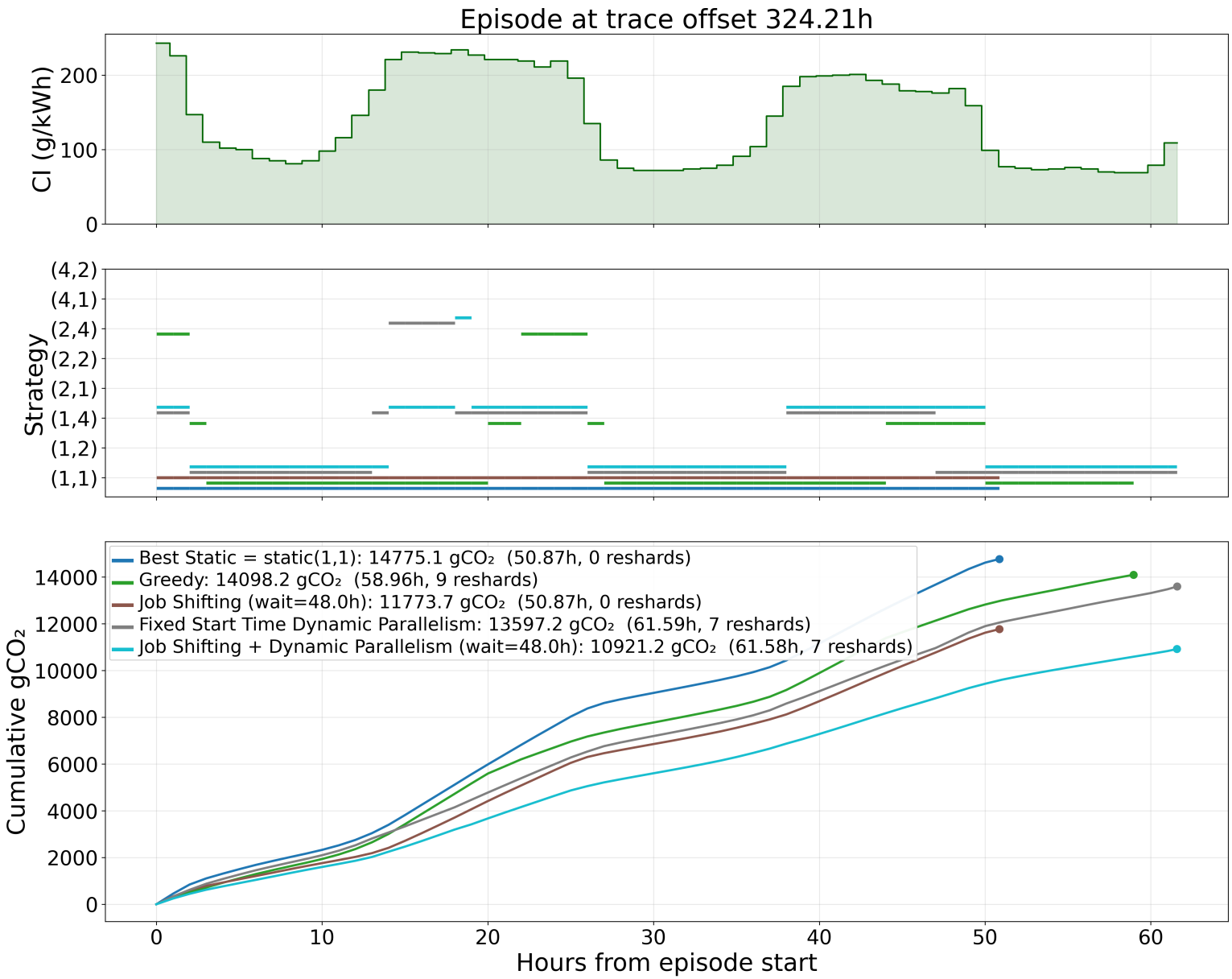
Results

Savings are largest when carbon intensity varies over time; `us-east-2` shows little benefit.

Example Episode



Example Episode



Conclusion

- Parallelism has a coupled, non-monotonic effect on power and throughput.
- Carbon-aware scheduling dynamically switches between lower-power and higher-throughput configurations as carbon intensity changes, while meeting the deadline.
- Savings reach **3.87%** over the best feasible static configuration.

Future Work: Expanding the Search Space

More training knobs

- Scale to larger models and more multi-node experiments.
- Add sequence parallelism and expert parallelism (Li et al. 2022; Fedus et al. 2022).
- Also consider batch size and placement strategies.

Efficient profiling

Adding these knobs quickly creates a huge search space.

Can we find good configurations without profiling every possibility?

Thank You!

Questions?

References I

- Dodge, Jesse et al. (2022). "Measuring the carbon intensity of ai in cloud instances". In: *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pp. 1877–1894.
- Fedus, William, Barret Zoph, and Noam Shazeer (2022). *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. arXiv: 2101.03961 [cs.LG]. URL: https://arxiv.org/abs/2101.03961.
- Grattafiori, Aaron et al. (2024). "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783*.
- Hanafy, Walid A et al. (2023). "Carbonscaler: Leveraging cloud workload elasticity for optimizing carbon-efficiency". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 7.3, pp. 1–28.

References II

- Huang, Yanping et al. (2019). *GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism*. arXiv: 1811.06965 [cs.CV]. URL: https://arxiv.org/abs/1811.06965.
- Kang, Xueze et al. (2025). "ElasWave: An Elastic-Native System for Scalable Hybrid-Parallel Training". In: *arXiv preprint arXiv:2510.00606*.
- Li, Shenggui et al. (2022). *Sequence Parallelism: Long Sequence Training from System Perspective*. arXiv: 2105.13120 [cs.LG]. URL: https://arxiv.org/abs/2105.13120.
- Li, Zongbiao et al. (2024). *Automatically Planning Optimal Parallel Strategy for Large Language Models*. arXiv: 2501.00254 [cs.AI]. URL: https://arxiv.org/abs/2501.00254.

References III

- Narayanan, Deepak et al. (2021). "Efficient large-scale language model training on gpu clusters using megatron-lm". In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–15.
- Patterson, David et al. (2021). "Carbon emissions and large neural network training". In: *arXiv preprint arXiv:2104.10350*.
- Shoeybi, Mohammad et al. (2019). "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism". In: *arXiv preprint arXiv:1909.08053*.
- Wiesner, Philipp et al. (2021). "Let's wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud". In: *Proceedings of the 22nd International Middleware Conference*, pp. 260–272.