jh220036

大規模アプリケーションの高性能な実用的アクセラレータ対応手法

下川辺 隆史
東京大学
shimokawabe@cc.u-tokyo.ac.jp
http://www.cspp.cc.u-tokyo.ac.jp/shimokawabe/

共同研究者

額田 彰（筑波大学）
古村 孝志（東京大学）
羽角 博康（東京大学）
川崎 高雄（東京大学）
朴 泰祐（筑波大学）
高橋 大介（筑波大学）
勢見 達将（筑波大学）
山岸 孝輝（高度情報科学技術研究機構）
三木 洋平（東京大学）
塙 敏博（東京大学）
中島 研吾（東京大学）
星野 哲也（東京大学）
大森 拓郎（東京大学）
畠山 昂（東京大学）
佐久間 大我（東京大学）
Ziheng Yuan（東京大学）

1 研究背景と研究目的

近年、電力と設置面積の制約のもと、最大限に計算性能を高くするために、多くのスパコンで GPU などの演算加速装置が導入されている。今後この傾向はさらに加速すると予想され、将来のスパコンでは、大多数のアプリケーションで高い性能を引き出すために演算加速装置の利用が必須となる。このような中、東京大学情報基盤センターと筑波大学計算科学研究センターと共同で設置する最先端共同 HPC 基盤施設（JCAHPC）では、Oakforest-PACS（OFP）の後継機（OFP-II）を 2024 年 4 月に稼働予定であり、その OFP-II においても演算加速装置を導入する方向で検討を進めている。OFP をはじめとした両センターが運営に関わるスパコンでは、CPU に最適化された多数のアプリケーションが実行されている。OFP-II の稼働開始に向け、これらのアプリケーションを演算加速装置を導入した OFP-II へ効率的に移植していく指針の確立とその手法の構築が必須である。

本研究課題では、スパコンで動作している CPU アプリケーションを演算加速装置を搭載したスパコンへ移植し、その移植の方法を確立することを目的とする。様々な分野の研究者が開発した CPU アプリケーションを移植することを念頭に、演算加速装置で高い性能を達成することを目指しながらも、高性能計算分野でない研究者も難なく扱えるように、演算加速装置専用の開発言語を多用せず、標準的・汎用的手法を用いて移植を実現する。既存コードの原型を残しつつ、最小限の修正で移植を行い、最大限の性能向上を行うことを目指す。また高い可搬性を保持するコード開発を目指す。

2 研究計画

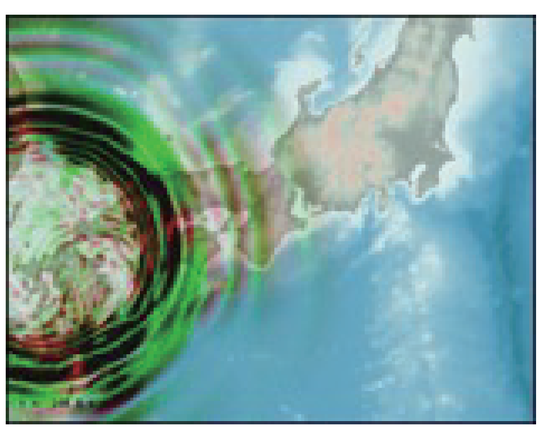
（1）実用的な方法による演算加速装置への移植方法の検証と性能評価

演算加速装置としては NVIDIA A100 GPU を対象とし、実用的な移植方法として、近年成熟してきている OpenMP、OpenACC、Fortran における DO CONCURRENT などの並列化の標準構文、C++ 言語の std::execution による実行ポリシーなど標準化された並列化手法の利用を検討する。ベンチマーク問題としてよく扱われる拡散方程式や我々自身が直接開発に携わりコードやデータ構造に詳しい流体計算を行う格子ボルツマン法（LBM）のコードを上記の方法で移植し、それを通して、コンパイラなどに問題ないかを確認し、上記の方法の可搬性や達成できる性能を検証し、最適化手法の確立を目指す。

（2）実用的な方法による実 CPU アプリケーションの移植とその方法論の確立

(1) で実用的な手法による移植に関する知見が得られたのちに、海洋大循環モデル COCO * と地震波伝播・強震動シミュレーションコード OpenSWPC ** の 2 つスパコンで動作する実 CPU アプリケーションを対象に移植を行い、性能評価を行う。最終的に演算加速装置への移植方法を確立する。また、様々な演算加速装置への性能可搬性を検証するため、JHPCN で提供される NVIDIA GPU に加え、自前で保有している AMD MI100 GPU においても開発したコードを実行し性能評価を行う。複数の演算加速装置において、移植性、達成できる性能、プログラミング上の制限などを総合的に検証する。最終的に、これらの研究開発を通して得た知見を、OFP-II および将来の演算加速装置が導入されたスパコンへの移植方法の指針として広く公開する。

* https://ccsr.aori.u-tokyo.ac.jp/~hasumi/COCO/
** https://github.com/tktmyd/OpenSWPC



OpenSWPC の計算例
https://github.com/tktmyd/OpenSWPC

3 拡散方程式の標準規格による高速化

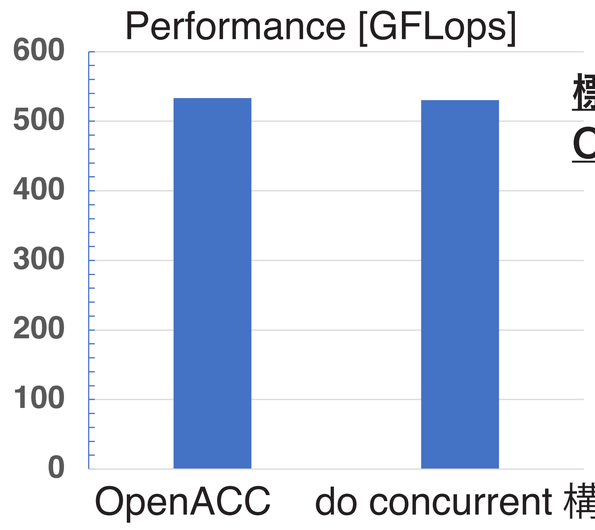
Fortran の標準構文による GPU 移植コードの性能を確認するため、ベンチマーク問題としてよく用いられる拡散方程式（7 点ステンシル）を Fortran 標準規格の do concurrent 構文で書き換え、単一の NVIDIA A100 で性能測定を行う。性能比較のため OpenACC による実装と比較する。

計算条件

計算領域 128^3

do concurrent による拡散方程式の GPU コードの例

```
do concurrent (k = 1 : nz, j = 1 : ny, i = 1 : nx)
    w = -1; e = 1; n = -1; s = 1; b = -1; t = 1;
    if(i == 1)  w = 0
    if(i == nx) e = 0
    if(j == 1)  n = 0
    if(j == ny) s = 0
    if(k == 1)  b = 0
    if(k == nz) t = 0
    fn(i,j,k) = cc * f(i,j,k) + cw * f(i+w,j,k) &
     + ce * f(i+e,j,k) + cs * f(i,j+s,k) + cn * f(i,j+n,k) &
     + cb * f(i,j,k+b) + ct * f(i,j,k+t)
end do
```



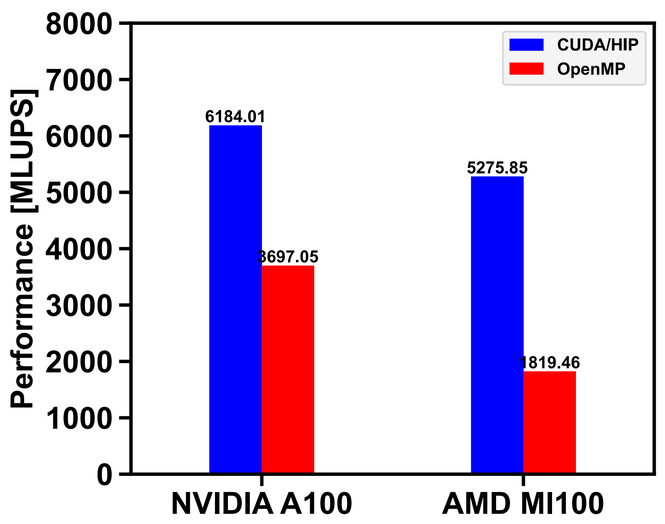
標準規格の do concurrent 構文と OpenACC による性能比較
OpenACC: 533 GFlops
do concurrent 構文 : 530 GFlops

4 LBM の OpenMP による高速化

C++ で記述された格子ボルツマン法（LBM）を用い、NVIDIA A100 および AMD MI100 の環境にて、OpenMP による GPU 化の性能を確認する。性能比較のため、A100 では CUDA、MI100 では HIP による実装と比較する。単一 GPU を用い、一秒間あたりの更新される格子点数 MLUPS (Mega Lattice Update Per Second) で比較する。

計算条件

格子ボルツマン法 D3Q27 モデル
計算領域 512 x 256 x 128



NVIDIA GPU における CUDA と OpenMP、AMD GPU における HIP、OpenMP による性能比較

5 まとめと今後の研究計画

本課題では、スパコンで動作している CPU アプリケーションを演算加速装置を搭載したスパコンへ移植し、その移植方法の確立を目指す。Fortran 標準規格の do concurrent 構文、OpenACC、OpenMP による GPU 化の方法と実行性能を確認した。今後は、実アプリケーションである COCO と OpenSWPC を Fortran 標準規格による GPU 化を進める。標準規格で、期待する性能が達成できない場合は、OpenACC や OpenMP による GPU 化を検討する。