Enabling OpenACC programming on Multi-hybrid Accelerated with GPU and FPGA

Tsunashima, Ryuta[1)] Kobayashi, Ryohei[2)1)] Fujita, Norihisa[2)] Nakamichi, Ayumi[1)] Boku, Taisuke[2)1)]
Lee, Seyong[3)] Vetter, Jeffrey[3)] Murai, Hitoshi[4)] Sato, Mitsuhisa[4)]

1) Graduate School of Systems and Information Engineering, University of Tsukuba, Ibaraki, Japan
2) Center for Computational Sciences (CCS), University of Tsukuba, Ibaraki, Japan
3) Oak Ridge National Laboratory (ORNL), TN, USA
4) Riken Center for Computational Science (R-CCS), RIKEN, Hyogo, Japan

Although the GPU is main player for accelerated computation in HPC, some category of applications are not suitable for it. For example, partially poor parallelism, non-regular computation (warp divergence) or frequent inter-node communication strongly degrade the performance in parallel GPU computing. On the other hand, FPGAs have been emerging in HPC. FPGA enables us to program the logic device in true co-designing manner. On April 2019, CCS in University of Tsukuba introduced a new GPU+FPGA hybrid accelerated cluster named Cygnus[1]. However, currently users have to describe programs in two languages, CUDA for GPU and OpenCL for FPGA to utilize both devices effectively and it causes heavy effort for users. It is much better if we can provide a uniform framework to program both devices at a single code. Then we are implementing a meta-compiler to apply OpenACC[2] for both devices, based on background compilers for GPU and FPGA.

We assume to use two background compilers, PGI OpenACC compiler for GPU and OpenARC[3] compiler for FPGA. As shown in Figure 1, the meta-compiler splits the corresponding OpenACC-directed parts out of original code into two parts for GPU and FPGA. Then these parts are compiled by corresponding backend compilers. Finally, two object files are linked to a single executable file by PGI compiler. We use Omni compiler[4] developed by RIKEN R-CCS and CCS of University of Tsukuba to implement the meta-compiler. OpenARC is a compiler to enable OpenACC for FPGA programming developed in ORNL. It translates OpenACC code in C to OpenCL with C++, then compiles OpenCL code by backend compiler, Intel FPGA SDK for OpenCL.

Since the meta-compiler is under development, we applied a hand-compilation in our assumed manner from single OpenACC code, then compiled them by PGI compiler and OpenARC. To evaluate our method, we compared the performance and source code size (lines and characters) with a currently available programming method with CUDA (for GPU) and OpenCL (for FPGA). We examined a synthetic code (not real application) where GPU performs a matrix-matrix multiply, the result is transferred to FPGA, then FPGA performs a CG method by this result matrix. Figure 2 shows the comparison between our OpenACC-only way and CUDA+OpenCL for the code size (a) and (b), and execution time (c). Here, "Others" of (a) includes miscellaneous parts such as initialization, validation function, etc. It is shown that our approach can reduce the number of characters and lines in the source code to approximately 50% and 30%, respectively. However, the performance of both devices are degraded (GPU: 3.4x worse, FPGA: 1.67x worse). We need more performance tuning both on code description and compilers.

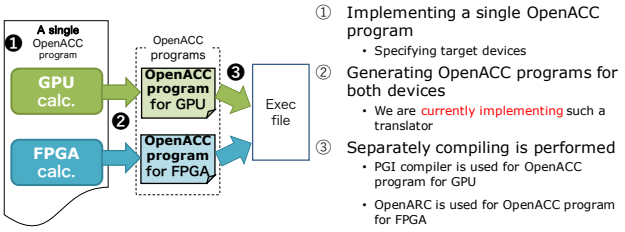

Figure 1: Our approach overview

(a) The number of characters (b) The number of lines

(c) Execution time

Figure 2: Programming cost comparison

As future works, we will complete the meta-compiler, improve the performance especially for FPGA programming by OpenACC, and apply our method to real applications.

ACKNOWLEDGMENTS

This research is partially supported by "Communication-Computation Unified Supercomputing" project under MEXT's "Next Generation Supercomputer R&D" program and Collaborative Research between CCS, R-CCS and ORNL.

REFERENCES

[1] Supercomputers - Center for Computational Science. https://www.ccs.tsukuba.ac.jp/eng/supercomputers/#Cygnus
[2] homepage | OpenACC https://www.openacc.org
[3] Lee, S. et al., OpenACC to FPGA: A Framework for Directive-based High-Performance Reconfigurable Computing, IPDPS2016, pp 544-554
[4] Omni Compiler https://omni-compiler.org/