Using hardware metrics to understand performance of the RAJA Performance Suite kernels in different GPU modes on MI300A

Amr Akmal Abouelmagd[1], Stephanie Brink[2], Michael McKinsey[2], David Boehme[2], Jason Burmark[2], Brian S. Ryujin[2], Tom Scogland[2], Anthony Skjellum[1], Olga Pearce[2]
[1]Tennessee Technological University [2]Lawrence Livermore National Laboratory

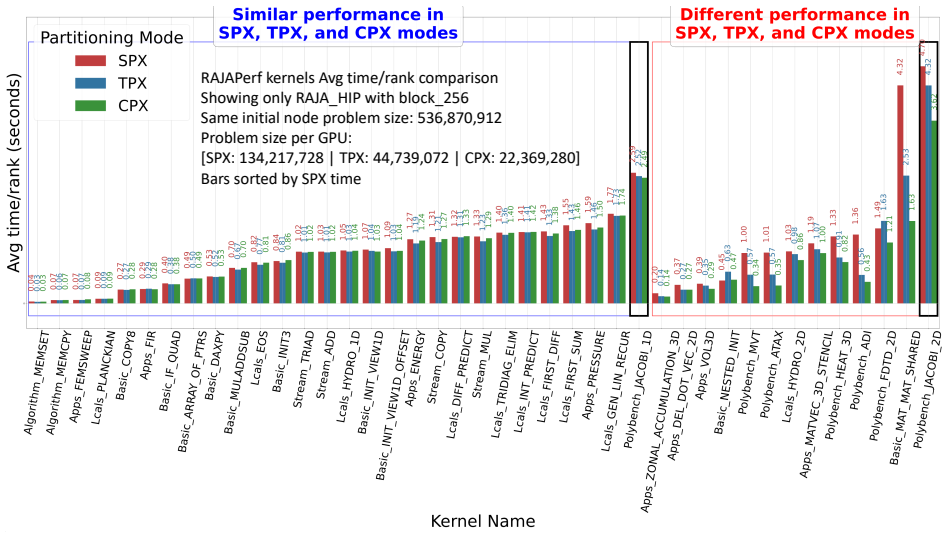

Figure 1: Runtime for RAJAPerf kernels in SPX, TPX, and CPX modes (same problem size per APU). Kernels are sorted by those with similar (*e.g.*, Polybench_JACOBI_1D) and different (*e.g.*, Polybench_JACOBI_2D) performance across the modes.

ABSTRACT

Modern GPUs play a crucial role in accelerating a wide range of computational workloads. However, their performance is often limited by the memory access patterns of the kernels they execute. AMD's MI300A APU supports multiple logical GPU partitioning modes to optimize compute resource allocation, offering new opportunities for performance tuning. In this work, we evaluate how different GPU kernels from the RAJA Performance Suite perform in various partitioning modes. Using hardware counters, we compare two kernels with identical computational complexity but different data layouts, highlighting how memory organization can influence performance outcomes. The results demonstrate that data layout and access patterns have a significant impact on runtime performance across different partitioning modes, even when computational complexity and problem size remain constant.

KEYWORDS

Hardware counters, performance analysis, GPUs

1 INTRODUCTION

The El Capitan system at Lawrence Livermore National Laboratory uses AMD Instict MI300A APUs (Accelerated Processing Units) [1]. While similar to other GPUs, MI300As introduce new capabilities which we want to understand to fully leverage the APUs. One such capability is flexible compute partitioning, allowing users to divide the GPU resources into different number of logical devices to explore tradeoffs of parallel execution and resource efficiency.

We investigate how the MI300A partitioning modes impact the performance of different GPU computational kernels. Initial results show that data layout and access patterns significantly affect kernel runtime across partitioning modes, even when computational complexity and problem size are constant. We use the RAJA Performance Suite's [2] diverse set of GPU kernels designed to evaluate performance across a broad spectrum. RAJA Performance Suite (RAJAPerf) includes compute-bound and memory-bound kernels, and kernels with a variety of data layouts and access patterns.

Partitioning Mode	Ranks/APU	Problem Size/Rank
SPX	1	13,4217,728
TPX	3	44,739,242
CPX	6	22,369,621

Table 1: Experimental setup using SPX, TPX, and CPX modes.

2 BACKGROUND

The AMD MI300A introduces flexible partitioning capabilities by allowing its six accelerator complex dies (XCDs) to be logically grouped based on three distinct modes. In Single Partition X-celerator (SPX) mode, all six XCDs are combined to function as a single, large GPU, maximizing performance for unified workloads. The Triple Partition X-celerator (TPX) mode divides the XCDs into three groups, each consisting of two XCDs, effectively creating three independent GPUs. Finally, the Chiplet-Partitioned X-celerator (CPX) mode assigns each XCD to its own partition, resulting in six separate GPU instances. This partitioning flexibility enables the MI300A

SC '25, November 16-21, 2025, Saint Louis, MO, USA
2025. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

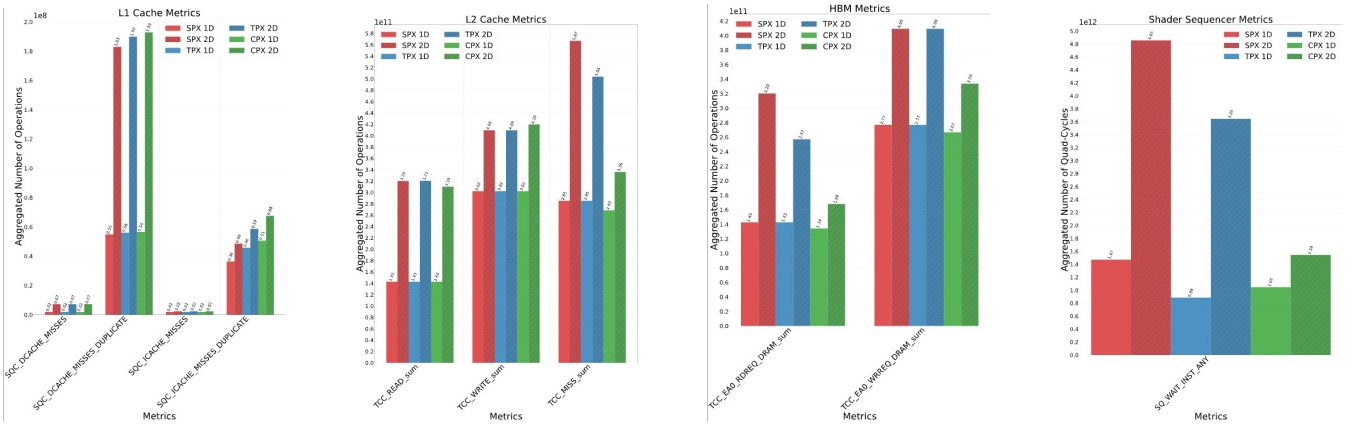

Figure 2: Hardware counters for four subsystems for two RAJAPerf kernels.

to efficiently support a wide range of workload requirements.We explore the MI300A GPU modes using RAJAPerf computational kernels with the HIP programming model (implementations also available in OpenMP, CUDA, and other programming models).

3 METHODOLOGY

We conduct our experiments using the RAJA Performance Suite on the LLNL Tuolumne machine, where each node is equipped with four AMD MI300A APUs. To ensure load balance and consistent resource utilization, we fully populated each node and maintained a fixed problem size per node by running four ranks in SPX mode, 12 ranks in TPX mode, and 24 ranks in CPX mode. For all configurations, the total problem size per node was set to 536,870,912. Table 1 summarizes the problem sizes we used for the different modes. We employed a block size of 256 threads. We utilize AMD's rocprofv3 profiling tool to collect low-level metrics and hardware counters. This includes monitoring GPU resources such as L1 and L2 cache usage, high-bandwidth memory (HBM) activity, and Shader Sequencer (SQ) performance. We used Caliper [3] to record the kernel runtimes in all three partitioning modes.

4 PERFORMANCE OF RAJA PERFORMANCE SUITE KERNELS FOR GPU MODES

We use Benchpark [4] to execute RAJAPerf across the three partitioning modes of the AMD MI300A APU, maintaining a consistent total node problem size. We observed significant difference in execution time among the partitioning modes for a subset of the kernels, despite the identical APU workload. Figure 1 shows the performance (in average time per rank) of RAJAPerf kernels for different GPU partitioning modes. The kernels are organized into two categories, a set of kernels that have similar performance across the modes and a set of kernels that have a wide performance difference. This result raises important questions about the underlying factors that influence performance. In particular, we study role of data layout and memory access patterns in the performance of the memory hierarchy of the MI300A.

To isolate the impact of memory access pattern on performance, we compare two kernels with O(n) computational complexity: one that traverses memory in an *in-order* fashion (Polybench_JACOBI_1D traverses a one-dimensional array of size N) and another that employs a *2D stencil* access pattern (Polybench_JACOBI_2D traverses a two-dimensional array of size $\sqrt{N} \times \sqrt{N}$, which also contains N elements), leading to different caching behavior. Polybench_JACOBI_1D shows the same runtime and identical hardware counter values for SPX, TPX, and CPX modes. In contrast, Polybench_JACOBI_2D shows more cache misses and memory requests than Polybench_JACOBI_1D because it needs to access data above and below each cell in the stencil, resulting in data access across three rows per iteration. In Polybench_JACOBI_2D, the number of L2 cache misses decreases from SPX to TPX, and further decreases from TPX to CPX mode. Figure 2 shows hardware counters values for the two kernels across the three partitioning modes.

5 CONCLUSIONS AND FUTURE WORK

Our results show that kernels with in-order memory access patterns exhibit similar performance across SPX, TPX, and CPX modes. In contrast, kernels employing 2D or 3D stencil memory access patterns demonstrate significant performance variation: they perform worst in SPX mode, better in TPX mode, and best in CPX mode. We attribute this improvement primarily to increased cache line reuse, similar to the effects of tunable blocking in matrix algorithms [5].

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 24-SI-005 (LLNL-ABS-2010191).

REFERENCES

[1] Alan Smith et al. Realizing the AMD Exascale Heterogeneous Processor Vision: Industry Product. In *Intnl Symp on Computer Architecture (ISCA)*, 2024.
[2] Olga Pearce et al. RAJA Performance Suite: Performance Portability Analysis with Caliper and Thicket. In *Intnl Conf on HPC, Network, Storage, & Analysis*, SC-W '24.
[3] David Boehme et al. Caliper: performance introspection for hpc software stacks. In *Intnl Conf for HPC, Networking, Storage & Analysis (SC'16)*.
[4] Olga Pearce et al. HPC Benchmarking: Repeat, Replicate, Reproduce. In *ACM Conf on Reproducibility and Replicability*, 2025.
[5] Hyuk-Jae Lee et al. Generalized Cannon's Algorithm for Parallel Matrix Multiplication. In *International Conference on Supercomputing*, 1997.