6J-07

OpenACC を用いた GKV ベンチマークの並列化

森下 誠† 大島 聡史‡ 片桐 孝洋‡ 永井 亨‡

名古屋大学 工学部電気電子・情報工学科† 名古屋大学 情報基盤センター‡

1. はじめに

次世代のエネルギー源として磁場閉じ込め核融合が注目されている．これは，およそ 1 億度の燃料プラズマを強力な磁場で閉じ込めて核融合反応を生じさせるものであるが，その際に引き起こされるプラズマの乱流によって閉じ込め性能が左右される．この乱流現象を解明するために核融合科学研究所ではプラズマ乱流解析コード GKV (GyroKinetic Vlasov code) [1]が開発された．GKV は，スーパーコンピュータ「京」を用いた大規模シミュレーションにより，電子/イオンスケール乱流の両者の相互作用を発見した研究[2]などに活用されている．そのため，GKV の高速化は当分野の研究に大きく貢献することが期待される．

本研究では，GKV の高速化に向けて，GKV のカーネル部分を抜き出して作成した GKV ベンチマークを並列化した．OpenMP によるマルチコア CPU 向けの並列化は行われていたが，GPU 向けの並列化は行われていなかったため，OpenACC による GPU 向けの並列化を行った．

2. GKV ベンチマーク

GKV ベンチマークは，GKV をもとに名古屋大学の片桐・渡邊らが開発したものである．

GKV ではシミュレーションの支配方程式として，運動論的ブラソフ・ポアソン・アンペール方程式系を解いている．ブラソフ方程式に現れる空間微分の数値計算手法として，(x,y)座標については FFT を用いたスペクトル法，($z, v_{||}, \mu$)座標については有限差分法を用いている．また，時間発展には Runge-Kutta-Gill 法を適用してプラズマ粒子の分布関数を求める．得られた分布関数を数値積分してプラズマの密度を求め，ポアソン・アンペール方程式により静電・ベクトルポテンシャルを計算する．

これら GKV の主要な処理を抽出しベンチマークとしてまとめたものが GKV ベンチマークである．

表 1 に GKV ベンチマークの概要を示す．

表 1 GKV ベンチマークの概要

kernel1_fft	FFT と MPI_Alltoall に関するベンチマーク
kernel2_intgrl	速度空間積分に用いる台形則の演算およびリダクションに関するベンチマーク
kernel3_diff45	4 次元および 5 次元の有限差分法の演算に関するベンチマーク
kernle4_diff123	1 次元，2 次元および 3 次元の有限差分法に関する演算およびリダクションに関するベンチマーク

本研究では 1 プロセス実行の kernel2，kernel3，kernel4 の 3 つを取り扱った．

3. 実装

OpenACC による基本的な並列化は，OpenMP と同様にループ制御構文 (Fortran では do 文) の直前にディレクティブを挿入することで行われる．GKV ベンチマークは既に OpenMP により並列化されているため，基本的にはその部分を置き換えることで OpenACC 化が可能である．さらに OpenACC ではハードウェア構成を考慮した並列化を行うことができ，外側ループの並列性を gang，最も内側のループの並列性を vector と指定することで高い性能が出やすくなる．

例として kernel2 のプログラムにおける並列化コードの一部を図 1 に示す．図 1 では，並列計算の方法を指定する loop 指示文，強制的に並列実行を指定する independent 指示節，多重ループをまとめて並列化する collapse(n)指示節，そして並列性を gang/vector で与えている．

kernel3 も同様に，主要なループ処理部の指示文を置き換え・追加することで OpenACC 化を行った．

Parallelization of GKV benchmark using OpenACC

† Makoto Morishita, Electrical and Electronic Engineering and Information Engineering, School of engineering, Nagoya University

‡ Takahiro Katagiri, Satoshi Ohshima, Toru Nagai, information Technology Center, Nagoya University

```
!$OMP do collapse(3) reduction(+:ww)
!$acc loop independent gang collapse(3)
do im = 0, nm
  do iv = 1, 2*nv
    do iz = -nz, nz-1
      do my = 0, ny
!$acc loop independent vector
        do mx = -nx, nx
          ww(mx,my,iz) = ww(mx,my,iz) &
            +wf(mx,my,iz,iv,im) &
             *vp(iz,im)*dvp(iz)*cef
        end do
      end do
    end do
  end do
end do
```

図 1 kernel2_intgrl_v03.f90 の一部

kernel4 は OpenMP 並列化されていた箇所のサブルーチン呼び出しの階層が深く，単純にディレクティブ置き換えるだけではメモリ不足による実行時エラーが発生する． そこで kernel4 では，サブルーチン呼び出しを辿り，全体の実行時間に対して特に時間がかかる部分のみ並列化を行った．また，インライン展開により，並列化部分の計算を GPU でまとめて行わせることで更なる高速化を図った．

4. 性能評価結果

名古屋大学情報基盤センターに設置されている GPU サーバ sx40 にて性能評価を行った．実行環境とコンパイルオプションを以下に示す．

実行環境

- CPU : Intel Xeon Gold 5122 (3.6 GHz, 4 コア 8 スレッド, 460.8 GFLOPS) ×2 ソケット
- GPU : NVIDIA Tesla V100 SXM2 (倍精度演算性能 7.8 TFLOPS) ×4 枚
- メモリ : DDR4-2666 (384 GB)
- ノード数 : 1
- コンパイラ:PGI compiler 19.4

コンパイルオプション

- OpenMP :-fast -mp -tp=skylake
- OpenACC :-fast -mcmodel=medium -acc -Minfo=accel -ta=tesla,cc70 -tp=skylake

逐次，CPU によるスレッド並列，GPU 並列の 3 つの形態で実行し性能を比較した．結果を図 2 に示す．いずれも 1 プロセス実行であり，GPU 並列も 1GPU による実行である．

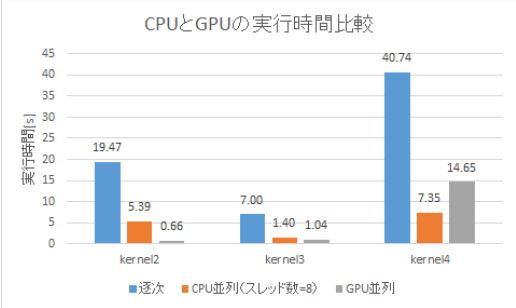

図 2 CPU と GPU の実行時間比較グラフ

CPU によるスレッド並列では，スレッド数=1, 2, 4, 8 それぞれの場合の実行時間を計測し，一番早かったスレッド数=8 のときの実行時間を比較対象としてグラフに載せている．

逐次と GPU 並列の実行時間と比較すると，kernel2 で約 29.5 倍，kernel3 で約 6.7 倍，kernel4 で約 2.8 倍高速化された．kernel2 および kernel3 では CPU 並列と比べても約 8.1 倍および約 1.3 倍高速化されており，GPU により多く性能向上することが確認できた．

5. まとめ

OpenACC を適用することにより GKV ベンチマークの並列化を行った． kernel4 のようなサブルーチン呼び出しが複雑なプログラムを GPU 化する際は，サブルーチンの切り分けによる並列化領域の考慮やデータ転送の最適化等の工夫が必要であるといえる．

今後の発展としては，GKV 本体に OpenACC を適用することが考えられる．その際は，GPU 搭載のスーパーコンピュータを実行環境とし，マルチ GPU 実行できるようにコードを書き換え，OpenMP/MPI ハイブリッド並列の場合の実行時間と比較し，性能を評価することが必要である．

謝辞

本研究は JSPS 科研費 JP18K19782 の助成を受けたものです．

参考文献

[1] T-H. Watanabe, and H. Sugama, “Velocity-space structures of distribution function in toroidal ion temperature gradient turbulence”, Nucl. Fusion 46, 24(2006)

[2] S. Maeyama, T-H. Watanabe, Y. Idomura, M. Nakata, M. Nunami, A. Ishizawa, “improved strong scaling of a spectral/finite difference gyrokinetic code for multi-scale plasma turbulence”, Parallel Comput. 49(2015) 1-12