TECHNICAL PAPERS

PHANTOM

Phase-Aware Power Management for Energy-Efficient LLM Training

Qi Luo, Kunlin Li, Changxi Liu, Dongsheng Wang, Yun Chen*
HKUST (Guangzhou) · National University of Singapore · Tsinghua University



SC26 hpc unites | Chicago

Outline

1	**Background**	why GPU power matters for LLM training
2	**Motivation**	energy-hungry training · unstable power · reactive control falls short
3	**PHANTOM**	predict power, set the clock early
4	**Evaluation**	two H20 nodes, 16 GPUs
5	**Conclusion**	limitations and takeaways

BACKGROUND

LLM training keeps thousands of GPUs busy for weeks

and runs clusters close to their power and cooling limits

Thousands
of GPUs for weeks, per model (GPT, Llama)

Tens of MWh
per training run

Recurring
retraining and fine-tuning

60–75%
of node power is GPU

Control GPU power → control training energy



NVIDIA Grace Hopper GPU racks · image: broadberry.com

BACKGROUND

Fast power swings threaten reliability

> 5 W/ms

power ramps: 100 W within 20 ms at a phase change

> 50 mV

voltage excursion: about the whole timing guardband

Thousands

of GPUs switch together: rack-level power surges

The consequences

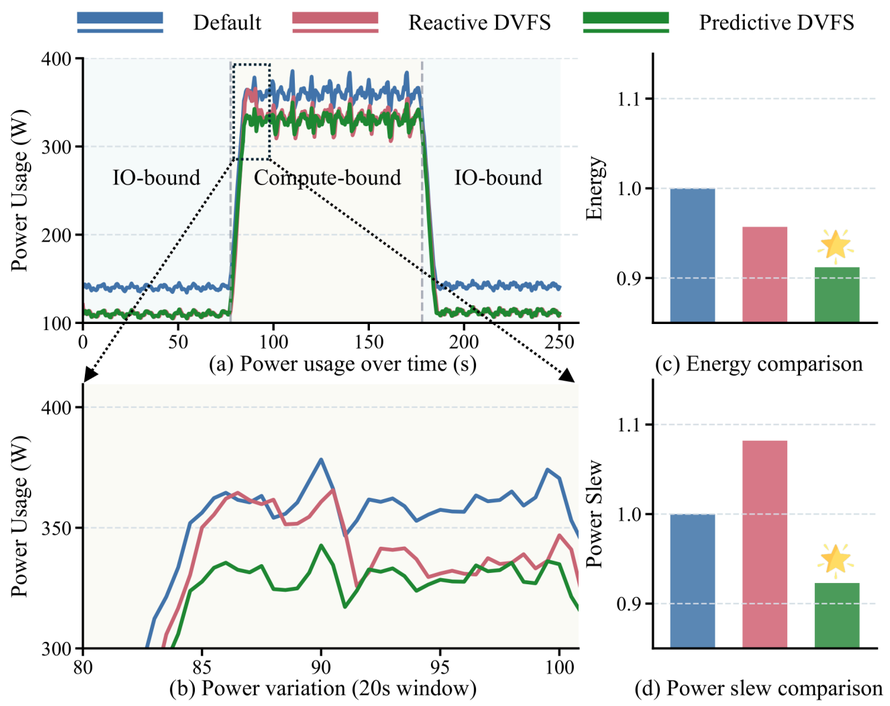
voltage droop, throttling, faults, aging → reload a checkpoint and retrain

SC26

BACKGROUND

Most GPU controllers today react too late

DVFS: a lower clock draws less power, but runs longer



Default vs. reactive vs. predictive DVFS: power over time (a, b), energy (c), power slew (d)

Reactive

acts after power moves: lag and overshoot

Predictive

acts before the phase changes

Result

lowest energy and least power slew

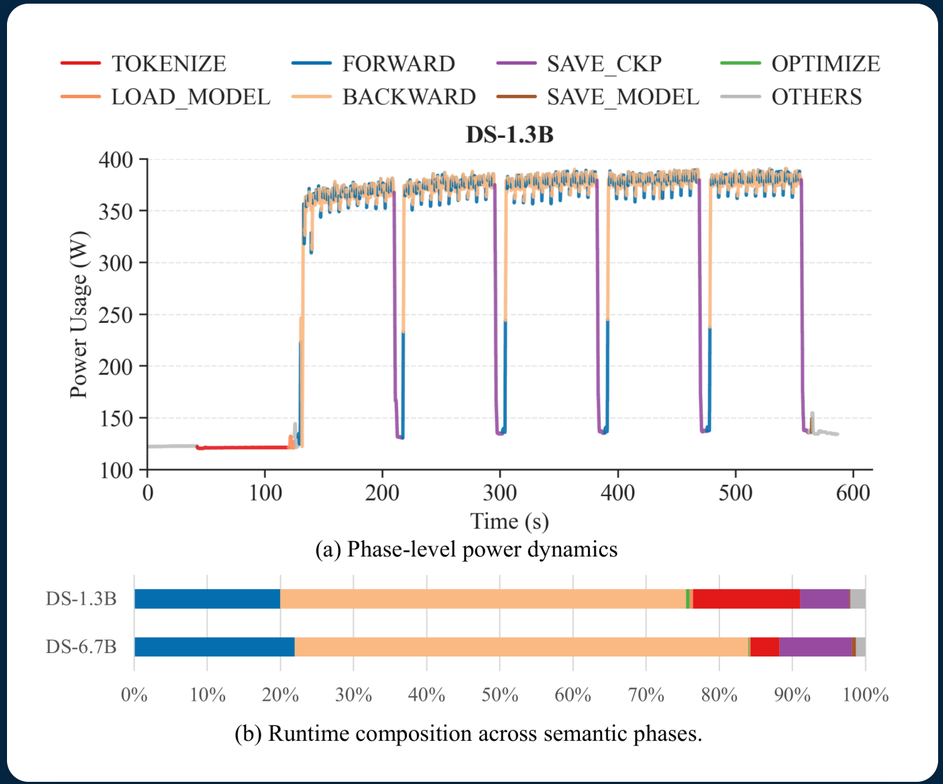
BACKGROUND

Existing approaches do not fit LLM training

APPROACH	EXAMPLES	WHY IT FALLS SHORT
Hardware DVFS	NVIDIA GPU Boost, AMD APM	reactive; blind to phases
Transient damping	power capping, buffering, power burners	after the fact; costs energy
App / cluster level	DEPO, PowerFlow	reactive, or far too coarse
Inference DVFS	μ-Serve, Ma et al.	built for request SLOs, not training
PHANTOM	this work	predictive, phase-aware, energy + stability

MOTIVATION

GPU power jumps at every phase change



Measured GPU power while training DS-1.3B on one H20 (a), and runtime share of each phase (b)

300–400 W
compute phases

100–150 W
I/O phases

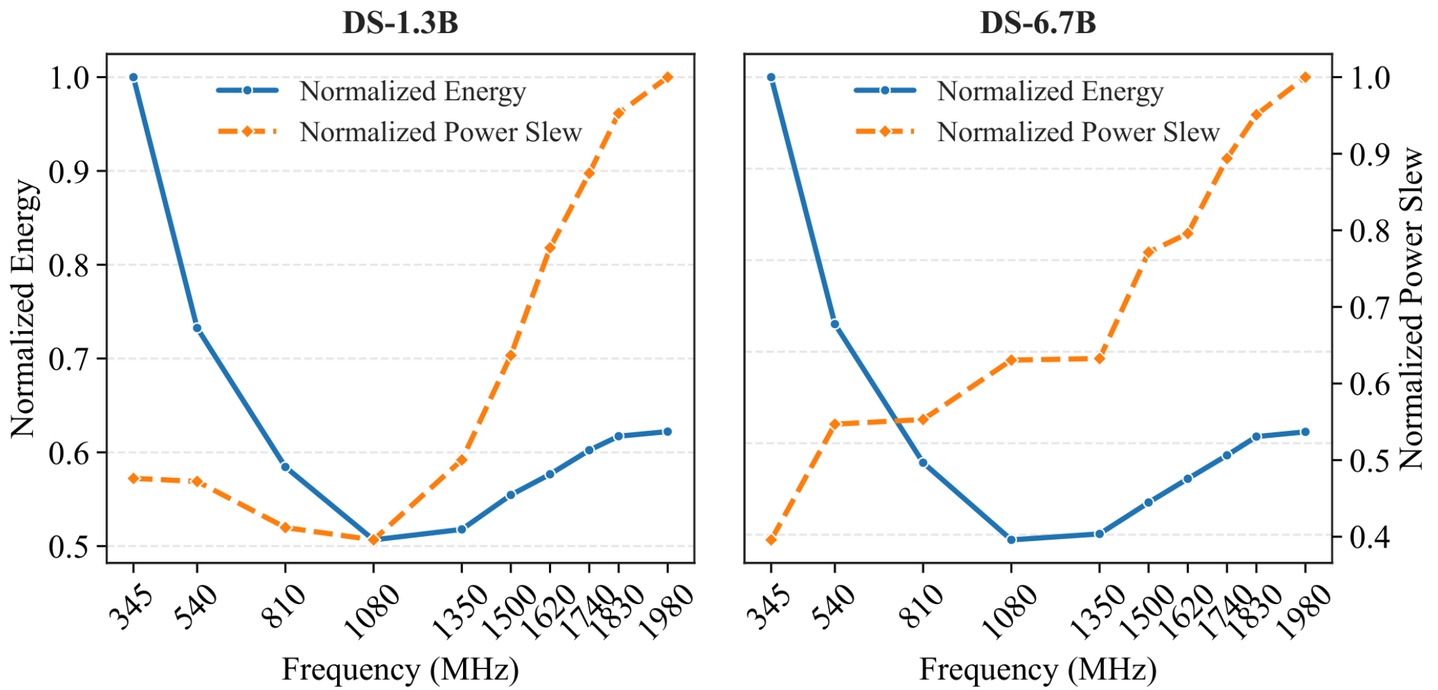
76%
of runtime is compute (b)

Every GPU
in every iteration

MOTIVATION

No fixed clock wins

the minimum-energy clock costs 33% more runtime



Energy and power slew at ten fixed GPU clocks, 50 training steps each

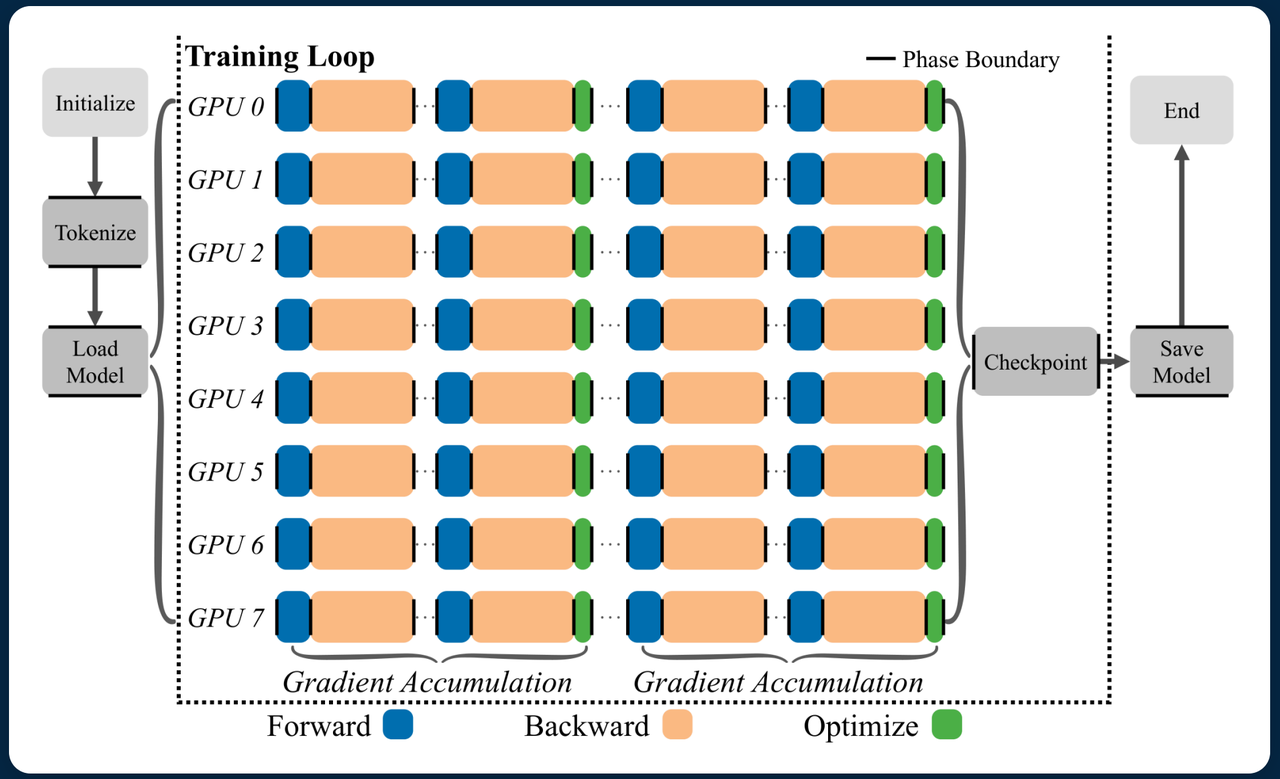
Energy: U-shaped
lowest near 1080 MHz

Power slew
keeps falling with the clock

Fixed 1080 MHz
+33% runtime (EngOpt)

KEY INSIGHT

Phases repeat, so the next one is predictable



Phase	Description
`TOKENIZE`	Data preprocessing and tokenization
`LOAD_MODEL`	Loading model weights from storage
`FORWARD`	Forward propagation
`BACKWARD`	Backward propagation (async All-Reduce)
`OPTIMIZE`	Parameter update via `optimizer.step()`
`SAVE_CHECKPOINT`	Intermediate checkpoint saving
`SAVE_MODEL`	Final model saving
`OTHERS`	e.g., initialization and miscellaneous setup

Training phases on 8 GPUs over time (left), and what each phase does (right)

Same order on every GPU

in every iteration

→ set the next phase's clock early

CHALLENGES

Three challenges for training-time DVFS

1 · Three goals, ms decisions

energy vs. stability vs. runtime

phases last tens of ms

→ Pareto optimizer within τ

2 · Act before the swing

swings unfold in milliseconds

feedback is always late

→ predict power 100 ms ahead

3 · Know the phase

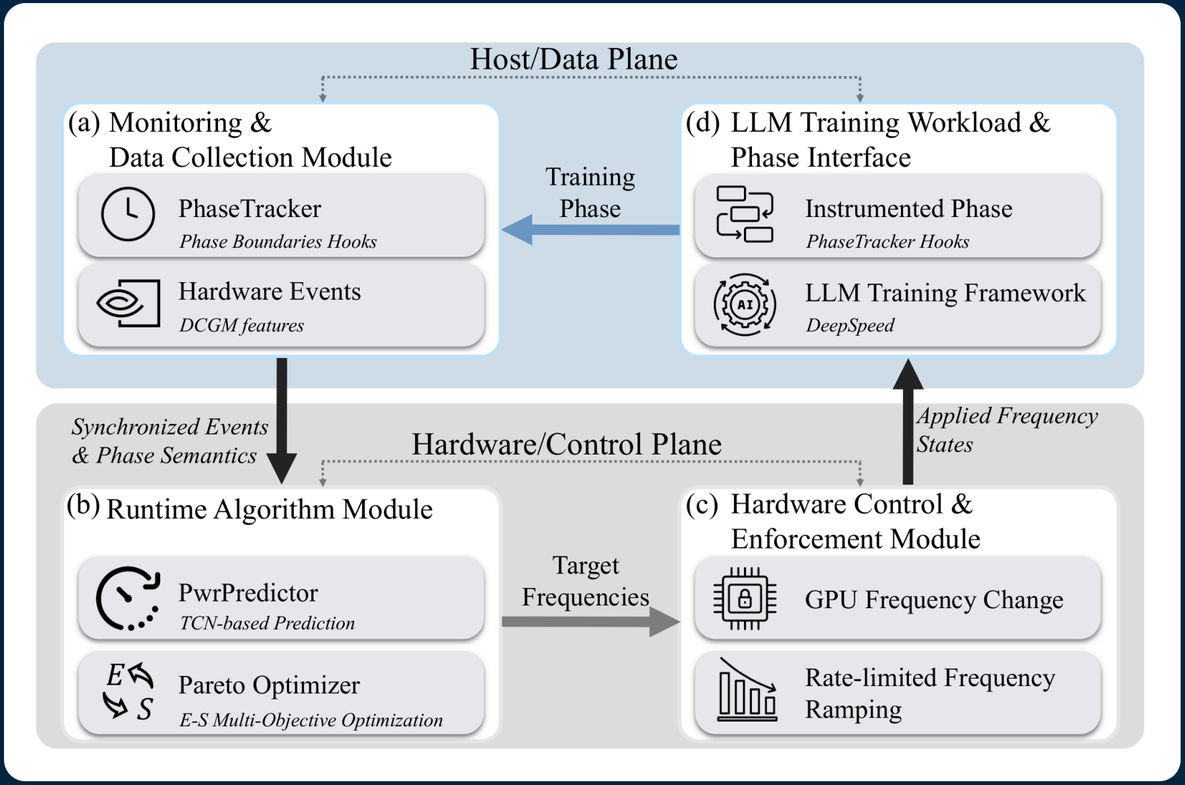
counters alone miss intent

jumps vary by model, GPU

→ phases + counters as input

DESIGN

PHANTOM sets the clock before each phase change



PHANTOM architecture: (a) monitor, (b) decide, (c) act, (d) the training job

(a) Monitor
phase hooks + 8 GPU counters

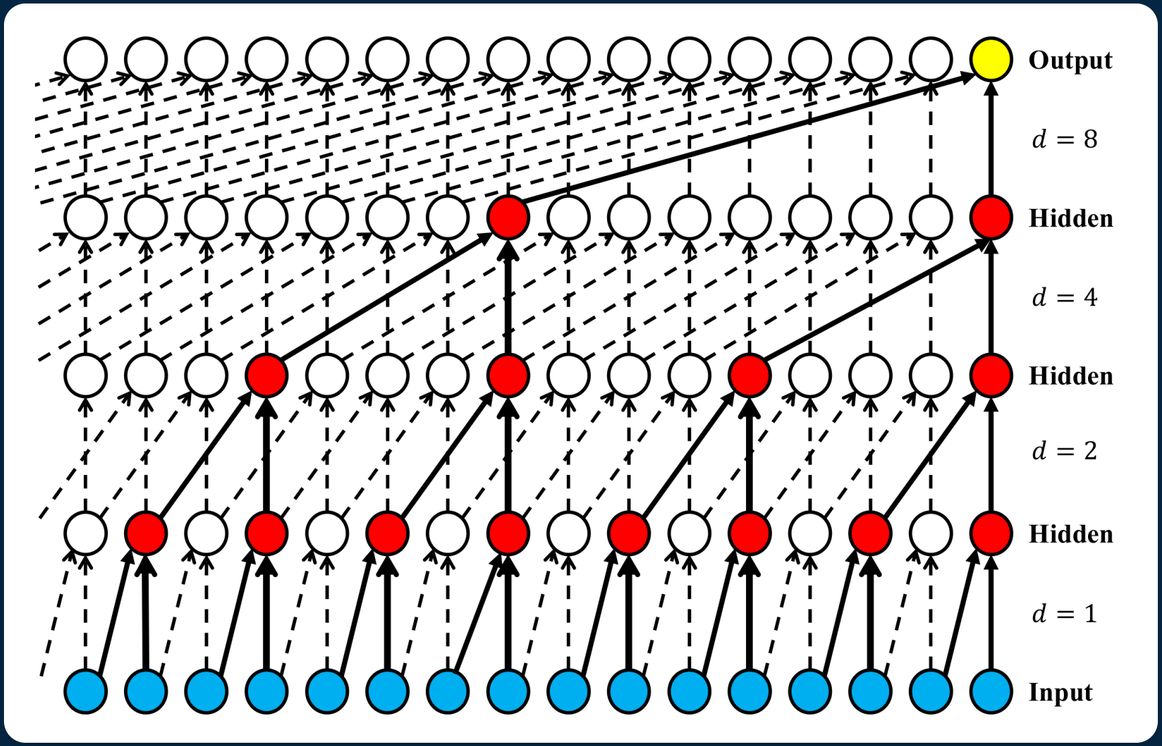
(b) Decide
TCN predicts, Pareto picks

(c) Act
NVML clock, rate-limited

(d) Train
DeepSpeed + ≈12 lines of hooks

DESIGN

A small TCN predicts power 100 ms ahead



Dilated causal convolutions: the output (top right) sees the whole window along the red path

Input

8 counters (clock ← candidate *f*), phase, GPU and model ids

Output

power 100 ms ahead at *f*

Model

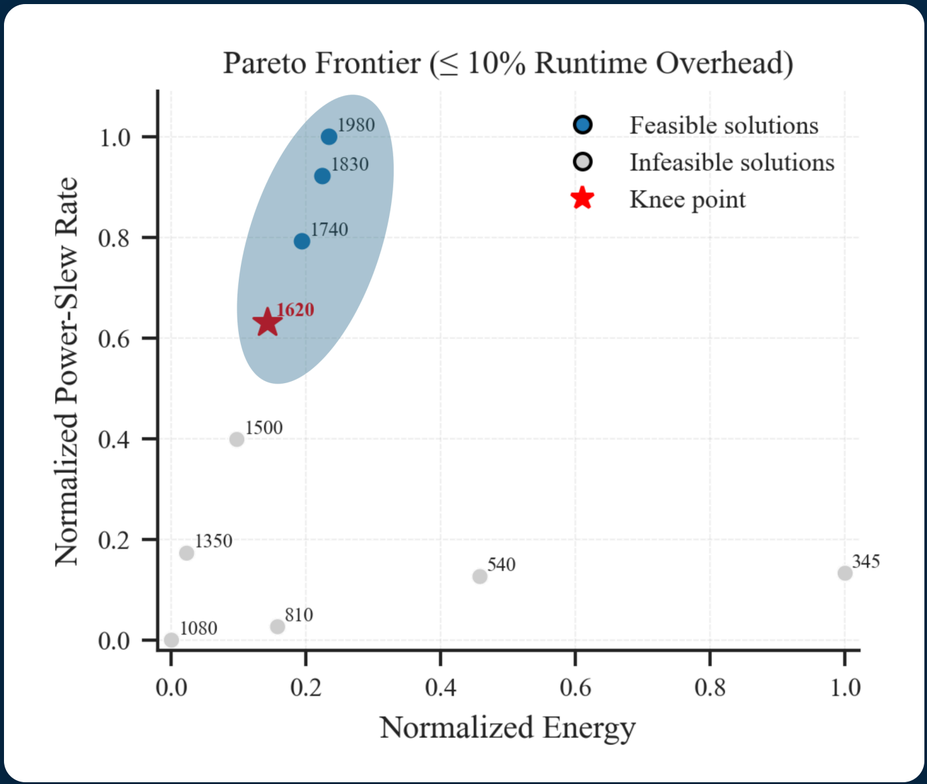
32 samples · 64 channels · 5 levels

Accuracy

R^2 0.997 · MAE 1.56 W

DESIGN

Pick the best balance within a slowdown bound τ



One compute phase, τ = 10%: gray clocks are too slow, blue are feasible, the star is chosen

1 · Predict for every clock *f*

$$E(f) = \hat{P}(f)\,\Delta t \qquad S(f) = \left|\hat{P}(f) - P_{\text{current}}\right|$$

2 · Drop clocks that are too slow

$$\hat{T}(f,p) \le (1+\tau)\,T_{\text{base}}[p]$$

compute phases only

3 · Take the knee of the Pareto front

$$f^{*} = \arg\min_{f \in \mathcal{P}} \sqrt{\left(\alpha_p\,\tilde{E}(f)\right)^2 + \left(\beta_p\,\tilde{S}(f)\right)^2}$$

compute favors stability, I/O energy

DESIGN

The control loop fits in 100 ms

STEP	MEDIAN	P99
Decision path	5.4 ms	7.0 ms
of which TCN inference	0.77 ms (mean)	
End-to-end actuation	11.0 ms	30.1 ms
Prediction window	100 ms	

≈ 70 ms
slack, even at p99

Rate-limited
big jumps become ramps

< 0.1%
runtime overhead

EVALUATION

Setup: up to 16 NVIDIA H20 GPUs

Platform

8× H20 per node

1 or 2 nodes

DCGM at 50 ms

Workloads

DeepSeek-Coder 1.3B, 6.7B

DeepSpeed ZeRO-3

batch 128, seq. 4096

Baselines

Default GPU Boost

Static low clock in I/O

EngOpt 1080 MHz

TimeCon within τ

Metrics

runtime, energy

NPSR: power slew

EDP = energy × time

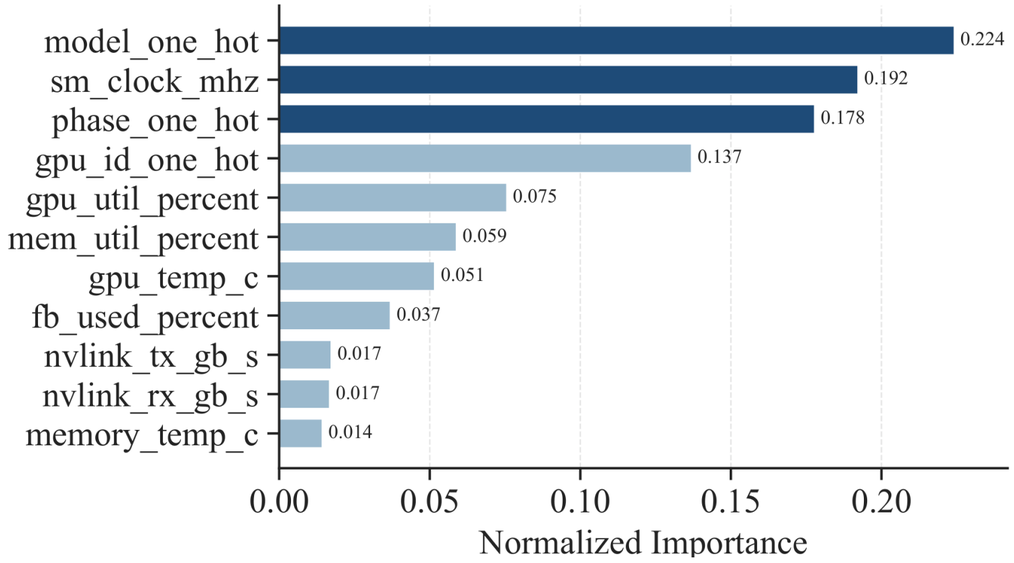
vs. Default, lower is better

EVALUATION

TCN: most accurate, 10× faster than XGBoost

Model	MAPE (%)	Latency (ms)
Linear Regression	19.42	**0.05**
XGBoost	1.23	7.89
LSTM	1.45	2.48
TCN	**1.08**	0.77

Metric	Value
Mean Absolute Error (MAE)	2.669 W
Root Mean Square Error (RMSE)	8.847 W
Mean Absolute Percentage Error (MAPE)	1.08%
Mean Measured Power	290.665 W
Mean Predicted Power	290.306 W



Phase one-hot ranks **3rd of 11**: phases matter

Error and latency of four predictors (top), online accuracy (bottom), feature importance (right)

EVALUATION

One node: 15–16% less energy, 5–6% more time

Method	Single Node (8× H20 GPUs)							
	DS-1.3B				DS-6.7B			
	Duration ↓	Energy ↓	NPSR ↓	EDP ↓	Duration ↓	Energy ↓	NPSR ↓	EDP ↓
Default	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Static	1.03	0.91	1.10	0.95	1.02	0.88	1.13	0.90
EngOpt	1.33	0.82	0.80	1.09	1.33	0.81	0.87	1.07
TimeCon	1.12	0.92	0.83	1.03	1.09	0.90	0.90	0.97
PHANTOM	1.06	0.84	0.85	0.89	1.05	0.85	0.91	0.90

1 node, 8× H20 · normalized to the default GPU policy (1.00), lower is better

−15–16% energy

−9–15% power slew (NPSR)

+5–6% runtime

−10–11% energy × delay (EDP)

EVALUATION

Two nodes: 16–20% less energy, 1% more time

Method	Two Nodes (16× H20 GPUs)							
	DS-1.3B				DS-6.7B			
	Duration ↓	Energy ↓	NPSR ↓	EDP ↓	Duration ↓	Energy ↓	NPSR ↓	EDP ↓
Default	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Static	1.09	0.96	1.24	1.04	1.02	0.90	1.15	0.92
EngOpt	1.11	0.77	0.81	0.85	1.04	0.78	0.90	0.81
TimeCon	1.09	0.92	0.99	1.00	1.05	0.82	0.92	0.86
PHANTOM	1.01	0.84	0.85	0.84	1.01	0.80	0.88	0.81

2 nodes, 16× H20 · normalized to the default GPU policy (1.00), lower is better

–16–20% energy

–12–15% power slew (NPSR)

+1% runtime

–16–19% energy × delay (EDP)

DISCUSSION

Limitations and future work

NVIDIA H20 only
generic telemetry should transfer; untested

Up to two nodes
larger scale is future work

Power sensor not characterized
update period and averaging unknown

Data parallelism only
tensor and pipeline parallelism next

CONCLUSIONS

Predict the phase, set the clock before it arrives

Power is phase-structured
so it is predictable

Predict, then act
set the clock before the phase changes

Optimize jointly
energy and stability, within τ

15–16%
less energy

9–15%
lower power slew

10–11%
better energy × delay

< 0.1%
control overhead

vs. the default GPU policy, on one 8-GPU node

PHANTOM · THANK YOU

Q & A

Phase-Aware Power Management for Energy-Efficient LLM Training

Qi Luo · qluo615@connect.hkust-gz.edu.cn
Yun Chen (corresponding) · yunchen@hkust-gz.edu.cn



SC26 hpc unites | Chicago