Poster: Determining code segments that can benefit from execution on GPUs

Ashay Rane
University of Texas at Austin
ashay.rane@tacc.utexas.edu

Saurabh Sardeshpande
University of Texas at Austin
saurabh.ss@gmail.com

James Browne
University of Texas at Austin
browne@cs.utexas.edu

ABSTRACT

Graphics Processing Units (GPUs) are a low cost, low power means of exploiting large-scale parallelism. Source-to-source transformation tools for mapping CPU code to GPU code (e.g. PGI Accelerator) are available. But identification of those code segments in an application that, when run on a GPU will attain significant performance enhancement, requires expert knowledge of algorithms, architectures, compilers and the program structure which many application developers may not possess. This poster demonstrates a process for identifying the code segments in programs optimized for multicore chip execution that are candidates for GPU execution and ranking these code segments by probable speedup. The identification and ranking are based on measurements of the programs by the PerfExpert tool and a new tool MACPO, which measures execution properties of data structures. The poster describes the identification and ranking process, gives the results of applying the process to the Rodina parallel benchmarks and gives the underlying assumptions for and the limitations of the process.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

Keywords

Graphics Processing Units, GPUs

1. INTRODUCTION

Graphics Processing Units (GPUs) provide a low cost, low power means of exploiting large-scale parallelism and are now widely available. The models of parallel execution implemented for applications executing on multicore chips (usually SPMD or MIMD) and in GPUs (usually some variant of SIMT) are quite different. Attainment of the full performance benefit from a GPU may require not only recoding but reformulation of the computations. However, it is commonly the case that code representing a significant fraction of an application can be recoded in the programming model or language of a GPU. Indeed, source-to-source transformation tools for mapping CPU code to GPU code (e.g. PGI Accelerator [1]) are available. The barrier to this mode of attaining performance benefit from GPUs is that identification of those code segments in an application that can be straightforwardly recoded to execute efficiently on a GPU requires expert knowledge of algorithms, architectures, compilers and the program structure which many application developers may not possess.

This poster presents a simple process of identifying the code segments (functions and loops) that which can be recoded to execute efficiently on a GPU and ranking those code segments by probable performance gain. The underlying assumption is that the code has already been optimized for efficient execution on multicore chips. The process is based on characterizing the functions and loops of an application as described in the next section. The characterization is accomplished by using the PerfExpert [2, 3] tool for optimization of performance on multicore chips and the MACPO [4] tool for measuring the memory access characteristics of code segments. PerfExpert analyzes the execution of the user's source code using hardware performance counters. MACPO is a low-overhead tool that captures memory traces and computes metrics for the memory access behavior of source-level (C, C++, Fortran) data structures. Both performance counter data and the data structure metrics are used to produce a score, which is an approximate metric of the degree of confidence in suitability of the code for GPU execution. The identifications and rankings are validated by comparing the code segments identified and ranked by our process with the code segments in the Rodina benchmarks that have been manually selected by experts for GPU execution.

Section II details the approach and process. Section III gives the results of applying the process to the Rodina benchmark set using the results of expert manual identification and ranking as comparison. The predictions are accurate in five of the six cases.

The process is based on the observation that SPMD parallelism can be mapped to SIMT parallelism.

2. APPROACH

The requirements for a code segment such as a loop to be directly recoded for GPU execution are:

- It must be computationally intensive,
- It must possess reasonable data locality,

SC'11 Companion, November 12–18, 2011, Seattle, Washington, USA.
ACM 978-1-4503-1030-7/11/11.

- The parallelism must be mappable to SIMT and
- Data access strides should be small and regular.

The measurements from PerfExpert consider the data access patterns, branching behavior and computational efficiency. Some properties (like parallelism) are inferred rather than directly measured. We normalize the measurements to the length of the program execution by calculating Local Cycles Per Instruction (LCPI) [2] values. The recommendation to port certain code segments to run on the GPU is a two-step process:

1. Eliminating those code segments from consideration that exhibit characteristics that are not suited for GPU execution
2. Ranking the remaining code segments based on characteristics that help in efficient execution on the GPU

The elimination process uses the following rules:

$$data\ access\ LCPI < 0.3 \tag{1}$$

$$TLB\ LCPI < 0.2 \tag{2}$$

$$branch\ LCPI < 0.3 \tag{3}$$

The code segments that pass this test are scored based on the following characteristics:

$$x = C_1 \cdot \frac{Achieved\ FLOP}{Max\ FLOP} \tag{4}$$

$$y = C_2 \cdot \frac{Achieved\ IOP}{Max\ IOP} \tag{5}$$

$$z = C_3 \cdot \left(1 - \frac{Observed\ stride}{Max\ stride}\right) \tag{6}$$

$$\mathbf{score} = x + y + z$$

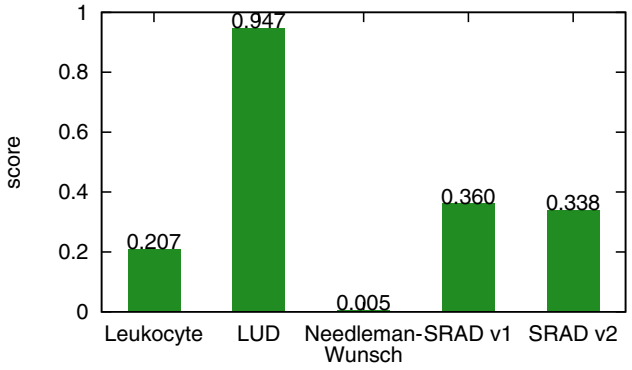
The constants (C_1, C_2 and C_3) in the above equations were determined from training samples, as described in Section 3. These constants are architecture specific.

3. RESULTS

We used 12 programs from the Rodinia benchmark suite [5] for testing the validity of the recommendations. The Rodinia suite is a collection of computational codes implemented in both OpenMP and CUDA. The presence of both OpenMP and CUDA versions of these applications made it possible to directly compare the results generated from our technique with the functions rewritten by the maintainers of the Rodinia benchmarks to use CUDA. To test if our methodology would include false positives, we included an additional application (GNU `tar`) that we know would not run efficiently on GPUs because its algorithm itself is not suited to the GPU architecture. Half of the Rodinia benchmark suite was used as the training set and the remaining half, in combination with GNU `tar`, formed the verification set. The constants in equations 4-6 were calculated from measurements on the training set. Out of the six computational codes, our technique identified the functions and loops that had been converted to use CUDA in five codes. For the sixth code (StreamCluster), the process failed to recognize the appropriate code segment because the data access metric is high. This may be the result of incomplete optimization for multicore execution.

The calculated scores for the remaining five codes are shown in Figure 1.

Figure 1: Calculated scores indicating suitability of GPU execution



4. CONCLUSIONS AND FUTURE WORK

This poster has presented a simple process which appears to yield reliable identification and ranking of functions and loops which are candidates for efficient execution on GPUs. The process is built on existing tools and requires minimal knowledge of architectures or compilers. For future work, we will include a larger and more diverse set of benchmarks. We also plan to incorporate the identification and rankings reported in this poster to the optimization recommendation capability of PerfExpert [3] by including suggestions for running specific functions and loops on a GPU.

Acknowledgments

This project is funded in part by the National Science Foundation under OCI award #0622780.

5. REFERENCES

[1] M. Wolfe, "Implementing the pgi accelerator model." in *GPGPU*, 2010, pp. 43–50.

[2] M. Burtscher, B. D. Kim, J. Diamond, J. Mccalpin, L. Koesterke, and J. Browne, "PerfExpert : An Easy-to-Use Performance Diagnosis Tool for HPC Applications," in *Computer*. IEEE, 2010, pp. 1–11.

[3] O. A. Sopeju, M. Burtscher, A. Rane, and J. Browne, "AutoSCOPE : Automatic Suggestions for Code Optimizations using PerfExpert," *Evaluation*.

[4] A. Rane and J. Browne, "Performance optimization of data structures using memory access characterization," in *CLUSTER*. IEEE, 2011, pp. 570–574.

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," *2009 IEEE International Symposium on Workload Characterization IISWC*, vol. 2009, no. c, pp. 44–54, 2009.