Underutilization in Research GPU Clusters: SE Challenges

Krzysztof Kaczmarski
Warsaw University of Technology
Warsaw, Poland
krzysztof.kaczmarski@pw.edu.pl

Jakub Narębski
Nicolaus Copernicus University
Toruń, Poland
jakub.narebski@mat.umk.pl

Piotr Przymus
Nicolaus Copernicus University
Toruń, Poland
piotr.przymus@mat.umk.pl

Abstract

GPU clusters underpin modern deep learning, yet studies across industry and academia consistently report widespread GPU underutilization. Prior work and our own analysis indicate that inefficiency often stems from recurring patterns in code, job scripts, and runtime behaviour that users rarely detect. We argue that addressing this issue is a MSR challenge: it requires mining inefficiency patterns, combining static and dynamic signals for actionable feedback, validating job-submission artefacts, and developing privacy-aware datasets linking code, configuration, and runtime metrics.

ACM Reference Format:
Krzysztof Kaczmarski, Jakub Narębski, and Piotr Przymus. 2026. Underutilization in Research GPU Clusters: SE Challenges. In *23rd International Conference on Mining Software Repositories (MSR '26), April 13–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3793302.3793559

1 Motivation and Findings

Utilization of GPU resources in a cluster environment is returning topic of research. Hardware access patterns must evolve with more GPU devices, more users per node [26] and bigger jobs [25]. Understanding of users' behavior is crucial to improve resource utilization and plan future investments. Administrators monitor computational jobs with profilers [5]: for hardware [2], provided by vendors like *NSight* [16], framework-based like *PyTorch* profiler [8, 21], or for job schedulers like *SLURM* [9, 26]. These solutions usually focus only on one of the aspects of the problem, not considering the environment as a multilevel system containing users, processes defined in particular programming languages, job schedulers, nodes with different hardware configurations and GPU devices. Only a few tools provide a holistic view of the system performance [7, 19, 28] but users still do not know how to improve their code performance by locating processing bottlenecks. User surveys showed that even if users measure processing time [26], which is on a bill, they do not consider GPU utilization can be improved.

Inspired by industrial [6] and academic [14, 18, 26] studies showing that up to half of deep-learning jobs in production have $< 50\%$ utilization [13], we explored whether similar inefficiencies appear in small research-centric, user-driven environments. We analyzed GPU utilization logs from the WUT Eden cluster [11]. We identified several users whose long-running jobs repeatedly showed sustained GPU utilization below 40%. From this group, two users agreed to take part in exploratory interviews. Insights from these interviews informed a 41-item questionnaire sent to 50 active cluster users, yielding 33 valid responses [12] (66% response rate).

Ad-hoc and individualized workloads. Only 24% users *often* run replication packages, while 46% *rarely or never* do. This highlights the need for adaptive SE tools—such as environment introspection, configuration linting, and lightweight profilers that can analyze arbitrary user scripts without prior setup. *Semgrep* static analysis [17, 24] can be only a part of this process [6].

Awareness gap. Although queues are often congested, only 29% of users *often* monitor GPU utilization, and 47% do so *rarely or never*. Many assume "the scheduler ensures efficiency", or lack convenient metrics during long runs. Thus, inefficiency persists unnoticed — not from unwillingness, but from missing feedback.

Conditional readiness. Users express motivation to improve: 88% would adjust configurations if inefficiency was detected, 76% rated dashboards as *useful*, and 82% valued training materials. The issue is therefore not resistance but the absence of mechanisms.

2 Implications for MSR Research

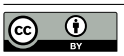
Low GPU utilization often stems from recurring, detectable patterns in code, configuration, and runtime behaviour [6, 26]. For the SE and MSR communities, the challenge is to build preventive, evidence-driven tools despite limited visibility into real cluster workloads.

(1) *Mining inefficiency patterns from code, scripts, and logs.* Gao et al. [6] show that $\approx$ 85% of inefficiencies correspond to identifiable static patterns (e.g., data-loader bottlenecks, remote reads, oversubscription). Pattern mining, commit-history analysis, and correlating edits with runtime traces—can extract such patterns from existing DL repositories and cluster logs. These mined motifs form the empirical backbone for static checks, underutilization predictors, and libraries of common fixes.

(2) *Combining static and dynamic signals for actionable feedback.* Users want explanations and concrete fixes [4, 20, 23]. Combining mined patterns with lightweight profiling enables concise guidance—e.g., "GPU idle → host-device transfer bottleneck; use async loading"—delivering explainable, actionable feedback.

(3) *Validating job scripts and configuration artefacts.* Misconfigured SLURM scripts remain a major cause of wasted GPU time [10, 22, 27]. Mining historical submissions [22] can reveal recurring mistakes and their impacts.

(4) *Need for datasets.* Several HPC datasets now include GPU metrics-and some even expose SLURM scripts or limited links to source code—but they still emphasize system-level resource traces over full code–job–runtime integration [1, 3, 8, 15]. These datasets offer a useful starting point, yet remain incomplete due to privacy and institutional constraints. Further progress requires privacy-preserving logging, synthetic or anonymized corpora, and closer collaboration with HPC centres to produce richer, code-linked workload data.



MSR '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2474-9/2026/04
https://doi.org/10.1145/3793302.3793559

References

[1] 2025. Alibaba/Clusterdata. Alibaba, https://github.com/alibaba/clusterdata.

[2] Rajeev Alur, Joseph Devietti, Omar S. Navarro Leija, and Nimit Singhania. 2017. GPUDrano: Detecting Uncoalesced Accesses in GPU Programs. In *Computer Aided Verification*, Rupak Majumdar and Viktor Kunčak (Eds.). Vol. 10426. Springer International Publishing, Cham, 507–525. doi:10.1007/978-3-319-63387-9_25

[3] Francesco Antici, Andrea Bartolini, Jens Domke, Zeynep Kiziltan, and Keiji Yamamoto. 2025. F-DATA: A Fugaku Workload Dataset for Job-centric Predictive Modelling in HPC Systems. *Scientific Data* 12, 1 (July 2025), 1321. doi:10.1038/s41597-025-05633-1

[4] Charles M. Curtsinger. 2016. *Effective Performance Analysis and Debugging*. Ph. D. Dissertation. College of Information and Computer Sciences, University of Massachusetts Amherst. doi:10.7275/8415585.0 https://scholarworks.umass.edu/dissertations_2/632.

[5] Paul Elvinger, Foteini Strati, Natalie Enright Jerger, and Ana Klimovic. 2025. Measuring GPU Utilization One Level Deeper. doi:10.48550/arXiv.2501.16909 arXiv:2501.16909 [cs]

[6] Yanjie Gao, Yichen He, Xinze Li, Bo Zhao, Haoxiang Lin, Yoyo Liang, Jing Zhong, Hongyu Zhang, Jingzhou Wang, Yonghua Zeng, Keli Gui, Jie Tong, and Mao Yang. 2024. An Empirical Study on Low GPU Utilization of Deep Learning Jobs. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 96, 13 pages. doi:10.1145/3597503.3639232

[7] James Gleeson, Moshe Gabel, Gennady Pekhimenko, Eyal de Lara, Srivatsan Krishnan, and Vijay Janapa Reddi. 2021. RL-scope: Cross-stack Profiling for Deep Reinforcement Learning Workloads. In *Proceedings of the Fourth Conference on Machine Learning and Systems, MLSys 2021, virtual, April 5-9, 2021*, Alex Smola, Alex Dimakis, and Ion Stoica (Eds.), Vol. 3. 783–799. arXiv:2102.04285

[8] Shuo Hong, Hailong Sun, Xiang Gao, and Shin Hwei Tan. 2024. Investigating and Detecting Silent Bugs in PyTorch Programs. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, Rovaniemi, Finland, 272–283. doi:10.1109/SANER60148.2024.00035

[9] Thomas Jakobsche, Nicolas Lachiche, and Florina M. Ciorba. 2023. Investigating HPC Job Resource Requests and Job Efficiency Reporting. In *2023 22nd International Symposium on Parallel and Distributed Computing (ISPDC)*. 61–68. doi:10.1109/ISPDC59212.2023.00024

[10] JobCritic Slurm Scripts. 2025. JobCritic: SLURM Job Efficiency Analysis Scripts. https://github.com/SJTU-HPC/slurm-scripts.

[11] Krzysztof Kaczmarski. 2025. Eden Cluster in 2025. https://hpcgpu.mini.pw.edu.pl/eden-in-2025/.

[12] Krzysztof Kaczmarski, Piotr Przymus, and Jakub Narębski. 2025. Utilization of WUT Eden Cluster – User's Questionnaire (Oct 2025 to Nov 2025). doi:10.6084/m9.figshare.30688940

[13] Cheng Li, Abdul Dakkak, Jinjun Xiong, Wei Wei, Lingjie Xu, and Wen-mei Hwu. 2020. XSP: Across-Stack Profiling and Analysis of Machine Learning Models on GPUs. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 326–327. doi:10.1109/IPDPS47924.2020.00042

[14] 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 *High Performance Computing*, Abhinav Bhatele, Jeff Hammond, Marc Baboulin, and Carola Kruse (Eds.). Springer Nature Switzerland, Cham, 297–316. doi:10.1007/978-3-031-32041-5_16

[15] Joshua McKerracher, Preeti Mukherjee, Rajesh Kalyanam, and Saurabh Bagchi. 2025. Fresco: A Public Multi-Institutional Dataset for Understanding HPC System Behavior and Dependability. In *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration (PEARC '25)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/3708035.3736090

[16] NVIDIA Corp. 2025. NVIDIA Nsight Systems: System-wide Performance Analysis Tool. https://developer.nvidia.com/nsight-systems. Accessed 2025-11-07.

[17] Opengrep. 2025. Opengrep - The open-source code security engine. https://www.opengrep.dev/. Accessed 2025-11-20.

[18] Tirthak Patel, Zhengchun Liu, Raj Kettimuthu, Paul Rich, William Allcock, and Devesh Tiwari. 2020. Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification, and Implications. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–17. doi:10.1109/SC41405.2020.00088

[19] Weiyu Peng, Jinghao Wang, Tianyu Wo, and Renyu Yang. 2024. PrecisionProbe: Non-intrusive Performance Analysis Tool for Deep Learning Recommendation Models. In *2024 IEEE International Conference on Joint Cloud Computing (JCC)*. 17–20. doi:10.1109/JCC62314.2024.00010

[20] Peter Pirkelbauer and Chunhua Liao. 2025. CompilerGPT: Leveraging Large Language Models for Analyzing and Acting on Compiler Optimization Reports. *CoRR* abs/2506.06227 (2025). doi:10.48550/ARXIV.2506.06227 arXiv:2506.06227

[21] PyTorch Documentation. 2025. PyTorch Profiler — PyTorch Tutorials 2.9.0+cu128 Documentation. https://docs.pytorch.org/tutorials/recipes/recipes/profiler_recipe.html.

[22] Siddharth Samsi, Matthew L Weiss, David Bestor, Baolin Li, Michael Jones, Albert Reuther, Daniel Edelman, William Arcand, Chansup Byun, John Holodnack, Matthew Hubbell, Jeremy Kepner, Anna Klein, Joseph McDonald, Adam Michaleas, Peter Michaleas, Lauren Milechin, Julia Mullen, Charles Yee, Benjamin Price, Andrew Prout, Antonio Rosa, Allan Vanterpool, Lindsey McEvoy, Anson Cheng, Devesh Tiwari, and Vijay Gadepally. 2021. The MIT Supercloud Dataset. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–8. doi:10.1109/HPEC49654.2021.9622850

[23] Marija Selakovic, Thomas Glaser, and Michael Pradel. 2017. An actionable performance profiler for optimizing the order of evaluations. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017*, Tevfik Bultan and Koushik Sen (Eds.). ACM, 170–180. doi:10.1145/3092703.3092716

[24] Semgrep. 2025. Semgrep App Security Platform. https://semgrep.dev/. Accessed 2025-11-20.

[25] Feiyi Wang, Sarp Oral, Satyabrata Sen, and Neena Imam. 2019. Learning from Five-year Resource-Utilization Data of Titan System. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. 1–6. doi:10.1109/CLUSTER.2019.8891001

[26] Le Mai Weakley, Scott Michael, Laura Huber, Abhinav Thota, Ben Fulton, and Matthew Kusz. 2025. Monitoring and Characterizing GPU Usage. *Concurrency and Computation: Practice and Experience* 37, 3 (2025), e8341. doi:10.1002/cpe.8341 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.8341.

[27] HPC Wiki. 2025. Automated Benchmarking using a Job Script. https://hpc-wiki.info/hpc/Benchmarking_%26_Scaling_Tutorial/Automated_Benchmarking. Accessed 2025-11-07.

[28] Qidong Zhao, Hao Wu, Yueming Hao, Zilingfeng Ye, Jiajia Li, Xu Liu, and Keren Zhou. 2025. DeepContext: A Context-aware, Cross-platform, and Cross-framework Tool for Performance Profiling and Analysis of Deep Learning Workloads. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (ASPLOS '25)*. Association for Computing Machinery, New York, NY, USA, 48–63. doi:10.1145/3676642.3736127