Compiler Transformation of Nested Loops for GPUs

Xiaonan Tian, Rengan Xu and Barbara Chapman
Department of Computer Science, University of Houston

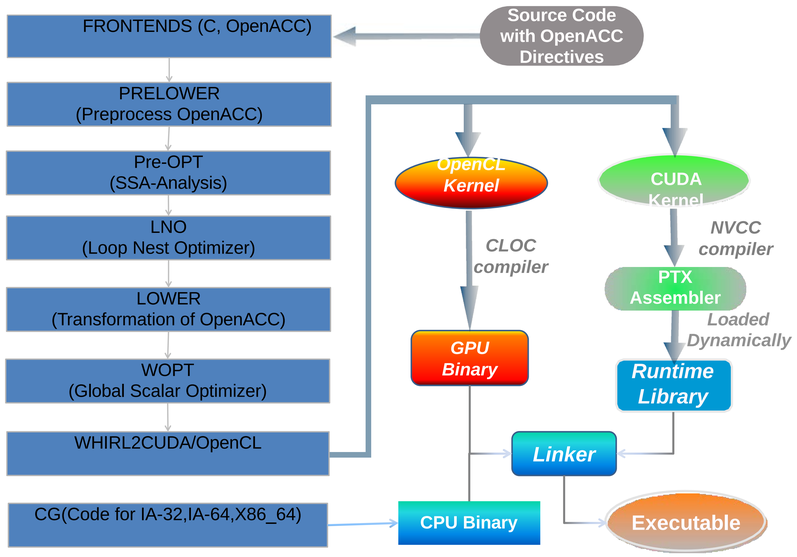
Introduction

Manycore accelerators have the potential to significantly improve performance of scientific applications when offloading computationally intensive program portions to accelerators. Directive-based high-level programming models, such as OpenACC and OpenMP, are used to create applications for accelerators through annotating regions of code meant for offloading. OpenACC is an emerging directive-based programming model for programming accelerators that typically enable inexperienced programmers to achieve portable and productive performance within applications. In this paper, we present our research in developming challenges and solutions when creating an open-source OpenACC compiler in an industrial framework (OpenUH as a branch of Open64). We then discuss in detail techniques we developed for loop-scheduling reduction operations on GPGPUs. The compiler is evaluated with NAS parallel benchmarks and self-written micro-benchmarks for reduction operations. This implementation has been designed to serve as a compiler infrastructure for researchers to explore advanced compiler techniques, extend OpenACC to other programming models, and build performance tools used in conjunction with OpenACC programs.

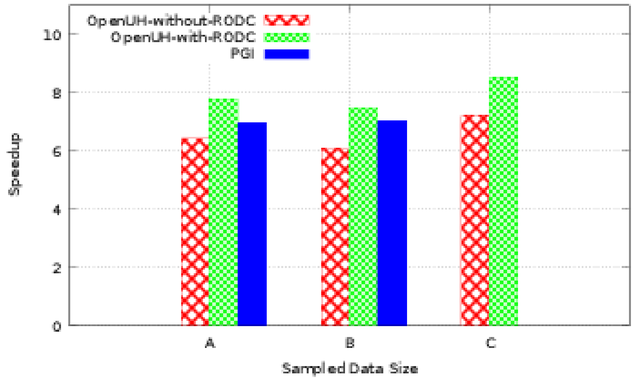
Contribution

- We deliver an open source OpenACC research compiler based on the robust Open64 industry-level compiler framework. Thus, the experiences could be applicable to other compiler implementation efforts. The OpenUH compiler adopts a source-to-source approach and generates readable CUDA source code for GPGPUs. This gives users an opportunity to understand the applications of loop mapping mechanisms which can be used to further manually optimize the code, if need be. It also allows the user to leverage the advanced optimization features offered by CUDA in the backend.
- We propose a rich set of loop-scheduling strategies within the compiler to efficiently distribute kernels or parallel loops to the threading architectures of GPGPU accelerators.Our findings provide guidance for users to adopt suitable loop schedules depending on the application requirements.
- We present the compile-time read-only array/pointer detection for the read-only cache optimization in the latest Nvidia Kepler architecture.
- We evaluate our novel strategies and its implementations in OpenUH OpenACC compiler using NPB benchmarks. Then comparisons against CUDA code version and PGI compiler are presented. The results show that OpenUH generates competitive performanceto CUDA and modest performance gains over PGI.

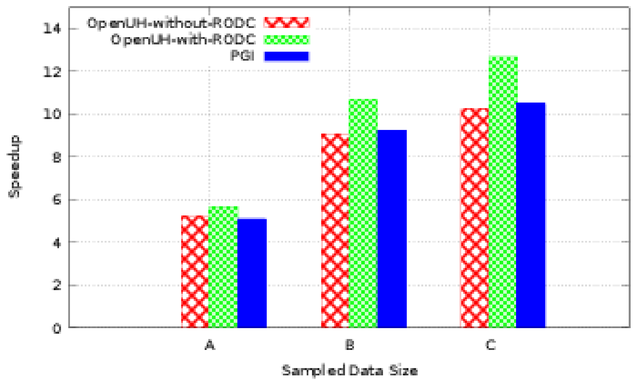
OpenUH OpenACC Compiler Infrastructure



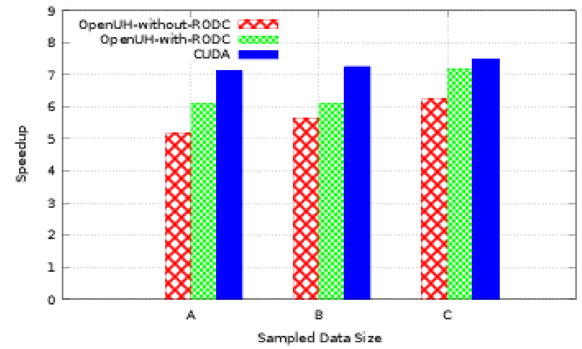
Results



NAS MultiGrid



NAS Conjugate Gradiant



NAS Block Tridiagonal

Conclusion

- In this poster, we presented a robust OpenACC implementation in the OpenUH compiler. We describe our compiler framework, which provides a rich set of loop scheduling strategies and discuss the Read-Only Data Cache optimization. We also deliver compile-time detection mechanisms to recognize read-only array/buffer use in the application, taking advantage of the read-only data cache in the hardware for OpenACC parallel and kernels regions.
- We used the NPB applications to evaluate our design and implementation. A number of accomplishments were presented. First, we provided compiler support for Read-Only Data Cache optimization which dramatically improved cases when there was a large number of read-only data/buffer in the offloaded compute regions. Second, our compiler support for a directive-based model, OpenACC, not only helped maintain source code consistency but also achieved performance close to that of a well-tuned CUDA code for NPB applications (BT). Evaluation for these solutions demonstrated that our compiler could yield competitive performance compared to that of some of the existing vendor compilers.

Reference

1. Xiaonan Tian, Rengan Xu, Yonghong Yan, Sunita Chandrasekaran, and Barbara Chapman. **Compiler Transformation of Nested Loops for GPGPUs**. Submitted to *Concurrency and Computation: Practice and Experience*

2. Xiaonan Tian, Rengan Xu, Yonghong Yan, Zhifeng Yun, Sunita Chandrasekaran, and Barbara Chapman. "Compiling a High-level Directive-Based Programming Model for GPGPUs," in The 26th International Workshop on Languages and Compilers for High Performance Computing (LCPC 2013), 2013.

3. Rengan Xu, Xiaonan Tian, Yonghong Yan, Sunita Chandrasekaran, and Barbara Chapman. **Reduction Operations in Parallel Loops for GPGPUs**. in the 2014 International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM 2014), February, 2014, Orlando, Florida, USA