Studying Spatial and Temporal effects of running ML workloads across GPUs and across Nodes

Tanmay Anand, Akanksha Chaudhari, Aboli Pai, Lakshika Rathi

1 Introduction

Machine learning has facilitated tremendous advancements across a wide range of application domains, including image classification [9], translation, language modeling [14], and video captioning [15], climate modeling [12], anomaly detection [11], drug discovery [5], and is thus being widely deployed in many shared-resource distributed environments. Since, ML workloads are typically *memory-intensive*, *compute-intensive*, and *long-running*, they often require parallel execution across multiple hardware devices, such as GPUs and other general-purpose accelerators. In order to ensure efficient execution and sharing of resources, it is important to consider how these jobs are allocated (i.e., spatially co-located) and scheduled (i.e., temporally co-located) across multiple nodes and devices on the cluster.

When jobs are *spatially co-located*, i.e. executing in parallel on neighbouring devices of the same node in a shared-resource distributed environment, the execution of one job may affect the performance of another job executing on a neighbouring node due to high contention for shared resources. For instance, consider a scenario where multiple memory-intensive jobs are co-located on neighbouring devices of the same node, the performance of all co-located jobs could be affected due to increased wait times resulting from parallel memory accesses blocking each other. Previous works [7] present interesting results on the effect of spatial co-location on GPU scheduling. We plan to take these results into consideration while also studying how co-location affects power and thermal efficiency.

Similarly, when jobs are temporally co-located, i.e. executing sequentially on the same device of the same node, the execution characteristics of a job currently running on the device may affect the performance of the next job scheduled to run on this node. For instance, consider a scenario where a compute-intensive job is executing at a high frequency on device 0 of a node – this would typically result in higher power consumption and thermal dissipation levels, resulting in increased device temperature. The device would then either have to idle or run at a lower frequency while it cools down – which can affect the performance of the next job scheduled on this device. Some works [4], [8], investigate the relation between power consumption and performance. However, these studies were conducted on an older generation of GPUs and focus on workload performance when running on a single GPU. While later works such Sinha et al. [13] present a comprehensive study on how a GPU's power, frequency, and temperature can impact its performance – they only evaluate the performance when running a single application, leaving the spatial and temporal effects of running multiple applications for future work.

This project. In this project, we aim to investigate the effects of spatially and temporally co-locating different ML workloads across GPUs and across nodes on a cluster. We begin by broadly categorizing the workloads we evaluate into two categories – memory-intensive workloads and compute-intensive workloads – based on the profiling metrics gathered using NSight Compute. We then study the spatial and temporal effects by considering different placements for both memory-intensive and compute-intensive workloads on TACC Frontera. In our experiments, we measure the power, temperature, GPU Utilization and frequency for each run using nvidia-smi. We also present experiments that introduce cooldown periods between different workload runs to check if that can improve the thermal efficiency and overall system performances. results, we then comment on the effectiveness of different placement strategies for memory-intensive and compute-intensive workloads.

Outline The rest of this project report is organized as follows. In Section 2, we discuss relevant background material and related work required to understand this work. In Section 3, we describe our evaluation methodology detailing the kinds of workloads we consider and the insights we hope to gather from our experiments. In Section 4, we first describe the setup used in our evaluations and then present the results of our co-location experiments. We then conclude in Section 5 by summarizing the results and insights reported here, and attempt to make reasonable recommendations on what co-location policies would be best suited for a given workload characteristic.

Metric	Pagerank	SGEMM	BERT	Resnet
Double-precision FU utilization (in %)	0	0	0	0
Single-precision FU utilization (in %)	1	10	3.61	9.60
Half-precision FU utilization (in %)	0	0	0.34	0
Tensor-precision FU utilization (in %)	0	0	1.21	0
Special FU utilization (in %)	0	0	2.54	1.22
DRAM utilization (in %)	1.00	4.09	7.30	8.54

Table 1: Profiling metrics for the workloads we evaluate

2 Background and Related Work

There are many papers that optimize application co-location on GPUs by measuring performance for a few latency-sensitive applications. Xiao, et al. [16] present *Gandiva* that improves job performance by time-slicing the GPU across multiple Deep Learning training jobs. It dynamically migrates jobs to achieve better cluster utilization for compute intensive applications. Jeon, et al. [7] present a detailed Deep Neural Networks workload characterization, and study how different factors affect the cluster utilization. They show that low GPU utilization can be caused as a result of disregarding locality when scheduling jobs and due to contention of shared resources when jobs are co-located. Both these works majorly focus on performance, not including power and thermal effects.

Coplin, et al. [4] study the effects of changing the core and memory clock frequencies on the power drawn, energy consumption and runtime of a GPU. It focuses on both memory-bound and compute-bound codes but limits the analysis to a single GPU. Additionally, it does not consider any temporal effects.

Chen, et al. [3] present *Prophet* that predicts performance degradation due to application co-location on accelerators. They considered the performance degradation due to contention of memory bandwidth. This work does not assess the job performance impact for a mix of different compute and memory intensive workloads.

Jiao, et al. [8] also investigated the correlation between power consumption and performance for different memory and compute workloads. However, these studies focus on the performance of a single GPU that only runs one application at a time. Further, they do not study the temporal effects (i.e., execution order of jobs on the same GPU).

Sinha, et al [13] studied the impact of a GPU's power, frequency, and temperature on it's performance. They perform experiments on different GPU clusters for a variety of workloads. We treat that paper as a foundation and aim to explore how spatial and temporal co-location of jobs affect power, frequency and temperature for different workloads on various GPU clusters.

Prior works have looked into temperature-aware job placements for CPU-based HPC systems. Menon, et al. [10] uses Dynamic Voltage and Frequency Scaling (DVFS) to control the temperature and perform load balancing. Ge, et al. [6] provides insights on the impact of DVFS on application performance and energy efficiency for compute-bound workloads. Zhang, et al. [17] predicts the thermal condition of the system based on application mapping.

3 Methodology

In this section, we describe the details of our methodology used to evaluate spatial and temporal effects of running ML workloads across GPUs and across nodes

3.1 Workload Characterization

Machine learning (ML) workloads may stand out due to their prolonged nature and high demands for memory and computational resources. These demands, whether memory-intensive, compute-intensive, or a combination of both, can significantly influence the performance of co-located workloads. So, we first need to understand the workload characteristics.

These key characteristics can affect the performance of other workloads in different ways. Spatially co-locating memory intensive workloads can cause high contention for the shared memory resources. Running a compute-intensive workload can lead to high power consumption and thermal dissipation. This might degrade the performance of workloads that are running after it. Long-running ML workloads can further exacerbate the thermal efficiency by increasing core temperatures.

In order to understand and reason how these factors influence the results of our experiments, we begin by characterizing our workloads into two different categories based on the profiling metrics which we collect using NSight Compute. Specifically, we record the FU utilizations for different compute units and take the maximum of these to denote the degree of compute-intensiveness. We also record the DRAM utilization to measure the memory-intensiveness of our work-

loads. We plot the normalized (compute-intensiveness, memory-intensiveness) tuples as shown in Figure 1 and classify all the workloads above the x = y line as compute-intensive and all the workloads below the x = y as memory-intensive[1]. We gather these metrics for four workloads as shown in Table 2. According to the described methodology, we classify `sgemm` and `resnet` as compute-intensive workloads and `pagerank` and `bert` as memory-intensive workloads.

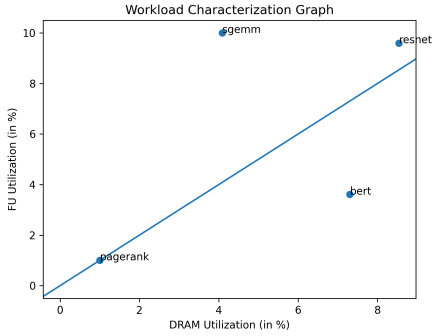


Figure 1: FU & DRAM Utilization tuple plot for our workloads of interest

3.2 Designing Experiments

Based on the insights we gather from the background and prior works presented in Section 2, we propose the following hypotheses and correspondingly design experiments to explore the validity of each of them.

Hypothesis #1(a): Temporally co-locating compute-intensive ML workloads may result in performance degradation.

Rationale: Temporal co-location of two ML workloads on a single device can lead to performance degradation, especially for the second co-located workload. This is because, when the first compute-intensive workload executes, the frequency and power consumption levels of the device go up to enable higher performance. However, this can also result in high levels of thermal dissipation and increased device temperature, especially in case of long-running workloads. In extreme cases, this can also cause thermal throttling in the system (as described in Section 2) further exacerbating the thermal efficiency. In such cases, the system recovers by lowering the operating frequency to allow the device to cool down and come back to normal operating ranges. This reduction in frequency can degrade the performance of the next workload to execute.

Hypothesis #1(b): Introducing a cooldown period between sequential executions of workloads can mitigate performance degradation resulting from overheating of the device and/or thermal throttling.

Rationale: An alternative mechanism to recover a thermally overloaded/throttled system is to let the device idle for a short duration between two runs, allowing it to cool down. This would mean that the system does not have to operate at a reduced frequency for too long in order to recover, thereby mitigating the performance degradation seen for the next workload to execute.

In order to support our hypotheses, we temporally co-locate compute-intensive workloads, demonstrating the performance degradation resulting from thermal throttling (see Figure 3). We then repeat this experiment by introducing different cooldown periods between consecutive runs, demonstrating a slight improvement in performance for subsequent runs (see Figures 4 and 5).

In addition to this, we also conduct temporal co-location experiments for memory-intensive workloads (see Figure 6) to show that our hypotheses hold specifically for the compute-intensive workloads.

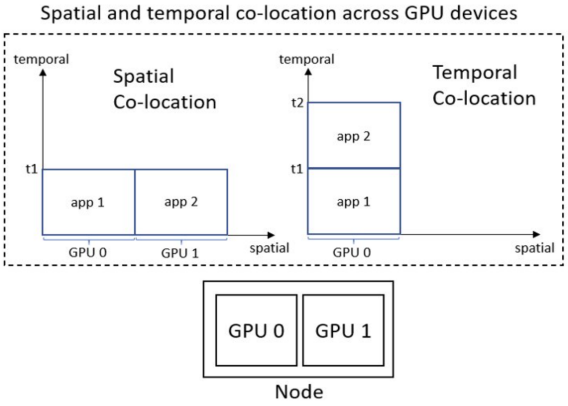


Figure 2: Spatial and Temporal Co-location

Hypothesis #2: Co-locating memory-intensive workloads might lead to increased completion times.

Rationale: Spatially co-locating two memory-intensive workloads on different devices of the same node (shared resource, distributed environment) can result in high contention for memory bandwidth. Moreover, depending on the workload access patterns the two spatially co-locating workloads may result in blocking memory accesses and increased wait times, adding to the overall run time for one or both of the workloads.

In order to support our hypothesis, we temporally co-locate compute-intensive workloads demonstrating the performance degradation resulting from thermal throttling (see Figure 3).

[1]We understand that a better way to characterize these workloads would be to use roofline models or more sophisticated methodologies (such as [2]). However, given the timeline for our project, and that our primary focus is to explore the effects of spatial and temporal co-location, we choose to go with this simpler strategy.

4 Evaluation

In this section, we describe the details of our experimental evaluation. We first begin by giving the details of our experimental setup, workloads and evaluation metrics in Section 4.1. We then present the results of our temporal and spatial co-location experiments in Section 4.2.

4.1 Setup

Cluster Configuration: In order to deploy our workloads, we used Frontera (TACC) [1] with following configuration:

- **GPU**: RTX 5000
- **Number of GPUs per Node**: 4
- **Cooling Mechanism**: Mineral Oil

However, in order to evaluate temporally co-located compute-intensive workloads with long running kernels such as SGEMM, we ran our workloads over a physically accessible standalone machine with NVIDIA Titan V GPU. Moreover, idle node unavailability over weekdays on TACC motivated us to also evaluate SGEMM over a standalone single GPU machine.

Workloads: To ensure that our study provides a holistic view with regards to co-location of ML workloads across devices and nodes, we chose ML workloads which are compute-intensive, memory-intensive, or balanced.

Metrics for workload classification:

- FU Utilization
- DRAM Utilization

Our workloads of interest include:

- ResNet-50
- SGEMM
- BERT

Evaluation Metrics: In order to fetch detailed performance metrics across all our experiments, we used NVIDIA-SMI to get detailed system level insights, running guided analysis across performance metrics & compare results across workloads.

Metrics to be collected:

- GPU CU/SM Utilization (%)
- GPU CU/SM temperature (°C)
- GPU CU/SM power consumption (in Watts)
- GPU CU/SM frequency (in MHz)

4.2 Results

In this section, we present the results of our co-location experiments.

4.2.1 Temporal Co-location Experiments

In this section, we present the results of the temporal co-location experiments we conduct on both compute-intensive and memory intensive workloads.

In order to evaluate effects of cooldown over temporal co-location of compute intensive workloads, we dispatched SGEMM workloads (each job running 100 kernels with Matrix Size: 25536) and Bert workloads. Our evaluation scheme focuses on gathering insights over Peak & average value of each of the temperature, power & frequency metrics across workload runtime.

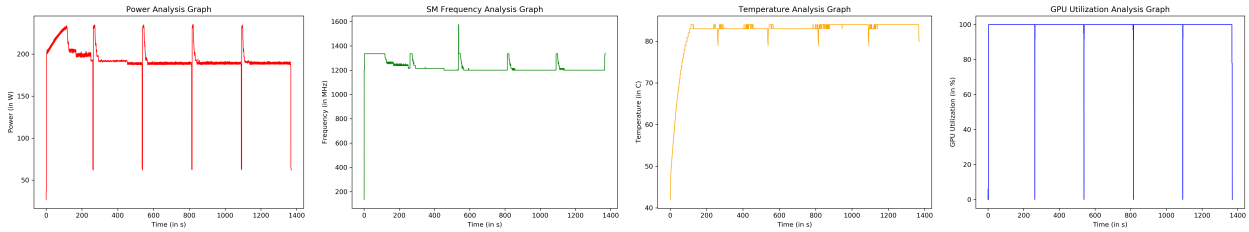
In the first experiment, we temporally co-locate the SGEMM compute-intensive workloads, demonstrating the performance degradation resulting from thermal throttling (see Figure 3) in support of Hypothesis #1(a). We observe in the figure that when workloads are run without any cool-down period, a peak power consumption of 238 W is observed, with sharp dips lowering the consumption. The power consumption reduces when GPU utilization dips as shown in Figure 3(d). The system reaches a maximum temperature of 85°C, which stays almost constant thereafter. The system does not get a chance to cool down, degrading the overall performance.

We then repeat this experiment by introducing different cooldown periods between consecutive runs to show a slight improvement in performance for subsequent runs (see Figures 4 and 5) in support of Hypothesis #1(b). When we introduce a cooldown of 120 seconds between runs, we observe a 4.5% reduction in job runtime and with a cooldown of 240 seconds, we see a reduction of 4.8%. During the cooldown periods, the power consumption reduces gradually, allowing reduction in the system temperature and SM frequency. The GPU utilization also flattens down during those time-periods. This leads to better performance of the system.

We also ran the experiment with the BERT memory-intensive workloads, with 8 jobs running sequentially without a cooldown period. We observe that temporal co-location does not degrade the performance in this case, as shown in Figure 6. As the GPU utilization fluctuates continuously, it gives a chance to the system to cool down. Hence, we do not observe thermal throttling for memory-intensive workloads.

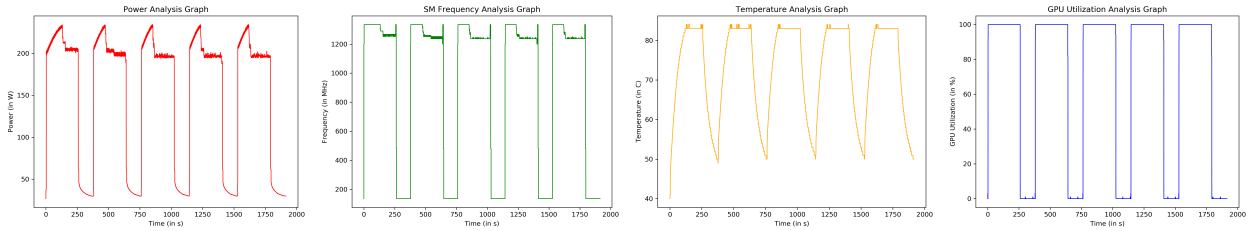
4.2.2 Spatial Co-location Experiments

In this section, we present the results of the spatial co-location experiments we conduct on both compute-



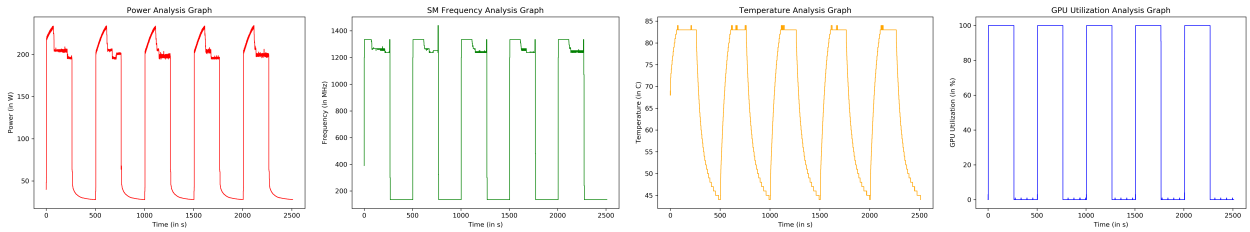
(a) (b) (c) (d)

Figure 3: Temporally co-locating compute-intensive workloads without cooldown (a) Power [Peak: 238.0 W, Avg: 192.51 W (b) SM Frequency [Peak: 1470.0 MHz, Avg: 1219.91 MHz] (c) Temperature [Peak: 85.0 °C, Avg: 82.75 °C], (d) GPU SM Utilization [Peak: 100%, Avg: 99.54%]



(a) (b) (c) (d)

Figure 4: Temporally co-locating compute-intensive workloads with a 120-second cooldown (a) Power [Peak: 234.44 W, Avg: 159.12 W] (b) SM Frequency [Peak: 1335.0 MHz, Avg: 979.01 MHz] (c) Temperature [Peak: 84.0 °C, Avg: 73.58 °C] (d) GPU SM Utilization [Peak: 100%, Avg: 72.14%]

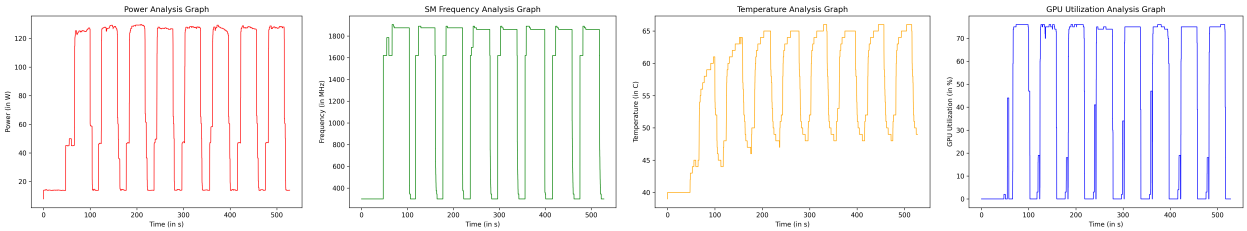


(a) (b) (c) (d)

Figure 5: Temporally co-locating compute-intensive workloads a with 240-second cooldown (a) Power [Peak: 234.4 W, Avg: 131.01 W] (b) SM Frequency [Peak: 1905.0 MHz; Avg: 1339.64 MHz] (c) Temperature [Peak: 84.0 C; Avg: 67.47 °C] (d) GPU SM Utilization [Peak: 100%, Avg: 56.26%]

intensive and memory intensive workloads.

In our experiments, we spatially co-locate BERT (a memory-intensive workload) across two GPUs of the same node and compare it against its baseline run on a single GPU. We observe that the average runtime of BERT increases by 31% over its baseline run when spatially co-located on the same node – this is in accordance with our expectations stated in Hypothesis #2. Since BERT is a memory-intensive workload, spatial co-location can lead to increased contention for memory bandwidth and increased number of blocking accesses, resulting in longer wait times. Hence, the latency of the



(a) (b) (c) (d)

Figure 6: Temporally co-locating memory-intensive workloads without cooldown (a) Power [Peak: 129.7 W, Avg: 78.64 W] (b) SM Frequency [Peak: 1905.0 MHz, Avg: 1339.64 MHz] (c) Temperature [Peak: 66.0 °C, Avg: 55.76 °C] (d) GPU SM Utilization [Peak: 76.0%, Avg: 40.99%]

Evaluation metrics	**Resnet-50**	**BERT**
Avg. Runtime on spatial co-location (in s)	70.16	77.49
Avg. Runtime on a single GPU (in s)	61.96	59.05
Peak. Power on spatial co-location (in W)	191.92	159.0
Peak. Power on a single GPU (in W)	193.59	129.7
Avg. Power on spatial co-location (in W)	97.01	97.19
Avg. Power on a single GPU (in W)	98.77	69.68

Table 2: Evaluation metrics for spatial co-location runs (each workload dispatched over 2 GPUs on a single node) compared against workload run over a single GPU

workload increases when it is spatially co-located. We also observe that the average and peak power in case of spatially co-located runs is higher than the average and peak power in case of baseline runs – we suspect that this could be due to the blocking memory accesses resulting in repeated evictions and higher power consumption levels.

Next, we spatially co-locate ResNet-50 (a compute-intensive workload) across two GPUs of the same node and compare it against its baseline run on a single GPU. We observe that the average runtime of ResNet-50 increases only by 15% over its baseline run when spatially co-located on the same node – this difference is not as significant as in the case of BERT. (Note that, while our methodology classifies ResNet-50 as compute-intensive – it is pretty close to the x = y line, suggesting that it also has a considerable memory-intensive characteristic which leads to this 15% increase in runtime. For a more compute-intensive workload such as SGEMM, we expect this difference to be much lower.) Since ResNet-50 is not as memory-intensive as BERT, spatial co-location has a weaker impact on the latency of the workload when it is spatially co-located. Unlike BERT, in this case, we observe that the average and peak power in case of spatially co-located runs is lower than the average and peak power in case of baseline runs.

Complete set of simulation results, nvidia-smi output files and source code for our analysis scripts can be found at: `https://github.com/saitama0300/CS744-Project-Results.git`.

5 Conclusions

We observed that the temporal co-location of compute-intensive workloads affects the thermal efficiency of the system to a greater extent than temporal co-location of memory-intensive workloads. To effectively co-locate temporally, we can:

- Interleave compute-intensive and memory-intensive workloads for better thermal efficiency.
- Introduce cooldown periods after compute-intensive runs to help the system recover faster and improve the overall performance.

The spatial co-location of memory-intensive workloads across GPUs affects the individual latencies due to high resource contention. To effectively co-locate spatially, we can:

- Avoid placing multiple memory-intensive workloads on the same node, place across different nodes if possible.

References

[1] TACC texas advanced computing center,. https://www.tacc.utexas.edu/, 2021.

[2] R. Adolf, S. Rama, B. Reagen, G. Wei, and D. Brooks. Fathom: reference workloads for modern deep learning methods. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–10, Los Alamitos, CA, USA, sep 2016. IEEE Computer Society.

[3] Q. Chen, H. Yang, M. Guo, R. S. Kannan, J. Mars, and L. Tang. Prophet: Precise qos prediction on non-preemptive accelerators to improve utilization in warehouse-scale computers. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, page 17–32, New York, NY, USA, 2017. Association for Computing Machinery.

[4] J. Coplin and M. Burtscher. Energy, power, and performance characterization of gpgpu benchmark programs. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1190–1199, 2016.

[5] S. Dara, S. Dhamercherla, S. S. Jadav, C. M. Babu, and M. J. Ahsan. Machine learning in drug discovery: A review. *Artif. Intell. Rev.*, 55(3):1947–1999, mar 2022.

[6] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *2013 42nd International Conference on Parallel Processing*, pages 826–833, 2013.

[7] M. Jeon, S. Venkataraman, A. Phanishayee, u. Qian, W. Xiao, and F. Yang. Analysis of large-scale multi-tenant gpu clusters for dnn training workloads. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '19, page 947–960, USA, 2019. USENIX Association.

[8] Y. Jiao, H. Lin, P. Balaji, and W. Feng. Power and performance characterization of computational kernels on the gpu. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*, pages 221–228, 2010.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[10] H. Menon, B. Acun, S. G. De Gonzalo, O. Sarood, and L. Kalé. Thermal aware automated load balancing for hpc applications. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–8, 2013.

[11] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab. Machine learning for anomaly detection: A systematic review. *IEEE Access*, 9:78658–78700, 2021.

[12] Prabhat, K. Kashinath, M. Mudigonda, S. Kim, L. Kapp-Schwoerer, A. Graubner, E. Karaismailoglu, L. von Kleist, T. Kurth, A. Greiner, A. Mahesh, K. Yang, C. Lewis, J. Chen, A. Lou, S. Chandran, B. Toms, W. Chapman, K. Dagon, C. A. Shields, T. O'Brien, M. Wehner, and W. Collins. ClimateNet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather. *Geoscientific Model Development*, 14(1):107–124, Jan. 2021.

[13] P. Sinha, A. Guliani, R. Jain, B. Tran, M. D. Sinclair, and S. Venkataraman. Not all gpus are created equal: Characterizing variability in large-scale, accelerator-rich systems. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 01–15, 2022.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator, 2015.

[16] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou. Gandiva: Introspective cluster scheduling for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 595–610, Carlsbad, CA, Oct. 2018. USENIX Association.

[17] K. Zhang, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman. Minimizing thermal variation across system components. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 1139–1148, 2015.