TSUBAME 共同利用 平成28年度 学術利用 成果報告書

利用課題名 高性能・高生産性を達成する垂直統合型アプリケーションフレームワーク
英文：High Performance, Highly Productive Application Frameworks

丸山直也
Naoya Maruyama

理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
http://mt.aics.riken.jp/

邦文抄録（300 字程度） 本課題では気候モデルを対象として性能と生産性を両立するためのドメイン特化型フレームワークの評価を行う。気候モデルとしては正 20 面体による全球モデルを対象とし、フレームワークとしては GridTools を中心に用いる。 TSUBAME2 の GPU を用いた性能評価を行い、フレームワークによる生産性と性能の評価を実施する。

英文抄録（100 words 程度） / Abstract (100 words) This research is focused on performance portability for climate models. To that end, we investigate using a high-level framework to enable a performance portable and highly performant version of an atmospheric icosahedral model. With GPU-accelerated supercomputers being a strong candidate architecture for exascale, we conduct experiments using large-scale heterogeneous supercomputers such as TSUBAME2.

Keywords:

背景と目的 / Background

Traditionally, climate models are one of the main class of applications that run on HPC systems. Supercomputers are becoming increasingly complex and divergent in architectures. As a consequence, one of the main challenges to advancing climate models, and other scientific applications at large, is to guarantee performance portability over supercomputers that have very different architectures.

概要 / Overview

In this project, our target is to navigate the challenges of achieving performance portability for climate models. More specifically, we target NICAM, an icosahedral atmospheric model chosen as a target application for the current and next-generation supercomputers in Japan (i.e. Oakforest-PACS, Tsubame3, and Post-K).

NICAM codebase is more than 50K lines of code. Hence, maintaining different codebases for optimized for different target architectures is infeasible.

To that end, we consider a high-level framework based on C++ template programming as a pathway for achieving performance portability while maintaining a single code base. The high-level framework, GridTools, provides a robust and performant backend for GPUs (based on CUDA), and other architectures.

結果および考察 / Results

To evaluate the efficiency of GridTools, we use a set of kernels extracted from NICAM's dynamical core; the costliest component of NICAM. Figure 1 shows the execution time of preliminary runs, on a single GPU, of three kernels: diffusion, divdamp, and vi_rhow_solver.

Benchmark	CUDA (Nvidia K20x)		
	Manual	Manual_opt (coal, sh_mem, occu, reg_pres)	GridTools
Diffusion	5.83	0.575 (5.23x OMP=10)	0.61 (4.93)
Divdamp (vgrid40_600m_24km)	35.23	3.15 (4.83x OMP=10)	3.17 (4.80x)
Vi_rhow_Solver (vgrid40_600m_24km)	0.91	0.288 (6.14x OMP=10)	0.311 (5.69x)

Figure 1: Execution time (Seconds)

The GridTools generated code is close in performance, and speedup over OpenMP CPU code, to the manually optimized CUDA code. It is important to note that hand-written optimizations for CUDA code is a complex task and requires significant time.

まとめ、今後の課題 / Summary and Future Plans

Our experimentation with kernels extracted from NICAM's dynamical core show that the high-level framework GridTools can generate code comparable in performance to manually optimized code. One main challenge remains to be porting the entire dynamical core to C++ from the original Fortran code. Another challenge is the introduction of convoluted code, typical of template programming in C++. Finally, a main point of work is to investigate the performance of other backends provided by GridTools, namely OpenMP for Intel's KNL.