Tuning Heterogeneous Computing Architectures Through Integrated Performance Tools

Robert Lim, Boyana Norris and Allen Malony
Performance Research Lab, High-Performance Computing Lab, University of Oregon

1 Motivation

Heterogeneous computing presents challenges in optimizing performance across diverse architectures, high-speed networks and programming methods present in these systems. For instance, GPU performance measurements can be captured using events injected into the stream immediately before and after the computation kernel, but it does not give a way to look at the details of kernel performance. Observing GPU hardware performance counters, collected either through instrumentation or sampling, elucidates kernel execution, but does not provide a means to correlate dense activity regions with source line information. The difficulty of attributing which tasks are on the CPU vs the GPU makes understanding heterogeneous parallel applications a very complicated task.

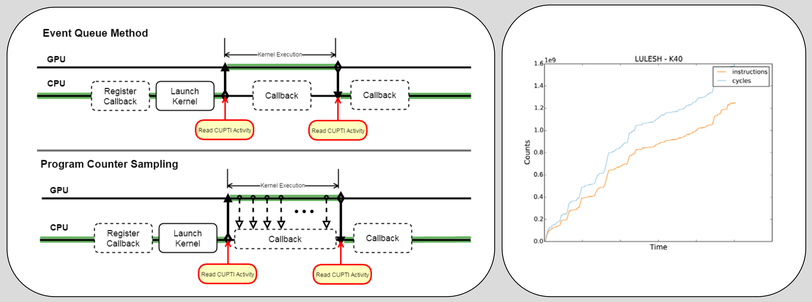

Figure: Event queue method and program counter sampling (left). Hardware counter sampling (right).

GPU accelerators specialize in executing SIMD (single instruction, multiple data) in lock-step, where threads that do not satisfy branch conditions are masked out. Mapping structured/unstructured control flow in SIMD architectures complicates matters because the application can only proceed once all warps have synchronized. Control flow graphs (CFG) can model program control flow and dependencies that may exist in a program. Calculating the execution frequency of code blocks provides an understanding of the program structure. In deriving the trip counts of the CFG, one can determine how an application of size N will perform on a GPU without having to compile or run the application.

2 Objectives

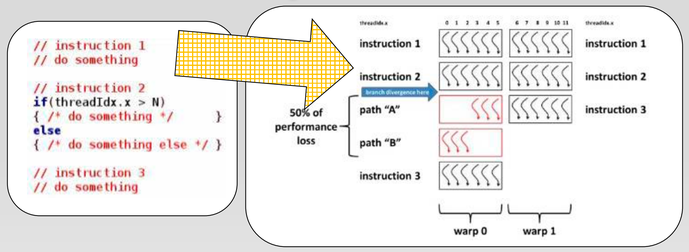

Figure: Branch divergence example which demonstrates loss of performance.

- Address divergent branch problem (% loss in performance)
- Facilitate in mapping structured/unstructured control flow in SIMD architectures
- Predict performance of kernel execution
- Discover whether a kernel benefits from executing on a GPU or CPU

PROPOSED METHODOLOGY

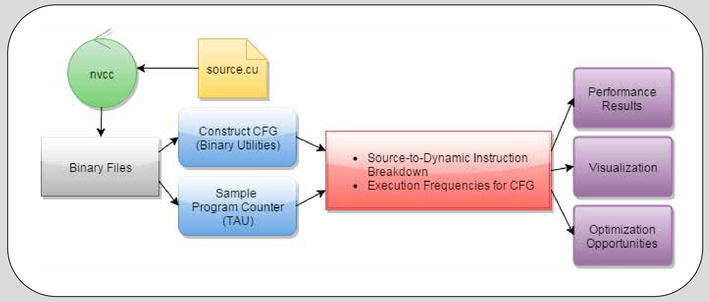

Figure: Overview of our proposed methodology.

3 Methodology

We construct control flow graphs statically and sample the program counter dynamically using the TAU Parallel Performance System, which provides instruction operations executed and source code location, among other information. Our methodology provides a more accurate characterization of the application kernel [1], when compared with other approaches such as hardware performance counters.

- Sample program counter. Collect instruction mixes, threads executed
- Construct CFG, calculate trip counts for each kernel
- Predict execution time for a given input size based on trip counts

Control Flow Graph

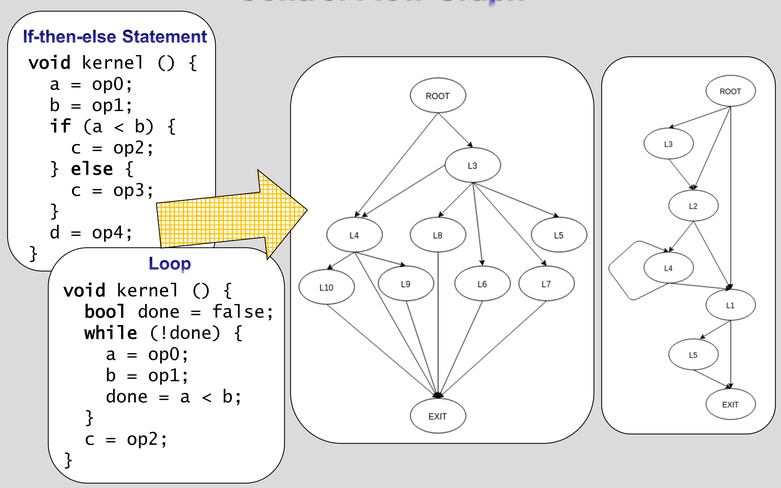

Figure: Control flow graph from Gaussian and Streamcluster kernel applications.

* op# represents instruction operations

4 Evaluation

Kernel instructions and execution frequencies were measured on three GPUs: M2090 (Fermi), K40M (Kepler), M6000 (Maxwell). The left plot shows instruction mixes for LULESH, where CKE, CMG CE2 displays more compute-intensive operations. The right plot shows a control flow graph for Gaussian, where each individual bar represents code block regions, and the three block sets represent input sizes from small to large N. Results show that each GPU creates its own version of a control flow graph. M2090 has a more balanced trip count across all blocks, while K40M and M6000 show higher execution frequencies for selected blocks (L_6, L_8).

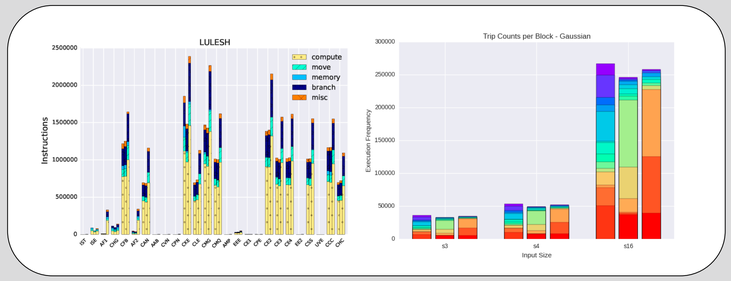

Figure: Execution frequencies of individual blocks for Gaussian and BFS kernels.

5 Future Work

- Apply Bayesian techniques to predict optimal performance parameters (block size, thread counts)
- Continue integrating CUDA/OpenACC/PGI support in TAU [2]
- Provide information from control flow graph and instruction mix sampling to Orio [3] to narrow parameter search space for autotuning

6 References

[1] Lim, R., et. al. "Identifying Optimization Opportunities within Kernel Execution for GPU Codes." *HeteroPar, in conjunction with EuroPar* (2015).
[2] Shende, S. and Malony, A. "The TAU parallel performance system." *International Journal of High Performance Computing Applications* (2006).
[3] Hartono, A., Norris, B. and Sadayappan, P.. "Annotation-based empirical performance tuning using Orio." *Parallel & Distributed Processing* (2009).
[4] Che, S., et. al. "Rodinia: A benchmark suite for Heterogeneous computing." *International Symposium on Workload Characterization* (2009).

7 Acknowledgements

R. Lim is currently a recipient of the Department of Defense SMART Fellowship Program. We want to thank NVIDIA for providing early access to CUDA 7.5 and to the PSG Clusters. This work is supported by the Department of Energy (Award #DE-SC0005360) for the project "Vancouver2: Improving Programmability of Contemporary Heterogeneous Architectures."