Input-Dependent Power Usage in GPUs

Theo Gregersen[1], Pratyush Patel[1], Esha Choukse[2]
[1]University of Washington, [2]Microsoft Azure Research - Systems

***Abstract*—GPUs are known to be power-hungry, and due to the boom in artificial intelligence, they are currently the major contributors to the high power demands of upcoming datacenters. Most GPU usage in these popular workloads consists of large general matrix-matrix multiplications (GEMMs), which have therefore been optimized to achieve high utilization of hardware resources.**

In this work, we show that modifying the input data to GEMMs, while maintaining the matrix shapes and sizes can notably change the power consumption of these kernels. We experiment with four kinds of input variations: value distribution, bit similarity, placement, and sparsity, across different data types. Our findings indicate that these variations can change the GPU power usage during GEMM by almost 40%.

We hypothesize that input-dependent power usage variations occur due to changes in the number of bit flips in the GPUs. We propose leveraging this property through compiler and scheduler optimizations to manage power and reduce energy consumption.

***Index Terms*—GPUs, power, energy, sparsity**

I. Introduction

Power demand for datacenters, supercomputers, and machine learning is exploding, mainly driven by the growth in large language models (LLMs) [1]–[4]. Recent estimates forecast substantial annual increases in datacenter energy consumption and raise concerns about demand exceeding grid capacity in next few years [3], [5], [6]. With the industry pushing for more compute, addressing power consumption is key to sustainability [7]–[9]. Previous work has explored and utilized various techniques for managing power, such as power capping [10], [11], frequency scaling [9], approximation [12], [13], and batching [14], [15]. Our work shows that input data can also be utilized to manage power.

We focus on compute kernels and datatypes applicable to a variety of accelerator workloads. Accelerators such as GPUs are central to modern machine learning. For instance, Meta plans to amass around 600,000 H100 GPUs by the end of 2024 [16], Microsoft has enabled OpenAI training and inference with large cluster deployments of A100 and H100s [17], and the top supercomputers leverage GPUs [18]. GPUs are also very power hungry. For instance, recent NVIDIA H100 SXM5 GPUs have a total power draw of 700 watts [19], and a DGX-H100 consisting 8 GPUs needs provisioning of 10,000W [19].

Research on reducing accelerator peak power draw or energy often focuses on hardware design and the surrounding infrastructure: system design and scheduling [14], [15], [20], [21], power control [9], [22], or efficient chips [23]–[25]. Instead, we target input data for general matrix multiplication (GEMM) kernels [26]–[28]. GEMM kernels comprise a large portion of machine learning cycles and are important operations for GPUs due to natural compatibility with parallel execution [24], [29], [30]. Prior work explores efficient implementations of GEMM [31]–[33] and the performance impacts of quantization [34]–[36], sparsity, and data ordering [37] on GEMM in LLMs. We demonstrate that GEMM input values and placement can have significant impact on GPU power as well.

While research has broadly characterized accelerator power during machine learning and high-performance computing (HPC) workloads [4], [8], minimal prior work measures GPU power consumption due to varying inputs. Previous work has shown that input data values can significantly impact GPU power, but only considered single instructions such as FMUL or IMUL [38]. Bhalachandra et al. investigated the effects of GEMM input patterning on power, but only looked at input value entropy for a single datatype and placement of zero versus non-zero values [39]. In addition to input value entropy, we explore the impact of other patterns such as sorting, bit-level sparsity, hamming weight, and similarity. We also compare across several datatypes and assess the effect of NVIDIA tensor cores. We find that GEMM input patterns can change GPU power consumption by up to 38%. This observation lends itself to a variety of potential future applications for power and energy efficiency: power-aware sparsity, data pruning for power capping, and efficient data placement algorithms.

II. Background

GEMM operations are fundamental to machine learning [30], [40] and many common computational tasks [41]. As such, GEMM is an important target for power efficiency.

GEMM is a fundamental linear algebra operation. For matrix A with dimensions (N, K), matrix B with dimensions (K, M), matrix C with dimensions (N, M), and scalars α and β, a standard GEMM execution typically computes the following matrix output [27]:

$$D = \alpha A \cdot B + \beta C$$

To reduce memory use, the D output matrix is often set to C and updated in-place. GPU makers such as NVIDIA typically provide proprietary kernels (i.e., compute routines) to execute operations like GEMM on their hardware [29]. Kernel libraries such as cuBLAS [26] and cuSPARSE [28] are available through public APIs, however the underlying implementations are black boxes. A more transparent alternative is CUTLASS [27], an open-source kernel library maintained by NVIDIA.

To improve performance, NVIDIA GPUs can utilize tensor cores. Tensor cores are specialized for matrix math operations

such as matrix multipy and accumulate (MMA) and provide acceleration for specific datatypes. For instance, the NVIDIA Ampere architecture provides 20× FP32 MMA throughput compared to the previous generation [42].

III. Experiment Setup

To explore the impact of GEMM inputs on GPU power, we run a series of GEMM operations on an NVIDIA A100 PCIe virtual machine (VM) hosted on Azure.[1] The NVIDIA A100 PCIe GPU has a maximum thermal design power (TDP) of 300 watts [42]. We use standard NVIDIA CUTLASS kernels [27] optimized for GEMM execution, measure power every 100ms with NVIDIA dcgm command-line tools [43], and measure elapsed time with C++ standard library high resolution clocks.

All experiments use 2048 by 2048 matrices. We selected 2048 as the largest power of two that did not consistently throttle the A100 GPU. During our experiments, the A100 GPU averaged 98.5% utilization. The C matrix is zeroed, and both A and B matrices use the same pattern with B transposed unless otherwise noted. Reported results are averaged over 10 seeds with 20k iterations each for FP16-T and 10k iterations each for the other datatypes. The A and B matrices use different seeds.

We explore four datatype setups: 32-bit floating point (FP32), 16-bit floating point (FP16), 16-bit floating point with tensor cores enabled (FP16-T), and 8-bit integer (INT8). For each datatype, we experiment with value similarity, physical (bit) similarity, data placement, and sparsity. All of the floating point experiments use the same generated FP32 values, with numeric conversion to their respective datatypes (round to nearest value). When generating input values, we use appropriate parameters to ensure that all values practically fall within each datatype's representation range.

During testing, we observed slight variations in power measurements depending on when experiments were run. Power measurements occasionally shifted by up to 10W when the VM instance changed, even when using the same configuration. We attribute this to process variation across GPUs. To minimize this effect, we executed all experiments on the same VM instance. We also trim the first 500ms of power measurements to account for warmup. Across all experiments for a given datatype, the average iteration runtime (Figure 1) was consistent to a microsecond-level; this is expected since each experiment uses the standard cutlass kernel. Figure 2 shows average iteration energy for a GEMM operation with Gaussian random variables. Note the identical patterns between the iteration runtimes and energy observations, showing that the power used with random variables is similar across the input types. In the rest of paper, we report power measurements rather than total energy, as power is the key bottleneck for large-scale machine learning [1], [2], [8].

[1]Experiment code and data can be found at https://github.com/theo-gregersen/input-dependent-power-sc24.

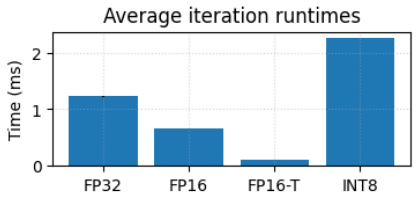

Fig. 1. Average iteration runtime by datatype for 2048x2048 GEMM across all A100 GPU experiments. Note that the error bars are a magnitude smaller; iteration runtimes are very consistent across experiments.

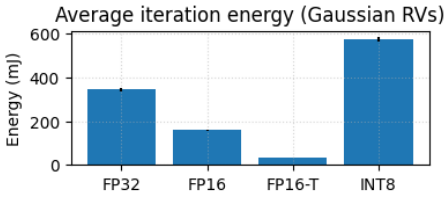

Fig. 2. Average iteration energy on A100 GPU for 2048x2048 GEMM filled with random variables from a Gaussian distribution. The distribution has a mean of 0 and a standard deviation of 2^{10} for FP and 2^5 for INT8.

IV. Input-Dependent Power Analysis

We profile different kinds of inputs to GEMM kernels and provide our takeaways (**Tn**).

A. Value Distribution

First, we explore the effects of input distribution on GPU power during GEMM.

Distribution standard deviation: Figure 3a shows the average power draw during GEMM when the A and B matrices are filled with Gaussian random variables with a fixed mean of 0 and varied standard deviation. ***T1:*** *Input distribution standard deviation does not significantly impact power.*

Distribution mean: For the results in Figure 3b, the A and B matrices are filled with Gaussian random variables with a fixed standard deviation of 1 and varied mean. ***T2:*** *Larger input value means can reduce power for FP datatypes.*

Inputs from a set: The third experiment (Figure 3c) fills A and B with values selected uniformly, with replacement, from a set of Gaussian random variables with a mean of 0 and standard deviation of 2^{10} for FP and 2^5 for INT8. ***T3:*** *Inputs from a small set of unique values decrease power consumption.*

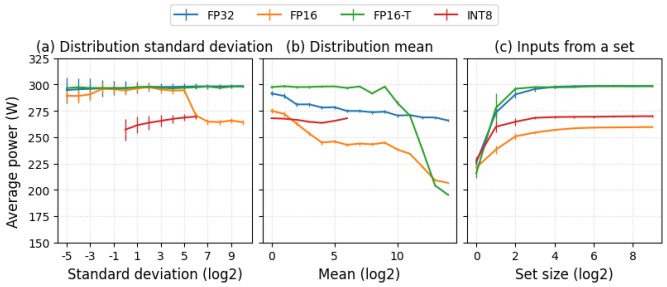

Fig. 3. Effects of input value distribution on GPU power.

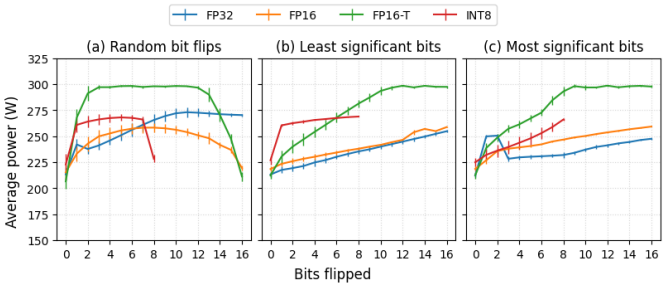

Fig. 4. Effects of bit similarity on GPU power.

B. Bit Similarity

Next, we consider input data bit similarity. In these experiments, the A matrix is initially filled with one random value and the B matrix is filled with another random value.

Random bit flips: Figure 4a illustrates how power changes when random bits are flipped in each element. ***T4:*** *Input data with highly similar bits uses less power.*

Least significant bits: Figure 4b shows how power changes as the least significant bits are randomized. ***T5:*** *As more least significant bits are randomized, power increases.*

Most significant bits: Figure 4c illustrates how power shifts when the most significant bits are randomized. ***T6:*** *As more of the most significant bits are randomized, power increases.*

Input data types: Figure 4 also shows that FP16-T on tensor cores has the highest power usage compared to the other data types. This is important to note, since the default data type in AI applications is FP16-T. Approximation related research tries to reduce the time and memory to run AI inference, but can also have a positive impact on power efficiency. ***T7:*** *FP16-T is the most power hungry data type.*

C. Placement Patterns

Next, we explore the impact of input data placement on GEMM power. Across each of the following experiments, the initial values for matrices A and B are constant. Both matrices are filled with random variables from a Gaussian distribution with a mean of 0 and standard deviation of 2^{10} for FP and 2^5 for INT8.

Sorted into rows: For this experiment (Figure 5a), we partially sort both matrices into rows. Sorting n percent means that the lowest n percent of values are sorted into the first n percent of indices (row-wise). The B matrix is not transposed. ***T8:*** *Sorting input values can decrease power consumption.*

Sorted and aligned: For Figure 5b, the matrices are partially sorted into rows again. However, this time the B matrix is transposed, so the lowest values in A are multiplied with the lowest values in B during GEMM. ***T9:*** *Aligning sorted values decreases power even more than just sorting.*

Sorted into columns: Figure 5c is a similar experiment to that of figure 5a, but the input values are sorted into columns rather than rows. ***T10:*** *Sorting values into columns can decrease power consumption.*

Sorted within rows: We also experiment with sorting within matrix rows and aligning across matrices. Figure 5d shows how power changes when the A and B matrix rows are partially sorted. ***T11:*** *Intra-row sorting can decrease power, but to a lesser extent than sorting fully.*

D. Sparsity

Next, we explore the impact of sparsity on GEMM power. These experiments use standard GEMM, not sparse GEMM.

General sparsity: Figure 6a shows power as the matrix is made sparser. ***T12:*** *Matrix sparsity decreases GEMM power.*

Sparsity after sorting: In Figure 6b, the initial A and B matrices are fully sorted before sparsity is added. With this patterning, power has a curve that peaks around $30 - 40\%$ sparsity for floating point datatypes. Although both sorting and sparsity decrease power in isolation, this trend indicates that they do not compound when paired. ***T13:*** *Sparsity applied to sorted matrices can actually increase power consumption.*

Sparsity in least significant bits: Finally, we consider sparsity in physical structure. Figure 6c is the result of setting each matrix item's least significant bits to zero. ***T14:*** *Zeroing least significant bits can reduce power.*

Sparsity in most significant bits: Figure 6d illustrates the effect of setting each matrix item's most significant bits to zero. ***T15:*** *Zeroing most significant bits can reduce power.*

E. Generalization

Our results hold across different GPU generations. We show this by replicating several of the experiments on an NVIDIA H100 80GB HBM3 GPU (TDP 700W, local cluster), NVIDIA Quadro RTX 6000 24GB GPU (TDP 260W, Chameleon cloud [44]), and NVIDIA Tesla V100-SXM2-32GB GPU (TDP 300W, Chameleon cloud [44]). We present results for FP16 runs of the *Distribution mean* experiment, *Most significant bits* experiment, *Sorted into rows* experiment, and *General sparsity* experiment. Figure 7 illustrates the results across GPUs. The matrix size was 512 by 512 for the RTX 6000 (it throttled at 2048 by 2048) For the V100, A100, and H100 GPUs, power consumption trends are consistent. The RTX 6000 has less prominent changes in power, likely because it is the oldest of the tested GPUs (e.g., uses GDDR6 memory rather than HBM) and has a lower TDP.

F. Bit Alignment and Hamming Weight

To investigate broader trends across experiments, we look at bit alignment between values multiplied during GEMM, as well as Hamming weights of the matrix values. Bit alignment between two values is 0 if all of the bits are opposite, and alignment is 1 if all of the bits are the same. Figure 8 illustrates average GPU power during GEMM in relation to the average bit alignment between the A and B matrices and average Hamming weight in the A matrix (B has similar weight because of shared patterns). Each dot represents one experiment configuration from the prior subsections. Across all floating point datatypes, there seems to be correlation with higher bit alignment or lower Hamming weight and decreasing

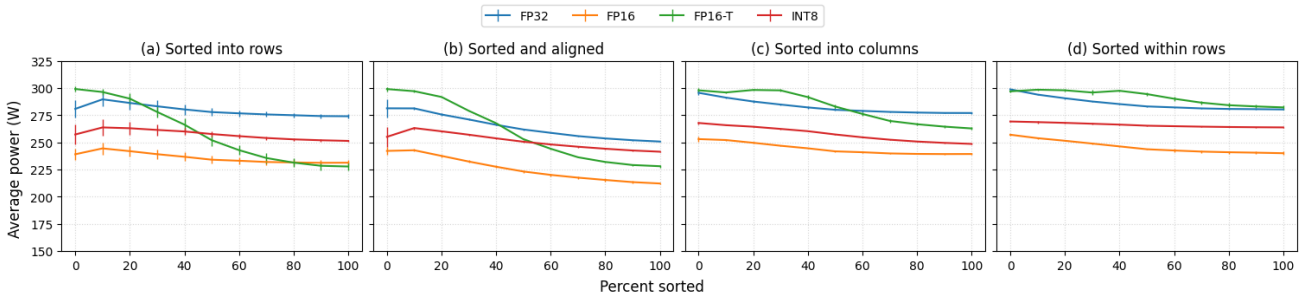

Fig. 5. Effects of input value placement on GPU power.

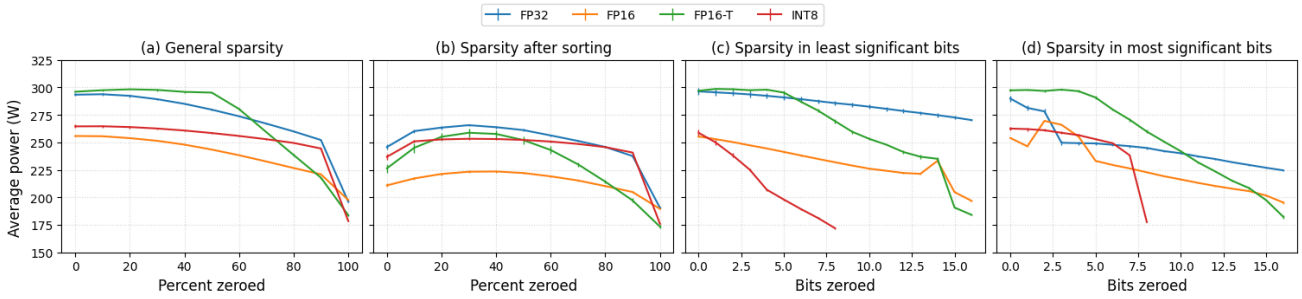

Fig. 6. Effects of input value sparsity on GPU power.

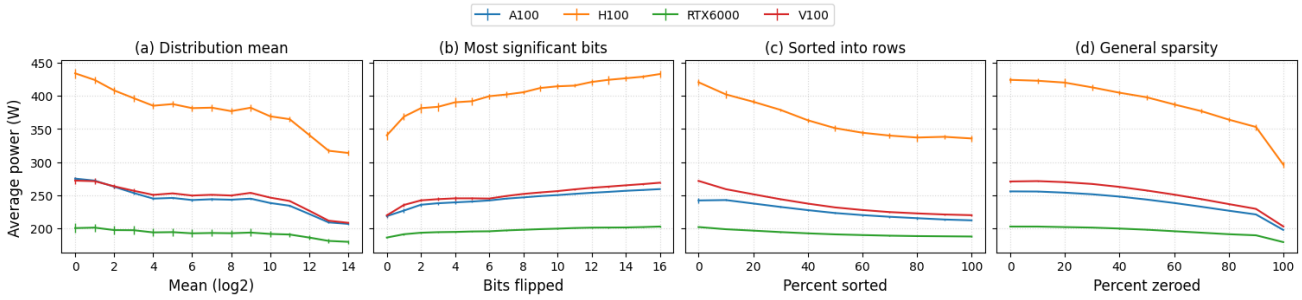

Fig. 7. Experiment results across different NVIDIA GPUs.

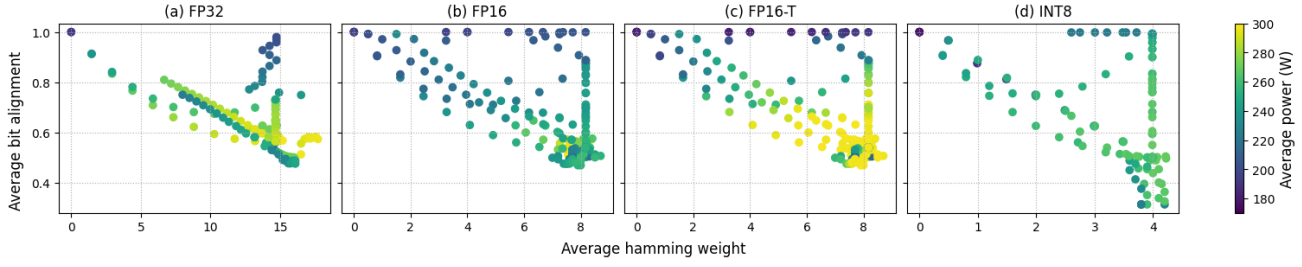

Fig. 8. Bit alignment and hamming weight of input values.

average GPU power during GEMM. However, this is not an entirely consistent trend.

V. Discussion and Future Work

Going forward, we plan to identify what causes input-dependent power usage variation in GPUs and develop practical techniques to improve the power and energy efficiency of GPU applications at scale. We list below future directions.

Identifying Causes. Based on prior work, we hypothesize that GPU power draw depends on inputs due to changes in the number of bitflips during computation [45], or how many bits are set [38]. For example, running a GEMM with zero matrices would incur no bitflips, and thus, it likely has lower power draw. Value similarity likely helps by reducing bit flipping at the hardware level. We plan to do extensive experiments to validate this hypothesis and investigate other hardware-level factors that might contribute to reduced power draw.

Input-dependent GPU Power Models. We are building input-dependent GPU power models to more precisely capture how input variations impact the GPU peak power draw. Such a power model would take in different data patterns as inputs (*e.g.*, specified via a domain-specific language), and estimate the power usage as output. Using such power models, future work could build power-aware compilers and optimizers to reduce the power draw of GPU applications that can tolerate input variations.

Power- and Energy-efficient Machine Learning. Since machine learning applications, like large language models, are very power intensive [8], a key future direction is to leverage input changes to drive down their power and energy usage. Specifically, we are exploring three different directions. First, we are trying to modify model weights into value ranges that use less power; for example, shifting the weight values towards larger averages could reduce the power draw as shown in Section IV. Second, we plan to investigate whether we can partially or fully sort neural network model weights so as to reduce power draw. Since weights within a layer correspond to independent neurons, rearrangement is computationally equivalent as long as each neuron processes its own input. Recent work leverages permutation invariant transformations to manipulate GPU tiles without changing the computation results [46]. Similar transformations could potentially be used to set patterns that reduce power. Finally, we would like to develop sparsity designs that reduce power usage while also optimizing performance, accuracy, and/or memory trade-offs [37]. While it is challenging to estimate/limit the impact of input variations on model accuracy, we are hopeful that the benefits will outweigh the pitfalls.

Acknowledgment

This work is supported by NSF CNS-2104548 and the UW Center for the Future of Cloud Infrastructure (FOCI). Some results in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

References

[1] L. Lin, R. Wijayawardana, V. Rao, H. Nguyen, W. E. Gnibga, and A. Chien, "Exploding ai power use: an opportunity to rethink grid planning and management," 2024.

[2] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Behram, J. Huang, C. Bai, M. K. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. S. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. K. Jain, M. G. Rabbat, and K. M. Hazelwood, "Sustainable ai: Environmental implications, challenges and opportunities," *ArXiv*, vol. abs/2111.00364, 2021.

[3] L. Lin and A. A. Chien, "Adapting datacenter capacity for greener datacenters and grid," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, e-Energy '23, (New York, NY, USA), p. 200–213, Association for Computing Machinery, 2023.

[4] Z. Zhao, E. Rrapaj, S. Bhalachandra, B. Austin, H. A. Nam, and N. Wright, "Power analysis of nersc production workloads," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23, (New York, NY, USA), p. 1279–1287, Association for Computing Machinery, 2023.

[5] S. Bangalore, A. Bhan, A. D. Miglio, P. Sachdeva, V. Sarma, R. Sharma, and B. Srivathsan, "Investing in the rising data center economy." https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy, 2023.

[6] K. Blunt and J. Hiller, "Big tech's latest obsession is finding enough energy." https://www.wsj.com/business/energy-oil/big-techs-latest-obsession-is-finding-enough-energy-f00055b2?st=ju6z5e4fcscj1ad&reflink=desktopwebshare_permalink, 2024.

[7] P. Patel, T. Gregersen, and T. Anderson, "An agile pathway towards carbon-aware clouds," in *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, HotCarbon '23, (New York, NY, USA), Association for Computing Machinery, 2023.

[8] P. Patel, E. Choukse, C. Zhang, I. Goiri, B. Warrier, N. Mahalingam, and R. Bianchini, "Characterizing power management opportunities for llms in the cloud," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2024.

[9] P. Patel, Z. Gong, S. Rizvi, E. Choukse, P. Misra, T. Anderson, and A. Sriraman, "Towards improved power management in cloud gpus," *IEEE Computer Architecture Letters*, vol. 22, no. 2, pp. 141–144, 2023.

[10] S. Li, X. Wang, X. Zhang, V. Kontorinis, S. Kodakara, D. Lo, and P. Ranganathan, "Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware power capping at scale," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 1241–1255, USENIX Association, Nov. 2020.

[11] C. Zhang, A. G. Kumbhare, I. Manousakis, D. Zhang, P. A. Misra, R. Assis, K. Woolcock, N. Mahalingam, B. Warrier, D. Gauthier, L. Kunnath, S. Solomon, O. Morales, M. Fontoura, and R. Bianchini, "Flex: High-availability datacenters with zero reserved power," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 319–332, 2021.

[12] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, mar 2016.

[13] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: approximate data types for safe and general low-power computation," in *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, (New York, NY, USA), p. 164–174, Association for Computing Machinery, 2011.

[14] J. You, J.-W. Chung, and M. Chowdhury, "Zeus: Understanding and optimizing GPU energy consumption of DNN training," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, (Boston, MA), pp. 119–139, USENIX Association, Apr. 2023.

[15] J.-W. Chung, Y. Gu, I. Jang, L. Meng, N. Bansal, and M. Chowdhury, "Perseus: Removing energy bloat from large model training," 2023.

[16] A. Heath, "Mark zuckerberg's new goal is creating artificial general intelligence." https://www.theverge.com/2024/1/18/24042354/mark-zuckerberg-meta-agi-reorg-interview, 2024.

[17] OpenAI, "Scaling kubernetes to 7,500 nodes." https://openai.com/index/scaling-kubernetes-to-7500-nodes/, 2024.

[18] TOP500, "Top 500 list." https://top500.org/lists/top500/2024/06/, June 2024.

[19] "Nvidia h100 tensor core gpu architecture." https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper.

[20] S. Choi, I. Koo, J. Ahn, M. Jeon, and Y. Kwon, "EnvPipe: Performance-preserving DNN training framework for saving energy," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, (Boston, MA), pp. 851–864, USENIX Association, July 2023.

[21] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024.

[22] T. Komoda, S. Hayashi, T. Nakada, S. Miwa, and H. Nakamura, "Power capping of cpu-gpu heterogeneous systems through coordinating dvfs and task mapping," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 349–356, 2013.

[23] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.

[24] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 58–70, 2020.

[25] M. Tibaldi and C. Pilato, "A survey of fpga optimization methods for data center energy efficiency," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 3, pp. 343–362, 2023.

[26] "cublas basic linear algebra on nvidia gpus." https://developer.nvidia.com/cublas, 2024.

[27] "Cutlass cuda templates for linear algebra subroutines and solvers." https://github.com/NVIDIA/cutlass, 2024.

[28] "cusparse." https://developer.nvidia.com/cusparse, 2024.

[29] "Matrix multiplication background user's guide." https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html, 2023.

[30] S. Pati, S. Aga, N. Jayasena, and M. D. Sinclair, "Demystifying bert: System design implications," in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 296–309, 2022.

[31] C. Hong, A. Sukumaran-Rajam, I. Nisa, K. Singh, and P. Sadayappan, "Adaptive sparse tiling for sparse matrix multiplication," in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, PPoPP '19, (New York, NY, USA), p. 300–314, Association for Computing Machinery, 2019.

[32] T. Gale, M. Zaharia, C. Young, and E. Elsen, "Sparse gpu kernels for deep learning," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, 2020.

[33] A. Mehrabi, D. Lee, N. Chatterjee, D. J. Sorin, B. C. Lee, and M. O'Connor, "Learning sparse matrix row permutations for efficient spmm on gpu architectures," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 48–58, 2021.

[34] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh, "Spqr: A sparse-quantized representation for near-lossless llm weight compression," 2023.

[35] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2023.

[36] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021.

[37] H. Tang, S. Yang, Z. Liu, K. Hong, Z. Yu, X. Li, G. Dai, Y. Wang, and S. Han, "Torchsparse++: Efficient training and inference framework for sparse convolution on gpus," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '23, (New York, NY, USA), p. 225–239, Association for Computing Machinery, 2023.

[38] J. Lucas and B. Juurlink, "Alupower: Data dependent power consumption in gpus," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 95–104, 2016.

[39] S. Bhalachandra, B. Austin, S. Williams, and N. J. Wright, "Understanding the impact of input entropy on fpu, cpu, and gpu power," *ArXiv*, vol. abs/2212.08805, 2022.

[40] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient processing of deep neural networks: A tutorial and survey," 2017.

[41] L.-H. Lim, "Tensors in computations," *Acta Numerica*, vol. 30, p. 555–764, May 2021.

[42] "Nvidia a100 tensor core gpu architecture." https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

[43] "Nvidia dcgm." https://developer.nvidia.com/dcgm, 2024.

[44] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*, USENIX Association, July 2020.

[45] G. Pekhimenko, E. Bolotin, N. Vijaykumar, O. Mutlu, T. C. Mowry, and S. W. Keckler, "A case for toggle-aware compression for gpu systems," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 188–200, IEEE, 2016.

[46] N. Zheng, H. Jiang, Q. Zhang, Z. Han, L. Ma, Y. Yang, F. Yang, C. Zhang, L. Qiu, M. Yang, and L. Zhou, "Pit: Optimization of dynamic sparse deep learning models via permutation invariant transformation," in *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, (New York, NY, USA), p. 331–347, Association for Computing Machinery, 2023.