2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 979-8-3315-0294-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISPASS64960.2025.00044

# ASLink: Modeling Multi-GPU Execution in Accel-Sim

Christin Bose[1], Cesar Avalos[1], Junrui Pan[1], Yechen Liu[1], Mahmoud Khairy[1], Clay Hughes[2], Timothy Rogers[1]
[1]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Sandia National Laboratories, Albuquerque, NM, USA

***Abstract*—Graphical processing units (GPUs) are widely used in numerous modern application domains, including modeling and simulation, machine learning, and data analytics. Many applications such as recommendation models and graph neural networks benefit from the use of multiple GPUs to scale up the size of the workload and increase throughput. While current open-sourced GPU architectural simulators can model multi-GPU workloads, doing so remains inefficient and challenging, limiting their broader applicability across various application domains. On the other hand, simulation tools used by the industry are often closed-source, thus hindering efforts to democratize architectural research. This paper proposes an open-source simulator design, ASLink, that extends Accel-sim to support multi-GPU configurations. We highlight the limitations of popular state-of-the-art GPU architecture simulators and propose mechanisms to improve user experience and modeling fidelity in multi-GPU systems. Finally, we validate our proposed infrastructure against kernels representative of real world workloads.**

***Index Terms*—Graphics Processing Units (GPUs), NVBit, Deep learning workloads, Interconnects**

## I. Introduction

Many applications such as recommendation models and graph neural networks often have memory footprints that are much larger than a single GPU can provide. Hence, these applications are often deployed in a multi-GPU system [5]. To enable hardware-software optimizations for modern workloads and explore novel architectural ideas, it becomes imperative to have a simulator that supports multi-GPU modeling without compromising simulation speed.

The architecture community has a long history of simulators that support modeling a single GPU ( [3], [4], [6], [8], [9], [19]–[21]) but support for multi-GPUs is limited. An ideal feature of a simulator would be to simulate kernels based on representative, unaltered GPU workloads that leverage programming frameworks such as Pytorch [17] or Tensorflow [1]. Such functionality would allow researchers to identify GPU bottlenecks in emerging workloads by simulating the entire stack of kernels produced from real-world code, a critical capability for understanding performance constraints and guiding architectural optimizations.

The multi-GPU arrangement introduces new features that must be considered, particularly inter-GPU communication patterns such as Peer-to-Peer (P2P), memory copies, and collective communication libraries like the NVIDIA Collective Communications Library (NCCL) [2]. A recent study [18] found that communication is expected to dominate runtime in distributed training setups, potentially consuming between 40% and 75% of the total execution time. Thus, a true multi-GPU simulator must incorporate these communication primitives to accurately model the behavior of distributed training or inference in machine learning models.

Corresponding author: chris241@purdue.edu

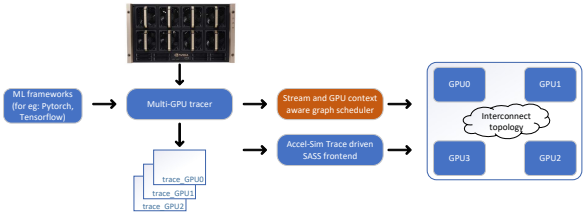


Fig. 1: High level view of the simulation framework .

We propose ASLink, a multi-GPU simulator based on the open-source GPU simulator Accel-Sim [8]. We extend the usage of the NVBit tracer [22] to trace multi-GPU workloads. Through our methodology, we show how any multi-GPU workload can be traced and executed on our proposed simulator. Based on the multi-GPU tracer, we propose the first open-sourced multi-GPU simulation framework that can simulate kernels end-to-end based on popular machine learning frameworks such as Pytorch or Tensorflow.

## II. Multi-GPU simulator framework

Figure 1 shows a high-level view of the simulation framework. The workload of interest (*e.g.*, a Pytorch program) is traced out by a multi-GPU tracer that generates traces for each GPU context. The kernels are then scheduled onto the respective GPUs while maintaining necessary synchronization requirements.

TABLE I: Multi-GPU configuration

| | |
|---|---|
| #GPUs | 4 |
| #SMs | 80 SMs per GPU |
| SM configuration | Volta-like SM, 64 warps, 4 warp scheds, 64KB shared memory, 64KB L1 cache, 1.4Ghz |
| Inter-GPU Interconnect | All-to-all topology, 46 GB/s per link (bi-directional) |
| Memory BW | 720 GB/s per GPU |

2766-0486/25/$31.00 ©2025 IEEE
DOI 10.1109/ISPASS64960.2025.00044

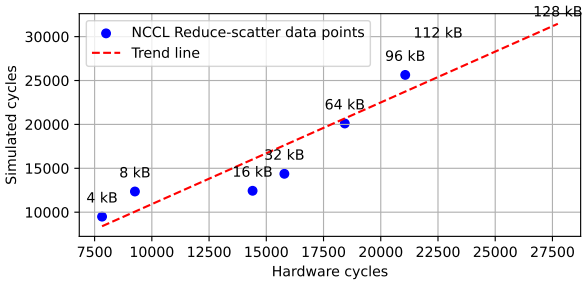


Fig. 2: Correlation on NCCL Reduce Scatter. Correlation: 0.9532. Mean Average Error: 18.06%.

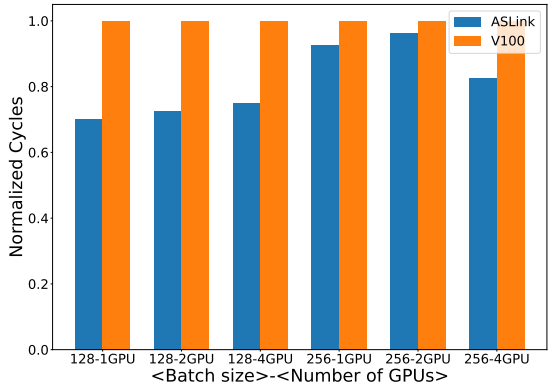


Fig. 3: End-to-end execution time comparison between ASLink and a multi-GPU V100 system. Correlation factor: 0.8582; Mean Average Error: 24.4%

## III. Simulation Validation

In this section, we discuss the results of validating ASLink against a multi-GPU Volta system [15] on several end-to-end workloads. The key parameters of the hardware are tabulated in Table I. For validation of end-to-end workloads, we use NVIDIA tools Nsight Compute [13] to report the runtime cycles.

*1) NCCL Reduce Scatter kernels:* Figure 2 shows the correlation on NCCL Reduce Scatter kernels ( [2], [12]) of various vector sizes. We note a high correlation factor of 0.9532 and a Mean Absolute Percentage Error of 18.06%.

*2) DLRM training iteration:* Figure 3 shows the performance of ASLink normalized to hardware performance for the DLRM training iteration [11] for various batch sizes and number of GPUs. End-to-end simulation in models like DLRM [7] involve hundreds of kernels per GPU along with necessary synchronization events to enable inter-GPU communication. We obtain a correlation factor of 0.8582 and a relative error of 24.4% for end-to-end correlation.

## IV. Case Study

### A. Embedding kernel in DLRMs

The embedding lookup phase in DLRM models entails retrieving multiple rows from embedding tables followed by a reduction operation on the retrieved values. The embedding bag operator [10] is used to process a given table. Figure 4 shows the correlation of a Peer-to-Peer (P2P) enabled Pytorch Embedding bag kernel for various values of batchsize and embedding pooling factors. Using P2P [14], an embedding kernel launched on one GPU can access data located on a neighboring GPU through an interconnect, such as NVlink [16]. The embedding pooling factor denotes the number of embedding table rows that are read for each sample of a batch. The strong correlation between silicon and simulation results validates the inter-GPU latency and bandwidth modeled in the simulator. As the embedding kernel does a series of lookups, it is natural to ask if traditional compiler techniques such as loop unrolling will help to improve the Memory Level Parallelism (MLP) of the kernel. ASLink can help to make informed optimization choices with various unrolling factors and projected bandwidth models for future GPU generations.

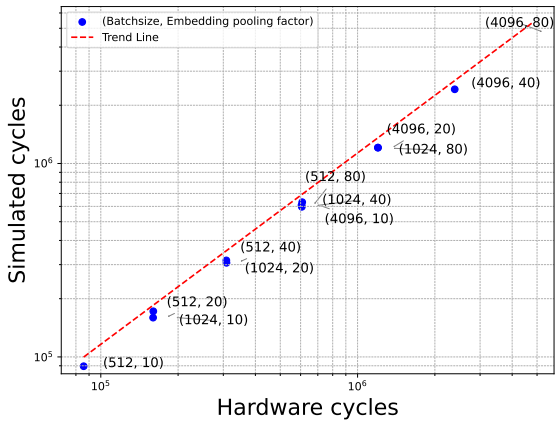


Fig. 4: Correlation on the Pytorch EmbeddingBag kernel for various pooling widths and batch sizes. Correlation coefficient:0.99994; Mean Average Error: 14.1%

## V. Conclusions

Current GPU architectural studies on multi-GPU systems are limited in flexibility outside the industry. This work proposes a design of a multi-GPU simulator based on Accel-sim [8] that adds much-needed flexibility in the interconnect topology, GPU architectural configuration parameters and can simulate any workload of interest based on actual traces collected from hardware. We propose a novel multi-GPU tracer [22] that can trace any workload of interest from silicon. We extend the Accel-sim frontend to incorporate a stream and context-aware kernel scheduler that can schedule kernels on to multiple GPUs while respecting the necessary synchronization primitives. We validate our proposed simulator on a suite of multi-GPU workloads and show strong modeling fidelty with a correlation as high as 99% and mean average error as low as 14%.

## REFERENCES


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[2] N. Corp., "Nccl library," https://docs.nvidia.com/deeplearning/nccl/index.html.

[3] P. Gera, H. Kim, H. Kim, S. Hong, V. George, and C.-K. Luk, "Performance characterisation and simulation of intel's integrated gpu architecture," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018, pp. 139–148.

[4] X. Gong, R. Ubal, and D. Kaeli, "Multi2sim kepler: A detailed architectural gpu simulator," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2017, pp. 269–278.

[5] U. Gupta, C.-J. Wu, X. Wang, M. Naumov, B. Reagen, D. Brooks, B. Cottel, K. Hazelwood, M. Hempstead, B. Jia, H.-H. S. Lee, A. Malevich, D. Mudigere, M. Smelyanskiy, L. Xiong, and X. Zhang, "The architectural implications of facebook's dnn-based personalized recommendation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 488–501.

[6] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeane, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain, and T. Rogers, "Lost in abstraction: Pitfalls of analyzing gpus at the intermediate language level," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 608–619.

[7] Kaggle, "Kaggle," https://www.kaggle.com/c/criteo-display-ad-challenge.

[8] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, "Accel-sim: An extensible simulation framework for validated gpu modeling," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 473–486.

[9] H. Kim, J. Lee, N. B. Lakshminarayana, J. Sim, J. Lim, and T. Pho, "Macsim: A cpu-gpu heterogeneous simulation framework user guide," in *Technical report, Georgia Institue of Technology*, 2015.

[10] Meta, "Pytorch embedding bag kernel," 2024. [Online]. Available: https://github.com/pytorch/pytorch/blob/main/aten/src/ATen/native/cuda/EmbeddingBag.cu

[11] M. Naumov, D. Mudigere, H. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C. Wu, A. G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy, "Deep learning recommendation model for personalization and recommendation systems," *CoRR*, vol. abs/1906.00091, 2019. [Online]. Available: http://arxiv.org/abs/1906.00091

[12] NVIDIA, "Nccl reducescatter," 2024. [Online]. Available: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html

[13] ——, "Nvidia nsight compute," 2024. [Online]. Available: https://docs.nvidia.com/nsight-compute/

[14] ——, "Nvidia unified virtual addressing (uva)," 2024. [Online]. Available: https://docs.nvidia.com/cuda/cuda-driver-api/group__CUDA__UNIFIED.html

[15] ——, "Nvidia volta whitepaper," 2024. [Online]. Available: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[16] ——, "Nvidia's data-center nvlink and nvlink switch," 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/nvlink/

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[18] S. Pati, S. Aga, M. Islam, N. Jayasena, and M. D. Sinclair, "Tale of two cs: Computation vs. communication scaling for future transformers on future hardware," in *2023 IEEE International Symposium on Workload Characterization (IISWC)*, 2023, pp. 140–153.

[19] Y. Sun, T. Baruah, S. A. Mojumder, S. Dong, X. Gong, S. Treadway, Y. Bao, S. Hance, C. McCardwell, V. Zhao, H. Barclay, A. K. Ziabari, Z. Chen, R. Ubal, J. L. Abellán, J. Kim, A. Joshi, and D. Kaeli, "Mgpusim: Enabling multi-gpu performance modeling and optimization," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 197–209. [Online]. Available: https://doi.org/10.1145/3307650.3322230

[20] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2sim: A simulation framework for cpu-gpu computing," in *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2012, pp. 335–344.

[21] O. Villa, D. Lustig, Z. Yan, E. Bolotin, Y. Fu, N. Chatterjee, N. Jiang, and D. Nellans, "Need for speed: Experiences building a trustworthy system-level gpu simulator," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 868–880.

[22] O. Villa, M. Stephenson, D. Nellans, and S. W. Keckler, "Nvbit: A dynamic binary instrumentation framework for nvidia gpus," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 372–383. [Online]. Available: https://doi.org/10.1145/3352460.3358307