Performance Profiling, Analysis, and Optimization of GPU-accelerated Applications

Keren Zhou*

Department of Computer Science, Rice University *Advisor: John Mellor-Crummey

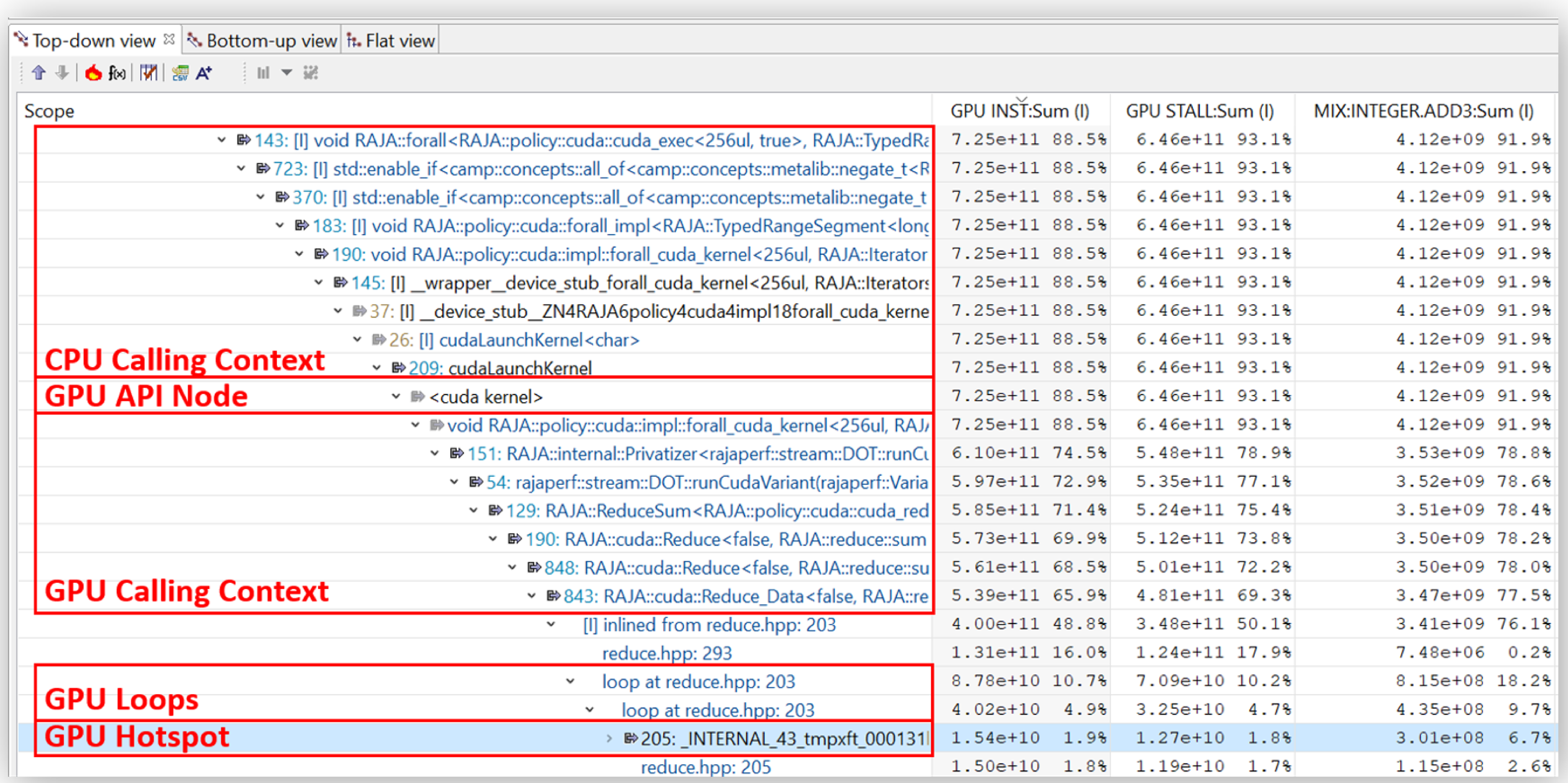
Abstract

We built tools to analyze the performance of GPU-accelerated applications. Our tools incorporate the following innovations:

- A **heterogeneous calling context view** for GPU programs
- An **automated performance advisor** that suggests effective optimizations
- A **value analyzer** that analyzes patterns of inefficiencies in GPU programs related to data values

Motivation

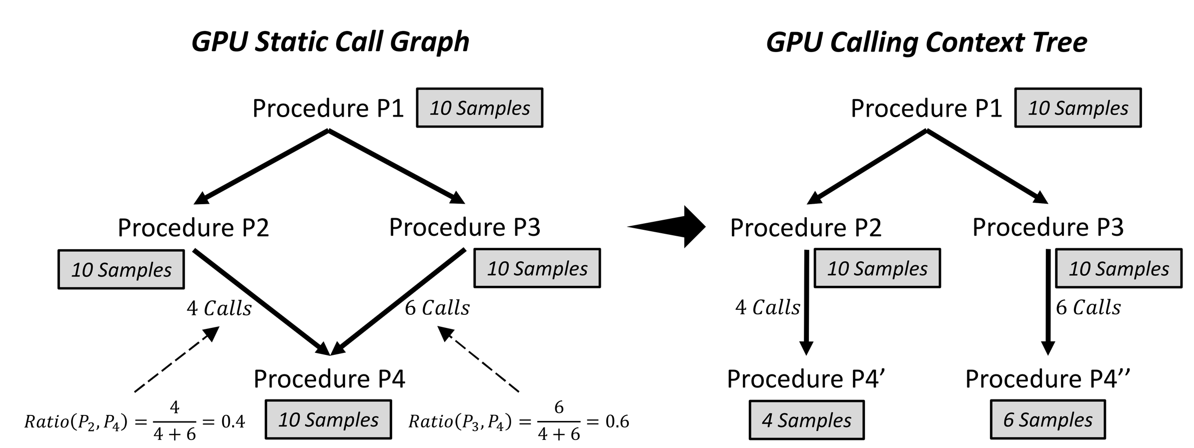
- The world's most powerful supercomputers are accelerated by GPUs
- Writing code well-suited to GPU architectures is critical for achieving peak performance
- GPU-accelerated applications may underutilize GPU resources due to program and data characteristics
- Pinpointing performance problems can be difficult as it often requires detailed analysis



Detailed Profile View of LLNL's Quicksilver

Heterogeneous Calling Context

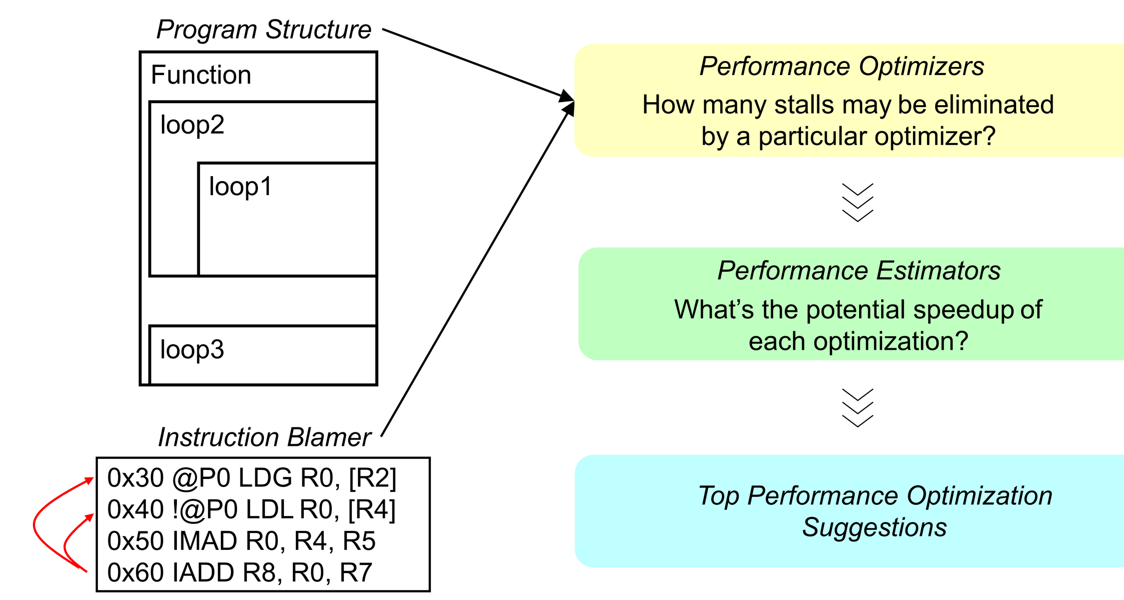
- The use of high-level programming models such as RAJA, Kokkos, and OpenMP can increase the difficulty of tuning GPU kernels for high performance by separating developers from many key details
- Our tool attributes metrics to GPU computations in the full heterogeneous calling context where they execute
- **CPU Calling Context:** Unwind the call stack at every GPU API call to associate GPU operations with the CPU calling context where they are initiated
- **GPU Calling Context:** Construct a GPU static call graph and apportion costs for each device function among its call sites according to the fraction of calls from each call site



Apportion P4's instruction samples among its call sites

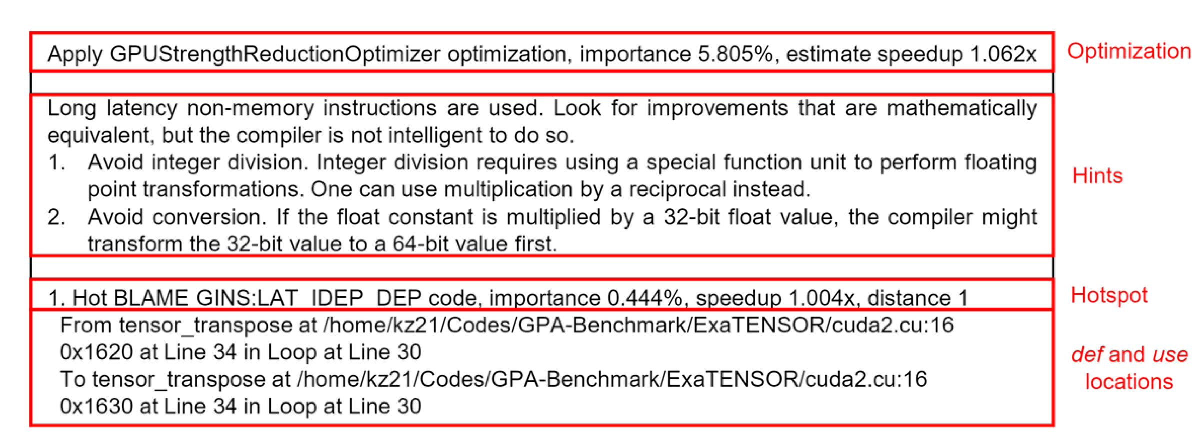
Performance Advisor

- Existing performance tools, such as Nsight Compute, only provide coarse-grained suggestions at the kernel level, if any
- GPA, our GPU performance advisor, suggests potential code optimizations at a hierarchy of levels, including individual *lines*, *loops*, and *functions*



The workflow of our performance advisor

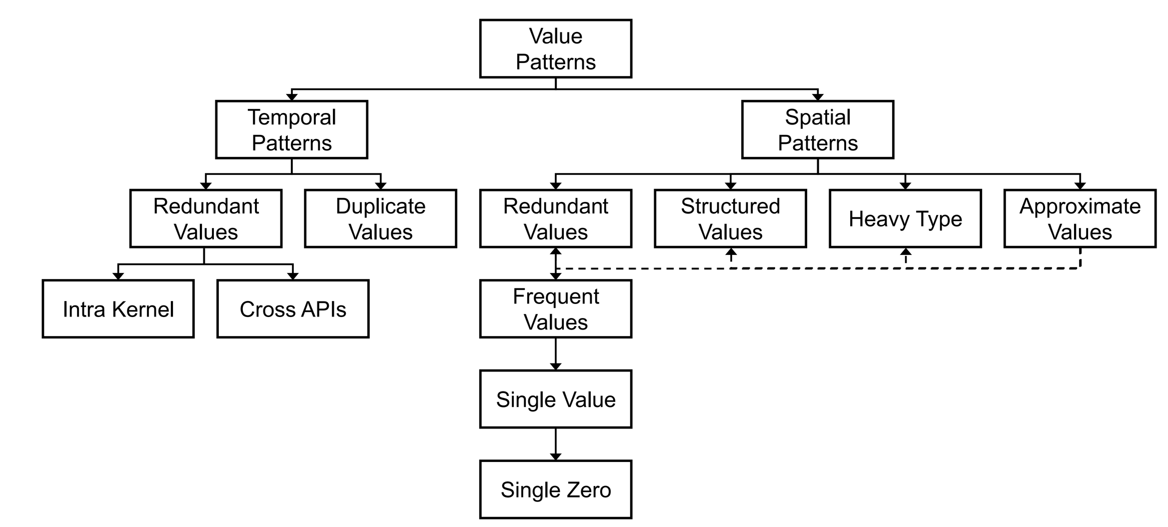
- **Instruction Blamer:** Attribute instruction stalls to their root causes by analyzing instruction dependencies
- **Performance Optimizers:** Associate instruction stalls with root causes to match inefficient code with suggestions for optimizations
- **Performance Estimators:** Estimate the potential speedup of each optimization
- **Advice Report:**
  - Lists several optimization suggestions ranked by speedups they may provide
  - Offers *hints* about code transformations to improve performance and *hotspots* where hints may apply
  - Supplies *program context*, *importance*, and *speedup* information for each hotspot



An example advice report for the ExaTensor code

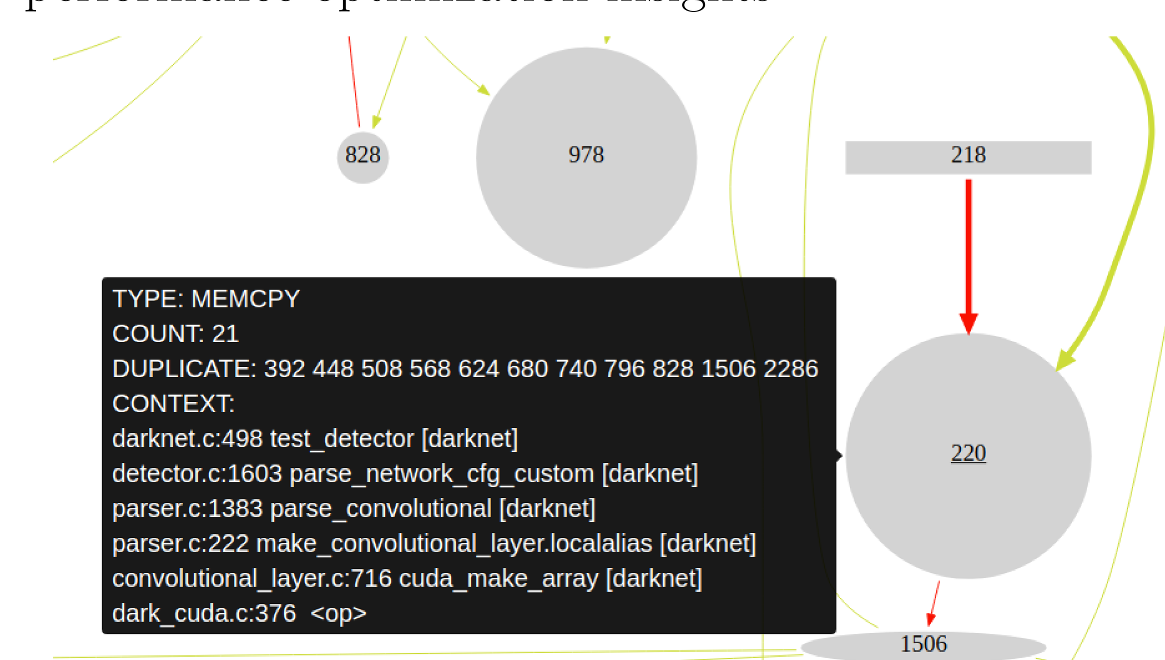
Value Analyzer

- Redundant computation and memory accesses involving the same values are pervasive problems exist in GPU-accelerated applications
- We developed value profiling and analysis tools to identify inefficient value access patterns in applications running on GPU-based clusters to address this issue
- **Efficient Profiling:** Employ *kernel filtering*, *kernel sampling*, *block sampling*, and *on-the-fly GPU-accelerated analysis* to reduce profiling overhead
- **Value Pattern Taxonomy:** Categorize common inefficient value patterns in GPU benchmarks and applications



Common inefficient value patterns

- **Value Flow Graph:** Visualize value changes across GPU APIs to provide value-related performance optimization insights



An example value flow graph for Darknet. The green edges denote benign value patterns, while the red edges denote the redundant values pattern. The thickness of edges quantifies the number of bytes accessed