SCALEのGPU対応の現状と課題

理化学研究所
西澤誠也

SCALE
Scalable Computing for Advanced Library and Environment

はじめに

- 海外では気象・気候モデルの GPU対応が進んでいる

- 東大スパコンが2024年度からGPU搭載機が予定されているなるなど、日本の研究用スパコンもGPU化の流れが加速
  - SCALEもGPU化が必須であると判断

実装方針

- 最新のコードを CPU でも GPU でも動かしたい
  とあるバージョンのGPU版リリースとか完全GPU移行は難しい
  - プロダクト実験と開発が密にリンク
  - 開発リソースを集約

→ CPU,GPUで共存できる実装を選択
(OpenACC vs OpenMP vs Fortran concurrent)

- 現時点では性能面(＋α)を考慮して OpenACC を選択
  実質NVIDIA GPUのみになってしまうので、将来的には OpenMP への移行も視野

SCALE GPU対応スケジュール

- 2022年度中にミニアプリをGPU化
  - 練習＋課題の洗い出し
- 2023年度中にSCALE全体をGPU化

ミニアプリ

- Warm-bubble 雷実験
  - 格子数: 60 x 40 x 40 (x,y,z)
    (格子数が小さく、あまりGPU向きではないことに注意)
- 主要コンポーネント
  - 力学: HE-VI
  - SGS乱流: 1次 (Smagorinsky) ※鉛直インプリシット
  - 雲微物理: 1-moment bulk (Tomita08)
  - 雷: Sato2019

- 行数: 約8.4万行
- ループ数: 約1.5千ループ

- 東大のGPUミニキャンプ(2022/12/12-19)中に大部分を実装
  - 参加者: 山浦, 足立, 河合, Kong, 佐藤, 西澤

- 雷スキームの一部以外 OpenACC化
  - 挿入ディレクティブ数: 約3千
- 初期値・I/O時、雷の一部計算時 のみデータ転送
- CUDA-aware MPI通信

- 結果の妥当性検証
  - CPU計算で複数の計算機、コンパイラで実行した結果のばらつきの範囲内にあるかどうかによって検証
  - 上記の確認をして、OK,NGという結果を出すチェックプログラムを用意

- CPU, GPUの比較 (最適化前、雷計算OpenACC化前)
  - CPU: Intel Xeon Platinum 8360Y (Wisteria-A)
    - 1 node, 6 process x 12 thread (=72 core); 理論ピーク 5.53 TFLOPS, 409.6 GiB/s
  - GPU: NVIDIA A100 (Wisteria-A)
    - 1 node, 6 process x 1 thread, 6 GPU; 理論ピーク 58.5 TFLOPS, 9,330 GiB/s

	CPU (72 core) [s]	GPU (6 GPU) [s]
Main Loop	17.3	23.3
Dynamics	2.1	5.3
Turbulence	0.2	1.9
Microphysics	0.3	1.0
Lightning	13.0	13.3 (※)

※ CPU計算 (CPU-GPU間のデータ転送を含む)

チューニング事例紹介

- カーネル毎経過時間 Top5 (1GPU 計算)
  NVCOMPILER_ACC_TIME=1 で計測

1. atmos_dyn_tstep_short_fvm_hevi: 3.33 s
   - 鉛直陰解法 3 重対角行列ソルバー
   - 全 dynamics 中 68%
2. atmos_phy_tb_calc_flux_phi: 1.87 s
   - 鉛直陰解法 3 重対角行列ソルバー
   - 全 turbulence 中 73%

最内ループに依存関係があり、
ベクトル化できない(のが原因と思っていた)
その後、別の要因が原因と判明

3. atmos_phy_mp_driver_calc_tendency: 1.15 s
   - 降水落下計算
   - 全 microphysics 中 61%

レジスタを大量に使用するため、
occupancy が低い(のが原因と思っていた)
その後、別の要因が原因と判明

4. file_read_var_realdp_3d: 0.24 s
   - 初期値データGPU転送
5. packwe_3d: 0.21 s
   - 通信パッキング

多数 (1万回以上) の kernel 起動。一回は短いが、塵積。
非同期 kernel (async) の利用で半分に

- 3重対角行列のソルバーの最適化
  - CPU版では、Thomas alogism を採用
    - 最内にキャッシュサイズ分の長さをとり(i方向からもってくる)、ベクトル化
      - 行列係数の計算時に転置
      - コードが複雑になってしまい、OpenACC ではj方向しか並列化できず
  - Cyclic Reduction 法、Parallel CR 法、cuda ライブラリ利用を新たに実装
    - cuda ライブラリ (cusparse) の場合は、3次元の行列係数配列を用意する必要があり、ループ (カーネル) を分割 (メモリ転送量増加)

	TA	CR	PCR	cuda lib	CPU (参考)
Main Loop	22.9	22.9	23.1	20.0	17.3
Dynamics	5.3	5.3	5.3	3.8	2.1
Turbulence	1.9	1.9	2.0	0.2	0.2

アルゴリズムの問題ではなく、次で述べるkernel内関数呼び出しが問題であることが判明

判明した課題 カーネル内関数呼び出し

- カーネル内から呼ばれた関数の中に作業配列 (自動配列)があると遅くなる
  - global memory アクセスが大量に発生 (atomic命令)
    - automatic 変数は stack ではなく heap に確保される！！(malloc が呼ばれる。giant lock が必要)
- 対処方法候補
  - 内部ループに依存関係が無い場合
    1. ループを融合し、一つの大きなループにすることで、作業配列をなくす(スカラーにする)
       - 富岳等の最適化とコンフリする
  - 内部ループに依存関係がある場合
    1. 作業配列を、呼び出し元で準備し、部分配列を引数で渡す
       - コードの煩雑化・利便性低下、作業配列の3次元化
    2. カーネル内から関数呼び出しをやめる、3次元ループをkernel化 (インライン展開)
    3. コンパイラのインライン展開機能を利用
       - 制限がたくさんある (e.g., 長さの上限をコンパイル時に設定)

  ➢プリプロセッサの利用 (コードが汚くなるのであまりやりたくはないが。。。)

  ➢コンパオルオプションで stack (alloca) を使えるように Nvidia に要望中

  ➢将来的に shared memory を使えるようになるとうれしい (願望)

1. 力学HE-VI の行列ソルバーを手動インライン展開
   - cusparse は、3重ループをループ分割 (係数行列計算と行列計算、後処理計算をそれぞれ別カーネルに)

TA	CR	PCR	cuda lib	展開前 (TA)
0.029	0.028	0.031	0.046	1.57

2. 降水落下計算関係のサブルーチンで、ループを融合し、中間配列をスカラー変数に
   - 0.274 → 0.010 s

- 雷計算のOpenACC化 (全体18秒)
  - 中和(落雷)計算部分 (6.3秒)
    - CPUに残すことに (大部分が逐次処理)
    - CPU-GPU間通信
  - 電場計算 (ポアソンソルバー) (11.3秒)
    - Bi-CGSTAB
      - 収束計算
        - 毎回袖通信が必要 (6秒)
      - 前処理 (対称 Gauss-Seidel)
        - カラーリングで並列化 (繰り返し回数増加)
          - OpenMP → y方向のみ並列化 (繰返し回数 平均 64回)
          - OpenACC → x, y方向を並列化 (繰返し回数 平均79回)

- CPU, GPU の比較 (最適化後)
  - 加えて、格子数が多い場合も計測 (雷は off) (メモリ40GB中34.4GB使用)

	CPU (72 core) [s]	GPU (6 GPU) [s] (最適化前)	GPU (6 GPU) [s] (最適化後)	CPU (72 core) [s]	GPU (8 GPU) [s] (比率)
問題サイズ	60 x 40 x 40 (50 step)			480 x 480 x 256 (100 step)	
Main Loop	17.3	23.3	23.2	1269.7	71.9 (17.6)
Dynamics	2.1	5.3	3.6	1008.8	33.7 (29.9)
Turbulence	0.2	1.9	0.2	100.4	3.2 (31.7)
Microphysics	0.3	1.0	0.1	89.3	15.0 (5.9)
Lightning	13.0	13.3 (※)	18.0		

※ CPU計算 (CPU-GPU間のデータ転送を含む)

まとめ

- 現状: ミニアプリをOpenACC 化
  - 判明した課題
    - カーネル内からの関数呼び出し
- スケジュール
  - 2023年度中に全モデルのフルGPU化を目指す
  - ポテンシャルボトルネック
    - 収束計算 (地表面フラックス、温度)