POSTER: Weaver: Diagnosing Extra Kernel and Synchronization Interference in GPU Workloads

Jingyuan Zhu, Menghao Zhang, Yuxuan Chen
Beihang University

Abstract

Modern GPU workloads use batching, asynchronous execution, kernel overlap, and GPU sharing to improve utilization, but these optimizations make kernel-level slowdown hard to diagnose. A target kernel may be blocked by extra synchronization or helper kernels, or slowed by concurrent kernels after it starts. Existing tools provide GPU visibility, but they either remain too heavy for continuous online use or focus on coarse-grained symptoms, leaving fine-grained kernel-level root causes to manual analysis. This poster presents Weaver, a low-overhead cross-layer diagnosis framework for GPU kernels. Weaver builds a semantic execution graph from operator, kernel timeline, and warp/block evidence, distinguishes blocked and slowed kernels, and reports an interpretable root-cause chain. Our prototype evaluation shows that Weaver continuously collects runtime evidence with low overhead and accurately localizes anomalies.

CCS Concepts

• **Networks** → **Network performance analysis**; *Network monitoring*.

Keywords

GPU Kernel Diagnosis; Cross-layer Profiling; Causal Root-cause Localization

ACM Reference Format:
Jingyuan Zhu, Menghao Zhang, Yuxuan Chen. 2026. POSTER: Weaver: Diagnosing Extra Kernel and Synchronization Interference in GPU Workloads. In *ACM SIGCOMM 2026 Posters and Demos (SIGCOMM Posters and Demos '26), August 17–21, 2026, Denver, CO, USA*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3789240.3830303

1 Introduction

As large language models (LLMs) continue to scale, GPU compute has become increasingly valuable in both training and inference [4]. To reduce GPU idle time and improve overall throughput, modern LLM training and inference systems commonly adopt utilization-oriented techniques such as batching, asynchronous execution, kernel overlap, and multi-process GPU sharing [9, 12, 15]. Although critical, they also make GPU execution more complex and fine-grained performance diagnosis more difficult.

This diagnostic difficulty mainly comes from kernel heterogeneity and complex dependencies. On the one hand, real executions contain diverse events, including General Matrix Multiplication (GEMM), NCCL communication or synchronization [7]. Different kernel types cannot be compared using a single metric: GEMM progress is better measured by FLOPs while NCCL and memory-copy operations are better characterized by bytes/s [11]. On the other hand, kernels may be constrained by data dependencies, stream ordering, synchronization operations, or framework scheduling decisions [1, 5]. In such highly concurrent execution, expected dependency relations among kernels may be disrupted by extra events, and a target kernel may also make slower progress due to concurrently running kernels. The key to indicate an anomaly is: (1) whether an extra event breaks the expected dependency relation and blocks the target kernel before execution; (2) whether it slows the target kernel during concurrent execution.

Existing tools provide useful GPU visibility, but they fail to simultaneously satisfy three requirements: online diagnosis, fine-grained kernel-level analysis, and root-cause explanation. Timeline profilers such as Nsight Systems [6] and PyTorch Profiler [8] mainly expose kernel execution time, stream ordering, and overlap relations, but they do not directly explain whether a timeline event is an anomaly or a root cause. Moreover, their overhead can make continuous online diagnosis difficult [2].

Framework- or cluster-level diagnosis systems usually focus on coarse-grained symptoms such as fail-slow workers, irregular iterations, hangs, or communication bottlenecks, and may miss concrete kernel-level root causes [1–3, 13, 14]. More importantly, existing methods lack causal analysis for extra synchronization, helper kernels, and harmful overlap. Users often still need additional tools and manual inspection.

To address these problems, we propose Weaver, a low-overhead cross-layer causal diagnosis framework for GPU kernels. Our key observation is that kernel slowdown can be explained by comparing the observed execution with a lightweight expected execution structure: an anomaly either introduces an unexpected predecessor that blocks a target before execution, or creates harmful overlap that slows its progress during execution. Based on this observation, Weaver does not merely display timelines; instead, it organizes cross-layer evidence into an interpretable execution graph and determines whether slowdown comes from blocking before execution or slowdown during execution. First, Weaver uses a lightweight collector to gather evidence from three layers. Second, Weaver builds a resource-aware execution graph that groups kernels by type and work size for fair progress comparison. Finally, Weaver performs causal localization: for blocked targets, it traces whether

This work is supported in part by the National Natural Science Foundation of China (No. 62402025), the Fundamental Research Funds for the Central Universities and Huawei-BUAA Joint Lab. Menghao Zhang is the corresponding author.

SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3830303

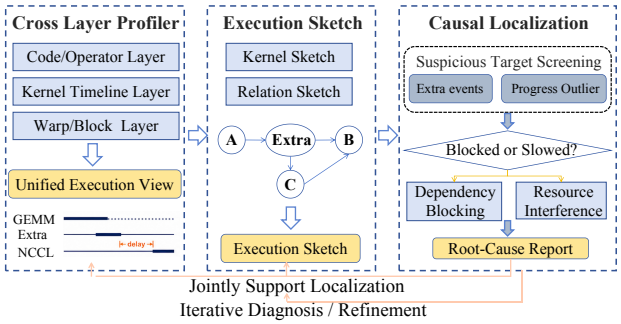

Figure 1: The Architecture of Weaver.

an extra kernel, synchronization, or event wait becomes an unexpected predecessor; for a target slowed during execution, it analyzes whether a concurrent extra kernel forms harmful overlap. We implement an initial open-source Weaver prototype [10] and validate its effectiveness, showing that Weaver can accurately localize the root causes of dependency and resource anomalies, and output the corresponding diagnosis report containing the target, anomaly type, root cause, and supporting evidence chain.

2 Weaver Design

Figure 1 shows Weaver's overall workflow with three modules: cross-layer profiling, execution sketch modeling, and causal root-cause localization. Weaver first collects cross-layer information, then builds an execution graph with dependency, overlap, and resource semantics, and finally performs causal root-cause localization to generate an evidence-backed kernel-level diagnosis report.

Low-overhead Cross-layer Evidence Collection. A single-layer timeline is insufficient to explain kernel slowdown, so Weaver adopts a low-intrusion cross-layer collection mechanism. Without modifying training code, Weaver collects lightweight evidence from three layers. At the operator layer, CPython profiling hooks capture key framework APIs, call stacks, and synchronization calls to locate the code source of GPU work. At the CUDA kernel layer, `LD_PRELOAD` hooks and asynchronous timing record kernel launches, streams, CPU enqueue time, GPU execution intervals, and NCCL events. At the warp/block layer, launch configurations and binary-level information provide block/warp scale to help explain internal kernel behavior. To reduce overhead, Weaver sends events asynchronously to a background daemon for collection and logging. This builds a three-layer view with low perturbation, providing evidence for dependency analysis and root-cause localization.

Semantic Execution Graph Construction. To localize abnormal dependencies and harmful overlap, Weaver builds an ExecutionSketch graph from lightweight annotations of key expected relations and stable patterns extracted from representative traces. The graph maintains execution relations among kernels, including kernel ordering, same-stream order, event waits, synchronization, and expected parallelism. By comparing the trace with the ExecutionSketch, Weaver identifies whether an extra kernel, synchronization event, or event wait becomes an unexpected predecessor, and whether expected overlap is broken.

The graph also attaches resource semantics to kernel nodes. Weaver classifies kernels into families such as matrix-compute, communication, memory or layout, reduction, and unknown kernels, and groups them into workload-size buckets. During diagnosis, Weaver compares progress only within the same family and bucket, supporting both dependency analysis and fair progress comparison among similar kernels.

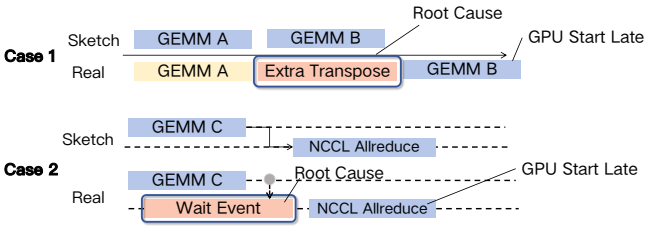

Figure 2: Examples of anomalies and diagnosis results.

Causal Root-Cause Localization. After identifying suspicious anomalies, Weaver diagnoses root causes by comparing the runtime trace with the ExecutionSketch. It first determines whether a target is blocked before execution or slowed during execution, then follows different localization paths.

Weaver screens for unexplained kernels, inserted memcpy, cast, layout, or sync events, lost expected overlap, and abnormal progress within the same family and bucket. It then classifies the target using CPU enqueue, GPU start, and GPU end: late CPU enqueue indicates CPU/runtime blocking; normal enqueue but late GPU start indicates dependency blocking; normal GPU start but late GPU end indicates execution-time slowdown. For blocked targets, Weaver compares expected and actual predecessors, and checks whether the actual start is later than the theoretical earliest start. For slowed targets, Weaver searches for overlapping kernels; if comparable kernels run faster without such overlap, Weaver prioritizes kernels outside the sketch as root-cause candidates. Warp/block evidence further distinguishes external interference from internal issues, such as a few slow warps/blocks delaying the whole kernel. The final report includes the target, anomaly type, root cause, and evidence chain.

3 Evaluation and Future Work

We implement an open-source prototype of Weaver [10], and evaluate it on collection overhead and diagnosis correctness. For overhead, we use an end-to-end training-style workload covering compute, memory/layout, backward, optimizer, and NCCL communication. The results show that Weaver continuously collects runtime evidence with low overhead, making it suitable for online diagnosis. For correctness, we build test cases with dependency and resource anomalies, i.e., unexpected synchronization or extra kernels before a target, and concurrent GEMM or memcpy kernels that induce slowdown. Figure 2 shows two anomaly examples and the corresponding root causes identified by Weaver. These experiments demonstrate that Weaver can distinguish blocked and slowed targets, and report the corresponding evidence chain.

In our current prototype, ExecutionSketch requires lightweight manual annotations, such as expected relations for key kernels. In future work, we will further automate its construction by mining stable kernel groups, ordering constraints, and overlap patterns from stable clean traces We also plan to further explore warp/block-level information for more fine-grained diagnosis.

References

[1] Wei Cui, Ji Zhang, Han Zhao, Chao Liu, Jian Sha, Bingsheng He, Minyi Guo, and Quan Chen. 2025. Flare: Anomaly Diagnostics for Divergent LLM Training in GPU Clusters of Thousand-Plus Scale.

[2] Sébastien Darche and Michel Dagenais. 2024. Low-Overhead Trace Collection and Profiling on GPU Compute Kernels. *ACM Trans. Parallel Comput.*

[3] Yangtao Deng, Lei Zhang, Qinlong Wang, Xiaoyun Zhi, Xinlei Zhang, Zhuo Jiang, Haohan Xu, Lei Wang, Zuquan Song, Gaohong Liu, Yangyang Bai, Shuguang Wang, Wencong Xiao, Jia jun Ye, Minlan Yu, and Hong Xu. 2025. Mycroft: Tracing Dependencies in Collective Communication Towards Reliable LLM Training. In *ACM SOSP 2025*.

[4] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jia jun Ye, Xin Jin, and Xin Liu. 2024. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. In *USENIX NSDI 2024*.

[5] Seonho Lee, Jihwan Oh, Junkyum Kim, Seokjin Go, Jongse Park, and Divya Mahajan. 2025. Characterizing Compute-Communication Overlap in GPU-Accelerated Distributed Deep Learning: Performance and Power Implications. In *IEEE ISPASS 2025*.

[6] NVIDIA. 2025. Nsight Systems. https://developer.nvidia.com/nsightsystems.

[7] Kishore Punniyamurthy, Khaled Hamidouche, and Bradford M. Beckmann. 2023. Optimizing Distributed ML Communication with Fused Computation-Collective Operations. In *SC 2024*.

[8] PyTorch. 2025. PyTorch Profiler. https://pytorch.org/docs/stable/profiler.html.

[9] Saeed Rashidi, Matthew Denton, Srinivas Sridharan, Sudarshan M. Srinivasan, Amoghavarsha Suresh, Jade Nie, and Tushar Krishna. 2020. Enabling Compute-Communication Overlap in Distributed Deep Learning Training Platforms. In *ACM/IEEE ISCA 2021*.

[10] WEAVER. 2026. GPU Kernel Diagnosis. https://github.com/Networked-System-and-Security-Group/Weaver.git.

[11] Samuel Williams, Andrew Waterman, and David A. Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*.

[12] Bingyang Wu, Zili Zhang, Zhihao Bai, Xuanzhe Liu, and Xin Jin. 2023. Transparent GPU Sharing in Container Clouds for Deep Learning Workloads. In *USENIX NSDI 2023*.

[13] Tianyuan Wu, Wei Wang, Yinghao Yu, Siran Yang, Wenchao Wu, Qinkai Duan, Guodong Yang, Jiamang Wang, Lin Qu, and Liping Zhang. 2025. GREYHOUND: Hunting Fail-Slows in Hybrid-Parallel Training at Scale. In *USENIX ATC 2025*.

[14] Zhiyi Yao, Pengbo Hu, Congcong Miao, Xuya Jia, Zuning Liang, Yuedong Xu, Chunzhi He, Hao Lu, Mingzhuo Chen, Xiang Li, Zekun He, Yachen Wang, Xianneng Zou, and Junchen Jiang. 2025. Holmes: Localizing Irregularities in LLM Training with Mega-scale GPU Clusters. In *USENIX NSDI 2025*.

[15] Gyeong-In Yu and Joo Seong Jeong. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *USENIX Symposium on Operating Systems Design and Implementation*.