Power Roofline

Michael Jaemin Kim[*,1], Doe Hyun Yoon[*,1], Jovan Stojkovic[1,2], Gefei Zuo[1],
Jong Bin Lim[1], Harsha Jagannati[1], Zhan Shu[1], Shobhit Kanaujia[1]
[1]Meta, USA
{michael604, dhyoon, gzuo, jongbinlim, harshaj, zhanshu, shobhitk}@meta.com
[2]The University of Texas at Austin, USA
jovan.stojkovic@utexas.edu

***Abstract*—Modern AI accelerators increasingly operate at their power limits, creating a widening gap between peak and attainable performance. On the other hand, the Roofline model, while widely adopted for reasoning about compute and memory bottlenecks, provides little insight when power is the binding constraint. To address this challenge, we propose *Power Roofline*, an extension that replaces operational intensity (ops/byte) with energy efficiency (ops/Joule), directly capturing how power budgets limit attainable performance. The model naturally generalizes to a 3D formulation that unifies compute, memory, and power bounds.**

We apply Power Roofline to GEMM and attention kernels across multiple generations of GPUs and uncover several insights that the conventional Roofline cannot express. First, we show that input data values affect energy efficiency and, consequently, throughput under power caps. Second, we identify an energy-efficiency optimal operating point, below the maximum TDP, due to the interplay between static power and voltage scaling. Third, we demonstrate that power excursion during short profiling runs can mislead kernel auto-tuning, causing it to select configurations that underperform under sustained operation. Finally, guided by Power Roofline, we optimize a decode attention kernel by eliminating wasted computation, improving performance under power-constrained settings by up to 19.2%. Together, these results show that energy efficiency is becoming a first-order performance metric for power-limited accelerators, and Power Roofline offers a practical lens to reason about it.

***Index Terms*—Roofline model, GPU, energy efficiency, DVFS**

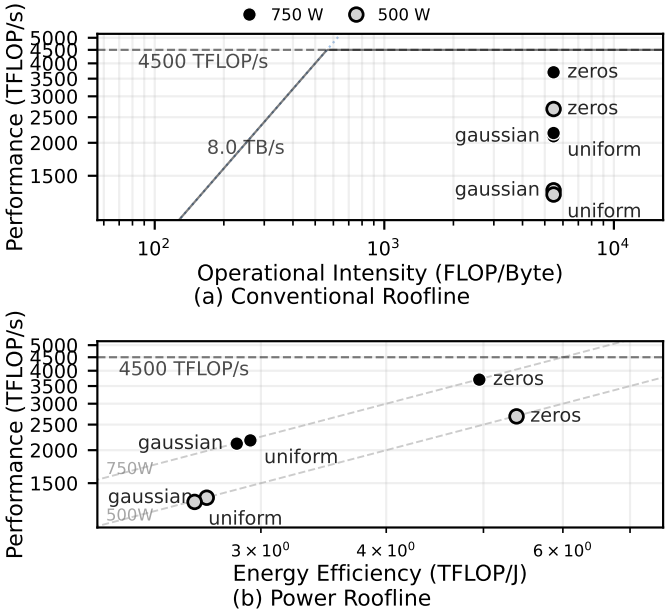

Fig. 1. (a) Conventional Roofline. (b) Power Roofline. FP8 8K-8K-8K M-N-K GEMM on B200 with varying power caps and data distributions.

I. Introduction

Despite all the sophisticated tools available to computer architects, the Roofline model [1] remains one of the most enduring and widely used. Its strength lies in its simplicity: by relating a program's operational intensity to the hardware's compute throughput and memory bandwidth, it provides an intuitive visual bound on attainable performance. One can sketch a Roofline on a whiteboard and immediately reason about whether a kernel is compute-bound or memory-bound. This simplicity has made it a staple in both academic research and industrial practice, from the evaluation of Google's TPUs [2], [3] to the optimization of HPC workloads [4], [5].

However, the Roofline model was designed for an era when compute throughput and memory bandwidth were the dominant performance limiters. As Dennard scaling has faded [6], [7], modern accelerators, particularly GPUs, have entered a regime where power is increasingly the binding constraint. A large GEMM running on a recent GPU can fall well short of the theoretical peak throughput, not because it lacks operational intensity, but because the chip cannot sustain the required clock frequency within its power envelope. The same kernel can exhibit different performance depending on the input data distribution, the duration of execution, or the power cap configured by the operator. None of these behaviors are captured by the conventional Roofline, which assumes a fixed peak performance and bandwidth.

Figure 1(a) illustrates this. An FP8 8K-8K-8K ($8K^3$) M-N-K shaped GEMM on B200 has high operational intensity and should, according to the Roofline, achieve near-peak throughput, yet measured performance varies significantly across power caps and data distributions, with no way for the model to explain *why*. This blind spot is becoming increasingly consequential. As datacenter power demand is increasing, outpacing power generation and grid capacity, power will constrain AI deployment [8]; hence, the inability to reason about power-bound performance leaves architects and system designers without a principled tool for a growing class of scenarios.

Prior work has incorporated power and energy into the

Roofline framework [9]–[11]. These approaches plot energy efficiency or power on the y-axis against the conventional operational intensity on the x-axis, enabling architects to reason about the energy balance between compute and memory subsystems and to identify the energy-optimal operating point along the operational intensity spectrum. However, they do not model the case where power itself becomes the performance bottleneck. When a GPU throttles its clock frequency because it has exhausted its power budget, performance degrades regardless of operational intensity, and these models provide no framework to predict or diagnose such behavior.

We propose *Power Roofline*, a new performance model that augments the Roofline framework to directly capture power-bound behavior. The core insight is to replace operational intensity (ops/byte) on the x-axis with energy efficiency (ops/Joule). This substitution comes from a fundamental analogy: just as bandwidth times operational intensity yields throughput in the conventional Roofline, power times energy efficiency yields throughput in the power-bound regime. The resulting model bounds attainable performance by the minimum of peak compute throughput and the product of the power budget and the kernel's energy efficiency. This forms a power "roof" analogous to the bandwidth slope in the original model. This can further be merged with the conventional Roofline into a unified 3D formulation that simultaneously expresses compute, memory, and power-bounds. Also, we propose a bandwidth-power variant (with bandwidth on the y-axis) that can be applied for memory-bound environments.

To demonstrate the utility of our model, we apply Power Roofline to various kernels and GPU generations, and we obtain several important insights that the conventional Roofline cannot express.

First, Power Roofline shows that the performance of compute-bound kernels under a power cap is determined by the computation's energy efficiency, which depends on the input data values. Sparse or low-activity inputs reduce dynamic power per operation, improving energy efficiency and, under a fixed power cap, enabling higher throughput. On the conventional Roofline, such data points appear as a puzzling vertical spread at the same operational intensity (Figure 1(a)); on the Power Roofline, they separate naturally along the energy efficiency axis, making the root cause visually apparent (Figure 1(b)).

Second, by sweeping the power cap on the Power Roofline, we can discover the energy-efficiency optimal operating point for a given kernel. This optimum arises from two competing effects: at low power, static power dominates the budget, so raising the cap improves energy efficiency; at high power, DVFS increases voltage alongside frequency, degrading energy efficiency. The crossover defines an optimal point below the maximum TDP, with direct implications for power-capped datacenter deployments aiming to maximize aggregate throughput across a fleet.

Third, Power Roofline exposes a pitfall in kernel auto-tuning caused by power excursion. Modern GPUs and accelerators allow brief power excursions above the configured TDP [12], and auto-tuning frameworks typically profile candidates with short burst runs that benefit from this transient boost [12], [13]. We show that the burst-phase ranking of kernels can differ from their sustained-operation ranking, because power excursion masks differences in energy efficiency. On the Power Roofline, the change in the ranking among kernels at different power cap is immediately visible.

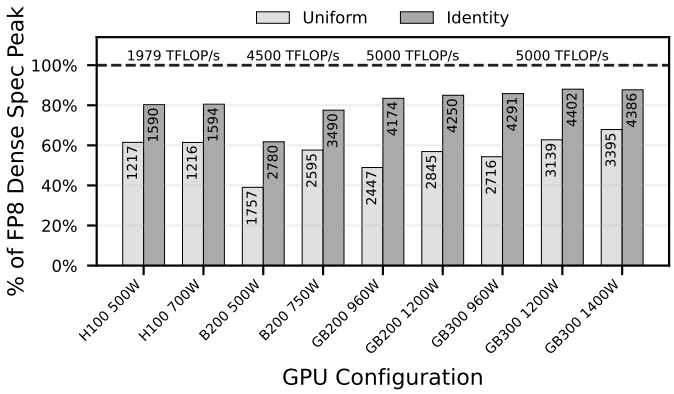

Fig. 2. Theoretical peak vs maximum achievable TFLOP/s.

Fourth, we use Power Roofline to diagnose an unexpected bottleneck in decode attention kernels. Decode attention is conventionally classified as memory-bound due to its low operational intensity [14], [15]. However, applying the Power Roofline under production power caps reveals that these kernels are in fact power-bound: they cannot saturate memory bandwidth because the power budget is insufficient. Guided by this diagnosis, we reduced wasted tensor core computation in the attention tile, improving energy efficiency and reclaiming bandwidth under the same power envelope.

Together, these case studies show that as accelerator power envelopes tighten relative to their theoretical peak, energy efficiency is no longer just a sustainability metric but a first-order determinant of attainable performance. Power Roofline enables architects, kernel developers, and system operators with a visual framework to reason about, diagnose, and optimize for this reality.

The key contributions of this paper are as follows:

- We propose Power Roofline, a performance model that replaces operational intensity with energy efficiency on the x-axis, capturing how power budgets constrain attainable performance.
- We generalize the model to a 3D formulation unifying compute, memory, and power-bounds.
- Using Power Roofline on GEMM kernels across multiple GPU generations, we characterize data-dependent performance, identify energy-efficiency optimal operating points, and expose auto-tuning pitfalls from power excursion.
- We show that decode attention, conventionally memory-bound, is power-bound under production power caps, and optimize the kernel accordingly to recover performance.

II. The Need for the Power Roofline

Roofline Model. The Roofline model [1] provides a visual upper bound on attainable performance by relating a kernel's *operational intensity*, *i.e.,* the ratio of arithmetic operations to bytes transferred from memory (ops/byte), to the hardware's peak compute throughput and peak memory bandwidth. Performance is bounded by the minimum of these two ceilings:

$$\text{Attainable Throughput} = \min(\text{Peak Compute}, \text{Peak Bandwidth} \times \text{Operational Intensity}) \quad (1)$$

On a log-log plot, this produces a flat *compute roof* at high operational intensity and a sloped *bandwidth roof* at low operational intensity. Their intersection, the *ridge point*, separates the compute-bound regime from the memory-bound regime. This simple decomposition has made the Roofline a staple for performance analysis in both academia and industry, from evaluating Google's TPUs [2], [3] to guiding HPC optimization [4].

Power-Bound Execution. Modern GPUs increasingly cannot sustain their theoretical peak throughput. Figure 2 illustrates this across four of NVIDIA GPUs (H100, B200, GB200, GB300) running FP8 GEMMs at varying power caps and data distributions. The gap between peak and attainable throughput has widened with each generation, reflecting the growing imbalance between theoretical peak compute and the power available to sustain it. This is a consequence of the end of Dennard scaling [6]. We refer to this regime as *power-bound*: performance is limited neither by compute throughput nor by memory bandwidth, but by the chip's sustainable power budget.

Several mechanisms underlie this behavior. GPUs operate within a power envelope defined by the *Thermal Design Power* (TDP), the maximum sustained power that the cooling and power delivery infrastructure is designed to support. Operators can further restrict this envelope by configuring a *power cap* below the TDP, which is a common practice in datacenters. When a workload's power draw approaches this limit, *Dynamic Voltage and Frequency Scaling* (DVFS) throttles the clock frequency to keep power within bounds. Because dynamic power scales as $P_{\text{dyn}} \propto \alpha C \cdot V^2 \cdot f$, where α is the activity factor, and DVFS raises voltage alongside frequency, increasing the clock increases power super-linearly. Thus, even modest frequency gains can push the chip past its power cap, triggering throttling that reduces throughput.

Two additional effects compound the problem. First, the power consumed by the arithmetic units depends on the *activity factor* of the data being processed: multiplying uniform-random values draws more power than multiplying zeros. This is visible in Figure 2, where the performance gap between identity and uniform-random inputs grows with each GPU generation as more compute is packed into the same power budget. Kernels processing different data distributions can thus exhibit different clock frequencies, and different throughput, under the same power cap, even when executing the identical instruction sequence. Second, GPU platforms permit brief *power excursions* [12] above the configured power cap, drawing on thermal capacitance before steady state is reached. During this transient window the GPU runs at higher clocks than it can sustain, inflating short-run throughput. Once thermal equilibrium is reached, DVFS throttles back and throughput drops accordingly. This behavior has direct implications for kernel profiling and auto-tuning (§IV-D).

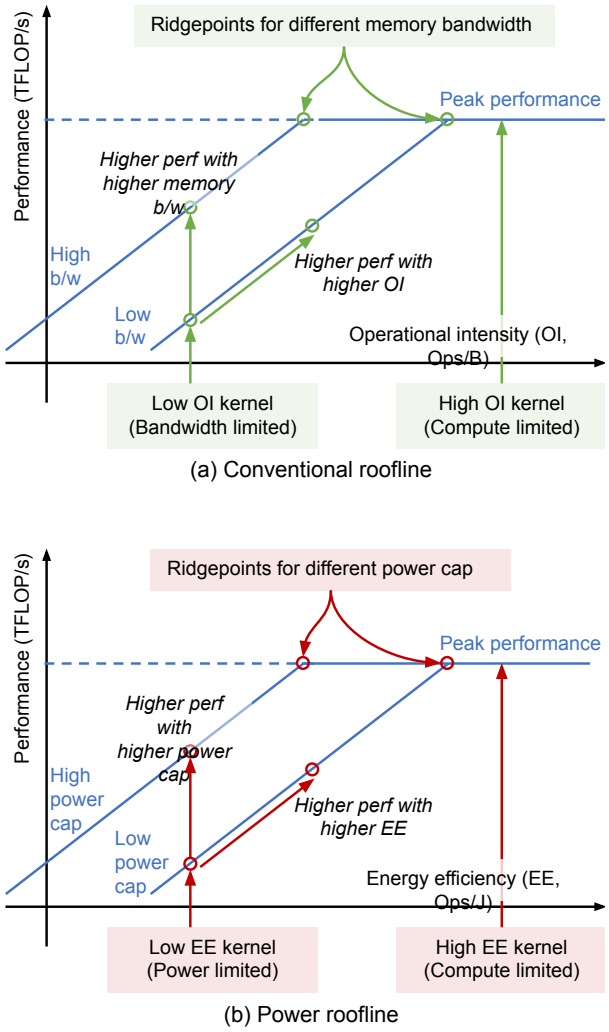

Fig. 3. (a) Conventional Roofline bounds throughput by the minimum of peak compute and bandwidth × operational intensity. (b) Compute-Power Roofline replaces operational intensity with energy efficiency, bounding throughput by the minimum of peak compute and power cap × energy efficiency.

Limitations of the Conventional Roofline. The conventional Roofline offers no mechanism to explain the phenomena described above. A compute-bound kernel, throttled by its power cap, simply appears as a data point below the flat compute roof, indistinguishable from a kernel that is poorly optimized or underutilizing the hardware. Multiple data points at the same operational intensity but different power caps or data distributions form a vertical scatter that the model cannot explain (Figure 1). Having a separate compute ceiling for

TABLE I
SYMBOLS AND TERMINOLOGIES USED THROUGHOUT §III

Symbol	Description
T	Throughput in ops/sec (*e.g.,* TFLOP/s).
P	Measured power consumption in Watts.
P_{cap}	Power cap set by the user.
T_{peak}	Hardware spec peak throughput.
BW	Achieved memory bandwidth in bytes/sec.
BW_{peak}	Peak memory bandwidth.
OI	Operational intensity (ops/byte).
EE	Energy efficiency (T/P) in ops/Joule.
BWE	Bandwidth energy efficiency (BW/P) in bytes/Joule.
EE_{ridge}	Ridge-point energy efficiency ($T_{\text{peak}}/P_{\text{cap}}$).

each different power cap is not practical for different data distribution. There is no axis on which to separate power-bound behavior from compute-bound inefficiency, and no way to reason about how changing the power budget or data distribution would shift attainable performance. The energy-aware Roofline extensions [9]–[11] do not address this gap either: they retain operational intensity on the x-axis and therefore model the energy *balance* between compute and memory, not the case where power itself becomes the binding constraint on throughput.

This motivates a new formulation that places power on equal footing with compute and memory as a first-class performance bound, by replacing operational intensity with energy efficiency on the x-axis.

III. POWER ROOFLINE

The conventional Roofline decomposes attainable throughput into bandwidth $\times$ operational intensity, separating a hardware parameter (peak bandwidth) from a kernel characteristic (operational intensity). We apply the same decomposition to the power-bound regime, replacing operational intensity with *energy efficiency* as the characterizing metric. This yields three complementary views: a Compute-Power Roofline for compute-intensive kernels, a Bandwidth-Power Roofline for memory-intensive kernels, and a unified 3D formulation that simultaneously expresses compute, memory, and power-bounds.

A. Compute-Power Roofline

Consider a kernel running at steady state, consuming power P and delivering throughput T (ops/sec) (Table I summarizes the symbols). We define its *energy efficiency* as the number of operations delivered per unit of energy consumed:

$$\text{EE} = \frac{T}{P} \quad [\text{ops/Joule}] \tag{2}$$

Rearranging gives $T = P \times \text{EE}$: throughput equals power times energy efficiency, just as in the conventional Roofline throughput equals bandwidth times operational intensity. This identity holds at any measured operating point. It becomes a *bound* when we introduce the power cap P_{cap}: the GPU's DVFS controller ensures $P \leq P_{\text{cap}}$ at steady state, so

$$T = P \times \text{EE} \;\leq\; P_{\text{cap}} \times \text{EE} \tag{3}$$

Combining with the compute ceiling yields the **Compute-Power Roofline**:

$$T \;\leq\; \min\big(T_{\text{peak}},\;\; P_{\text{cap}} \times \text{EE}\big) \tag{4}$$

On a log-log plot with EE on the x-axis and compute throughput on the y-axis (Figure 3b), this takes the familiar Roofline shape: a sloped *power roof* at low EE, where throughput is limited by the power budget, and a flat *compute roof* at high EE, where the kernel reaches peak compute without exhausting the power cap. The *ridge point* occurs at $\text{EE}_{\text{ridge}} = T_{\text{peak}}/P_{\text{cap}}$; kernels to its left are power-bound, and kernels to its right are compute-bound.

The structural parallel with the conventional Roofline is exact. The power cap P_{cap} plays the role of peak bandwidth: it is a system-level knob that sets the slope of the diagonal roof. Energy efficiency plays the role of operational intensity: it is the kernel-dependent quantity that determines where a data point falls along the x-axis. Just as lowering the available bandwidth shifts the bandwidth roof downward (making more kernels memory-bound), lowering the power cap shifts the power roof downward, making more kernels power-bound. Furthermore, just as improving a kernel's operational intensity moves it rightward toward the compute roof, improving its energy efficiency moves it rightward on the Power Roofline—a visual guide for optimization.

Tightness of the Power Roof. A notable difference from the conventional Roofline is that the Power Roofline can be *tight* when the configured power cap is the active limit: power-bound kernels sit *on or near* the diagonal rather than below it. In the conventional Roofline, kernels routinely fall below the bandwidth slope because achieving peak bandwidth requires favorable access patterns, and the gap quantifies memory-subsystem inefficiency. In the Power Roofline, a power-bound kernel's DVFS controller throttles the clock as needed to approach the power cap, so $P \approx P_{\text{cap}}$ and the bound $T = P_{\text{cap}} \times \text{EE}$ is approached.

B. Energy Efficiency as the Characterizing Metric

A natural concern is that Equation 4 is tautological: since $\text{EE} = T/P$, the bound $T \leq P_{\text{cap}} \times \text{EE}$ reduces to $T \leq P_{\text{cap}} \times T/P$, which merely restates $P \leq P_{\text{cap}}$. The same concern applies to the conventional Roofline: if operational intensity is measured as $T/\text{BW}_{\text{achieved}}$, then $\text{BW} \times \text{OI} = T$ is equally circular. In both models, the utility lies not in the arithmetic identity but in the *decomposition*: separating a system parameter (bandwidth or power cap) from a kernel-dependent characteristic (OI or EE), and visualizing the two regimes on a single plot.

Unlike operational intensity, which is largely fixed by the algorithm's ratio of operations to data movement, energy efficiency varies with the operating point. It depends on the power cap (via DVFS-induced voltage and frequency changes), the

data distribution (via activity-factor-driven dynamic power), and the kernel implementation (via utilization of functional units). This is not a limitation of the model—it is precisely the phenomenon the model is designed to expose. As the power cap changes, a kernel traces a *curve* on the Power Roofline, and the shape of that curve reveals the interplay of static power, voltage scaling, and data-dependent dynamic power. We explore these curves in detail in §IV-B.

Constructing the Power Roofline in practice. To plot a kernel on the Power Roofline, two measurements are needed at steady state: throughput T (e.g., from hardware timers or performance counters) and power draw P (e.g., from board-level power sensors via NVML or similar interfaces). Energy efficiency is then $\text{EE} = T/P$. The power roof is drawn as a line with slope equal to the configured power cap P_{cap} on a log-log plot, and the compute roof is a horizontal line at the hardware's peak throughput T_{peak} [1]. Because the model relies on measured steady-state values, profiling runs must be long enough to move past the power excursion window and reach thermal equilibrium—a point we return to in §IV-D.

C. Bandwidth-Power Roofline

For memory-bound kernels, bandwidth (bytes/sec) is often the more natural performance metric than compute throughput. We define *bandwidth energy efficiency* as the rate at which the kernel converts energy into data movement:

$$\text{BWE} = \frac{\text{BW}}{P} \quad [\text{bytes/Joule}] \tag{5}$$

By the same reasoning as the compute case, the **Bandwidth-Power Roofline** bounds attainable bandwidth by:

$$\text{BW} \;\le\; \min\big(\text{BW}_{\text{peak}},\;\; P_{\text{cap}} \times \text{BWE}\big) \tag{6}$$

This variant is useful when a kernel has low operational intensity and should, according to the conventional Roofline, be bandwidth-bound—yet is observed to fall short of peak bandwidth. The Bandwidth-Power Roofline can reveal whether this shortfall is due to power: the kernel may lack sufficient power budget to drive the memory subsystem at full bandwidth, even though it has no compute bottleneck. We apply this analysis to decode attention kernels in §V, where we find that kernels conventionally classified as memory-bound are in fact power-bound under production power caps.

D. Unified 3D Roofline

The conventional Roofline and the Compute-Power Roofline can be unified into a single formulation that simultaneously expresses compute, memory, and power-bounds:

$$T \le \min\big(T_{\text{peak}},\quad \text{BW}_{\text{peak}} \times \text{OI},\quad P_{\text{cap}} \times \text{EE}\big) \tag{7}$$

On a 3D plot with OI and EE on the two horizontal axes and throughput on the vertical axis, this defines three surfaces: a flat compute ceiling, a bandwidth plane sloped along the OI axis, and a power plane sloped along the EE axis. The lower envelope of these three surfaces forms the attainable performance surface. A kernel's position in this space immediately classifies its bottleneck: which surface it touches determines whether it is compute-bound, memory-bound, or power-bound.

While the full 3D visualization can be difficult to read on a printed page, it yields useful 2D projections: the conventional Roofline (T vs. OI, collapsing the EE axis), the Compute-Power Roofline (T vs. EE, collapsing the OI axis), and a *flattened* Roofline that overlays both slopes on a single plot with a shared y-axis. We use the flattened variant in §IV-C to simultaneously visualize compute-bound, memory-bound, and power-bound GEMM configurations across a range of shapes.

[1]We use spec numbers for theoretical peak throughput for each GPU.

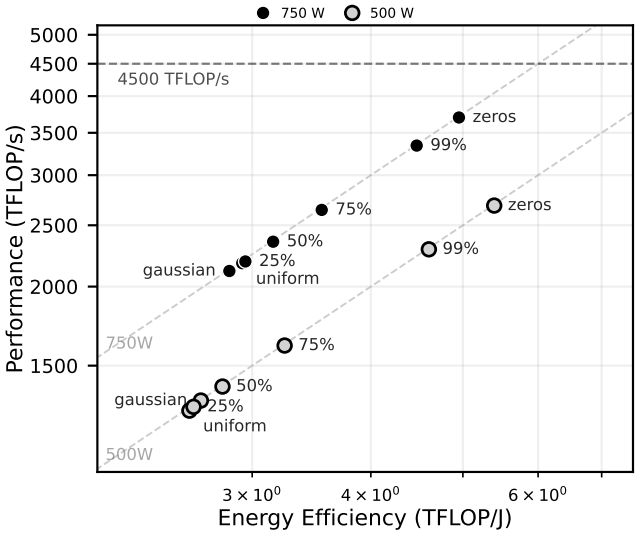

Fig. 4. Data distribution dependence. B200 500W and 750W, FP8 8K[3] GEMM.

TABLE II
B200 VS. GB200 IDLE POWER.

Metric	B200	GB200
Idle power	~140W	~192W
Max GPU power	750W	1200W
FP8 at 750W	2194.7 TFLOP/s	1753 TFLOP/s
Power usage	736 W	734 W
Energy efficiency	2.98 TFLOP/J	2.39 TFLOP/J
Frequency (during MatMul)	1042 MHz	790 MHz

IV. GEMM AND CONVOLUTION KERNELS

This section uses GEMM and direct 2D convolution kernels to demonstrate various behaviors that the Power Roofline can capture. We leverage both CUBLAS [16] and auto-tuned CUTLASS kernels [17] for our kernel selections, which we clarify on each cases. For each kernel, we run for 30sec runtime and use the measured performance/bandwidth and power, unless stated otherwise. We base our study primarily on B200 750W and GB200 1200W, but we also experiment on wider range of power settings or machines in certain cases.

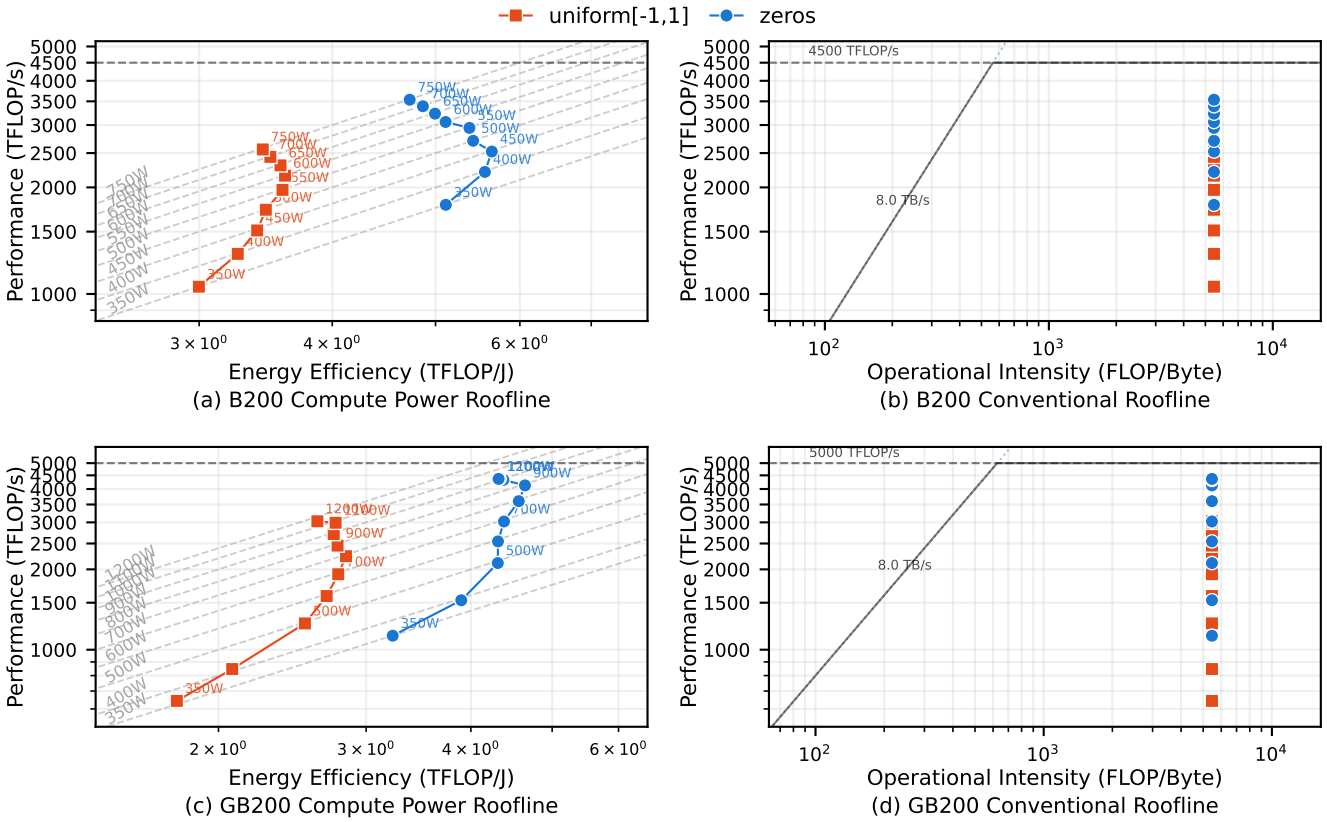

Fig. 5. FP8 $8K^3$ GEMM on B200 and GB200 power sweep. Shows EE optimal point and DVFS & Idle power effects.

While we demonstrate our study mainly on NVIDIA GPUs for clarity, similar observations can be found on different accelerator platforms.

A. Data Distribution Dependence

First, we examine the impact of input data distribution on FP8 GEMM performance using an 8K-8K-8K ($8K^3$) M-N-K shape on B200. Figure 4 shows Power Roofline at two power caps, 750W and 500W, for uniform random inputs (min=−1, max=1) with varying levels of unstructured sparsity (25%, 50%, 75%, 99%), an all-zeros, and a Gaussian random. We attribute the performance and energy efficiency gains at higher sparsity to reduced activity factors in compute and datapath, which lower dynamic power consumption. At 750W, the all-zeros input reaches 3701 TFLOP/s, a 1.70× improvement over uniform random (2178 TFLOP/s). At 500W, the relative gain is even larger. All-zeros achieves 2686 TFLOP/s versus 1274 TFLOP/s for uniform, a 2.11× improvement. In energy efficiency, all-zeros at 500W attains 5.39 TFLOP/J compared to 2.58 TFLOP/J for uniform, a 2.09× improvement.

This relationship is notably non-linear. Uniform, 25%, and 50% sparsity cluster tightly together at both power levels, indicating that moderate sparsity yields only marginal gains. The meaningful performance gain begins around 75% sparsity, beyond which each increment in sparsity produces a substantial jump in both performance and energy efficiency.

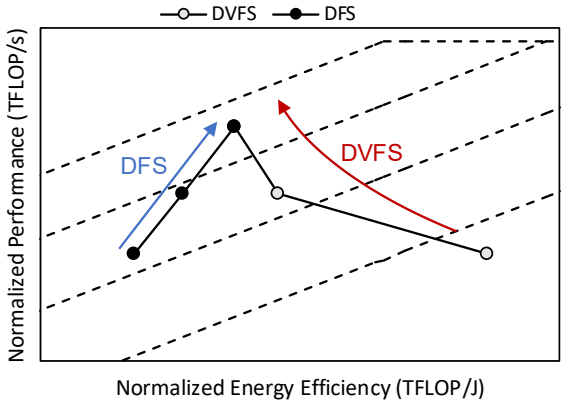

Fig. 6. DFS and DVFS effects on internal accelerator.

For instance, at 750W, moving from uniform to 25% sparsity gains only 13TF/s, whereas moving from 75% to 99% yields additional 697TF/s.

B. Energy-Efficiency Optimal Power Setting

By sweeping the *power cap* setting and expressing the data-points on Power Roofline, we identify an energy-efficiency optimal power setting for the tested kernels. On B200 and GB200, we sweep the power setting from 350W to 750W and 1200W, respectively, with all-zero and uniform-random data distribution. In Figure 5(a)(c), we can identify a curve, where

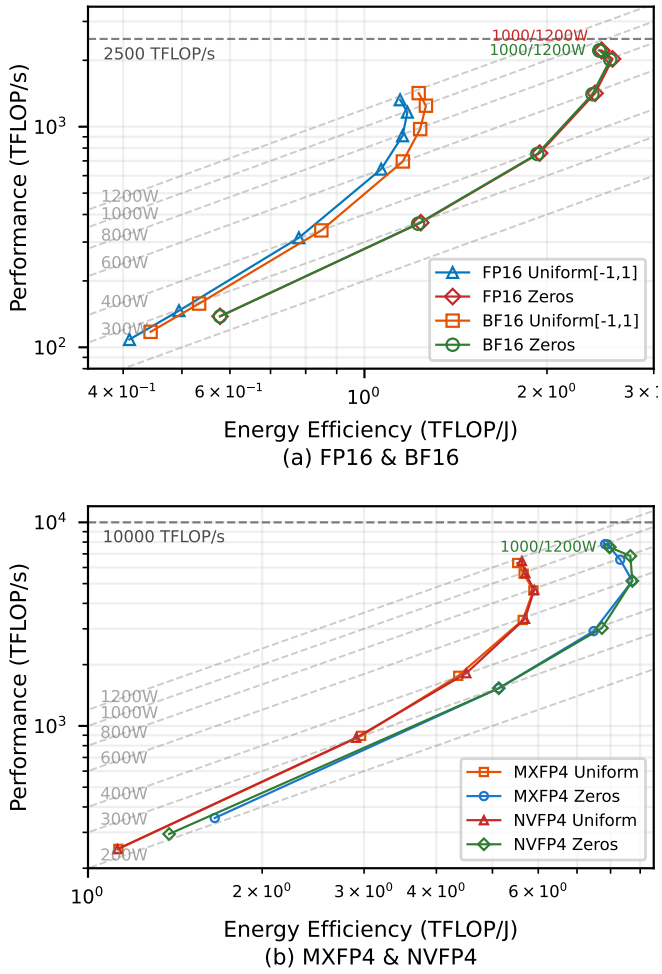

Fig. 7. GB200 $8K^3$ GEMM dtype and power sweeps.

there exists an energy-efficiency optimal point for the $8K^3$ FP8 GEMM. While this optimal point can vary across different kernels, this can be especially useful for the data centers with a limited power budget, as operating the accelerators at this point can provide the highest total performance.

We attribute the curve to the interplay of idle power and voltage increase from DVFS. On the lower power setting, the energy-efficiency increases as the portion of idle power decreases, allowing for larger power headroom for meaningful work. The EE increase on this phase is more dramatic for GB200, which aligns with the higher idle power trend on GB200, compared to B200 (Table II). In fact, when we fix the power setting for both B200 and GB200 to 750W, we can identify that the FP8 performance of GB200 is lower than B200, which can be attributed to the smaller headroom of GB200.

As we increase the power setting, the frequency and voltage increases allowing for higher performance. However, the EE decreases at a certain point. We attribute this to the voltage increase, which increases the power consumption quadratically, deteriorating the overall energy-efficiency once the idle power effect diminishes. To test this effect in isolation, we compare DFS (Dynamic Frequency Scaling) with the DVFS. Since the tested GPUs do not support DFS, we use a custom AI accelerator in which we can freely set the voltage and frequency values to emulate both DFS and DVFS. Figure 6 shows the operating points of the internal accelerator for FP8 $8K^3$ GEMM on the Power Roofline. With DFS, energy-efficiency increases because the idle power gets amortized, creating a larger headroom. However, with DVFS from low frequency to high frequency, the energy-efficiency drastically decreases due to the increased voltage.

C. Different Shapes and Datatypes

We extend our analysis to different shaped GEMMs and other datatypes. First, we express three more shapes of different OI (64x65536x8192, 128x32768x8192, 256x16384x8192) FP8 GEMMs on B200, with both all zero and uniform random data distribution. On the conventional Roofline (Figure 8(c)), they fall short of the bandwidth roof or compute ceiling. Expressing them on the Power Roofline reveals the power-bound nature of data points (Figure 8(a)). Both Roofline can also be expressed in a single consolidated 3D Roofline (Figure 8(b)), which shows three different roofs of compute, bandwidth, and power boundedness.

Second, we also express different data types of FP16, BF16, NVFP4, and MXFP4 $8K^3$ GEMMs on GB200, with varying power settings. In Figure 7(a), we can see that for uniform random case, BF16 has a higher EE than FP16, achieving a higher performance. This can be attributed to the fewer mantissa bit width of BF16, which consumes less energy during GEMM operations [18]. We can see that this gap is smaller for all-zero case. Meanwhile, such a gap is less visible between NVFP4 and MXFP4 (Figure 7(b)). The main difference between NVFP4 and MXFP4 is on the scaling factor and the block size, which can have smaller impact on the power consumption of the compute unit.

D. Varying Energy Efficiency of GEMM Kernels

Kernel Ranking Flip across Different Power. We investigate EE of GEMM kernels on Power Roofline, and demonstrate its implication on selecting the best kernel for different power setting, and possible impact on the auto-tuning when combined with the power-excursion effect. We leverage the CUTLASS profiler [17] to sweep a wide range of instantiated kernels, and narrow down to four different kernels for FP8 $8K^3$ GEMM. CUTLASS on Blackwell supports two SM modes for GEMM; 1sm and 2sm. With 1sm kernels, the CTA executes on a single SM and relies on large CTA clusters (*e.g.,* $4\times4\times1$) for cross-SM cooperation. For 2sm kernels on the contrary, each CTA spans a physically paired SM duo, doubling the per-tile register file and tensor core throughput, but using smaller clusters (*e.g.,* $2\times1\times1$).

In Figure 9, we plot four CUTLASS kernels, two 1sm and two 2sm kernels with varying tile size and CTA cluster sizes on Power Roofline. We measure the sustained performance and power consumption, after running each kernels for 30 seconds. At low power ($\leq$500W), 1sm kernels with large clusters

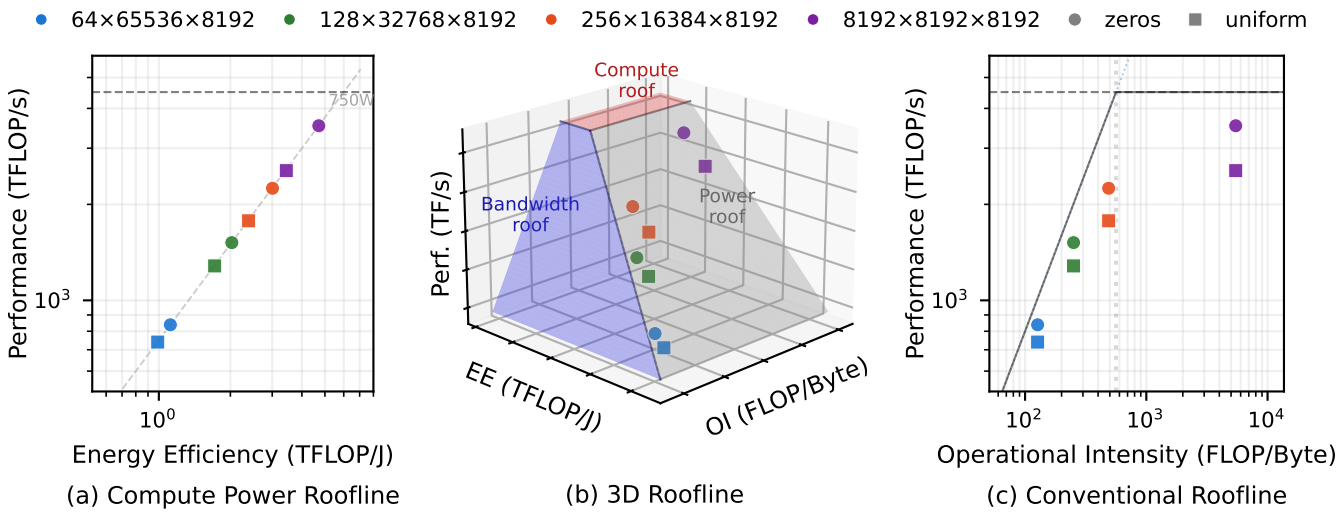

Fig. 8. Different shaped GEMMs. Compute-Power Roofline, conventional Roofline, and 3D Roofline. B200 750W.

achieve higher energy-efficiency and higher TFLOP/s, leading the best 2sm kernel by 6–17%. At high power (≥700W), the ranking *flips*: 2sm kernels dominate by 7–13%, with the gap widening at higher power caps. The crossover occurs near 600W. We attribute this to the interplay between cluster parallelism and per-tile efficiency: at low clocks (low power), 1sm kernels better utilize the limited frequency across many small cooperative CTAs, whereas at high clocks (high power), 2sm kernels extract more throughput per tile and are more energy efficient. This is consistent with 2sm kernels peaking in energy-efficiency at ∼2.78 TFLOP/J (at 900W), while 1sm kernels peak earlier at ∼2.51 TFLOP/J (at 700–800W) and decline thereafter.

Power-Excursion and Auto-Tuning. This ranking flip across different power setting can have a direct implications for auto-tuning. While the runtime for each iteration can vary, each kernel often runs for a short iteration during the auto-tuning phase when finding the best kernel. However, burst runs can experience *power excursion* [12], where the GPU can operate at an elevated power budget for a short period of time, without throttling the clock frequency. Table III shows an example electrical design point (EDP) excursion envelope from an Open Compute Project (OCP) specification [12], allowing up to 2× TDP for at most 20µs.[2] We can identify a similar effect on NVIDIA GPU, as demonstrated in Figure 11. Such a power excursion can unexpectedly cross the ranking flip point, disrupting in finding the best kernel.

Figure 10 demonstrates this flip between a short iteration (default iteration value of the CUTLASS profiler) and a long iteration (30 seconds). A ranking flip between the short and long run occurs at 350W and 500W, which we can attribute to the power excursion effect, which can increase the power budget beyond 500W, crossing the flipping boundary in Figure 9. The pattern is consistent: smaller tiles win at burst

[2]Note that this is an example from OCP spec. Different GPUs implement different excursion levels and time durations.

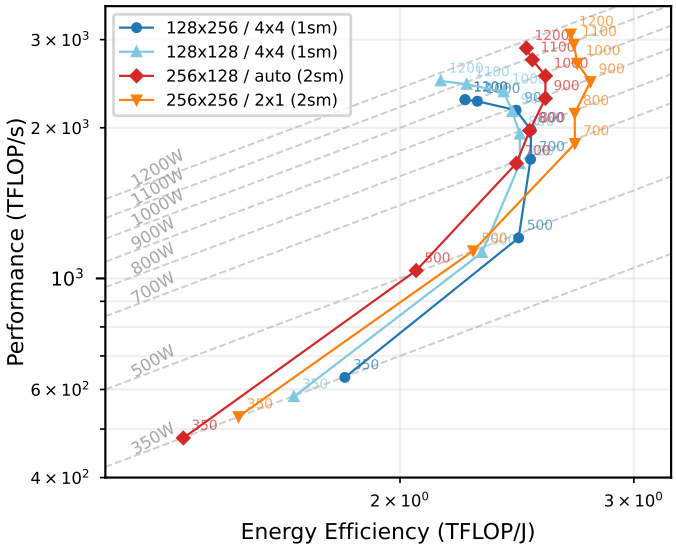

Fig. 9. Ranking Flip. GB200, $8K^3$ FP8 GEMM, 30 seconds run, uniform [-1,1].

runs by exploiting power excursion, while larger, more energy-efficient tiles win under sustained power-bound operation. Note that we cannot directly express short runs on the Power Roofline, because we cannot measure the power consumption in a fine-grained manner (§VI-C). As Figure 11 demonstrates, the power excursion effect decays within a 100ms, which is a shorter period than what is reliably measurable for the user.

E. Convolution Kernel

Figure 12 applies Power Roofline to direct 2D convolution [19], [20] using auto-tuned CUTLASS kernels [17]. Figure 12(a) shows the Compute-Power Roofline, which is the more relevant view because the tested case of convolution is compute-bound unless constrained by power. Figure 12(b) shows the Bandwidth-Power Roofline; it is less informative because neither memory bandwidth nor bandwidth energy

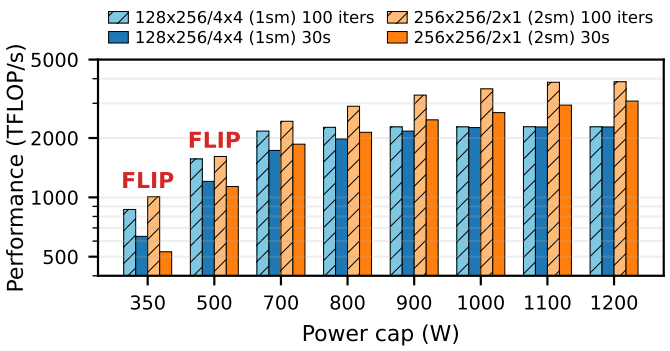

Fig. 10. Short run vs. Long run. GB200 $8K^3$ FP8 GEMM uniform [-1,1].

TABLE III
POWER EXCURSION EXAMPLE [12].

EDP	Duration
2× TDP	$\leq$ 20µs
1.6× TDP	$\leq$ 2ms
1.5× TDP	$\leq$ 5ms
1.2× TDP	$\leq$ 10ms
1.1× TDP	$\leq$ 20ms

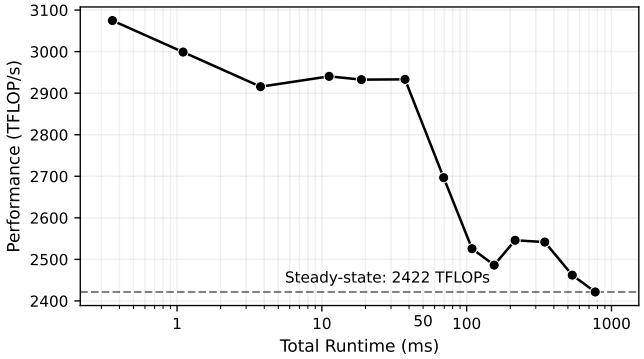

Fig. 11. Iteration sweep on B200, $8K^3$ FP8 GEMM, uniform [-1,1]. Demonstrates power excursion effect.

efficiency limits performance. As in the GEMM experiments, sweeping the power cap reveals an energy-efficiency-optimal operating point and data-distribution dependence. Compared with uniform-random inputs in $[-1, 1]$, all-zero inputs experience less throttling at the same power cap and enter the compute-bound regime at 1000W.

V. DECODE ATTENTION KERNELS

Before analyzing decode attention kernels, which are expected to be memory-bound, we first demonstrate that the saturating memory bandwidth itself as the baseline is constrained by power, even for a simple HBM read stream microbenchmark without any compute. Figure 14 plots the achieved read bandwidth against data transfer size on B200 and GB200 (dotted lines). Reading all-zero data consistently achieves higher bandwidth than uniform random data. On B200 at 750W, the peak read bandwidth reaches 7390 GB/s for all-zero data versus 6073 GB/s for random data, a 21.7% gap. On GB200 at 1200W, the gap narrows to 4.3% (7606 vs. 7292 GB/s). We attribute this to the data-dependent power consumption of the memory subsystem, from cache hierarchy to memory controller, HBM PHY and I/O interface. This establishes that power constrains not only compute-bound kernels (§IV), but also purely memory-bound workloads, motivating the Bandwidth-Power Roofline variant below.

A. Vanilla CUTLASS Decode Attention Kernel

We analyze the decode attention kernels [21] on GPU using the Bandwidth-Power Roofline. We primarily focus on grouped query attention (GQA) [22], which is widely adopted in modern LLMs. During the iterative decode stages of the multi-headed attention (MHA), each attention head maintains independent Q, K, and V projections, *i.e.,* head_q = head_kv. GQA shares a single K and V head across a group of G = head_q/head_kv query heads, reducing the KV cache size by $G\times$ while increasing the arithmetic intensity by $G\times$. We define the tile dimensions with respect to the attention computation ($QK^\top$ and PV) as follows (Figure 13): MTile tiles the query dimension, NTile tiles the key/value sequence dimension (*i.e.,* context length), and KTile tiles the head dimension (*e.g.,* 128).

We started off with the out-of-the-box CUTLASS Blackwell FMHA example kernel [23], a warp-specialized decode kernel targeting the Blackwell architecture. This kernel only provides MTile of 128, which we believe is the remnant of the prefill-oriented kernel design focusing on large batch-size queries. For brevity, we demonstrate the below experiments with $G = 5$, but similar observations were found in different settings as well. For decode, the data movement is dominated by reading the KV cache, yielding an operational intensity of approximately $2G$/element byte. This is 5 FLOP/Byte for BF16 attention with group size 5, placing the kernel firmly in the memory-bound regime [24], [25].

However, despite being memory-bound, the kernel was unable to saturate the memory bandwidth on B200 at 750W and GB200 at 1020W/1200W (Figure 14), even when we increase the KV cache size. The conventional Roofline (Figure 15(b)(d)) does not reveal much insight into the reason. All data points collapse to a single cluster at OI=5, without providing insight into why bandwidth is not saturated. The Bandwidth-Power Roofline (Figure 15(a)(c)), however, clearly demonstrates that these points fall on or near the power slope, confirming the power-bound regime of the experimented points.

B. Energy Efficient CUTLASS Attention Kernels

Given the power-bound nature of this kernel, we identified a possible source of energy inefficiency in MTile size of 128. The MTile 128 in theory consists of a warpgroup of four warps each computing 32 rows along the M dimension. Via warp specialization, we can focus on a single warp: with $G = 5$ (Llama-4), only 5 out of 32 rows per warp carry valid query heads, yielding an effective utilization of $5/32 = 15.6\%$. The remaining rows compute on invalid data that is ultimately

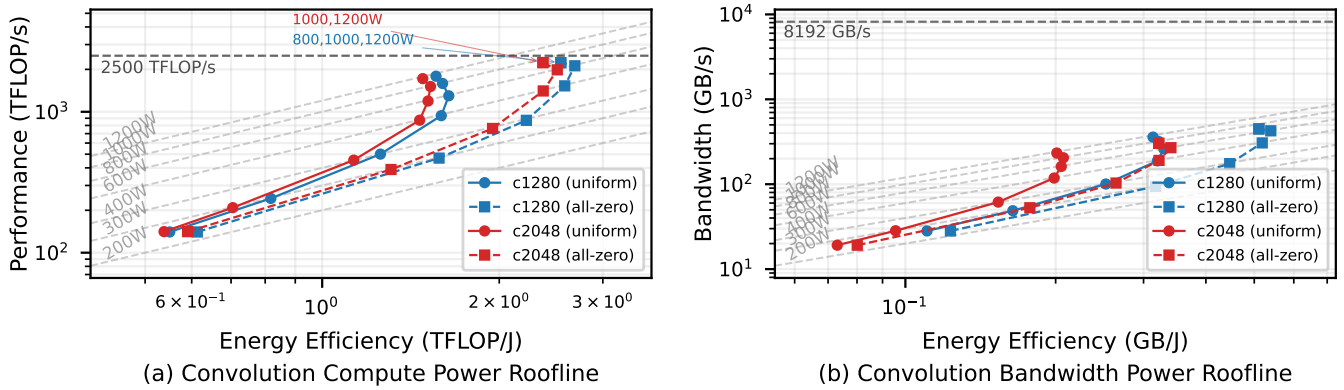

Fig. 12. Compute-Power and Bandwidth-Power Roofline for BF16 direct 2D convolution on GB200. We use a 3×3 filter, 1280/2048 input/output channels [19], [20], spatial height/width of 48, batch size of 16, and stride/padding of 1.

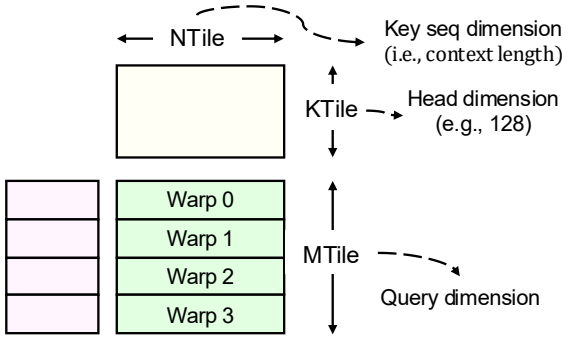

Fig. 13. Tile dimensions and relationship with the attention parameters.

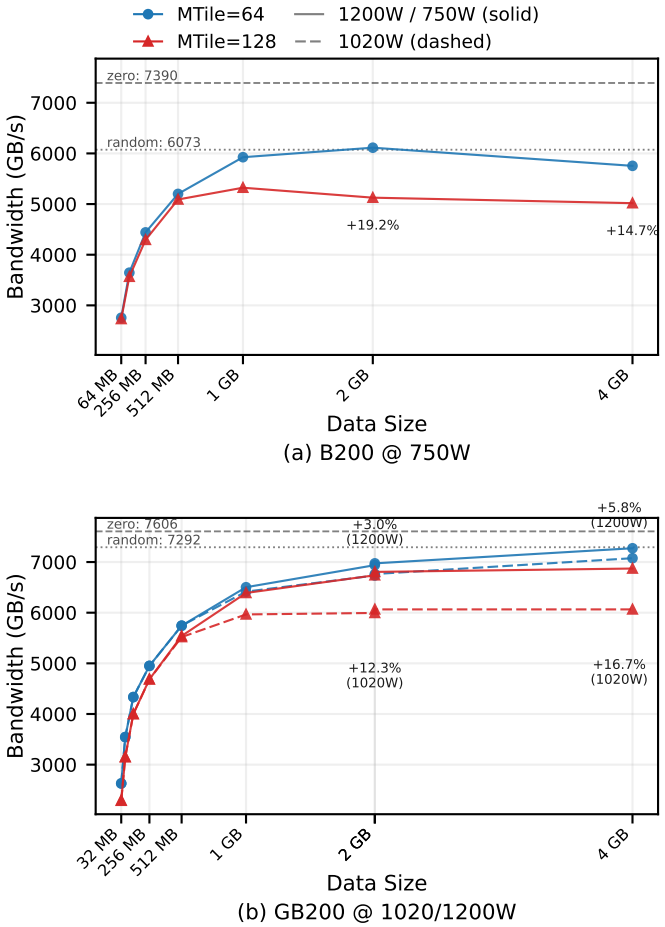

Fig. 14. Data size - bandwidth graph.

discarded, yet still draw tensor core power, consuming the limited power budget that would otherwise be available for sustaining high HBM bandwidth.

An effective optimization is to reduce the M-dimension tile size to 64, where a single warp now processes 16 rows and performs 2-thread reduction on a single row. This increases the utilization to $5/16 = 31.3\%$ effectively. With this optimization, we were able to achieve higher performance under the same power setting. On B200 at 750W, MTile=64 achieves 6113 GB/s compared to 5127 GB/s for MTile=128 at a 2 GB KV cache size, a 19.2% improvement in achieved bandwidth (Figure 14). On GB200 at a 2 GB KV cache size, the improvement is 12.3% at 1020W and 3.0% at 1200W. The diminishing gain at higher power caps is consistent with the Power Roofline prediction: at 1200W, the kernel operates closer to the bandwidth roof and further from the power slope, so reducing wasted compute yields less benefit. Conversely, at the more power-constrained 1020W setting, the kernel sits firmly on the power slope, and the MTile=64 optimization shifts it rightward along the energy efficiency axis, from 9.67 to 10.83 GB/J, translating directly into higher bandwidth.

VI. Discussion

A. Unstructured Sparsity for Energy Efficiency

§IV-A demonstrates that input data values directly affect energy efficiency. This motivates exploring software and hardware techniques that increase the sparsity of operands.

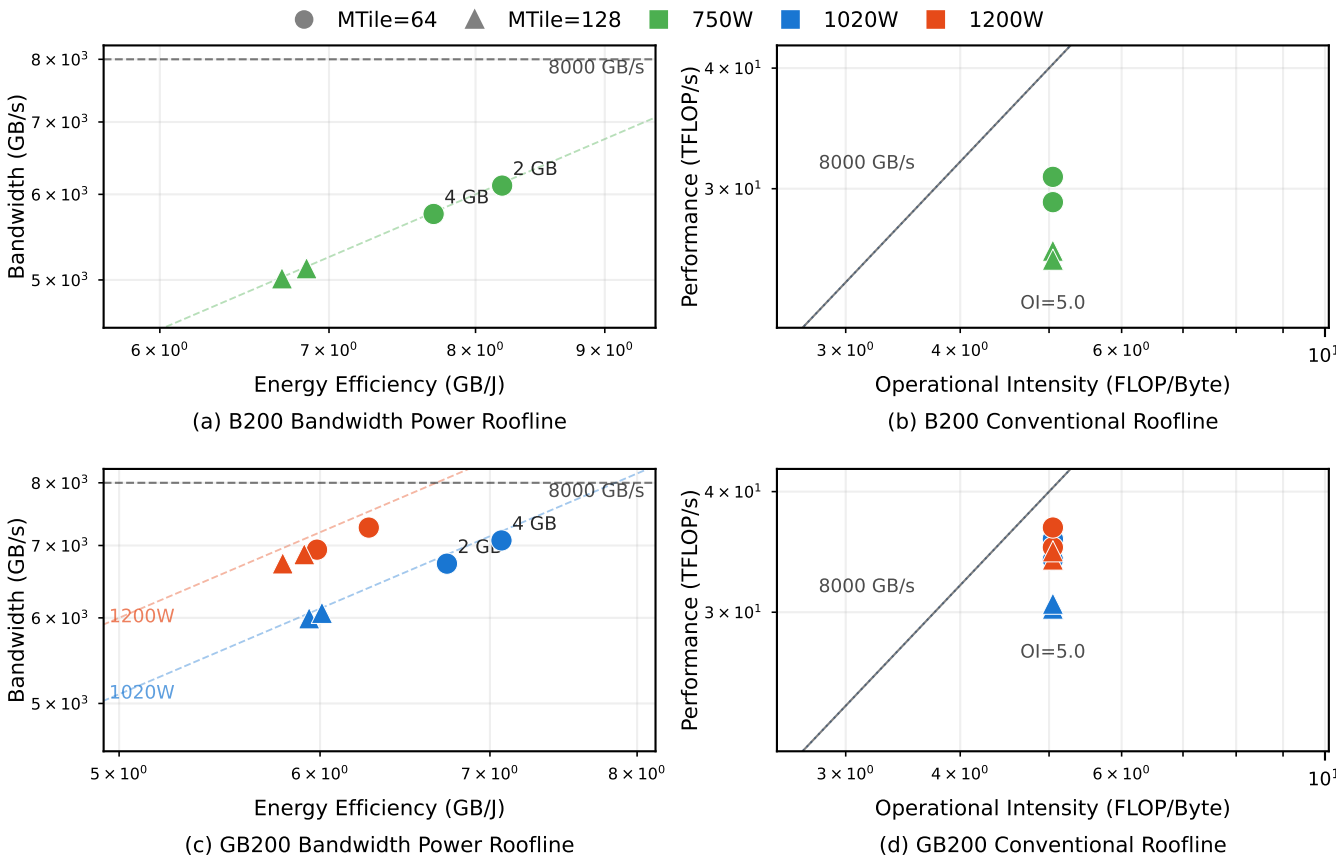

Fig. 15. Attention Power and Conventional Roofline for B200 750W, GB200 1020W & 1200W, with MTile=128 and MTile=64.

This includes structured sparsity [26] as well as unstructured sparsity [27], [28].

Sparsification Techniques. In the context of alleviating the power-bound, sparsification techniques [26]–[31], either lossy or lossless, do not have to create structured sparsity. Prior work on sparse acceleration has focused on the structured sparsity, which GPU hardware natively supports. However, the power-bound regime suggests that even unstructured sparsity can be beneficial. For severely power-bound kernels, increase in compute and memory from sparsification techniques is tolerable if the resulting zeros reduce enough power to boost overall performance.

Activation Function. Modern LLMs use smooth activations (GELU [32], SiLU [33]) that produce few exact zeros, but ReLU variants such as squared ReLU [34]–[36] can induce 90%+ sparsity with minimal accuracy loss. Since our measurements in §IV-A show that energy efficiency increases non-linearly beyond 75% sparsity, such activation functions can deliver substantial throughput improvements on power-bound accelerators.

B. Hardware Design for Power-proportional Computation

As shown in §V, tensor cores waste significant power when tiles are partially occupied. This affects any workload with partial tile occupancy, including small-batch GEMMs and narrow matrix dimensions. Our software fix (reducing MTile from 128 to 64) helps, but a more principled solution requires *energy-proportional* hardware, conceptually similar to energy proportionality in warehouse scale computing [37] that consumes power and energy in proportion to useful work. Possible directions include hardware support for unstructured sparsity [38], configurable tile dimensions, sub-tile power gating to disable unused rows or columns, and dedicated GEMV datapaths for decode-like workloads. Processing-in-memory (PIM) [25] is another promising direction, particularly for decode attention. By performing GEMV operations near or within HBM, PIM eliminates the data movement from memory to the processor. This improves energy efficiency by avoiding moving KV cache and weight data across the memory interface.

C. Measurement Determinism and Observability

Figure 16 shows time-series measurements of SM clock frequency and power consumption for repeated FP8 $8K^3$ GEMM runs on GB200 at a 1200W cap. For all experiments and Power Rooflines, we use pynvml and average the SM clock frequency and power consumption over a long kernel execution (typically 30 seconds), excluding samples at the beginning and end of each run. We leave the GPU idle for at least 5 seconds before each run. We observe no large run-to-run variation in steady state: the SM clock varies both across

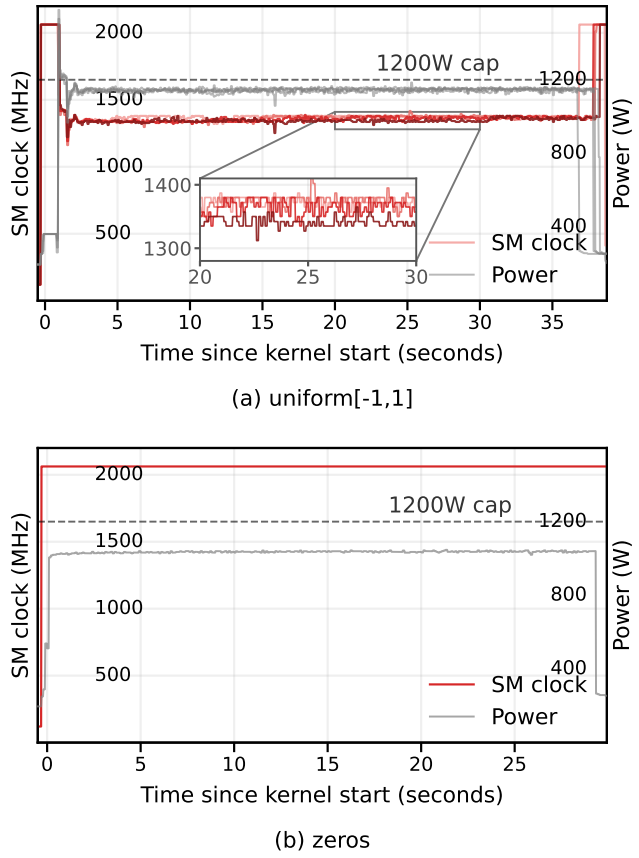

Fig. 16. Measured SM clock frequency and power consumption across repeated GB200 FP8 $8K^3$ GEMM runs at a 1200W power cap, sampled using pynvml.

runs and within a run, but remains within a limited range, as shown by the enlarged view in Figure 16(a).

We adopt this methodology because fine-grained power and clock measurements are unavailable, both temporally and spatially. GPU-level power readings are sampled at hundreds of milliseconds via NVML [39]. However, power-excursion effects begin to decay after tens of milliseconds and approach steady state over hundreds of milliseconds (§IV-D), making it difficult to accurately characterize burst-phase behavior. Spatially, only aggregate die-level power is available; there is no standard interface to decompose power into per-subsystem contributions (tensor cores, memory controllers, HBM PHY), which would enable the Power Roofline to distinguish compute-side from memory-side power constraints. Finer-grained measurement in both dimensions, combined with continuous hardware performance counter sampling, would provide the ground truth needed to validate and extend the Power Roofline model.

D. Kernel Interleaving

Power excursion can be further exploited to bypass power-bound by scheduling two different kernels with different EE characteristics. A kernel running below TDP builds up excursion headroom, and the next power-hungry kernel can run at higher clocks for a short time. This requires a way to track the remaining excursion budget at kernel boundaries, a scheduler to arrange kernels in a good order, and knowledge of the GPU's excursion behavior (duration, magnitude, cooldown). More generally, scheduling workloads with different power profiles together—whether across layers or across tenants—could improve total throughput without raising the power budget.

VII. Related Work

The seminal Roofline performance model [1], proposed by Williams et al., offers an intuitive visual framework for assessing hardware performance limitations by relating attainable performance to operational intensity. The model establishes two fundamental bounds – a flat compute throughput and a memory bandwidth slope – which intersect at the *ridge point* to delineate compute-bound from memory-bound regions. Roofline model has been widely adopted to establish performance upper bounds and compare different architectures, notably in the evaluation and analysis of Google's TPUv1 [2], TPUv4i [18], and TPUv4 [3]. We propose Power Roofline, built upon the foundation of Roofline model, by extending it to incorporate power as a bottleneck and analyze it using energy efficiency, for modern power-limited accelerators.

Researchers made extensions to the Roofline performance model to address other limiting factors. For instance, Cache-Aware Roofline model [40] introduces multiple bandwidth slopes to reflect the memory hierarchy; Wang et al. [41] incorporate a latency-bound region for more accurate bottleneck classification beyond compute and memory; DECA [42] introduces 3D Roofline that accounts for vector processing throughput in addition to memory bandwidth and compute throughput; and Gables [43] extends Roofline model to multi-accelerator SoCs. These extensions are orthogonal to our work, as our focus specifically is on analyzing how power constraints fundamentally limit the performance.

Given that power consumption and energy efficiency are now primary design constraints, several studies have adopted the Roofline concept for energy and power analysis:

- Roofline for energy [9] introduces an energy-based analogy of Roofline, visualizing energy efficiency against operational intensity.
- Ilic et al. [10] extend the cache-aware Roofline [40] to model power consumption and energy efficiency and show power and energy vary with operational intensity across cache levels.
- Nugteren et al. [44] utilize the kernel's position relative to the ridge point as a classifier to determine optimal Dynamic Voltage and Frequency Scaling (DVFS) strategies.
- Verhelst et al. [11] build an Energy Roofline (energy efficiency vs. operational intensity), similar to [9], noting that the throughput-optimized and energy-optimized operating points can diverge.

The aforementioned power and energy extensions primarily relate power or energy efficiency to operational intensity. In particular, the Energy Rooflines in [9], [11] reason about how

compute and memory energy combine as operational intensity changes, and show that throughput-optimal and energy-optimal operating points can differ. They do not, however, treat a fixed power budget as an active bound on attainable throughput. Power Roofline instead places throughput against energy efficiency and makes the bound $P_{\mathrm{cap}} \times \mathrm{EE}$ explicit. This distinction exposes insights that are not represented by prior models: performance variation at identical operational intensity due to input data, DVFS behavior under a power cap, short-versus-sustained kernel-ranking changes caused by power excursion, and nominally memory-bound kernels whose bandwidth is constrained by power.

Gregersen et al. [45] observe input-dependent GPU power usage. As discussed in §IV, Power Roofline helps reason how data distribution impacts performance.

VIII. Conclusion

We presented Power Roofline, a model that extends the classical Roofline by replacing operational intensity with energy efficiency on the x-axis, directly capturing how power budgets constrain attainable performance on modern accelerators. Through the Compute-Power Roofline, unified 3D Roofline, and an extended Bandwidth-Power Roofline, Power Roofline provides a visual framework that can express phenomena invisible in the conventional Roofline. Our analysis reveals data-distribution-dependent throughput and bandwidth under power caps, energy-efficiency-optimal operating points below the maximum TDP, kernel-ranking flips caused by power excursions during auto-tuning, and power-boundedness of memory-bound decode-attention kernels. As accelerator power envelopes continue to tighten relative to their theoretical compute capabilities, we believe Power Roofline offers architects, kernel developers, and datacenter operators a practical and intuitive tool to reason about, diagnose, and optimize performance in this increasingly power-limited regime.

References

[1] S. Williams, A. Waterman, and D. A. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

[2] N. P. Jouppi, C. Young, N. Patil, D. A. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-L. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the International Symposium on Computer Architecture*, 2017, pp. 1–12.

[3] N. P. Jouppi, G. Kurian, S. Li, P. C. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles, C. Young, X. Zhou, Z. Zhou, and D. A. Patterson, "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the International Symposium on Computer Architecture*, 2023, pp. 1–14.

[4] Z. Zhu, N. Bartelheimer, and S. Neuwirth, "An empirical roofline model for extreme-scale i/o workload analysis," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2023, pp. 622–627.

[5] Z. Jiang, L. Wang, X. Xiong, W. Gao, C. Luo, F. Tang, C. Lan, H. Li, and J. Zhan, "HPC AI500: The methodology, tools, roofline performance models, and metrics for benchmarking HPC AI systems," 2020. [Online]. Available: https://arxiv.org/abs/2007.00279

[6] H. Esmaeilzadeh, E. R. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the International Symposium on Computer Architecture*, 2011.

[7] J. Shalf, "The future of computing beyond Moore's Law," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378, no. 2166, p. 20190061, Jan. 2020. [Online]. Available: https://doi.org/10.1098/rsta.2019.0061

[8] D. Chen, J. Cong, A. Mirhoseini, C. Kozyrakis, S. Mitra, J. Xiong, C. Young, A. Anandkumar, M. Littman, A. Kirschen, S. Shao, S. Leef, N. Shanbhag, D. Milojicic, M. Schulte, G. Cauwenberghs, J. M. Chow, T. Dao, K. Gopalakrishnan, R. Ho, H. Kim, K. Olukotun, D. Z. Pan, M. Ren, D. Roth, A. Singh, Y. Sun, Y. Wang, Y. LeCun, and R. Puri, "Ai+hw 2035: Shaping the next decade," 2026. [Online]. Available: https://arxiv.org/abs/2603.05225

[9] J. W. Choi, R. Vuduc, R. Fowler, and D. Bedard, "A roofline model of energy," in *27th IEEE International Parallel and Distributed Processing Symposium (IPDPS'13)*, 2013, pp. 661–672. [Online]. Available: https://doi.org/10.1109/IPDPS.2013.77

[10] A. Ilic, F. Pratas, and L. Sousa, "Beyond the roofline: Cache-aware power and energy-efficiency modeling for multi-cores," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 52–58, 2017.

[11] M. Verhelst, L. Benini, and N. Verma, "How to keep pushing ml accelerator performance? know your rooflines!" *IEEE Journal of Solid-State Circuits*, vol. 60, no. 6, pp. 1888–1905, 2025.

[12] Open Compute Project, "Ocp accelerator module design specification v1.0," Open Compute Project, Technical Specification, February 2019, initial release by the Open Accelerator Infrastructure (OAI) subgroup. [Online]. Available: https://opencompute.org

[13] A. Inci, "Optimizing performance under power constraints," in *NeurIPS 2025 Tutorial: Energy and Power as First-Class ML Design Metrics*. Neural Information Processing Systems (NeurIPS), December 2025, session 2 presentation. [Online]. Available: https://ml.energy/tutorials/neurips25/resources/session-2.pdf

[14] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=mZn2Xyh9Ec

[15] A. K. Kamath, R. Prabhu, J. Mohan, S. Peter, R. Ramjee, and A. Panwar, "Pod-attention: Unlocking full prefill-decode overlap for faster llm inference," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2025, pp. 897–912.

[16] NVIDIA Corporation, "cuBLAS :: CUDA Toolkit Documentation," https://docs.nvidia.com/cuda/cublas/, 2026, accessed: 2026-04-02.

[17] ——, "CUTLASS: CUDA Templates for Linear Algebra Subroutines and Solvers," 2023, accessed: 2026-04-02. [Online]. Available: https://github.com/NVIDIA/cutlass

[18] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. C. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. A. Patterson, "Ten lessons from three generations shaped Google's TPUv4i: Industrial product," in *Proceedings of the International Symposium on Computer Architecture*, 2021, pp. 1–14.

[19] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," *arXiv preprint arXiv:2307.01952*, 2023. [Online]. Available: https://arxiv.org/abs/2307.01952

[20] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, vol. 30, 2017, pp. 5998–

6008. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[22] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebron, and S. Sanghai, "GQA: Training generalized multi-query transformer models from multi-head checkpoints," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 4895–4901. [Online]. Available: https://aclanthology.org/2023.emnlp-main.298/

[23] NVIDIA Corporation, "CUTLASS example 77: Blackwell FMHA," https://github.com/NVIDIA/cutlass/tree/main/examples/77_blackwell_fmha, 2026, gitHub repository, accessed 2026-04-02.

[24] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 16 344–16 359.

[25] J. Park, J. Choi, K. Kyung, M. J. Kim, Y. Kwon, N. S. Kim, and J. H. Ahn, "Attacc! unleashing the power of PIM for batched transformer-based generative model inference," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024, pp. 103–119.

[26] A. Mishra, A. Lym, K. Goswami, S. Chakradhar, A. Aji, M. Rosing, J. Pool, and A. Leung, "Accelerating sparse deep neural networks," *arXiv preprint arXiv:2104.08378*, 2021. [Online]. Available: https://arxiv.org/abs/2104.08378

[27] G. Jeong *et al.*, "Enabling unstructured sparse acceleration on structured sparse accelerators," *arXiv preprint arXiv:2403.07953*, 2024. [Online]. Available: https://arxiv.org/abs/2403.07953

[28] G. Jeong, S. Damani, A. R. Bambhaniya, E. Qin, C. J. Hughes, S. Subramoney, H. Kim, and T. Krishna, "VEGETA: Vertically-integrated extensions for sparse/dense GEMM tile acceleration on CPUs," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 259–272.

[29] M. Mahmoud, K. Siu, and A. Moshovos, "Diffy: A déjà vu-free differential deep neural network accelerator," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 134–147.

[30] S. Kim, H. Lee, W. Cho, M. Park, and W. W. Ro, "Ditto: Accelerating diffusion model via temporal value similarity," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 338–352.

[31] W. Kong, Y. Hao, Q. Guo, Y. Zhao, X. Song, X. Li, M. Zou, Z. Du, R. Zhang, C. Liu *et al.*, "Cambricon-d: Full-network differential acceleration for diffusion models," in *2024 ACM/IEEE 51st annual international symposium on computer architecture (ISCA)*. IEEE, 2024, pp. 903–914.

[32] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016. [Online]. Available: https://arxiv.org/abs/1606.08415

[33] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[34] D. R. So, C. Liang, and Q. V. Le, "Primer: Searching for efficient transformers for language modeling," in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022, pp. 20 365–20 377.

[35] J. M. Klusowski and A. R. Barron, "Approximation by combinations of relu and squared relu ridge functions with ℓ^1 and ℓ^0 controls," *arXiv preprint arXiv:1607.07819*, 2016. [Online]. Available: https://arxiv.org/abs/1607.07819

[36] Z. Zhang, Y. Song, G. Yu, X. Han, Y. Lin, C. Xiao, C. Song, Z. Liu, Z. Mi, and M. Sun, "ReLU^2 wins: Discovering efficient activation functions for sparse LLMs," *arXiv preprint arXiv:2402.03804*, 2024. [Online]. Available: https://arxiv.org/abs/2402.03804

[37] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[38] W. J. Dally, "Irregular sparse matrix multiply," Dec. 18 2025, U.S. Patent Application Publication No. 2025/0384107, Application No. 18/825,805. [Online]. Available: https://patents.justia.com/patent/20250384107

[39] NVIDIA Corporation, "NVIDIA Management Library (NVML)," https://developer.nvidia.com/nvidia-management-library-nvml, 2026, accessed: 2026-04-02.

[40] A. Ilic, F. Pratas, and L. Sousa, "Cache-aware roofline model: Upgrading the loft," *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 21–24, 2014.

[41] B. Wang, A. Kozhokanova, C. Terboven, and M. Mueller, "RLP: Power management based on a latency-aware roofline model," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2023, pp. 446–456.

[42] G. Gerogiannis, S. Eyerman, E. Georganas, W. Heirman, and J. Torrellas, "DECA: A near-core LLM decompression accelerator grounded on a 3D roofline model," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*, 2025, pp. 184–200.

[43] M. D. Hill and V. J. Reddi, "Gables: A roofline model for mobile SoCs," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 317–330.

[44] C. Nugteren, G.-J. van den Braak, and H. Corporaal, "Roofline-aware DVFS for GPUs," in *Proceedings of International Workshop on Adaptive Self-tuning Computing Systems (ADAPT)*, 2014, pp. 8–10.

[45] T. Gregersen, P. Patel, and E. Choukse, "Input-dependent power usage in GPUs," in *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '24. IEEE Press, 2024, pp. 1872–1877. [Online]. Available: https://doi.org/10.1109/SCW63240.2024.00235