



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

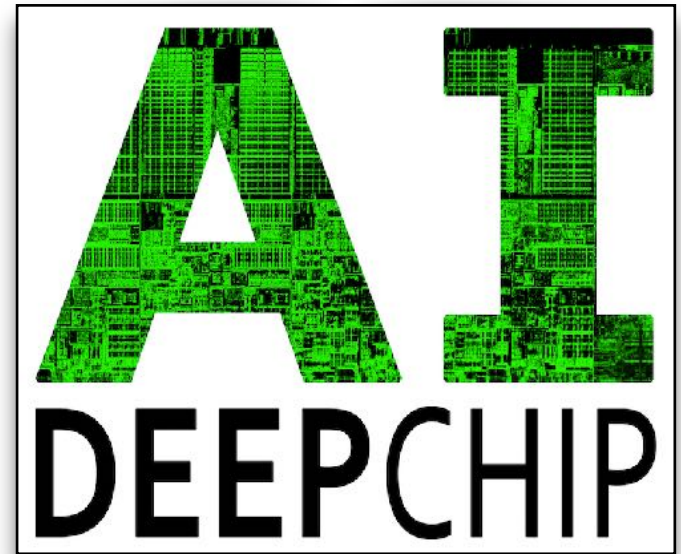


FROM PICOJOULES TO GIGAWATT-HOURS ENERGY-TO-COMPLETION AND GPU DVFS FOR LLM WORKLOADS

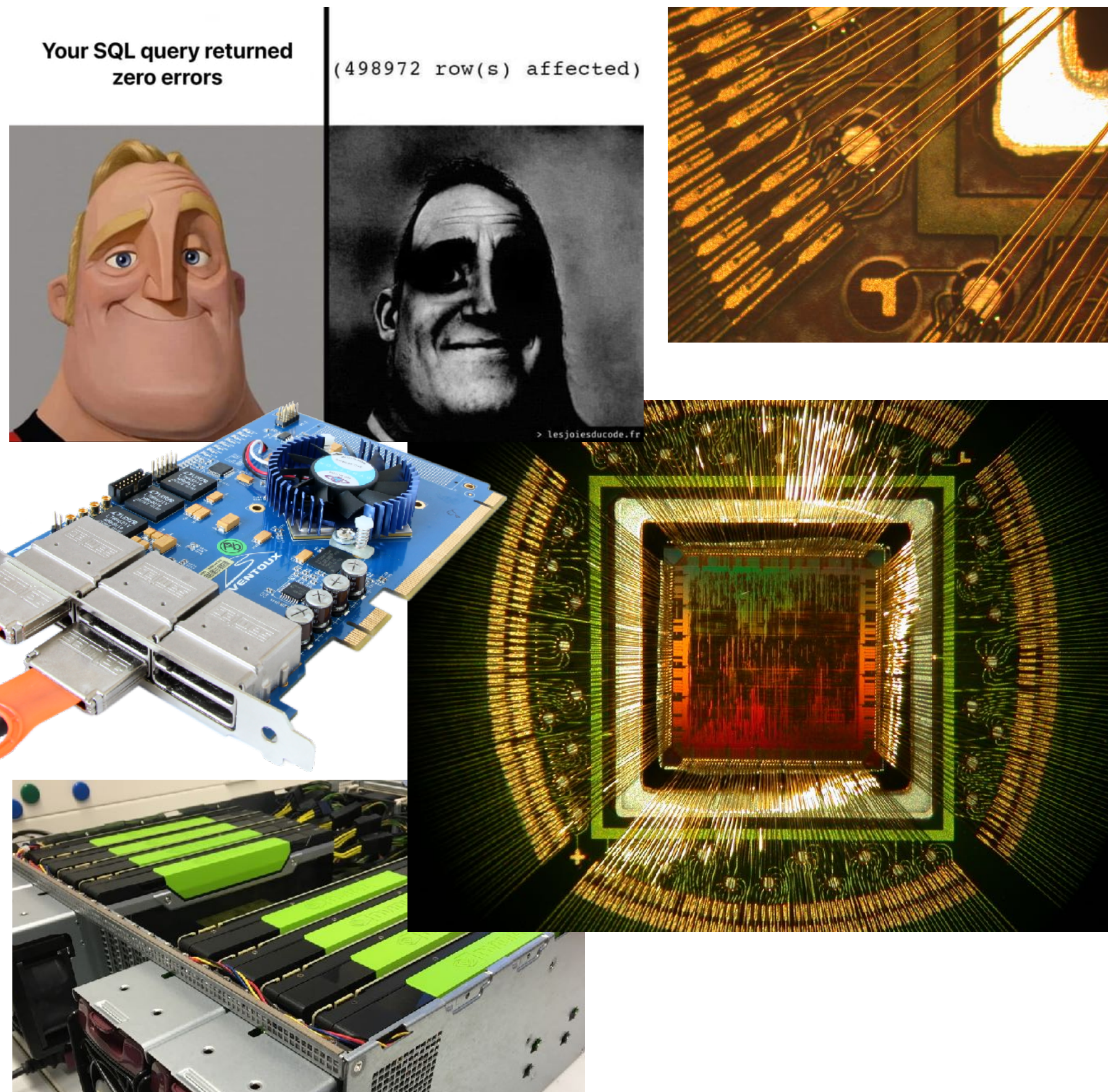
*Holger Fröning, Daniel Barley, Dominik Gausepohl - holger.froening@ziti.uni-heidelberg.de
Hardware and Artificial Intelligence (HAWAII) Lab - hawaii.ziti.uni-heidelberg.de
Heidelberg University*

NHR FAU, Jan 21, 2026

GROUP BACKGROUND: FROM HW TO ACCELERATORS TO ML



From: database engineer, HW designer
(ASICS, FPGA), HPC



$$x^l = \Phi(\mathbf{W} \oplus \mathbf{x}^{l-1} + b^l)$$



Neural Architectures



Compiler



Plethora of HW

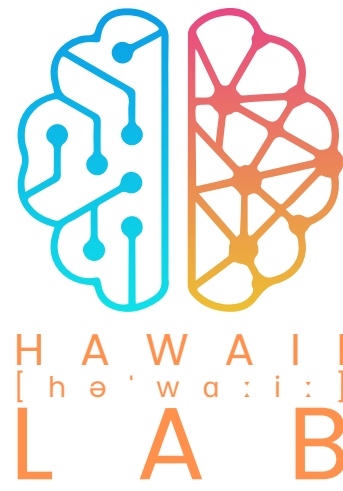
$$perf[\frac{\text{ops}}{\text{s}}] = p[\text{Watt}] \cdot e[\frac{\text{ops}}{\text{J}}]$$

$$P = afCV^2 + VI_{leakage}$$



To: vertically integrated approach to
efficient ML => HW systems for AI

HARDWARE & ARTIFICIAL
INTELLIGENCE



Holger Fröning
Prof. Dr.



Andrea Seeger
Office Assistance



Emma Seeger
Assistance's Assistance



Shigehiko Schamoni
M.A.



Kazem Shekofteh
Postdoc



Bernhard Klein
PhD Student



Alexandra Stehle
PhD Student



Hendrik Borrás
PhD Student



Daniel Barley
PhD Student



Aleksandra Poreba
PhD Student



Xiao Wang
Researcher



Robin Janssen
PhD Student



Andrej Meininger
Master Research Assistant



Anusha Chattopadhyay
Graduate Research Assistant



Arjan Siddhpura
BScI Thesis



Christian Heusel
BScI Thesis



Dennis Jakob
Bachelor Research Assistant



Dominik Gausepohl
MScDACS Thesis



Elizaveta Mironovich
TA



Fangling Du
MScSC Thesis



Hao Zhang
MScSC Thesis



Ilya Belkin
MScDACS Thesis



Jonathan Bernhard
MScDACS Thesis



Jonathan Leis
MScDACS Thesis



Leandro Borzyk
MScTI Thesis



Lukas Rapp
MScDACS Thesis



Max Mielke
Master Research Assistant, MScTI Thesis



Maximilian Burr
Master Research Assistant



Niklas Summ
Master Research Assistant, TA



Nils Kochendörfer
MScTI Thesis



Paul Kupper
MScDACS Thesis



Philia Jankov
Master Research Assistant



Philipp Hematty
MScTI Thesis



Runan Duan
TA



Shijie Sun
MScSC Thesis



Theo Stempel-Hauburger
MScDACS Thesis



Vimala Bauer
TA

CONTINUED DEMAND FOR MORE HARDWARE PERFORMANCE

Power scales GPU performance

GPUs are close to perfect for standard DNNs and LLMs, but can we do better?

Future scaling of GPUs comes at tremendous costs in terms of power consumption

Do GPUs create a bias towards “standard”-DNN-based solutions?

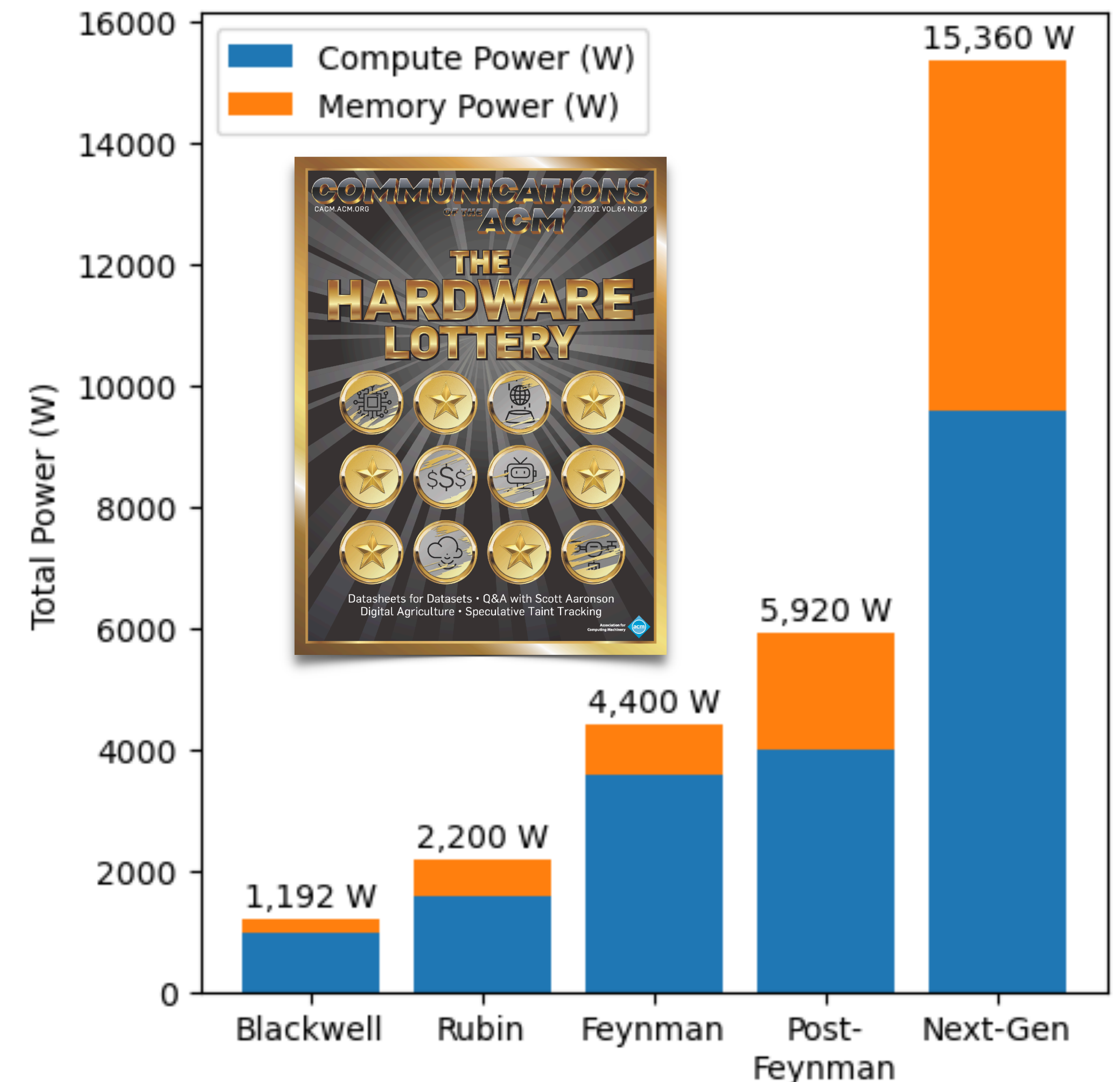
Hardware lottery: yes!

What are promising alternatives? ¹

Emerging HW often comes with imperfections

Noise, non-linearities, saturation effects, etc.

Analog computations (electrical, photonic), resistive memory



¹ Careful, billion-dollar question

POST-DENNARD CMOS PERFORMANCE SCALING

Peak-compute
throughput

$$\text{perf} \left[\frac{\text{ops}}{s} \right] = \text{IPC} \cdot f$$

Power-limited
throughput

$$\text{perf} \left[\frac{\text{ops}}{s} \right] = P [\text{W}] \cdot \eta \left[\frac{\text{ops}}{\text{J}} \right]$$

Energy is additive

$$= P \cdot \left(\frac{\text{ops}}{\epsilon_{\text{op}} + \epsilon_{\text{mem}}} \right)$$

Operation ← → Data movement

$$\epsilon_{\text{op}} = f_{\text{comp}}(t, b) [pJ]$$

GPU FPGA
CPU
CMOS-BASED, ARCH DEP.

$$\epsilon_{\text{mem}} = f_{\text{mem}}(l, d, b) [pJ]$$

SRAM HBM
WIRES DRAM
TECHNOLOGY DEP.

Frequency f , efficiency ϵ , data type $t[\{\text{float}, \text{int}\}]$, bit width b , distance $d[\text{mm}]$, memory level l

CMOS TECHNOLOGY TRENDS

45NM (2014) VS 7NM (2021)

	picoJoules	
Integer	45nm	7nm
Add		
8 bit	0.03	0.007
32 bit	0.1	0.03
Mult		
8 bit	0.2	0.07
32 bit	3.1	1.48

	picoJoules	
Float	45nm	7nm
FAdd		
16 bit	0.4	0.16
32 bit	0.9	0.38
FMult		
16 bit	1.1	0.34
32 bit	3.7	1.31

64-bit	picoJoules	
Mem	45nm	7nm
SRAM		
8kB	10	7.5
32kB	20	8.5
1MB	100	14
DDR4	1300 - 2600	1300 -
HBM2		250 - 450

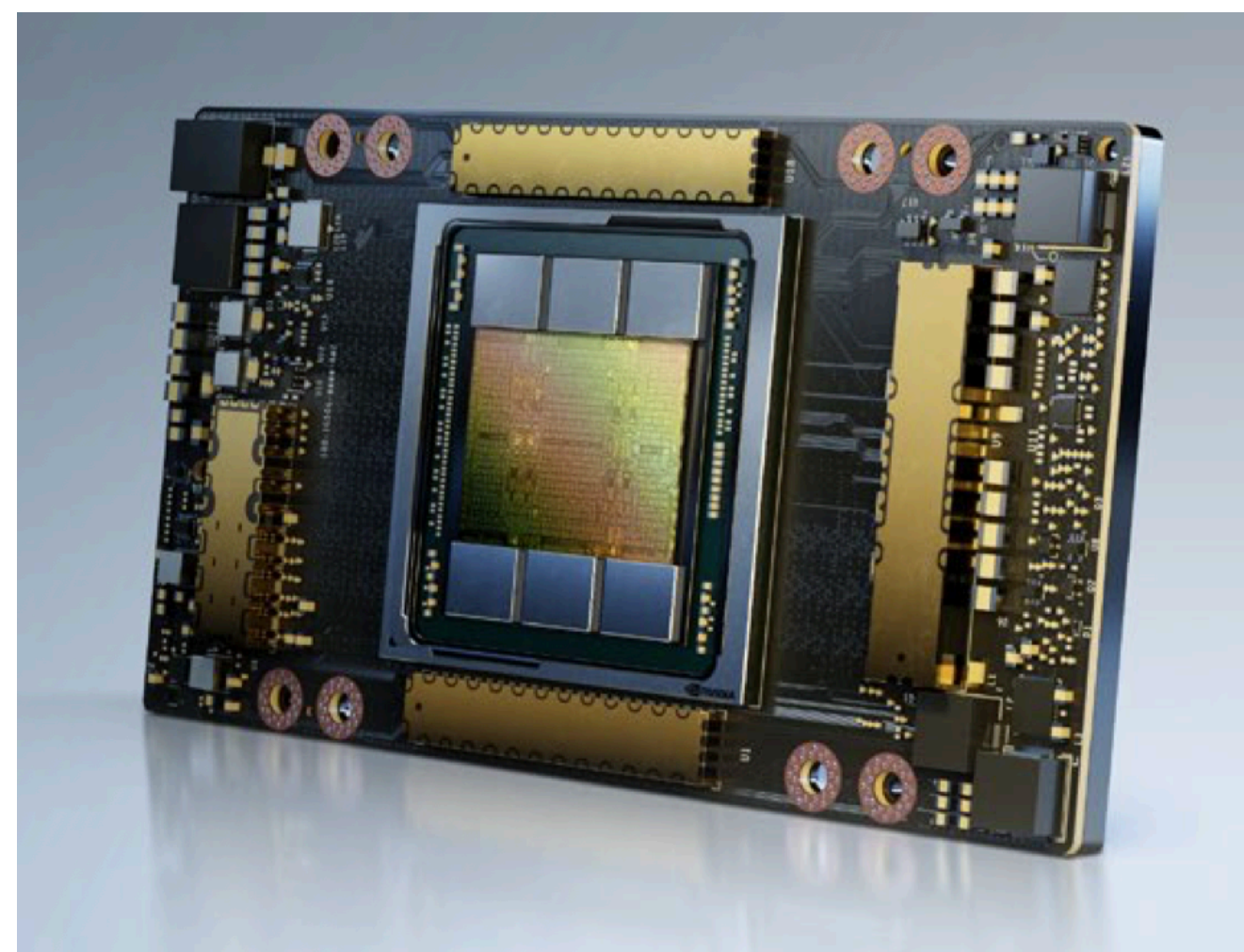
M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). doi: 10.1109/ISSCC.2014.675732

Norman P. Jouppi, et al. 2021. Ten lessons from three generations shaped Google's TPUv4i. ISCA. <https://doi.org/10.1109/ISCA52012.2021.00010>

A100 NUMBER CHECK

NVIDIA A100 SXM, 7nm, 400W

	A100 TOPS/s	A100 pJ/OP	Theory pJ/OP	Residual = memory pJ/bit
Vector FP32 [TF/s]	19.5	20.51	1.31	0.20
Matrix FP32 [TF/s]	156	2.56	1.31	0.01
Matrix FP16 [TF/s]	312	1.28	0.34	0.02
Matrix INT8 [TOP/s]	624	0.64	0.07	0.02



7nm Memory Energy

	64bit	1bit
SRAM 8kB	7.5	0.12
SRAM 32kB	8.5	0.13
SRAM 1MB	14	0.22
DDR4	1300	20.31
HBM2	450	7.03

PICOJOULES <=> GIGAWATT-HOURS

No matter the scale, same physics,
just aggregated differently

Energy is power integrated over time

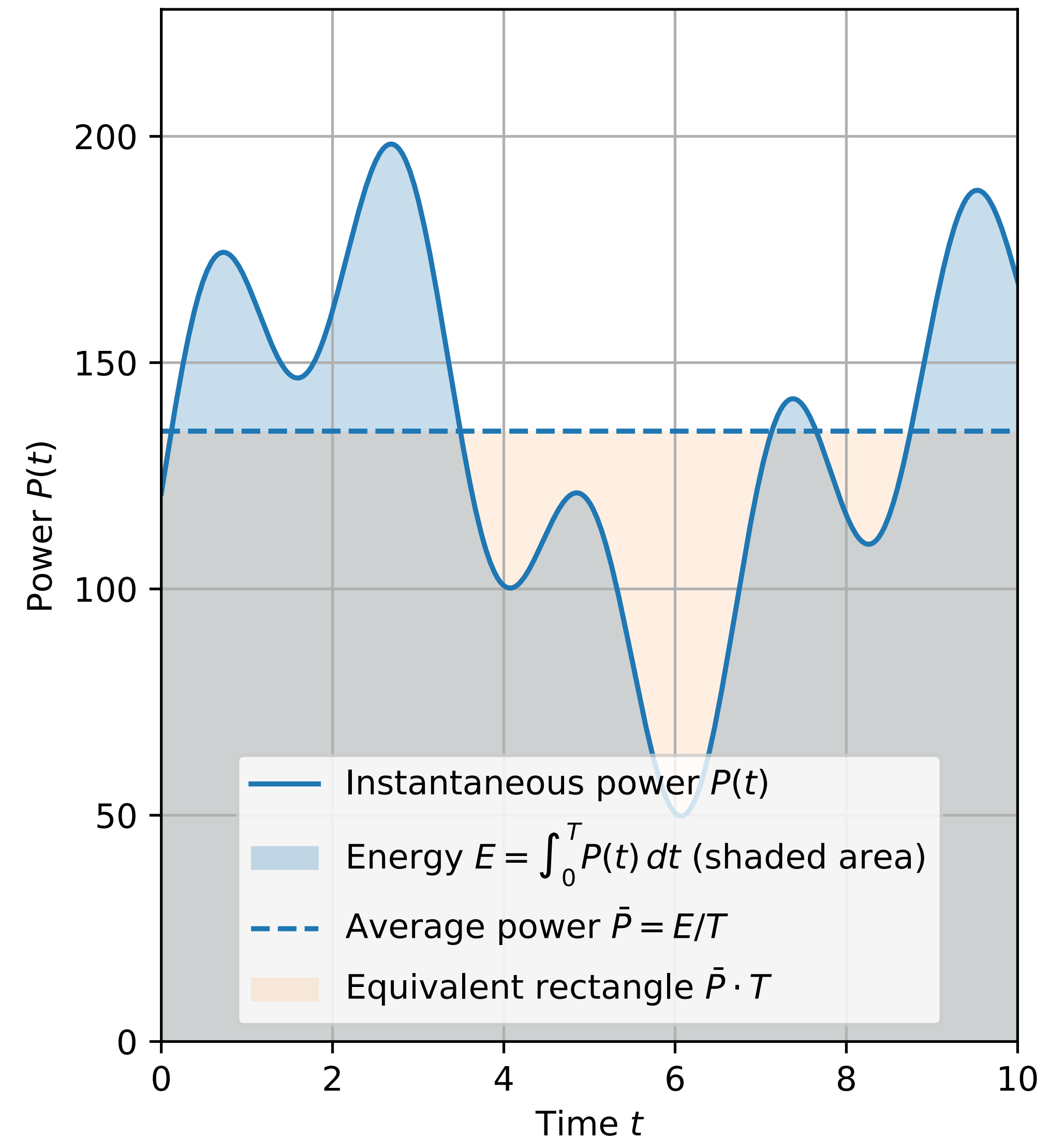
$$E = \int P(t) dt = \bar{P} \cdot t$$

Fixed workload

Then, DVFS is fundamentally a
trade-off curve

More power to finish faster, or,

Lower power and take longer



ON (IDEALIZED) FREQUENCY-INDEPENDENCE

Following Dennard yields frequency independency

$$E_{\text{dyn}} = P_{\text{dyn}} t = (\alpha C_{\text{eff}} V^2 f) \cdot (N_{\text{cycles}} / f) \approx \text{const} \cdot V^2$$

From first principles, dynamic energy per fixed work is roughly independent of frequency if voltage is constant and performance scales linearly. But:

Increasing f typically requires higher V for timing closure

$$E_{\text{dyn,overhead}}(f) \propto V(f)^2$$

Static/leakage and other time-proportional power

Real chips consume non-zero power even when not doing "useful" switching

$$E_{\text{static}} = P_{\text{static}} \cdot T \approx I_{\text{leak}}(V, T_{\text{die}}) V \cdot t$$

Architectural overheads and non-linear performance scaling

In practice, $\text{Perf}(f) \approx f$ does not hold true except for compute-bound kernels with fully utilized pipelines

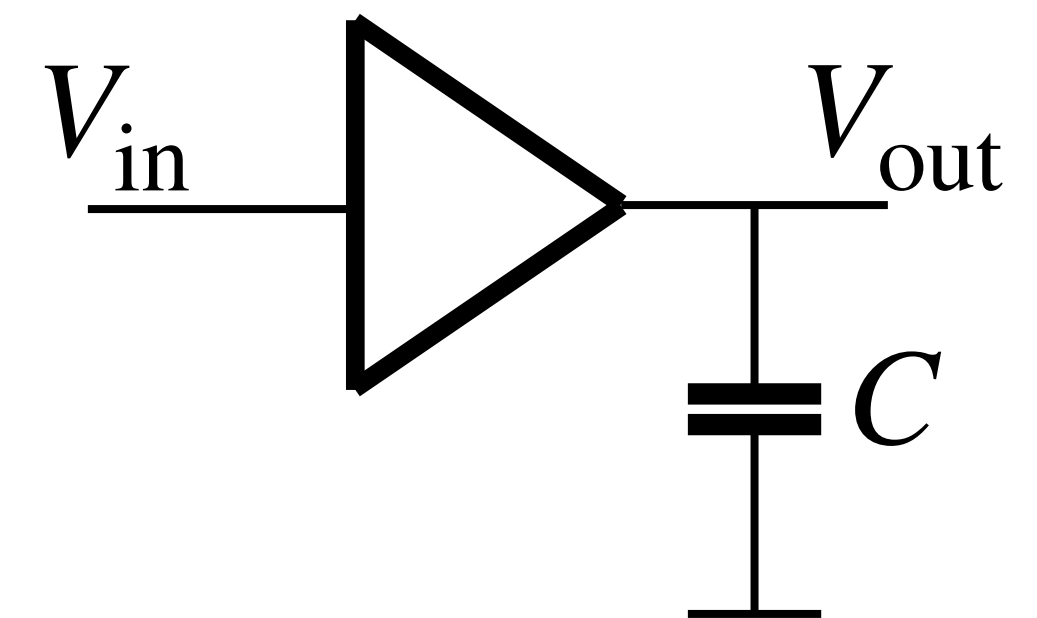
If $\text{Perf}(f)$ grows slower than $P(f)$, then total energy $E(f)$ rises

Switching activity α

$$C_{\text{eff}} = \sum C$$

$$N_{\text{cycles}} = t f$$

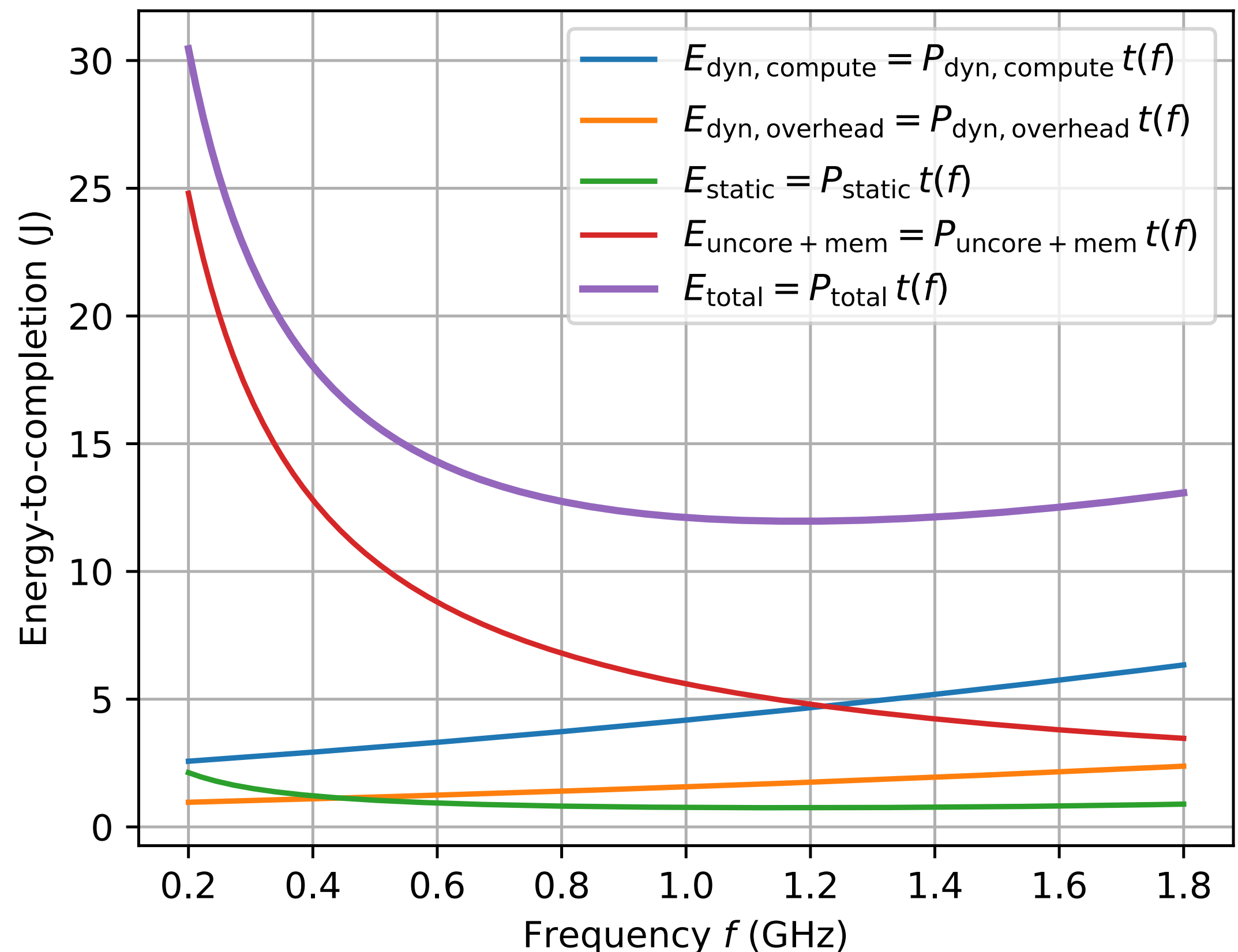
Temperature T_{die}



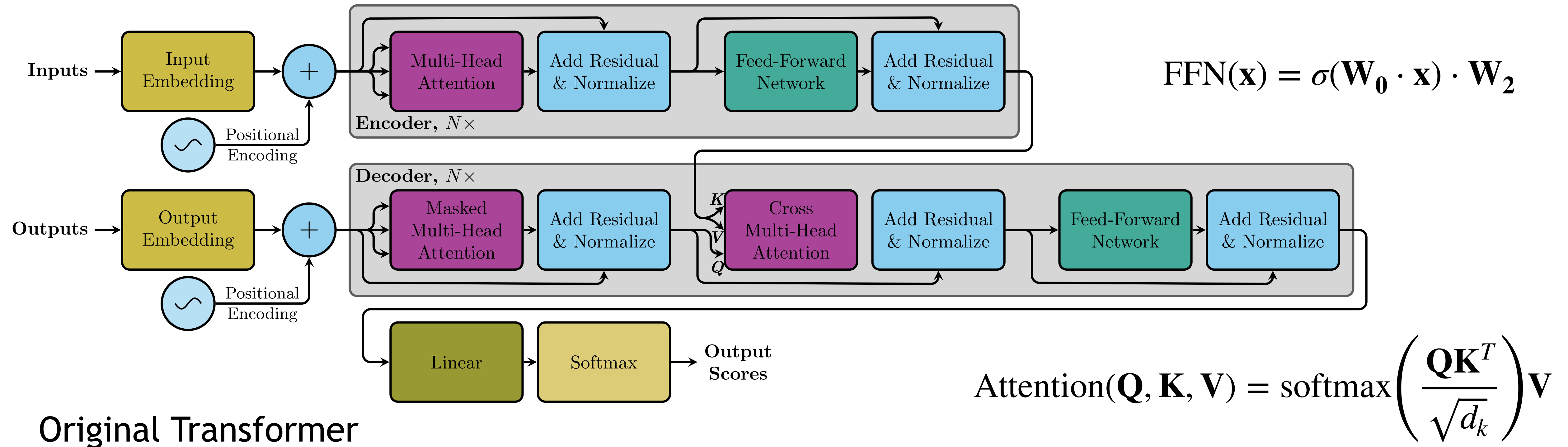
WHY REALITY BENDS IT: THREE DEVIATIONS

$$P_{\text{total}}(f) = \underbrace{P_{\text{dyn,compute}}(V(f), f)}_{\text{useful switching}} + \underbrace{P_{\text{dyn,overhead}}(V(f), f)}_{\text{clock/control}} + \underbrace{P_{\text{static}}(V(f), T_{\text{die}})}_{\text{leakage}} + \underbrace{P_{\text{uncore+mem}}}_{\text{weakly tied to SM } f}$$

1. Useful work
2. Clock/Control: DVFS requires $V = V(f)$ (the V^2 tax at high clocks)
3. Leakage: time-proportional power (the low-clock penalty)
4. Bottlenecks/overheads: $\text{Perf}(f)$ saturates (wasted watts at high clocks)
=> U-shaped energy-frequency behavior

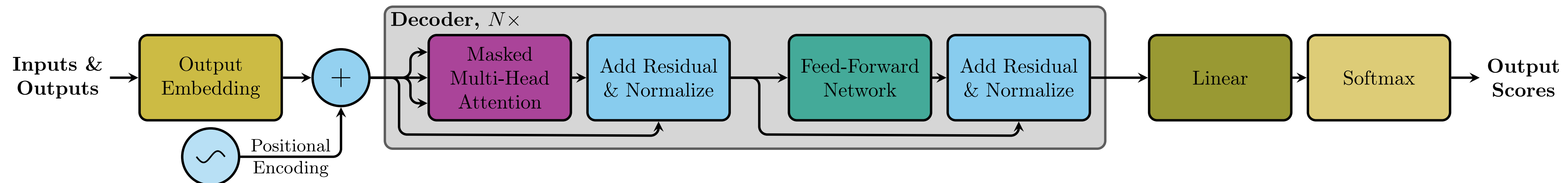


DECODER LAYER AS THE REPEATABLE UNIT



Original Transformer

Decoder-only Transformer



**MICRO-SCALE: ENERGY-TO-COMPLETION
+ GPU DVFS FOR DECODER LAYERS**

EXPERIMENTAL SETUP

Use NVIDIA Management Library (NVML) to set SM clocks and to measure power consumption

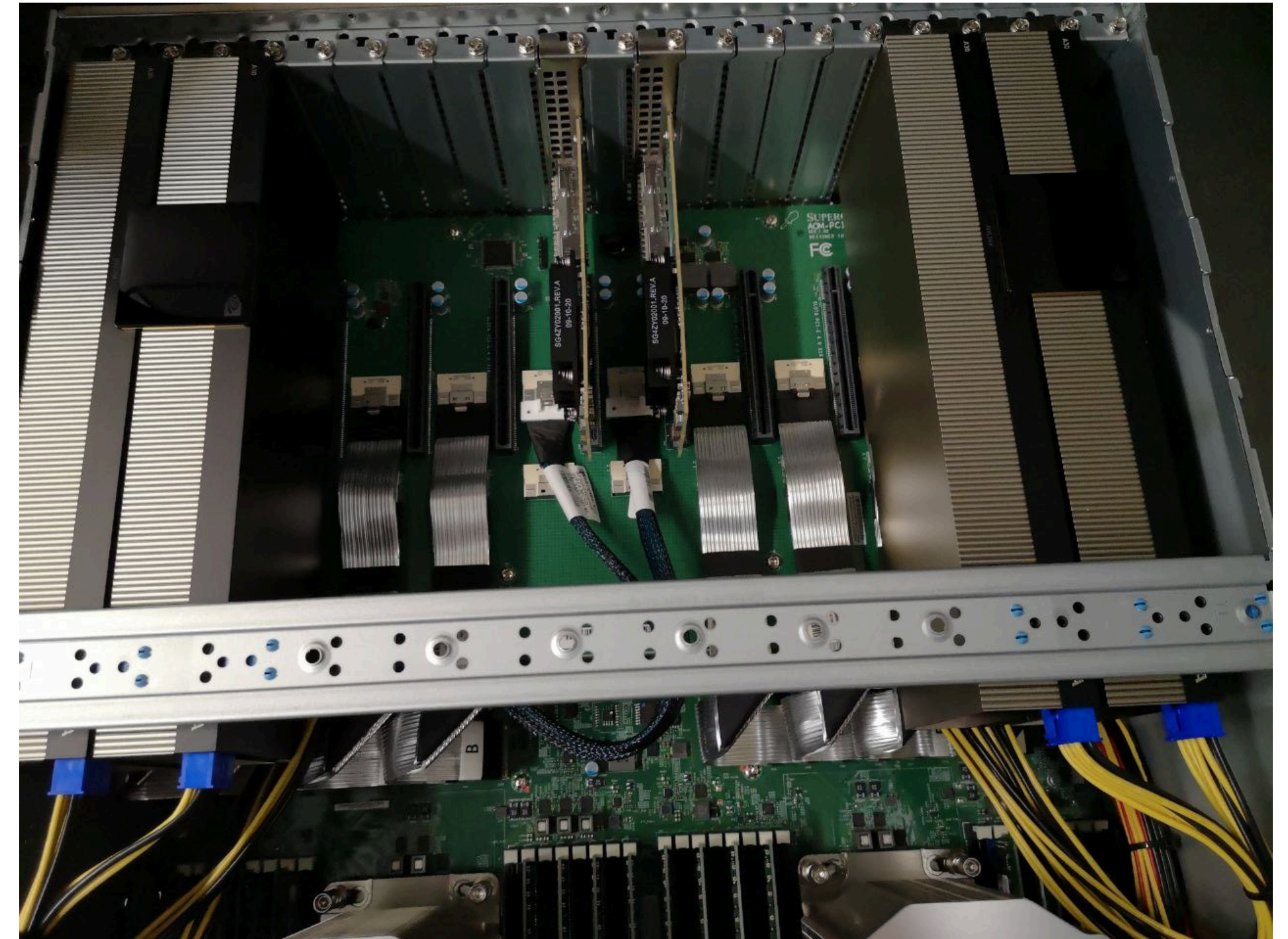
`nvmlDeviceGetSamples` is used with `NVML_TOTAL_POWER_SAMPLES` to get most recent power samples at ~2ms resolution

SM clock: from 210 to 1440 MHz in 15 MHz steps

HBM clock is fixed at 1215 MHz

Energy-to-solution is obtained by running a transformer layer in a loop until a stable runtime and power reading is obtained,

then $E = \overline{P} \cdot t$



Rivulet: 4 nodes, each 4x NVIDIA A30 GPU

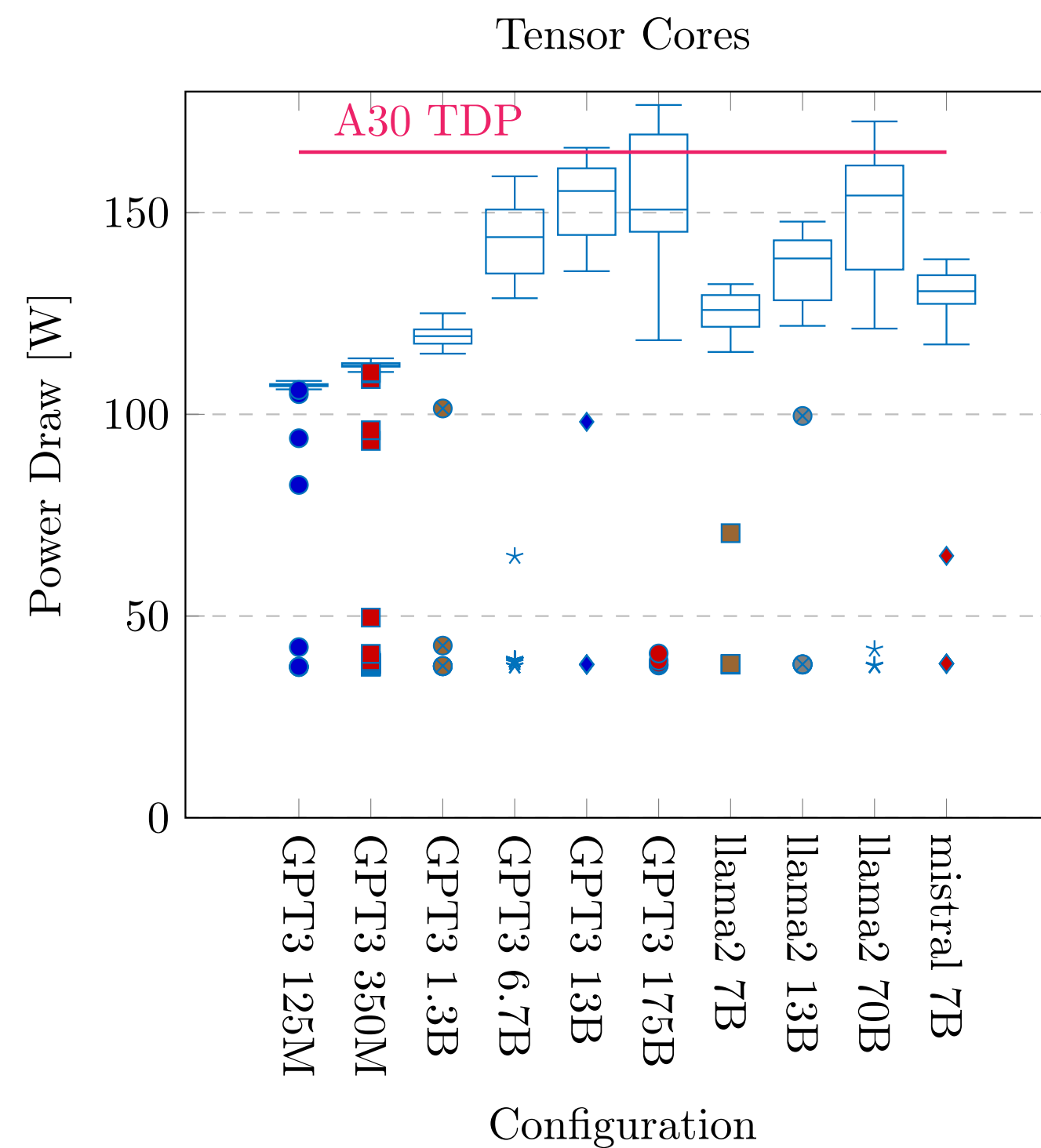
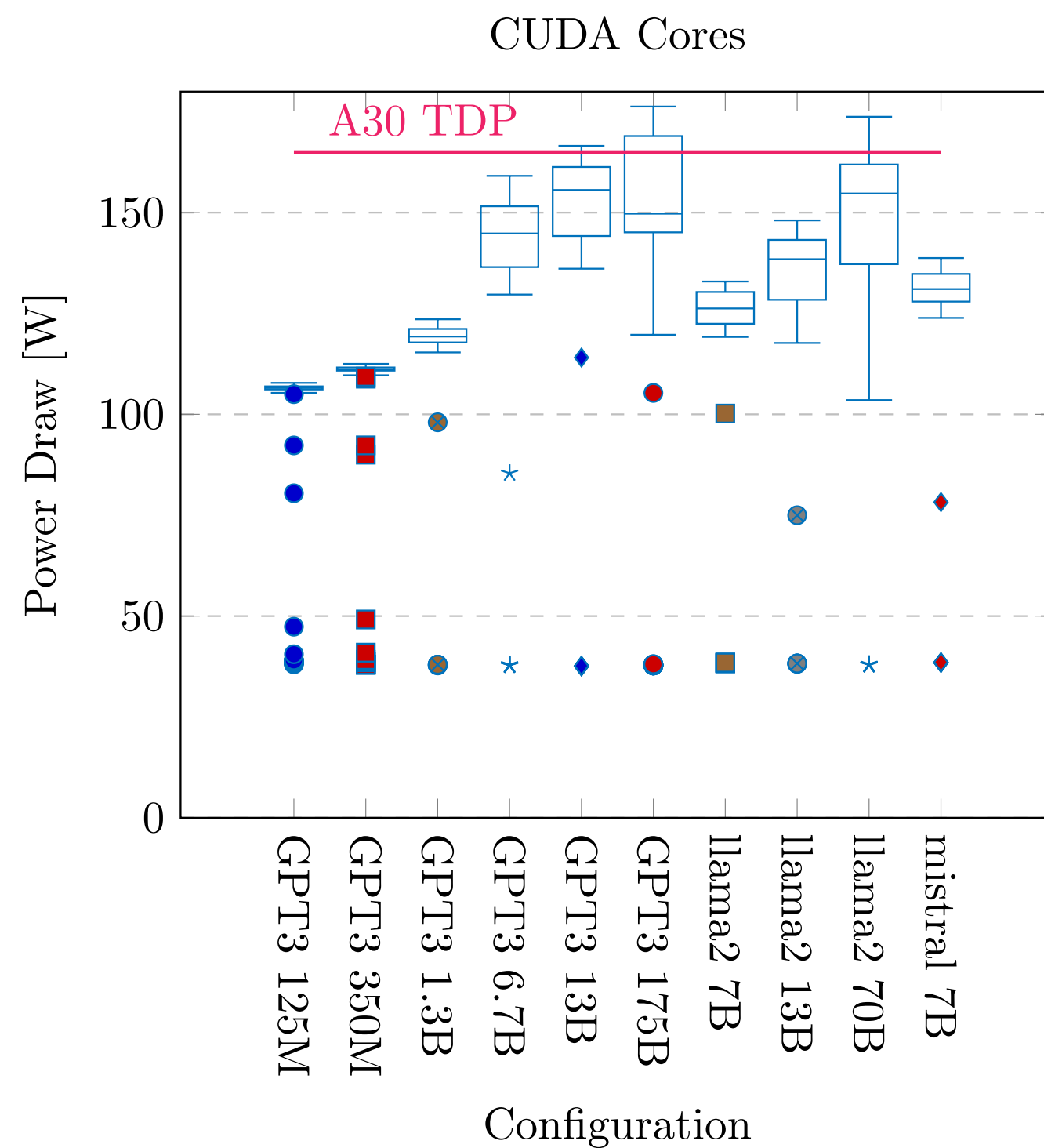
Gen3 TensorCores: 82 TFLOP/s TF32, 165 TFLOP/s FP16

24GB HBM2 at 933GB/s

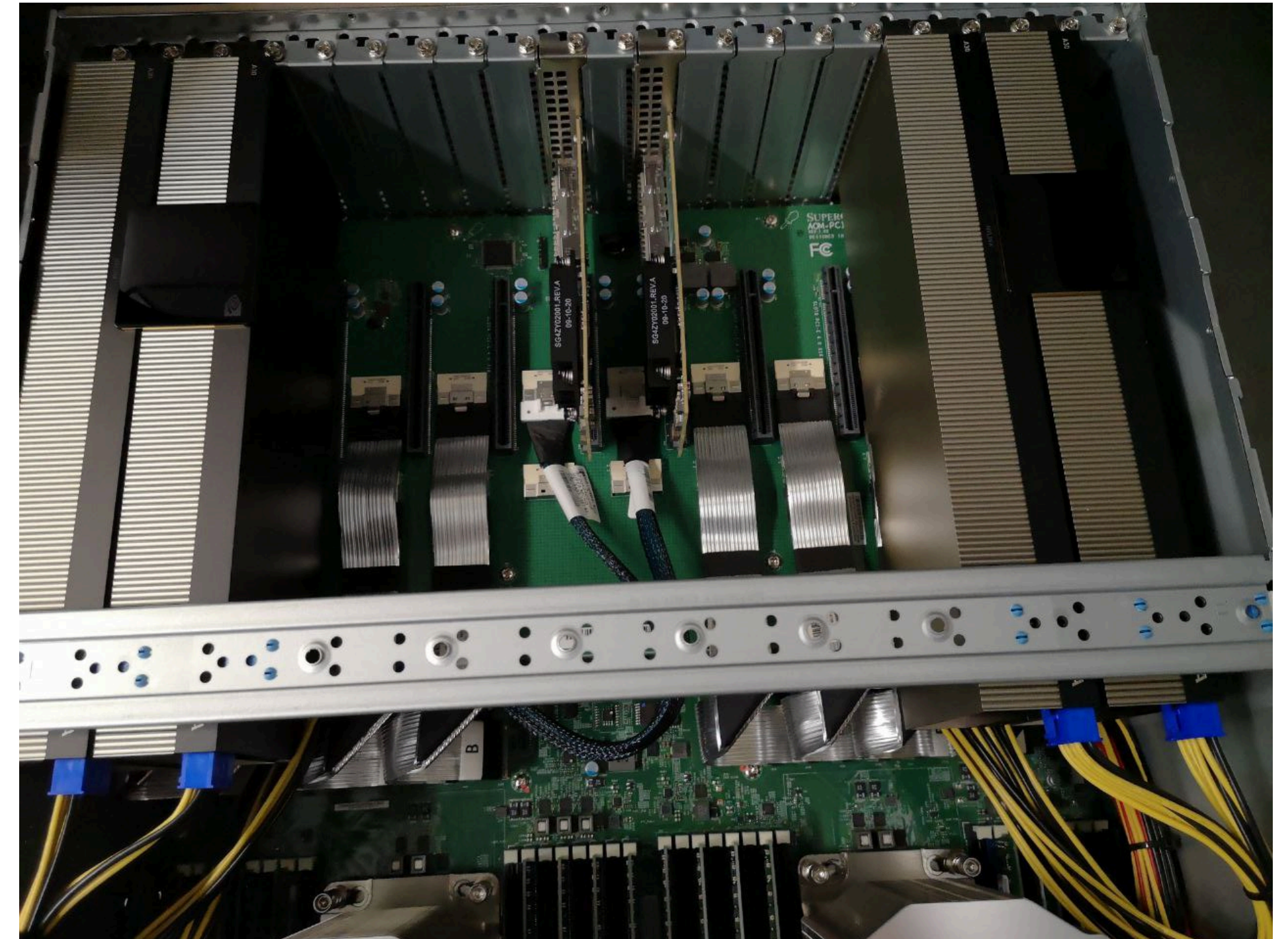
165W TDP

200GB/s NVLINK (for two neighboring GPUs)

EXPERIMENTAL SETUP



Example power measurement for various problem sizes at 1035 MHz



Rivulet: 4 nodes, each 4x NVIDIA A30 GPU

Gen3 TensorCores: 82 TFLOP/s TF32, 165 TFLOP/s FP16

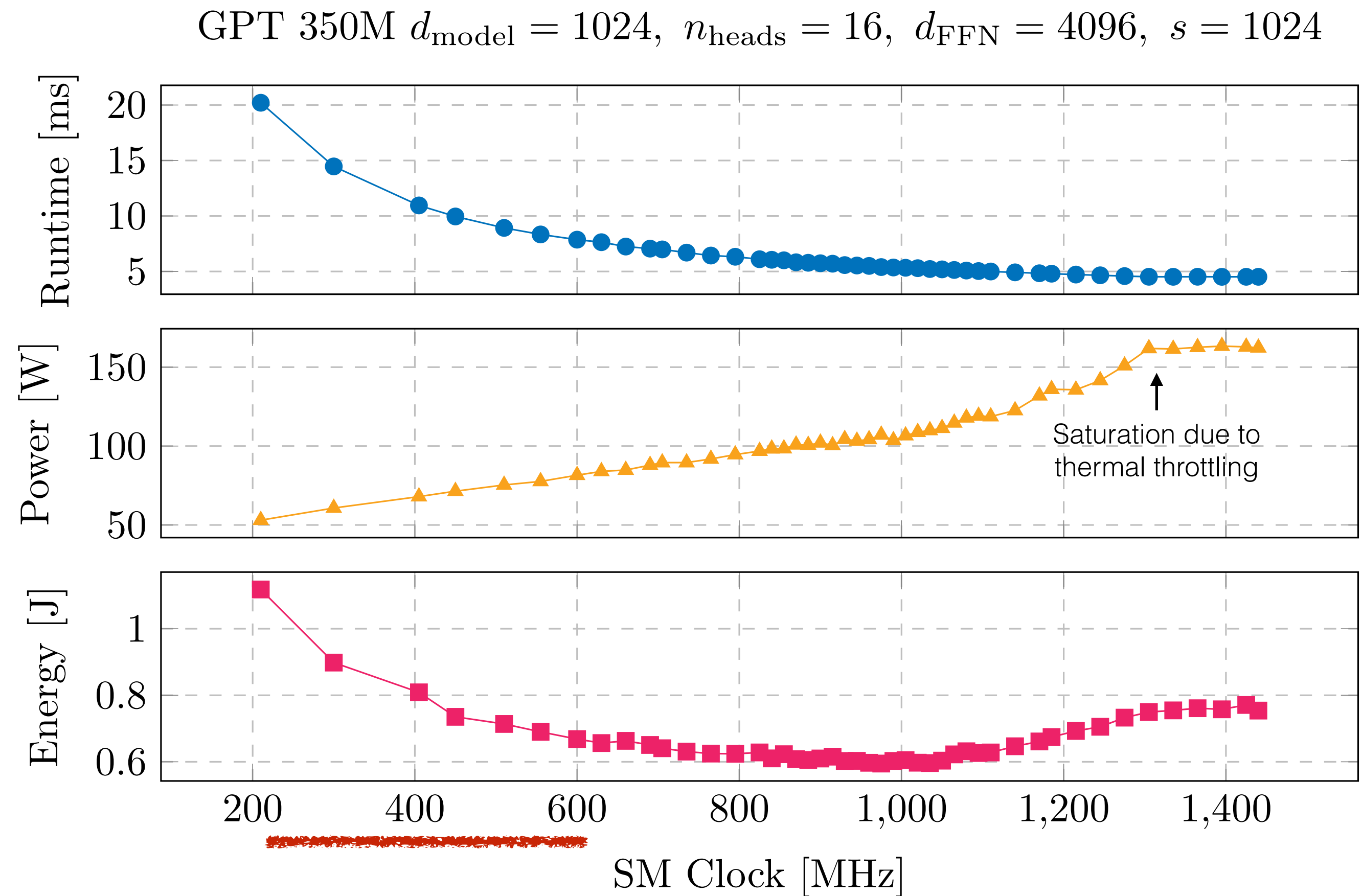
24GB HBM2 at 933GB/s

165W TDP

200GB/s NVLINK (for two neighboring GPUs)

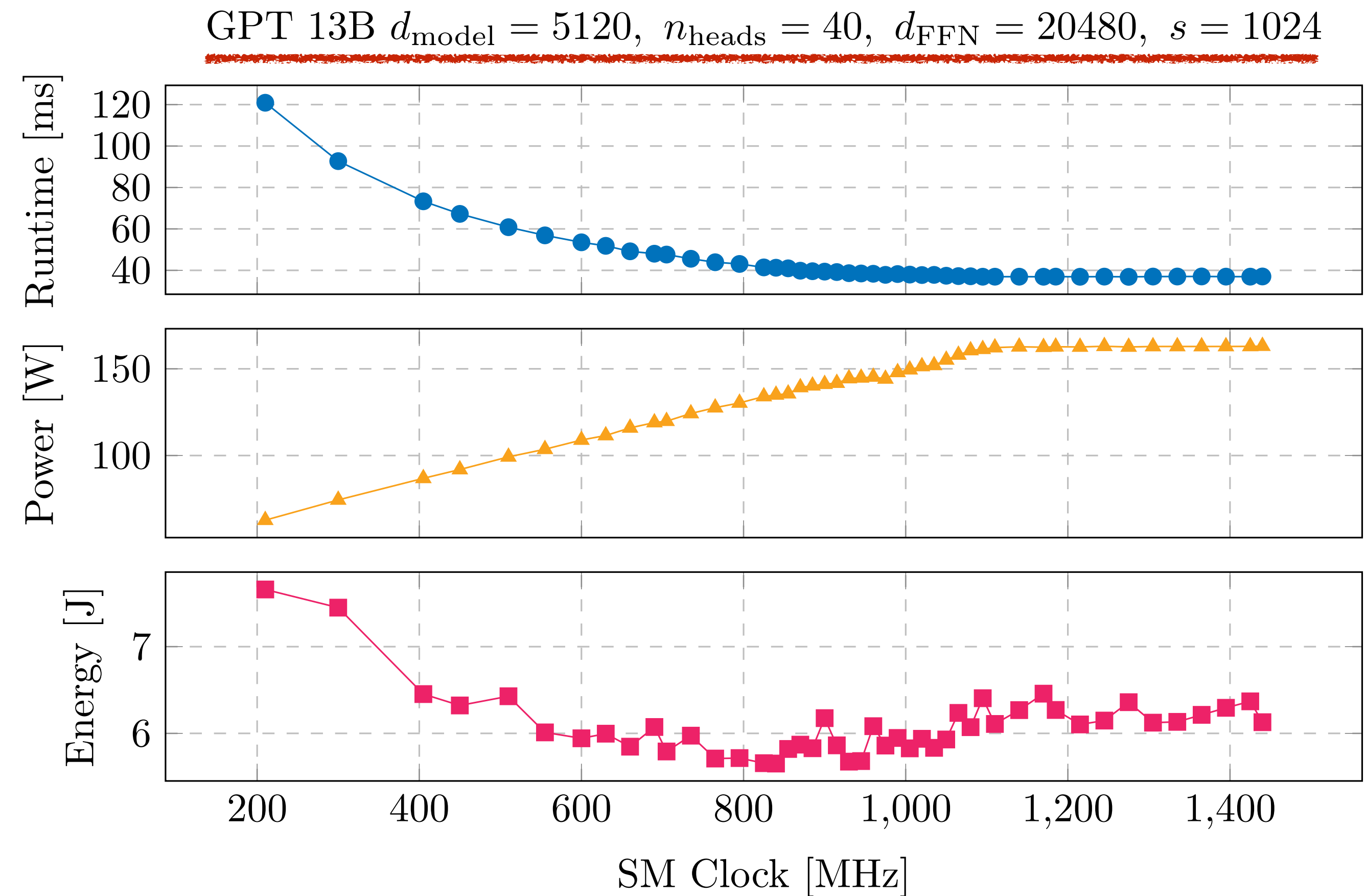
OBSERVATION 1: U-SHAPED ENERGY PER LAYER

1. If SM clock is too low runtime increases and thereby effects of static power $\rightarrow$ energy grows disproportionally
2. Mid-range clock settings offer the best tradeoff
3. If SM clock is too high power consumption increases disproportionally $\rightarrow$ energy grows disproportionally again

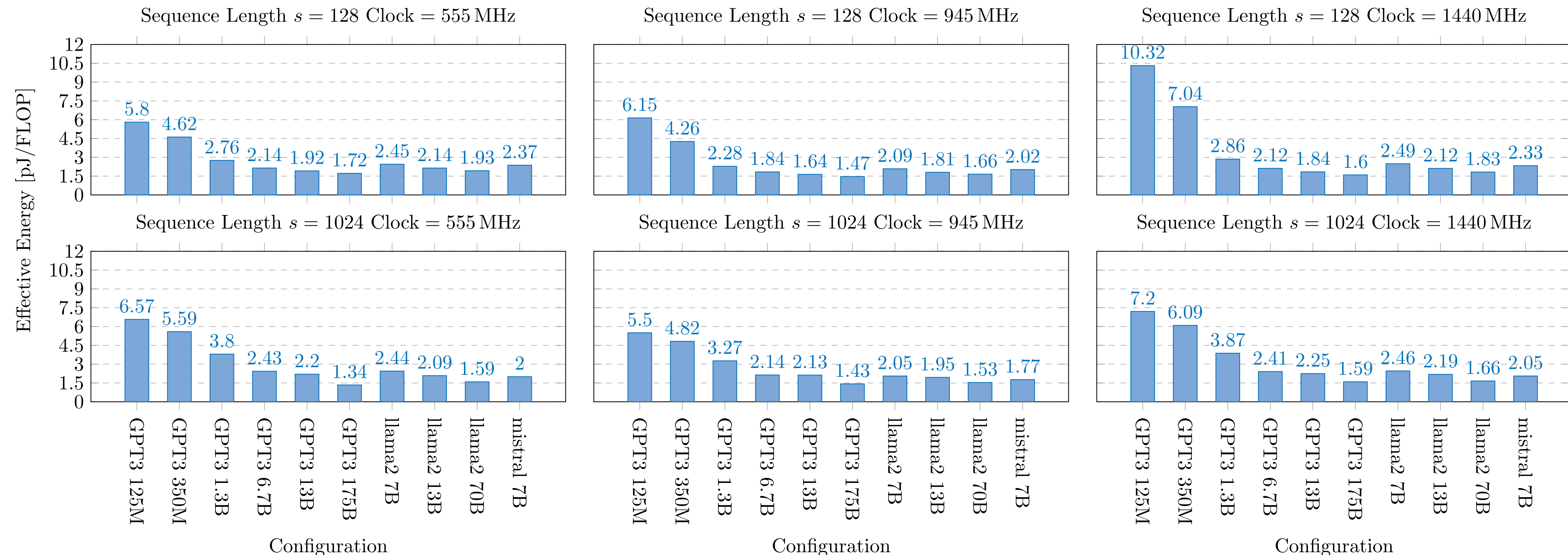


OBSERVATION 1: U-SHAPED ENERGY PER LAYER

1. If SM clock is too low runtime increases and thereby effects of static power $\rightarrow$ energy grows disproportionately
2. Mid-range clock settings offer the best tradeoff
3. If SM clock is too high power consumption increases disproportionately $\rightarrow$ energy grows disproportionately again
4. The optimal setting depends on workload properties



OBSERVATION 2: LAYER SIZE & EFFICIENCY



Larger layer configurations offer better “effective” energy per operation, i.e. energy required for the calculation including data/instruction fetch, scheduling, and baseline power draw

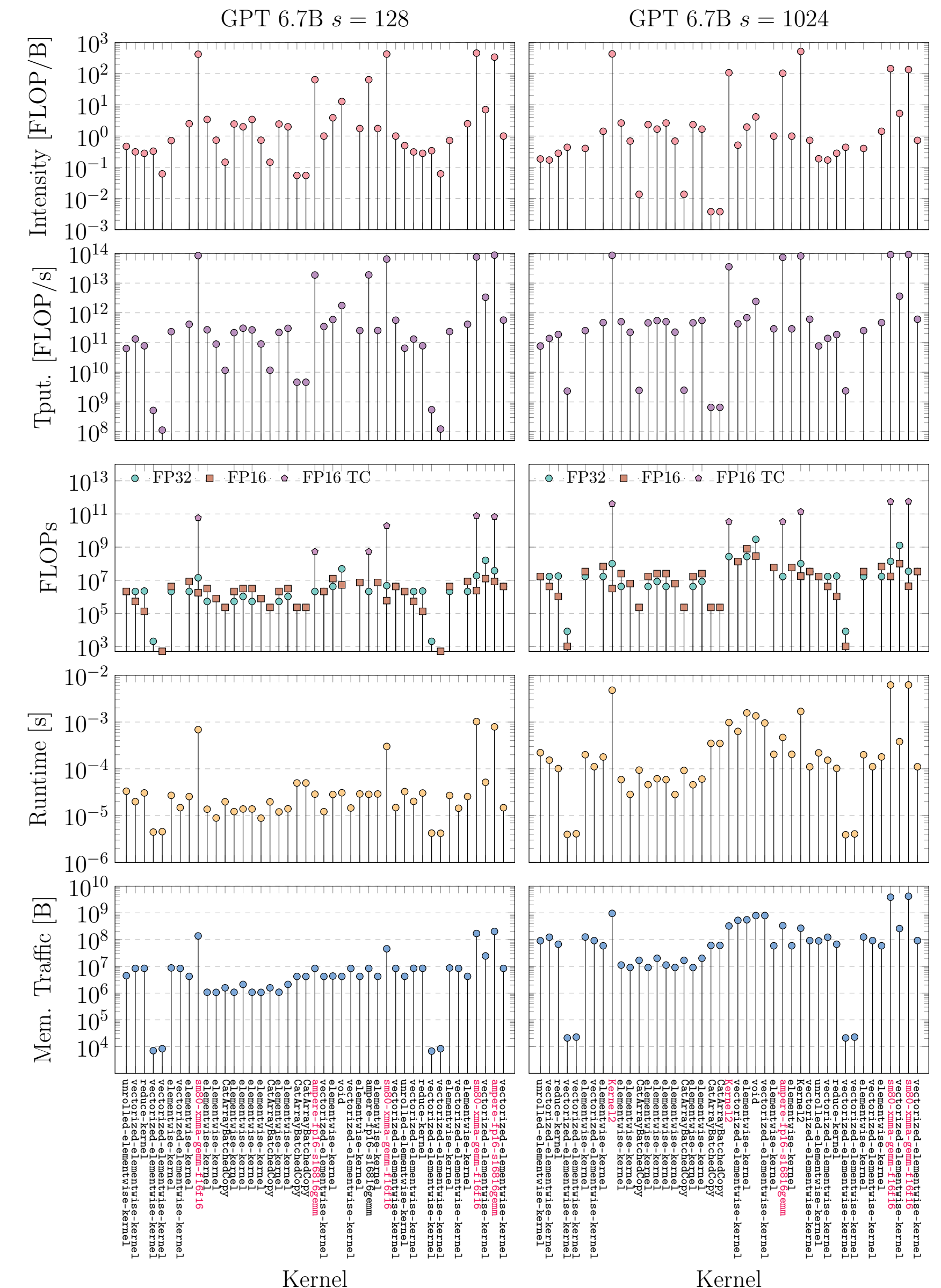
OBSERVATION 3: SEQUENCE LENGTH SHIFTS THE OPTIMUM

Sequence length is very important for energy per layer

Larger sequences result in more time being spent in low intensity memory heavy kernels that don't benefit from faster compute clocks

Compute bound kernels achieve very similar throughput in both cases

=> There is no single best clock for a layer, it depends on the input, too

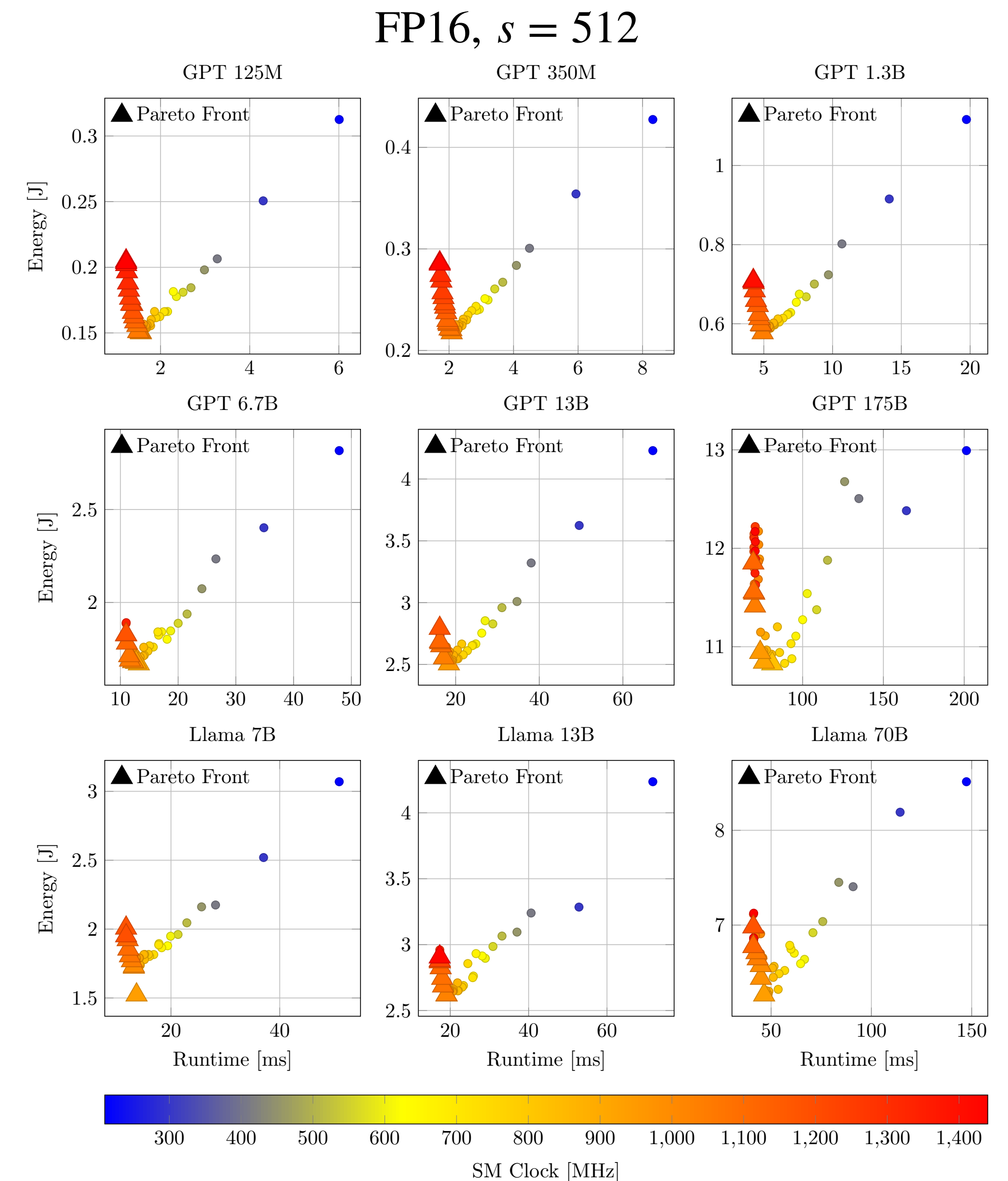


ENERGY VS RUNTIME PARETO FRONTS ACROSS MODELS

Optimizing for both energy and runtime

Characteristic “V”-shape for smaller layers that softens to “U”-shape for larger ones

Fastest runtime is not necessarily at highest clock



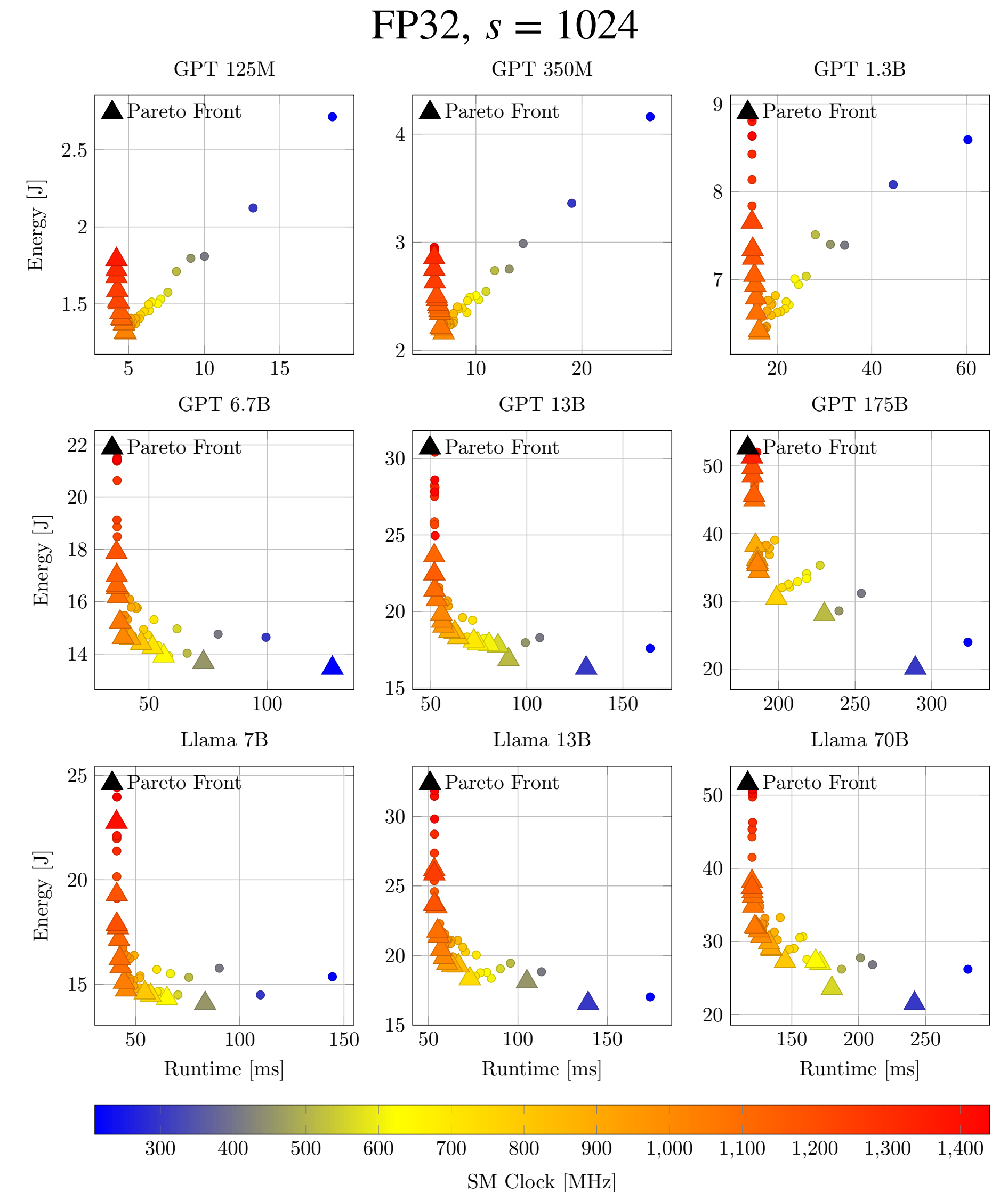
ENERGY VS RUNTIME PARETO FRONTS ACROSS MODELS

Optimizing for both energy and runtime

Characteristic “V”-shape for smaller layers that softens to “U”-shape for larger ones

Fastest runtime is not necessarily at highest clock

Softens even more for longer sequences and larger datatypes



PRACTITIONER TAKEAWAY: ~10-20% ENERGY SAVINGS FOR SIMILAR SLOWDOWN

Table shows comparison of fastest Pareto optimal runtime to lowest Pareto optimal energy

Again: fastest optimal runtime is not necessarily at highest clock

Relative energy savings and runtime penalties are often very close

Energy saving compared to maximum clock speed can be substantially higher for larger models

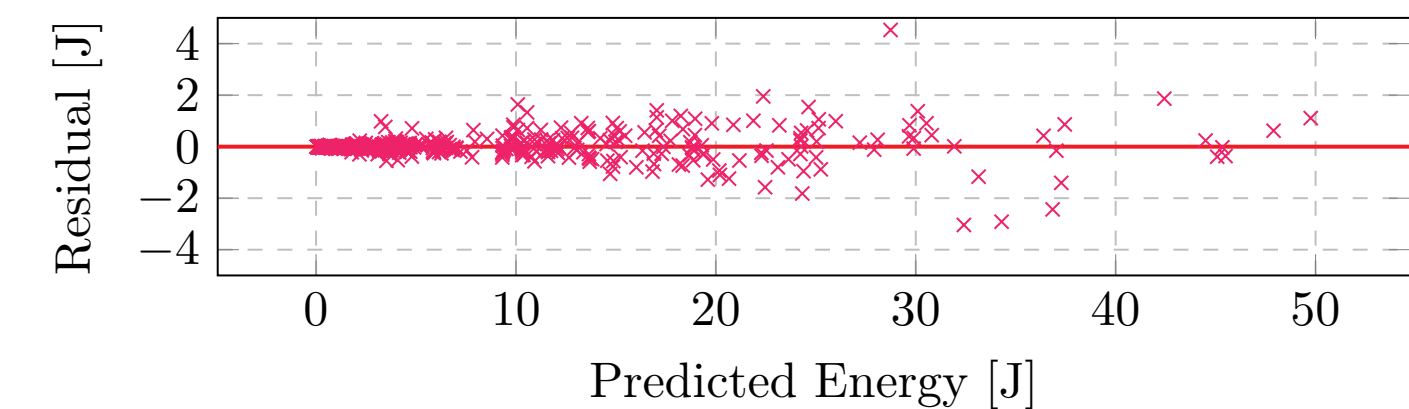
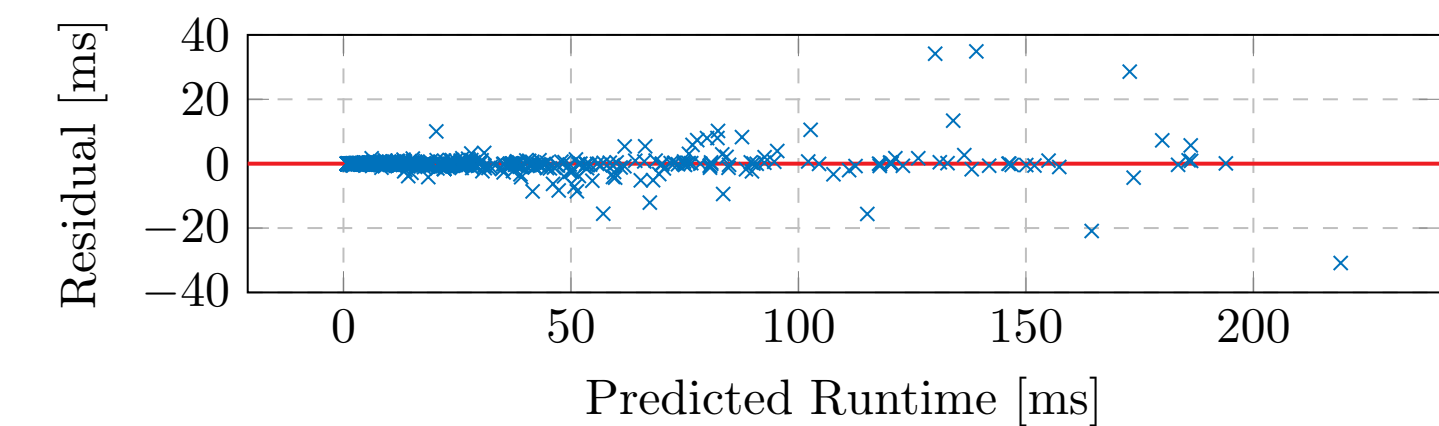
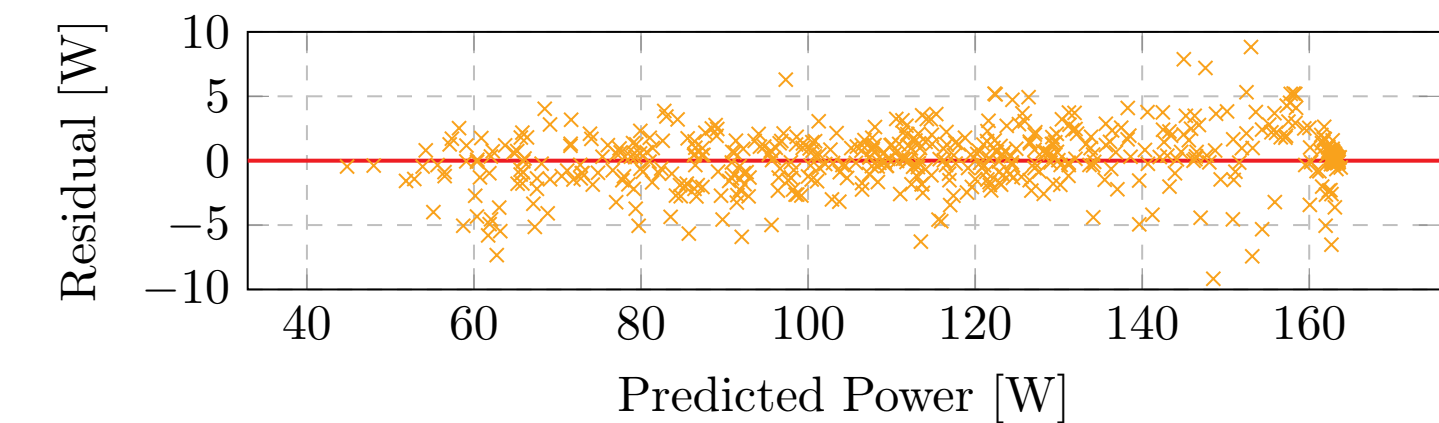
Layer	S	Energy Saving [%]	Runtime Penalty [%]
GPT 125M	128	5.3	1.6
	512	30.8	23.9
	1024	28.1	18.3
GPT 1.3B	128	24.5	27.4
	512	20.5	14.8
	1024	21.0	12.8
GPT 13B	128	14.6	14.9
	512	10.8	12.7
	1024	11.8	11.3
GPT 175B	128	13.5	11.3
	512	9.0	15.5
	1024	19.9	30.7

FEASIBILITY STUDY: PREDICTION MODEL

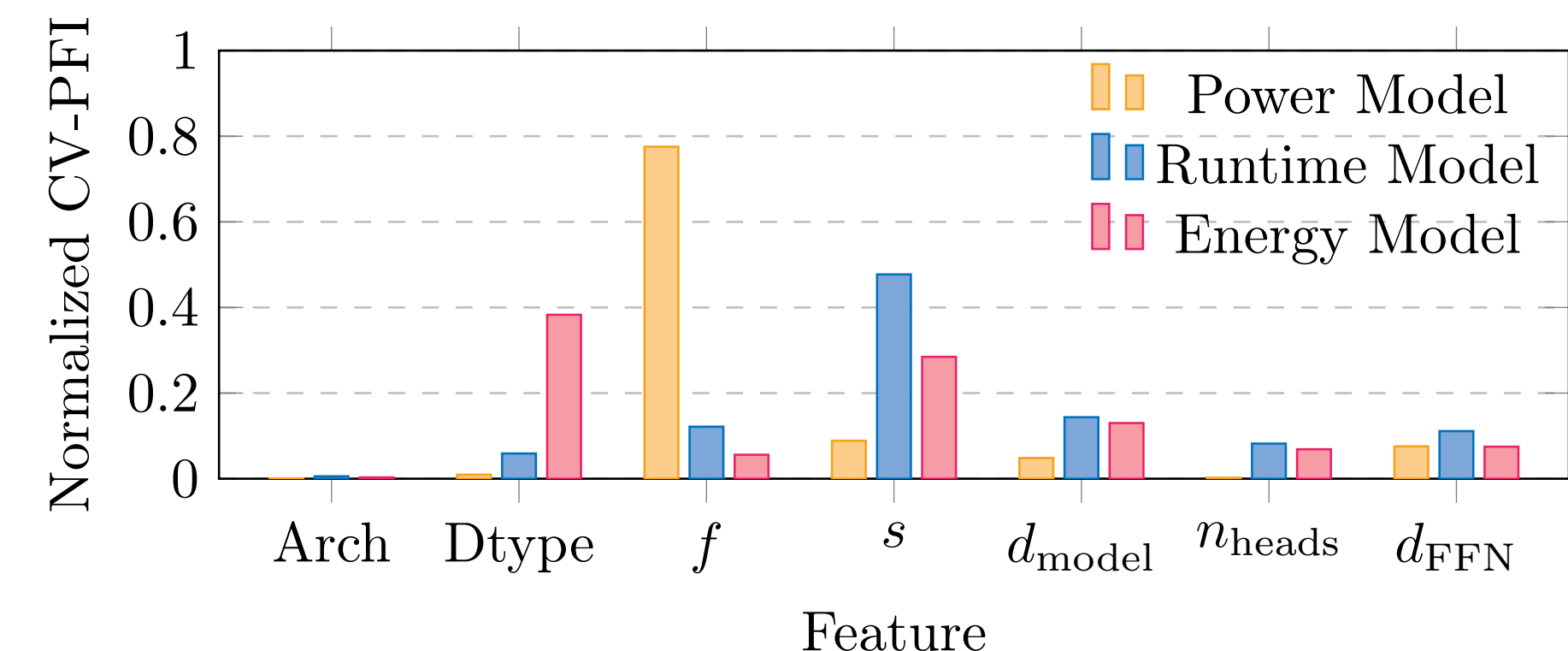
Simple Random Forest predictor achieves acceptable accuracy

Training data shifted to small values, more training data required

Feature importance tracks with our intuitions and observations



Error	Power	Runtime	Energy
RMSE	2.2W	3.6ms	0.5J
MAPE	1.5%	3.5%	3.0%
MAE	1.6W	1.1ms	0.2J
R ²	0.9965	0.9919	0.9974



MACRO-SCALE: MODELING LARGE-SCALE LLM TRAINING TIME

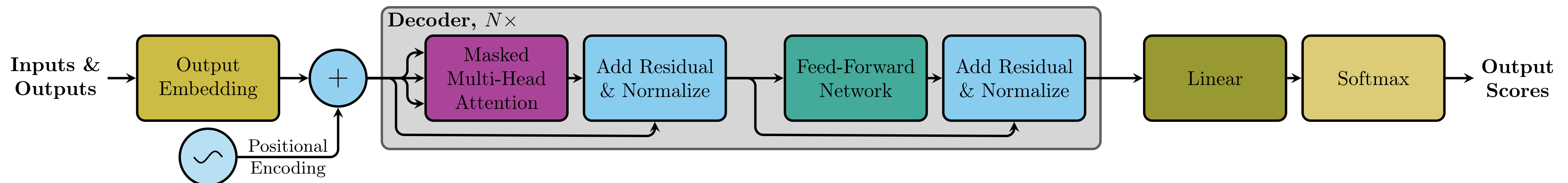
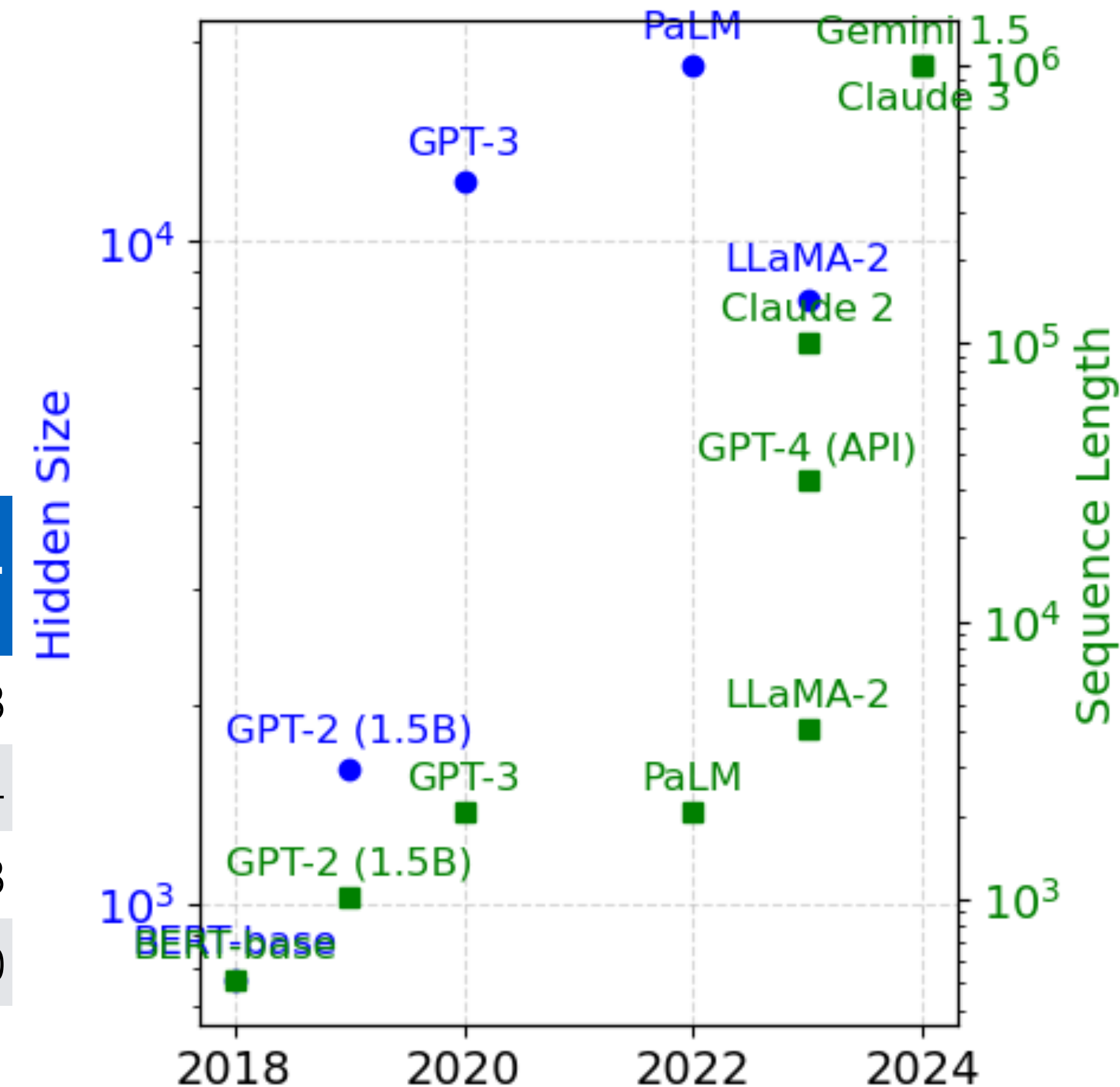
FROM LAYER ENERGY TO RUN ENERGY

Training LLMs is easy given 4k+ GPUs

Stacked decoder blocks

Plethora of training data

Model	Sequence length	Hidden Size	FFN Size	Layers	Heads	Vocab Size	Head Dim.	Hidden Act.	Year
Qwen/Qwen-72B	32768	8192	49152	80	64	152064	128	SwiGLU	2023
Mistral L1 2411	131072	12288	28672	88	96	32768	128	silu	2024
Megatron-1T	2048	25600	49152	128	160	51200	160	gelu	2023
GPT-3 175B	2048	12288	49152	96	96	50257	128	gelu	2020



WHY MODELING MATTERS - GRAPHCULON

Question: can we model the behavior of these 4k+ GPUs?

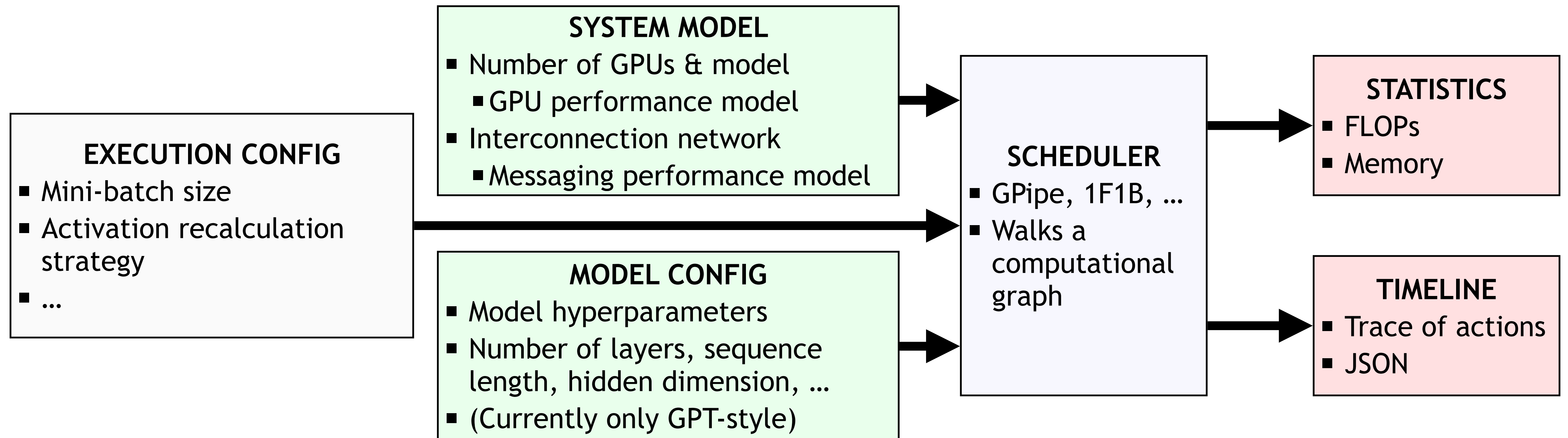
LLMs: one iteration perfectly predictable

Performance model: LLMs mainly based on linear algebra + element-wise operations

Idea: “walking the computational graph”, computation and communication tasks modeled as nodes

Critical-path walk or full traversal

Model various details: 3D parallelism, (collective) communication, memory consumption (fwd + activations + bwd + optimizer)



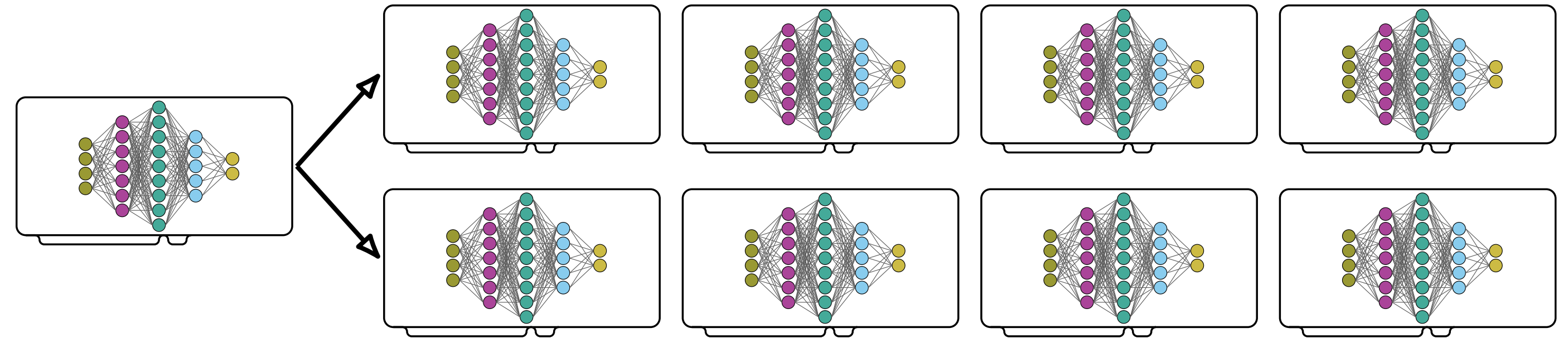
PARALLELISM TOOLBOX: DP / TP / PP

Data parallelism (DP)

Split data, not model

Different mini-batches go to different (sets of) GPUs

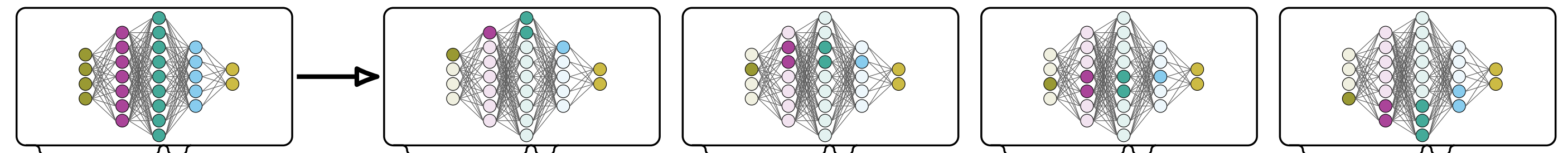
“Embarrassingly parallel”



Tensor parallelism (TP)

Split a single layer onto multiple GPUs

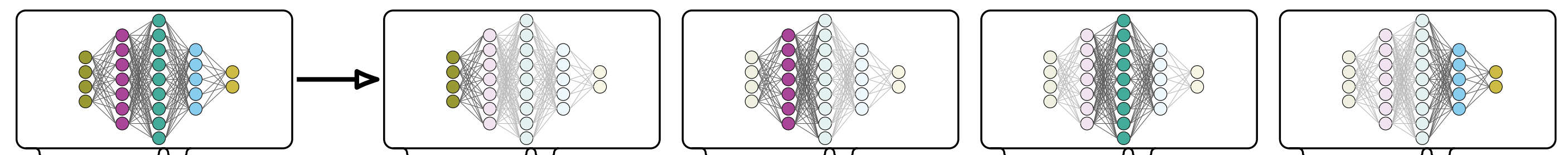
Collective communication -
allreduce & broadcast



Pipeline parallelism (PP)

Split mini-batch into micro-batches

Process multiple layers in a
pipelines fashion

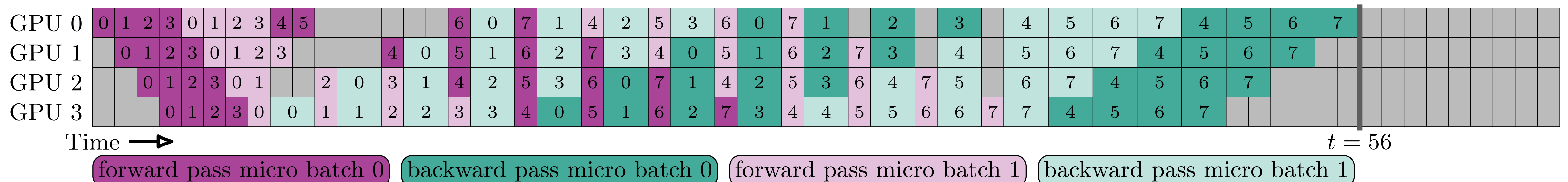
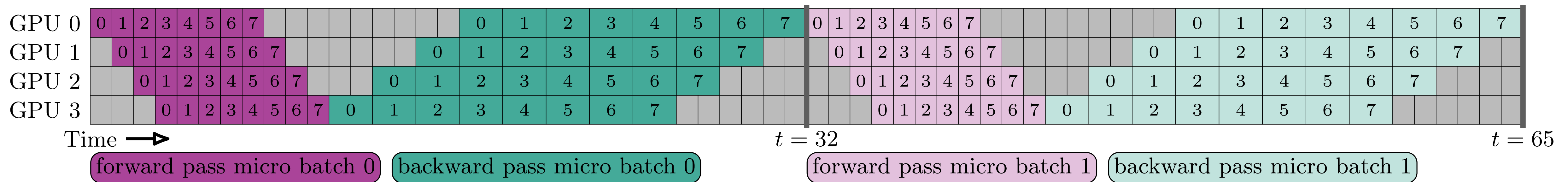


PIPELINE SCHEDULES & BUBBLE FRACTION INTUITION

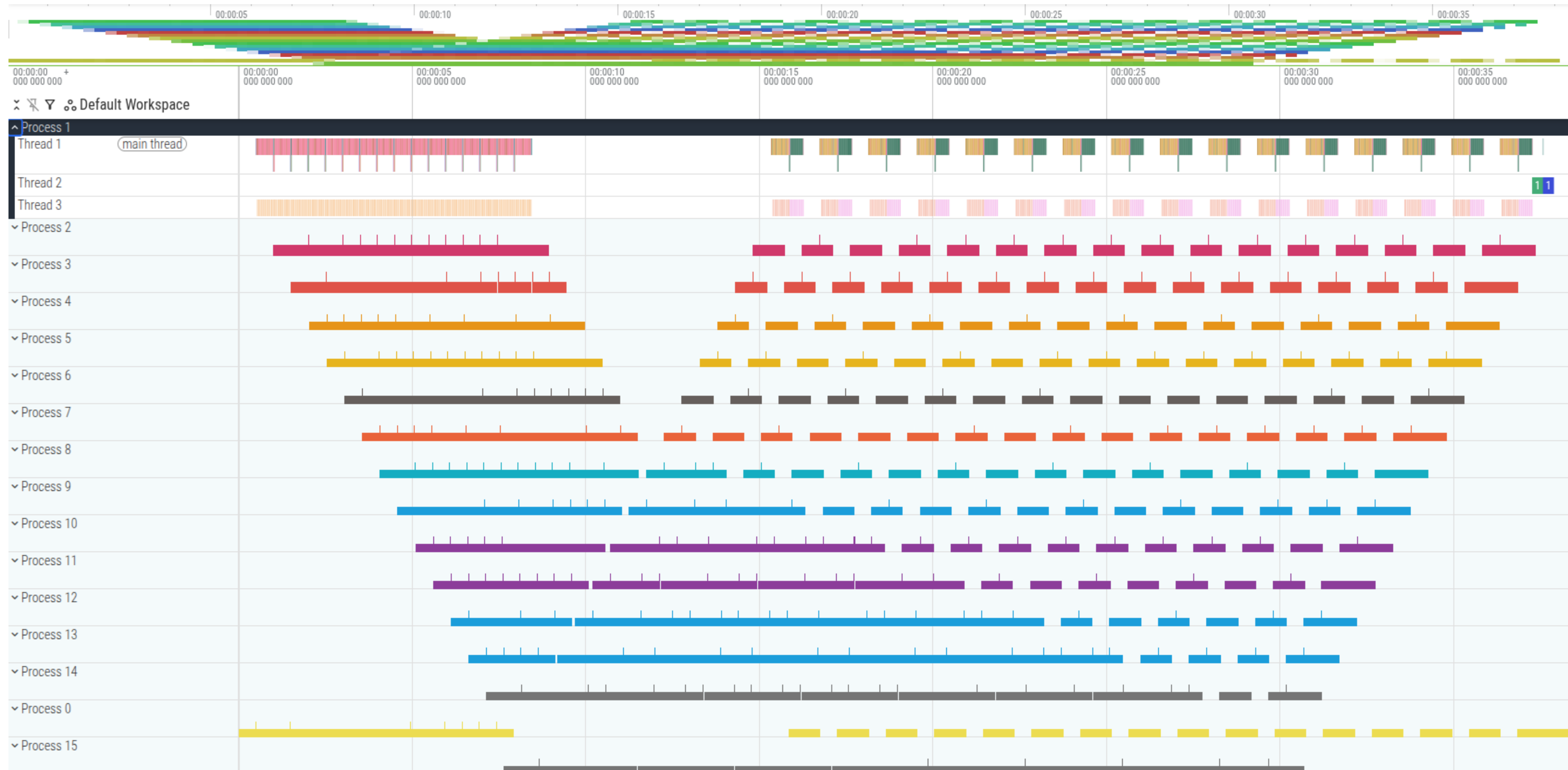
Pipelining is a powerful technique, but introduces bubbles due to synchronous SGD

GPipe as starting point, (interleaved) 1F1B as common optimization

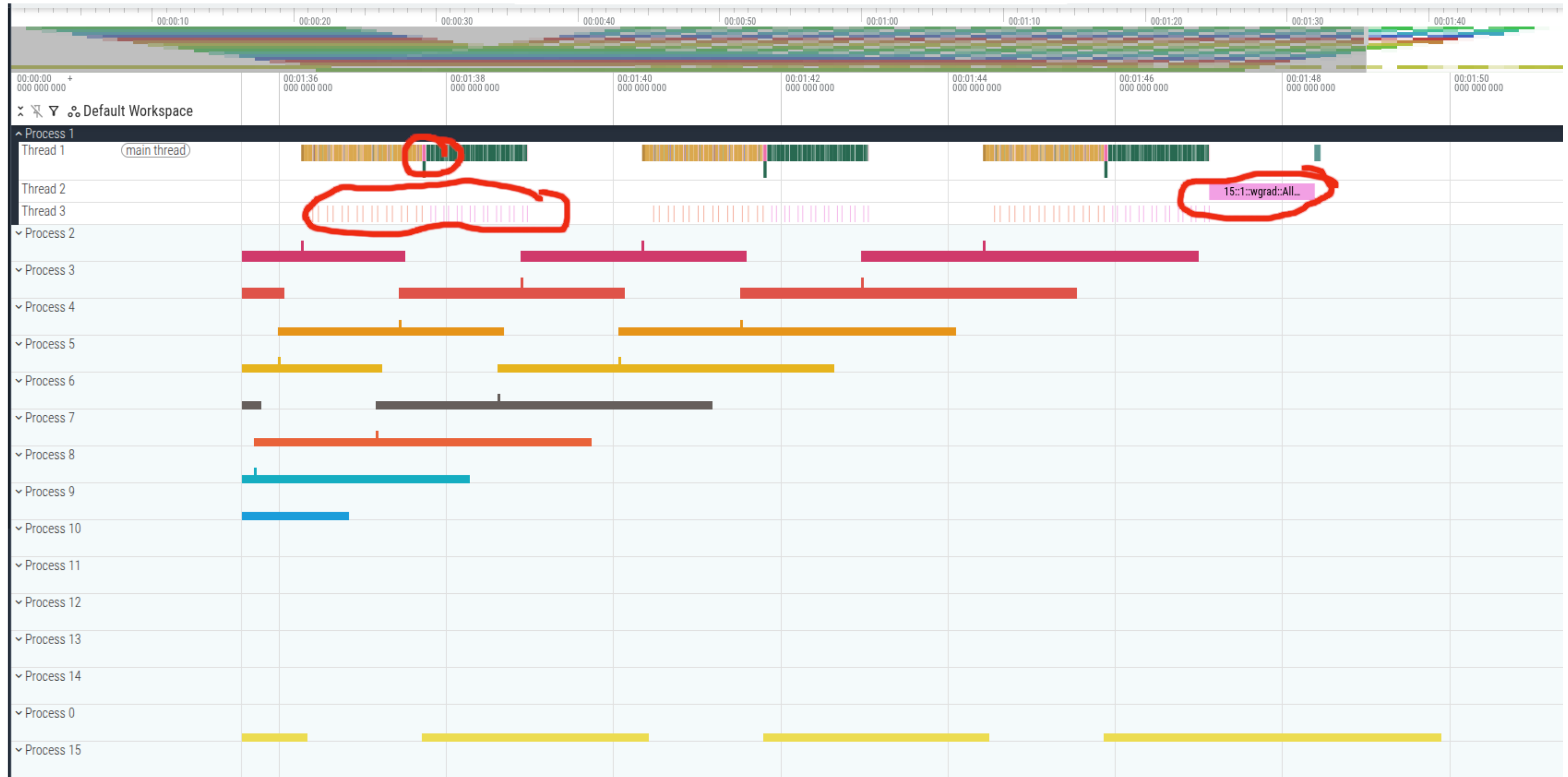
Variety of pipelines schedules proposed, with different trades (bubbles, memory, etc.)



GRAPHCULON: TIMELINE

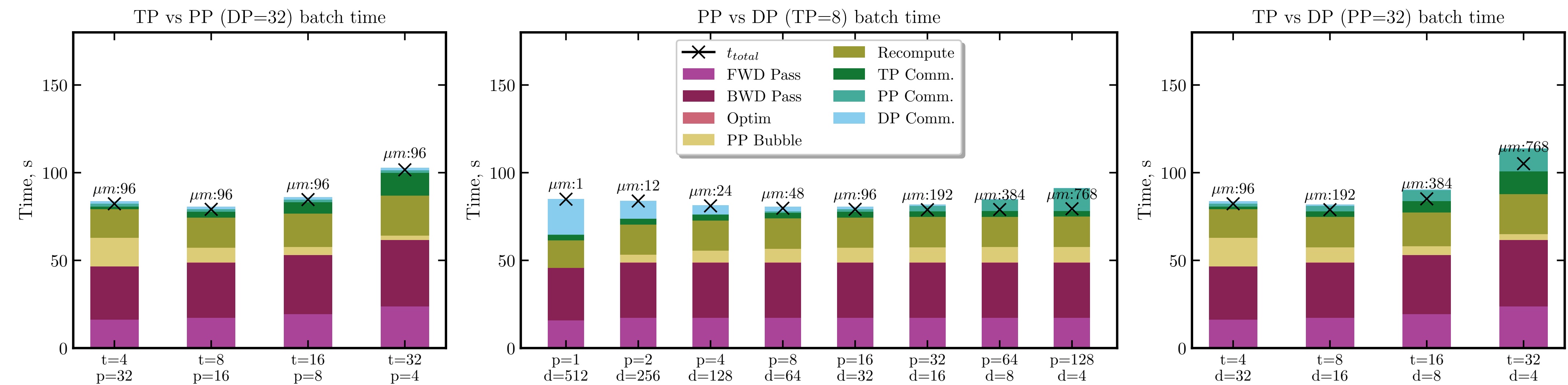


GRAPHCULON: TIMELINE



CLOSED-FORM ITERATION TIME BREAKDOWN

Megatron-1T single batch training on 4096 A100 GPUs with various parallelism strategies
($B = 3072, b_{\mu} = 1, v = 1$, recompute=full, optim sharding)



WHAT IT'S GOOD FOR TODAY

Use for trend exploration: bubble sensitivity, micro-batch scaling, where overlap could matter

Overall simulation behavior is internally consistent except for specific issues

Checks against (1) fundamentals (FLOP/parameter counts), (2) analytical model (same ballpark for 25k configurations), (3) calculon [1] results, (4) own on-premise experiments with Megatron

Current status: informal collaboration with NVIDIA research + UK-based start-up

Current goal: good enough (usability, accuracy) for publication (code+paper)

Under development, next major iteration expected mid-February

Future work

Modeling energy-to-completion

Comparing and optimizing pipeline schedules

Extensions for LLM inference - autoregressive aspects, prefill phase (initial fill) + decode phase (generation)

[1] Mikhail Isaev, Nic McDonald, Larry Dennison, and Richard Vuduc. 2023. Calculon: a methodology and tool for high-level co-design of systems and large language models. SC '23. <https://doi.org/10.1145/3581784.3607102>

WRAPPING UP

ENERGY-EFFICIENT SYSTEMS AND AI LAB (ESAIL)

Explore optimality

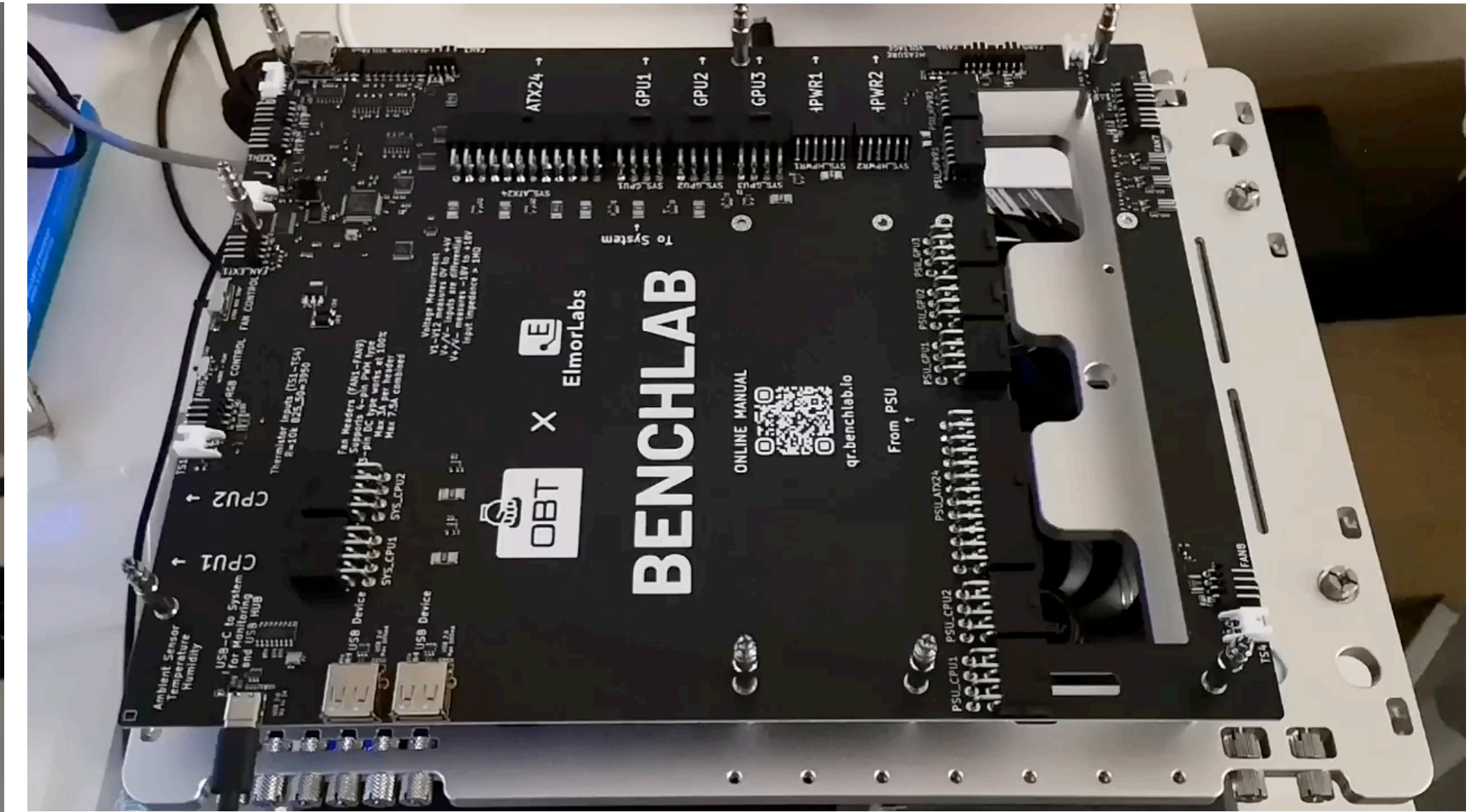
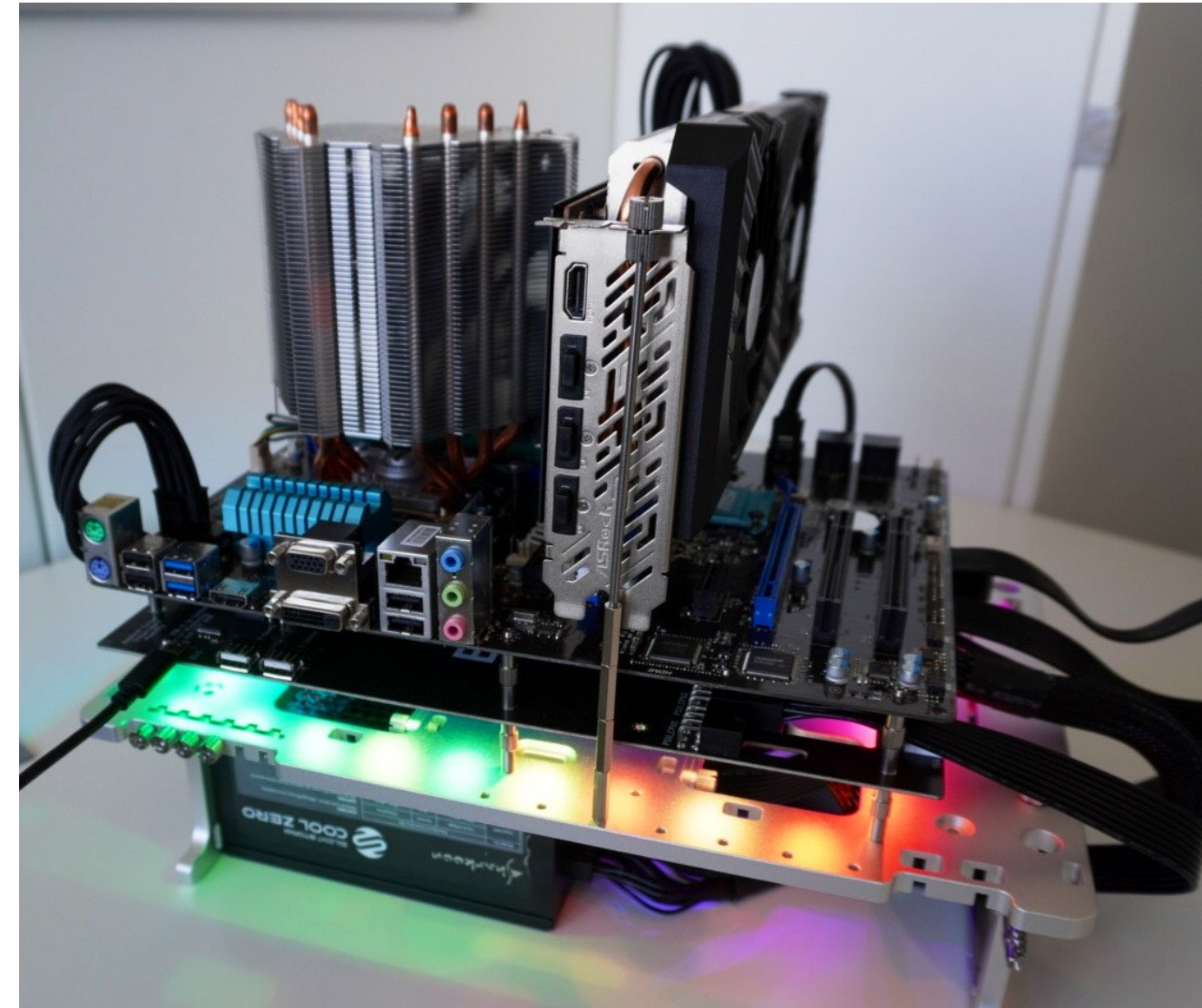
GPU type (consumer/
server)

DVFS

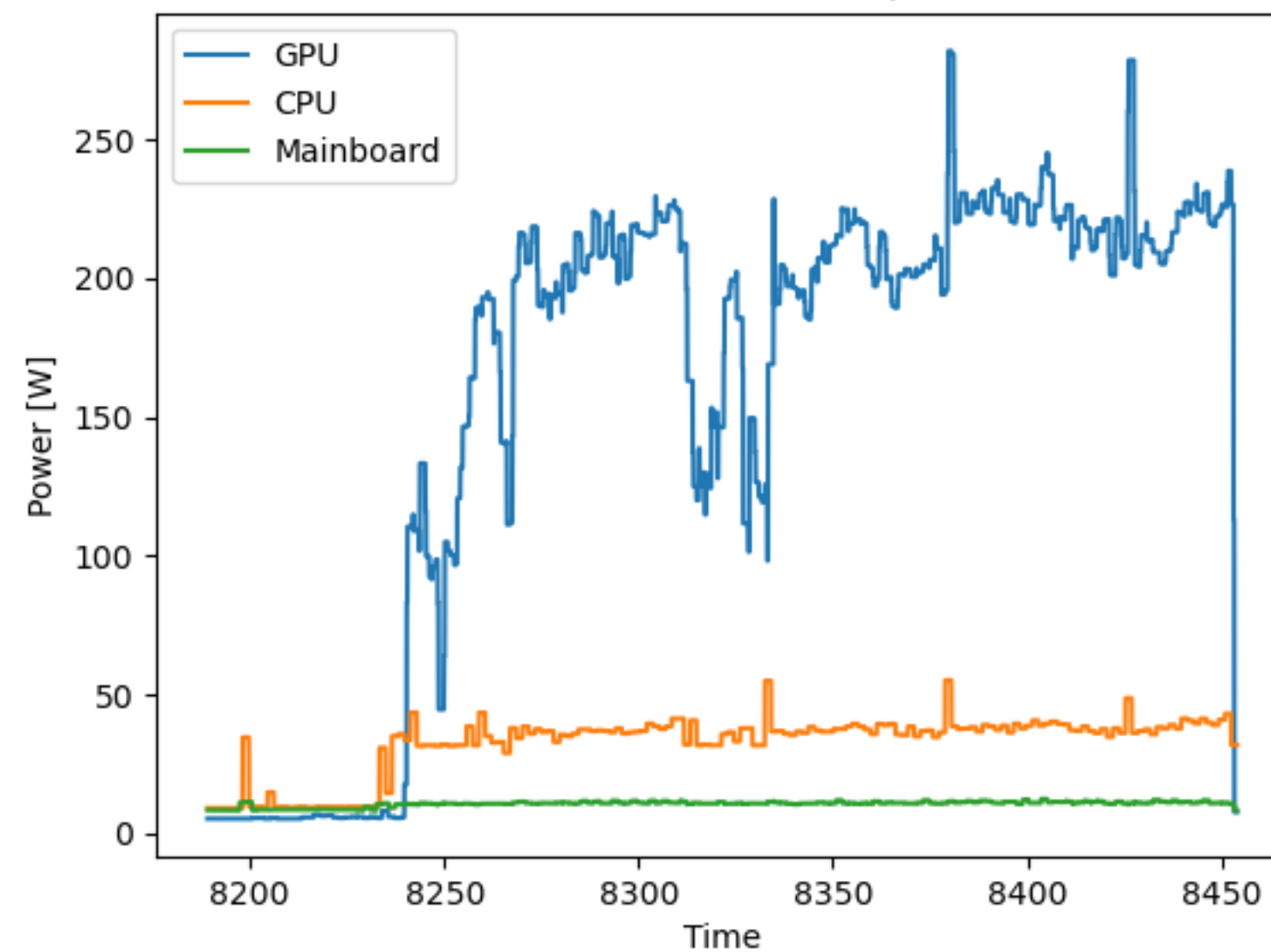
Other accelerators

System overhead

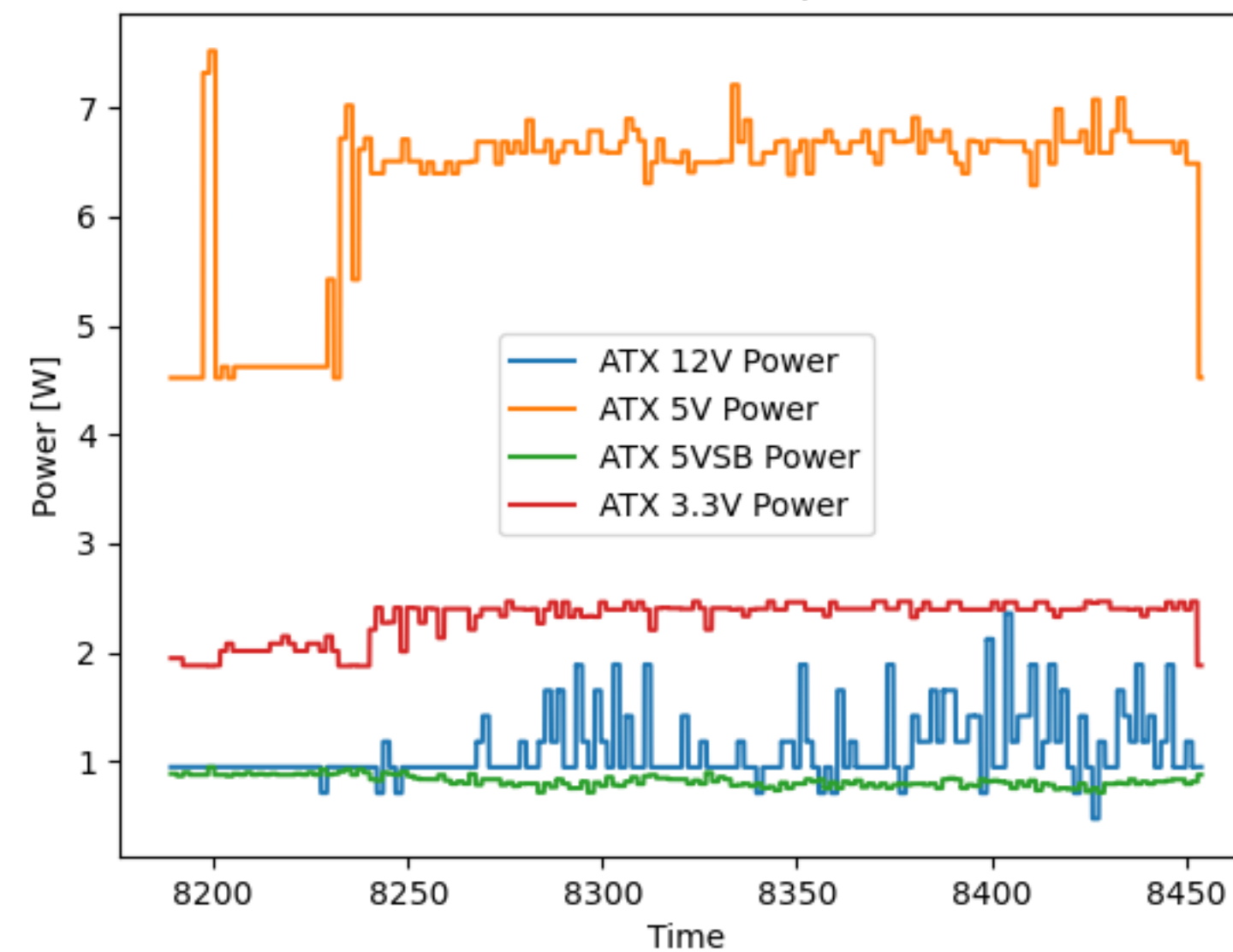
4x for comparisons



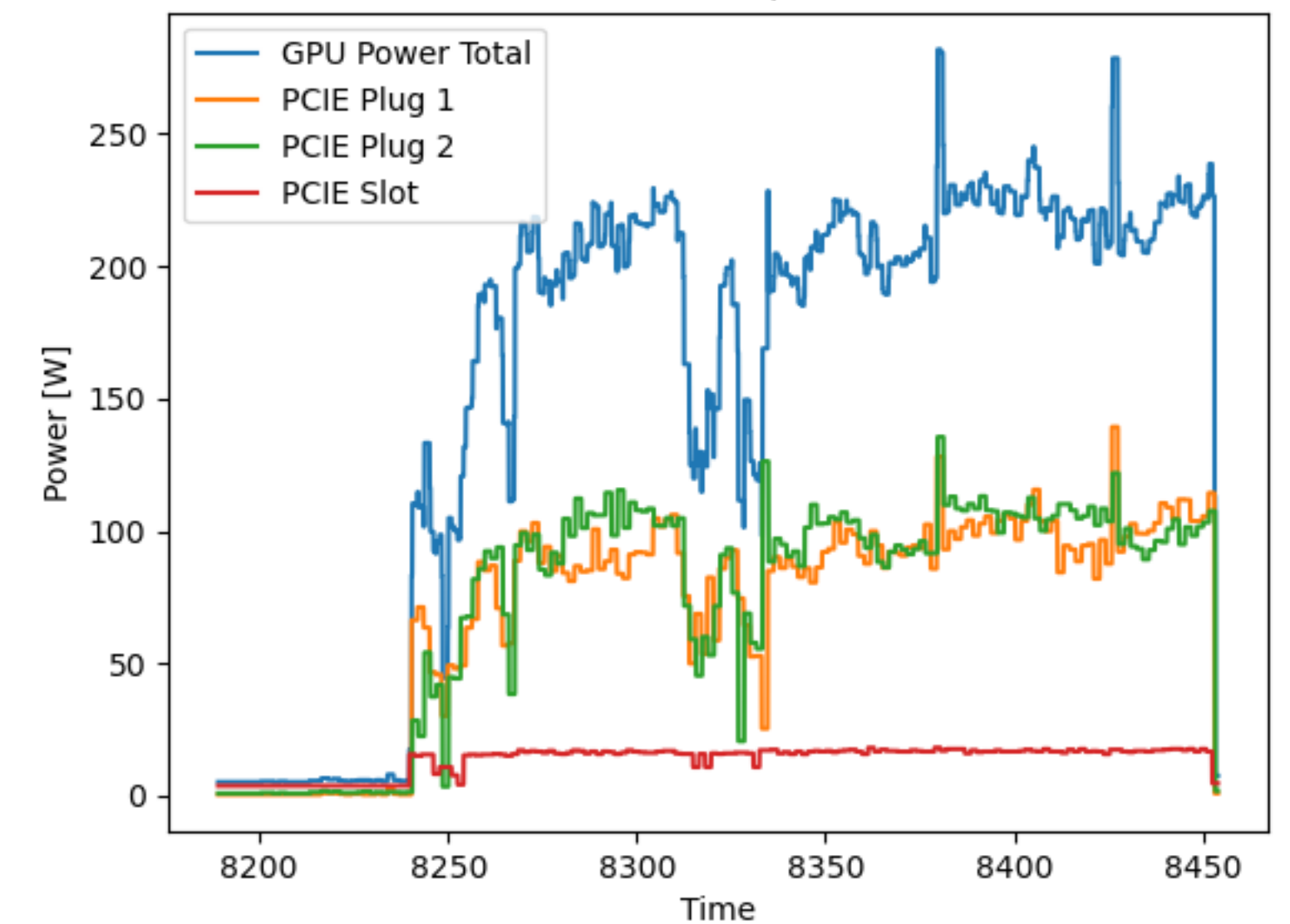
Overall Power Consumption



Mainboard Power Consumption Detailed



GPU Power Consumption Detailed



SUMMARY

Research on the intersection of accelerators and deep neural networks

Scalable ML -> “Green ML”, LLMs

Robust ML -> Build “Bayesian Machines”

ML Accelerators -> Diversification

Beyond today

Bayesian Machines - co-designed to reason about uncertainty (intractable workloads such as BNNs)

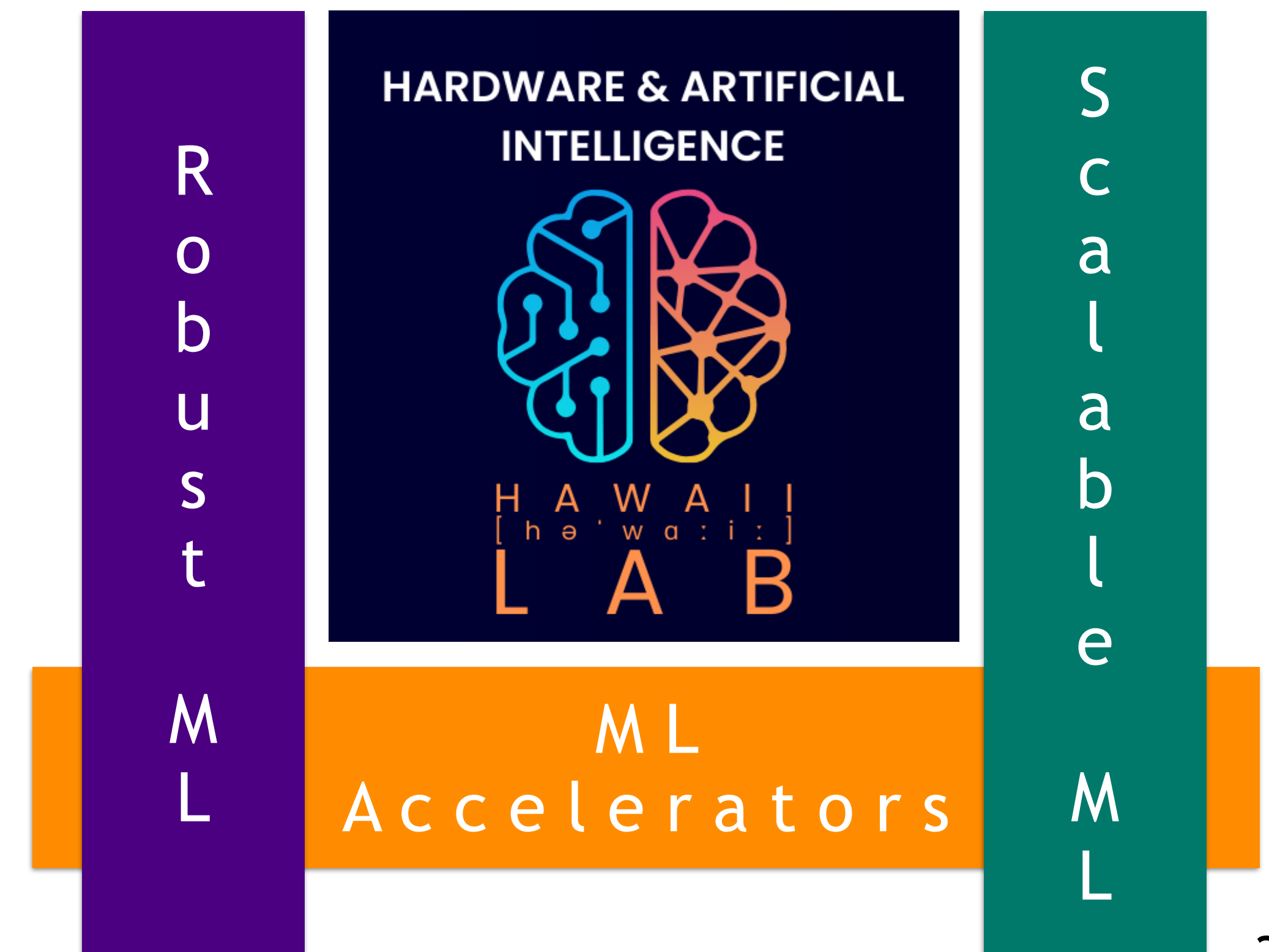
RRAM for LLMs - efficient activation storage but unstable devices

Repulsive ensembles - combining LLMs & uncertainty estimates

FPGAs as SVI accelerators

Greener training - compressing the backward pass

Energy-Efficient Systems and AI Lab (ESAIL)



SUMMARY

Energy-to-completion for LLMs is promising

V- and U-shaped pareto curves suggest that race-to-idle is not always the best strategy

DVFS-control was actually quite limited - no control over HBM

LLM training is unaffordable (but super interesting HPC case)

Simulators à la GraphCulon

Even dynamic features can often be modeled - e.g. MoEs

FlashAttention might require the use of statistics

Research questions

Predictability and granularity (frequency, time) of DVFS settings

Energy efficiency for all system components - interconnection network

Alternative forms of processing

