Scalable GPU Performance Variability Analysis framework

Ankur Lahiry[1], Ayush Pokharel[1], Seth Ockerman[3], Amal Gueroudji[2], Line Pouchard [4], Tanzima Z. Islam[1]
[1]Texas State University, [2]Argonne National Laboratory, [3]University of Wisconsin-Madison, [4]Sandia National Laboratory
vty8@txstate.edu, ssu22@txstate.edu, sockerman@cs.wisc.edu, agueroudji@anl.gov, lcpouch@sandia.gov, tanzima@txstate.edu

Abstract

Analyzing large-scale performance logs from GPU profilers often requires terabytes of memory and hours of runtime, even for basic summaries. These constraints prevent timely insight and hinder the integration of performance analytics into automated workflows. Existing analysis tools typically process data sequentially, making them ill-suited for growing trace complexity and volume. We introduce a distributed data analysis framework that scales with dataset size and compute availability. Instead of treating the dataset as a single entity, our system partitions it into independently analyzable shards and processes them concurrently across MPI ranks. This design reduces per-node memory pressure, avoids central bottlenecks, and enables low-latency exploration of high-dimensional trace data. We apply the framework to end-to-end Nsight Compute traces from real HPC and AI workloads and demonstrate its utility in diagnosing performance variability and uncovering the impact of memory transfer latency on GPU kernel behavior.

Goal

- Design a scalable and distributed analysis of large-scale GPU traces.
- Analyze GPU performance variability to identify performance bottlenecks.

Background

Profiling	Kernel	Memory	GPU	# of samples
Rank 0	842054	107045	4	~93M
Rank 1	842054	107099	4	~93M
Rank 2	842054	107045	4	~93M
Rank 3	842054	107045	4	~93M

Design

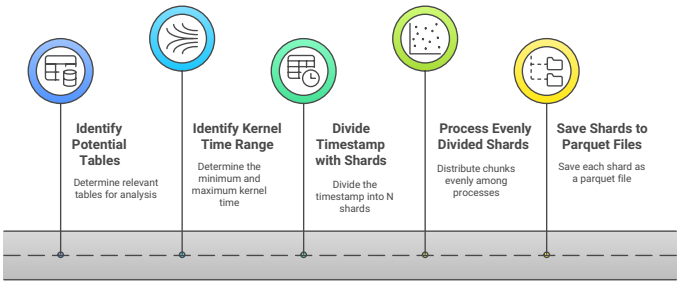


Figure: Data Generation

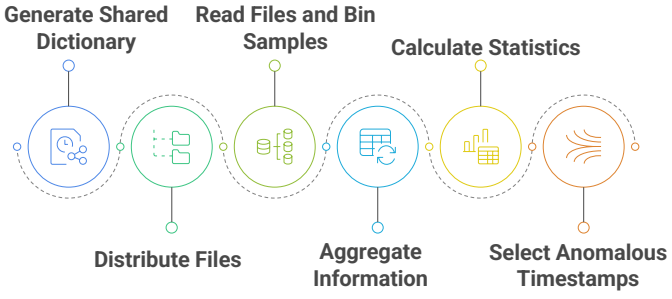


Figure: Data Aggregation

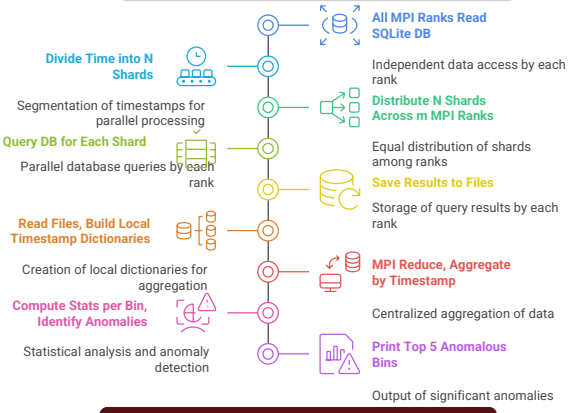
Data Generation:

- Identify relevant sqlite3 tables and extract kernel execution timestamps.
- Determine the minimum and maximum kernel time to define the full-time range.
- Split the time range into N shards, distribute them across P processes,
- Save each shard to a Parquet file.

Data Aggregation:

- Initialize a shared dictionary with timestamps as keys to organize samples.
- Assign each process N/P shards of data for parallel reading.
- Processes read their files and bin the samples based on kernel execution time into the shared dictionary.
- Aggregate binned data per rank to construct a complete view of the trace distribution.
- Perform statistical analysis on key columns to identify and rank the top K anomalous timestamps for further investigation.

Task Decomposition



Result

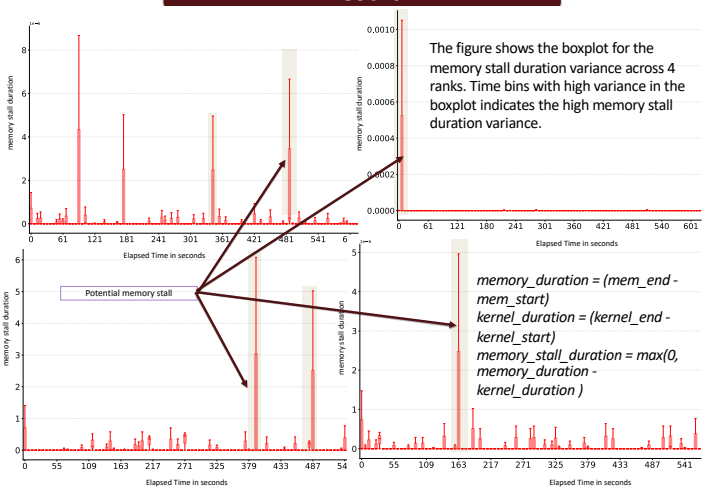


Figure 1: Memory Stall Duration vs Elapsed Time. The figure presents the variance in memory stall duration over time across four MPI ranks. The top two subplots correspond to Rank 0 and Rank 1, while the bottom two show Rank 2 and Rank 3. For each rank, the data is segmented into time bins of 1-second intervals, totaling up to 601 bins. Within each bin, the variance of memory stall durations is calculated and plotted to show the variability in memory stall.

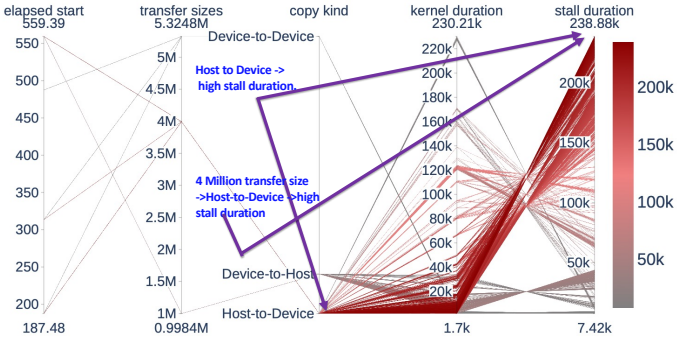
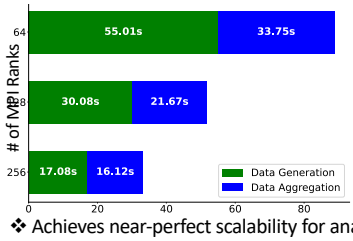


Figure 2: A parallel co-ordinate to visualize memory stall in Rank 2 for the top 5% of the highest stall duration in the bins. It shows the relation between the transfer size, copy kind and stall duration. Most of the high memory stalls happens in Host-to-device. The transfer sizes of 4 millions bytes results in Host-to-Device copy kind which results in high stall duration.



Left Figure shows Computation time over the MPI ranks for generating figure 1.

We see the speedup is around 1.8 ~ 2x if we double the number of MPI ranks.

- Achieves near-perfect scalability for analyzing large-scale data.
- Identified specific performance metrics responsible for memory stalls.

Future Work

- Improve the performance bottleneck of the pipeline by removing the I/O operations like file read or saving to the parquet files
- Analyze root cause for the variance in kernel

References

https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html
https://mpi4py.readthedocs.io/en/stable/

This material is based upon work supported by the U.S. Department of Energy, Office of Science under Award Number DE-SC0023173.