A tool for top-down performance analysis of GPU-accelerated applications

Keren Zhou, Mark Krentel, and John Mellor-Crummey

Department of Computer Science, Rice University

Abstract

We extended Rice University's HPCToolkit to measure and analyze GPU-accelerated applications. Our tool has the following key innovations:

- A measurement substrate that employs a wait-free data structure to coordinate measurement and attribution between each application thread and a GPU monitor thread.
- An approach to reconstruct an approximate calling context tree for GPU computations from flat GPU PC samples.
- A method to derive GPU performance metrics from PC samples and attribute them at all levels in a heterogeneous calling context.

Background

- A *trace view* shows a series of events that happen over time on each process, thread, and GPU stream.
- A *profile view* collapses out the time dimension and correlates performance metrics with program contexts.
- Most GPU performance tools, including nvprof, Nsight Systems, Nsight Compute, TAU, and Allinea Map, only provide a trace view for GPU API calls and/or a flat profile view.
- HPCToolkit attributes performance metrics to calling contexts that span both CPUs and GPUs.
- In the Figure below, we show HPCToolkit's heterogeneous calling context for a **fully optimized** GPU kernel that has 28 device functions.

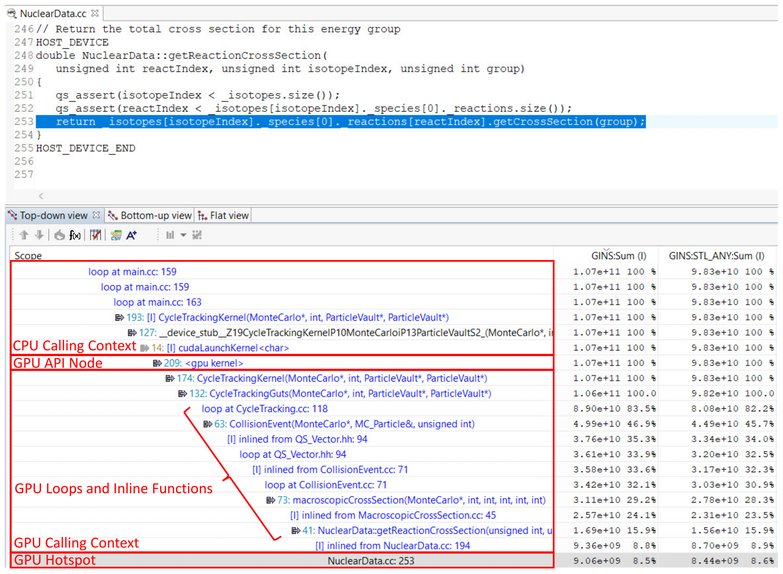

Figure: Quicksilver is a GPU-accelerated proxy application for LLNL's Mercury - a Monte Carlo neutron transport code.

Wait-free Communication

- We designed a data structure that supports wait-free communication of measurement and correlation records between the GPU monitor thread and each application thread.

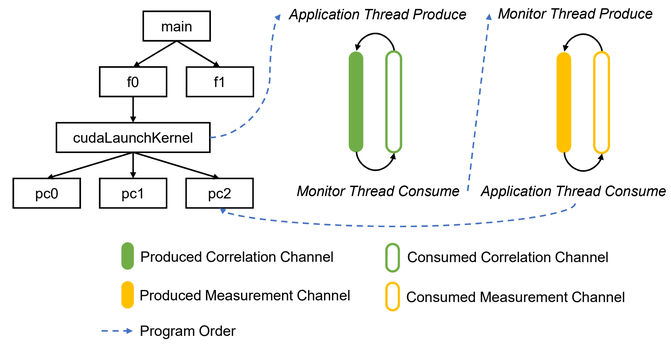

Figure: Interactions between the GPU monitor thread and application threads. pc0, pc1, and pc2 denote PC samples

GPU Calling Context Tree Reconstruction

- We first construct a static call graph based on function symbols and call instructions.
- We transform the call graph into a calling context tree by splitting call edges and cloning called procedures.
- We use a heuristic method to apportion samples to procedures called from multiple call sites.

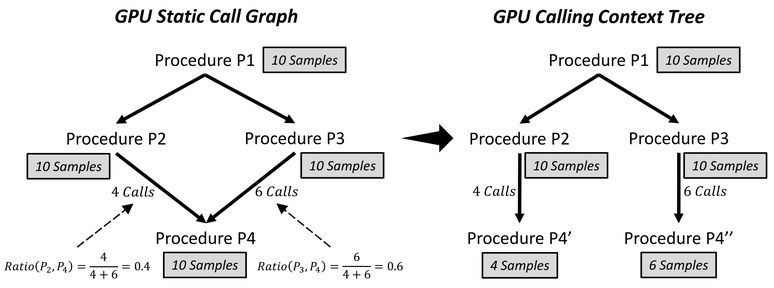

Figure: A procedures's costs are computed using the number of samples at each call site

Metrics Derivation

- Some performance metrics cannot be collected in the same pass with PC samples.
- Nsight-compute runs nine passes to collect all of these metrics for a GPU kernel.
- Our tool estimates important performance metrics based on PC samples and kernel statistics, including SM efficiency, SM busy rate, SM active ratio, occupancy, warp issue rate, IPC, issued instruction counts, instruction throughput, etc.

Case Studies

- We have used our tool to analyze HPC applications on the Summit supercomputer whose compute nodes are equipped with IBM POWER9 processors and NVIDIA Volta GPUs.
- We identified three costly memory copies in Laghos from the bottom-up profile view.
- Applying optimizations based on calling context and performance increases Laghos' GPU code by 25%.

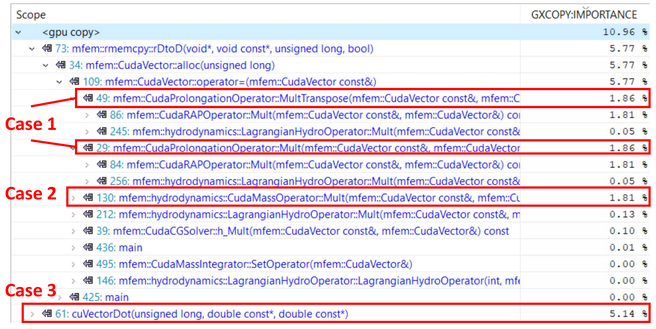

Figure: Laghos is a DOE mini-app that solves the time-dependent Euler equation of compressible gas dynamics.

- By combining PC sampling measurements with instruction mix analysis, our tool provides the metrics necessary to construct a Roofline model.
- We optimized Nekbone to 84% of the peak performance limited by memory bandwidth.

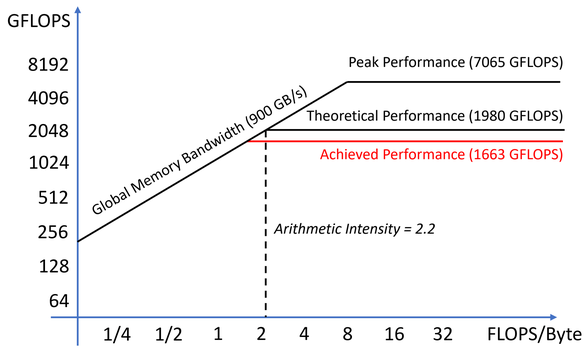

Figure: Nekbone is a lightweight subset of Nek5000 that mimics the computational characteristics of Nek5000, a high-order Navier-Stokes solver based on the spectral element method.

Next Steps

- We are continuing work on our tools with the aim of turning measurement data and metrics into high-level guidance for performance tuning.
- We have begun to apply our tool to MPI-based HPC applications and machine learning applications.