University of Victoria | Graduate Studies

Notice of the Final Oral Examination
for the Degree of Doctor of Philosophy

of

AHMAD LASHGAR

MSc (University of Tehran, 2012)
BSc (Jundi Shapor University of Technology, 2010)

"Addressing Software-Managed Cache Development Effort in GPGPUs"

Department of Electrical and Computer Engineering

Tuesday, August 8, 2017
10:00 A.M.
Engineering & Coomputer Science Building
Room 468

Supervisory Committee:
Dr. Nikitas J. Dimopoulos, Department of Electrical and Computer Engineering, University of Victoria (Supervisor)
Dr. Mihai Sima, Department of Electrical and Computer Engineering, UVic (Member)
Dr. Alex Thomo, Department of Computer Science, UVic (Outside Member)

External Examiner:
Dr. Xipeng Shen, Department of Computer Science, North Carolina State University

Chair of Oral Examination:
Dr. Neil Burford, Department of Chemistry, UVic

Dr. David Capson, Dean, Faculty of Graduate Studies

Abstract

GPU Computing promises very high performance per watt for highly-parallelizable workloads. Nowadays, there are various programming models developed to utilize the computational power of GPGPUs. Low-level programming models provide full control over GPU resources and allow programmers to achieve peak performance of the chip. In contrast, high-level programming models hide GPU-specific programming details and allow programmers to mainly express parallelism. Later, the compiler parses the parallelization notes and translates them to low-level programming models. This saves tremendous development effort and improves productivity, often achieved at the cost of sacrificing performance. In this dissertation, we investigate the limitations of high-level programming models in achieving a performance near to low-level models. Specifically, we study the performance and productivity gap between high-level OpenACC and low-level CUDA programming models and aim at reducing the performance gap, while maintaining the productivity advantages. We start this study by developing our in-house OpenACC compiler. Our compiler, called IPMACC, translates OpenACC for C to CUDA and uses the system compile to generate GPU binaries. We develop various micro-benchmarks to understand GPU structure and implement a more efficient OpenACC compiler. By using IPMACC, we evaluate the performance and productivity gap between a wide set of OpenACC and CUDA kernels. From our findings, we conclude that one of the major reasons behind the big performance gap between OpenACC and CUDA is CUDAs flexibility in exploiting the GPU software-managed cache. Identifying this key benefit in low-level CUDA, we follow three effective paths in utilizing software-managed cache similar to CUDA, but at a lower development effort (e.g. using OpenACC instead). In the first path, we explore the possibility of employing existing OpenACC directives in utilizing software-managed cache. Specifically, the cache directive is devised in OpenACC API standard to allow the use of software-managed cache in GPUs. We introduce an efficient implementation of OpenACC cache directive that performs very close to CUDA. However, we show that the use of the cache directive is limited and the directive may not offer the full-functionality associated with the software-managed cache, as existing in CUDA. In the second path, we build on our observation on the limitations of the cache directive and propose a new OpenACC directive, called the few directive, to address the shortcomings of the cache directive, while maintaining OpenACC productivity advantages. We show that the few directive overcomes the cache directive limitations and narrows down the performance gap between CUDA and OpenACC significantly. In the third path, we propose fully-automated hardware/- software approach, called TELEPORT, for software-managed cache programming. On the software side, TELEPORT statically analyzes CUDA kernels and identifies opportunities in utilizing the software-managed cache. The required information is passed to the GPU via API calls. Based on this information, on the hardware side, TELEPORT prefetches the data to the software-managed cache at runtime. We show that TELEPORT can improve performance by 32% on average, while lowering the development effort by 2.5X, compared to hand-written CUDA equivalent.