Understanding GPU Utilization Using LDMS Data on Perlmutter

Onur Cankur
ocankur@umd.edu
Department of Computer Science,
University of Maryland, College Park
College Park, Maryland, USA

Brian Austin
baustin@lbl.gov
Lawrence Berkeley National
Laboratory
Berkeley, California, USA

Abhinav Bhatele
bhatele@cs.umd.edu
Department of Computer Science,
University of Maryland, College Park
College Park, Maryland, USA

Abstract

GPGPU-based clusters and supercomputers have grown significantly in popularity over the past decade. While numerous GPGPU hardware counters are available to users, their potential for workload characterization remains underexplored. In this work, we analyze previously overlooked GPU hardware counters collected via the Lightweight Distributed Metric Service on Perlmutter. We examine spatial imbalance, defined as uneven GPU usage within the same job, and perform a temporal analysis of how counter values change during execution. Using temporal imbalance, we capture deviations from average usage over time. Our findings reveal inefficiencies and imbalances that can guide workload optimization and inform future HPC system design.

Keywords

resource utilization, workload characterization, monitoring

1 Summary

General-purpose Graphics Processing Units (GPGPUs) have become pervasive in compute nodes on high performance computing (HPC) systems. Gaining insights into their utilization is necessary for identifying inefficiencies and guiding optimization and future system design. This is made possible by gathering and analyzing system-wide monitoring data collected over extended periods across all compute nodes. Such data provides information about GPU usage along with job metadata (e.g., job duration, number of nodes/GPUs).

Tools such as the Lightweight Distributed Metric Service (LDMS) [2] enable longitudinal monitoring on large systems. However, the sheer volume of data makes extracting insights a formidable task. Previous works [3–5] analyze GPU monitoring data but are limited to a few counters. In this study, we conduct a comprehensive analysis of GPU-specific counters collected via LDMS on Perlmutter.

We focus on spatial and temporal imbalance in GPU workloads. Work distribution across GPUs can lead to spatial imbalance, where some GPUs are underutilized while others are heavily loaded. Our spatial analysis quantifies this imbalance to assess allocation efficiency. Counter values also fluctuate over time and we capture deviations from mean behavior using the temporal imbalance metric and reveal how consistently GPUs are utilized during a job.

Our analysis reveals several trends in GPU usage on Perlmutter. Many single-node jobs allocate all four GPUs but effectively use only one, due to non-shareable GPU nodes. FP64 cores are most frequently used, while FP16 remains rare. Tensor cores are often used alongside FP32 or FP64.

1.1 Methodology

We define three metrics: spatial imbalance, temporal imbalance, and overall utilization. These are computed from LDMS/DCGM samples aligned with Slurm job metadata.

1.1.1 Analyzing the Spatial Imbalance of Jobs. This metric captures uneven GPU usage within a job. For job j in time window w, let $TC(g,w) = \sum_{t=1}^{t_w} C_{g,t}$ be the sum of counter values for GPU g. Spatial imbalance is:

$$SI(j,w) = 1 - \frac{\sum_{g=1}^{g_j} TC(g,w)}{\max\limits_{1 \le g \le g_j} TC(g,w) \times g_j} \tag{1}$$

Values near 0 indicate balanced usage, and values near 1 indicate imbalance. Overall imbalance is the mean across windows. We use 1-minute windows to capture bursts.

1.1.2 Analyzing the Temporal Imbalance of Jobs. Temporal imbalance quantifies the variation in hardware counter values over a job's runtime. We adopt the definition from [7]:

$$TI(j,g) = 1 - \frac{\sum_{t=1}^{t_j} C_{g,t}}{t_j \times \max_{1 \le t \le t_j} C_{g,t}} \tag{2}$$

where $C_{g,t}$ is the hardware counter value for GPU g at time t. It compares observed values with the maximum possible. Low values indicate stable behavior; high values reflect fluctuations. A job's temporal imbalance is defined as the maximum across its GPUs.

1.1.3 Analyzing Overall Utilization of Jobs. To analyze the overall utilization of jobs, we calculate the job level mean, $M(j)$, which is computed by first averaging the counter values over time, t_j, for each GPU in a job, and then taking the mean across all GPUs, g_j, assigned to that job:

$$M(j) = \frac{1}{g_j} \sum_{g=1}^{g_j} \left(\frac{1}{t_j} \sum_{t=1}^{t_j} C_{g,t} \right) \tag{3}$$

where $C_{g,t}$ is the hardware counter value for GPU g at time t.

1.2 Monitoring Data Used in this Study

1.2.1 Data Sources. Perlmutter uses LDMS [2] to collect per-GPU counters via the DCGM plugin [6] (10-second sampling) and job metadata from Slurm [1]. Our dataset spans August–December 2023. FP16_ACTV, FP32_ACTV, FP64_ACTV, and TNSR_ACTV record the fraction of cycles when the respective GPU cores were active, while GPU_UTIL measures the fraction of time at least one kernel was executing. Since LDMS lacks job IDs, we aligned samples with Slurm by matching node IDs and job start–end times.

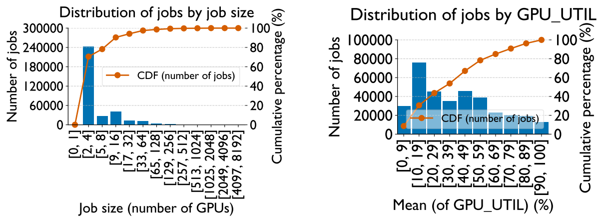

Figure 1: The plot shows the number of jobs by the number of GPUs used and by the mean of GPU_UTIL, with corresponding CDFs. Most jobs run on a single node (four GPUs), and 43% of jobs fall in the low utilization range (0–30%).

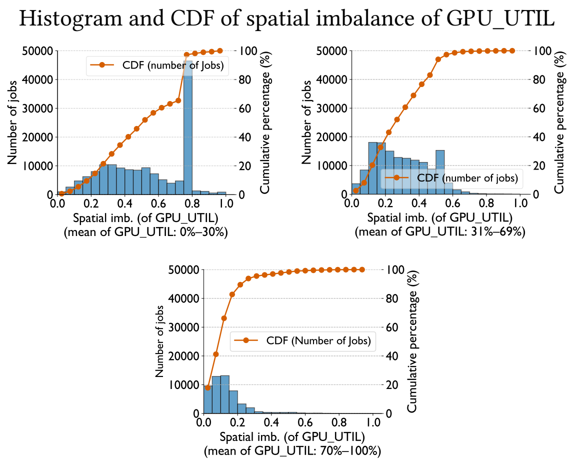

Figure 2: The plots show the distribution of spatial imbalance of GPU_UTIL for jobs grouped by mean of GPU_UTIL ranges (0-30%, 31-69%, 70-100%, left to right). Low-utilization jobs exhibit the highest spatial imbalance. 97.6% of high-utilization have below 0.5 imbalance.

1.2.2 Data Cleaning and Preprocessing. We excluded jobs on login nodes, non-GPU partitions, staff accounts, jobs shorter than three minutes, and incomplete jobs (0.28%). Counters with physically invalid values (e.g., GPU_UTIL >100%) and jobs with mean counter values below 1% were also removed.

1.3 Results

The Cumulative Distribution Function (CDF), shown by the orange line in both plots, represents the probability that a random variable X takes a value less than or equal to x.

The left plot in Figure 1 shows job distribution by GPU count. Red dots show mean values. Most jobs run on a single node (first two bars), and job counts decrease as GPU use exceeds one node (16 GPUs). The CDF shows 70.5% of jobs use a single node, most allocating all four GPUs.

The right plot in Figure 1 shows the distribution of jobs by mean GPU_UTIL. We observe that 53.8% of jobs fall in the 0–30% mean GPU_UTIL range.

Figure 2 shows spatial imbalance across utilization ranges. Low-utilization jobs show the highest imbalance (up to 0.75), with 45.7% below 0.5. Medium-utilization jobs are more balanced, with 83% below 0.5. High-utilization jobs exhibit the least imbalance, with 97.6% below 0.5, indicating that higher GPU_UTIL correlates with more uniform GPU use. We also observe that most four-GPU jobs (one node) actively use only one.

1.4 Conclusion

Many jobs allocate full nodes but use only one GPU. Over 40% of jobs run below 30% GPU_UTIL. Low-utilization jobs are sporadic or idle, whereas high-utilization jobs are consistent. FP64 dominates usage, FP16 is rare, and tensor cores are usually paired with FP32 or FP64. These trends point to opportunities for better workload design, balanced GPU use, and improved system efficiency.

References

[1] 2020. Slurm Workload Manager. https://slurm.schedmd.com/documentation.html

[2] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker. 2014. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 154–165.

[3] Xiaoyu Chu, Daniel Hofstätter, Shashikant Ilager, Sacheendra Talluri, Duncan Kampert, Damian Podareanu, Dmitry Duplyakin, Ivona Brandic, and Alexandru Iosup. 2024. Generic and ML Workloads in an HPC Datacenter: Node Energy, Job Failures, and Node-Job Analysis. *arXiv preprint arXiv:2409.08949* (2024).

[4] Baolin Li, Rohin Arora, Siddharth Samsi, Tirthak Patel, William Arcand, David Bestor, Chansup Byun, Rohan Basu Roy, Bill Bergeron, John Holodnak, et al. 2022. AI-enabling workloads on large-scale GPU-accelerated system: Characterization, opportunities, and implications. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 1224–1237.

[5] Jie Li, George Michelogiannakis, Brandon Cook, Dulanya Cooray, and Yong Chen. 2023. Analyzing resource utilization in an HPC system: A case study of NERSC's Perlmutter. In *International Conference on High Performance Computing*. Springer, 297–316.

[6] NVIDIA Corporation. 2023. NVIDIA Data Center GPU Manager (DCGM). https://developer.nvidia.com/dcgm. Accessed: 2024-10-15.

[7] Ivy Peng, Ian Karlin, Maya Gokhale, Kathleen Shoga, Matthew Legendre, and Todd Gamblin. 2021. A holistic view of memory utilization on HPC systems: Current and future trends. In *Proceedings of the International Symposium on Memory Systems*. 1–11.