jh260061

コード生成 AI を活用した 大規模 MPI+OpenMP コードの段階的 GPU 化プロセス

課題代表者：星野 哲也（名古屋大学情報基盤センター）
共同研究者：堀田 英之，加藤 雅也，坪木 和久（名古屋大学宇宙地球科学研究所），片桐 孝洋，椋木 大地（名古屋大学情報基盤センター）

研究背景

- スパコンのGPU化
  - AI需要の拡大により，計算環境のGPU化が加速
  - 例：富岳NEXT，Miyabi，不老・弐
- アプリGPU化人手不足
  - 大規模アプリのGPU移植は年単位のプロジェクト
  - CPU版+GPU版で保守・管理コスト倍増
  - 専門知識が必要で属人化傾向
- コード生成AIによるプログラムGPU化
  - ベンチマークカーネル単位で多数の成功例

    →大規模実アプリで実用するには？
- AI支援による大規模実アプリGPU化の課題
  - アプリGPU化は単なるコード変換ではなく巨大なワークフロー（解析・設計・実装・検証・評価）
  - コンテキスト長（AIの記憶容量）制約
  - セッションを跨いだタスク一貫性
  - AI実施内容の把握容易性・妥当性検証

研究手法

- 対象実アプリケーション
  - 雲解像モデルCReSS (http://www.rain.hyarc.nagoya-u.ac.jp/CReSS/fcst_exp.html)
  - 輻射磁気流体コードR2D2 (https://hottahd.github.io/R2D2-manual/)
- GPU化プロセスの適切な粒度（AI処理可能・人間認知可能）への段階的タスク分解
  - コード解析・プロファイリング・メタ情報付与
  - OpenMPループ単位のベンチマーク化
  - ベンチマークのGPU化
  - GPU化ループの本体への統合
  - 全体最適化
- 各段階タスクの仕様書記述による外化
  - 上記段階ごとに実装方法，検証方法など仕様書化
  - セッションを跨ぐAIタスクを安定化
- Human-in-the-loopワークフロー
  - シミュレーションの妥当性，AIエージェントの暴走検知，仕様書の適宜修正は人間が実施

AI 支援による CReSS の GPU 化

Phase	AIの役割	人間の役割
コード解析・プロファイリング	解析，コードへのメタ情報付与，レポート出力	以降の方針決定，後続Phase仕様書化
CPUベンチマーク化	カーネルベンチ作成，実行時ダンプデータ取得	安定動作の確認，結果確認
ベンチマークGPU化	カーネルごとにOpenACCコード生成	GPU化に伴う数値誤差の妥当性検証
本体への統合	#ifdef 付きで本体へOpenACCカーネル統合	シミュレーション全体の妥当性検証
最適化	（現時点では）カーネル単位の最適化	最適化に伴う数値誤差の妥当性検証

- CReSS GPU化プロジェクト概要
  - CReSSは25万行以上のFortran + MPI + OpenMPからなり，GPU化対象となるOpenMPループが387ある大規模アプリ
  - 実際にGPU化したのは，2022 年 9 月の西太平洋における台風シミュレーションを入力とした際に実行される167カーネル
  - 899×899×128 の格子（約 1 億格子点）
  - Claude Opus 4.5 ~ 4.6 を用いて，試行錯誤含む期間としては3ヶ月程（AI稼働の実時間は100時間超）で上記の 1 パスのGPU化達成
  - 性能はOpenACC + Unified Memoryとしては全般的に妥当（図 1 ）

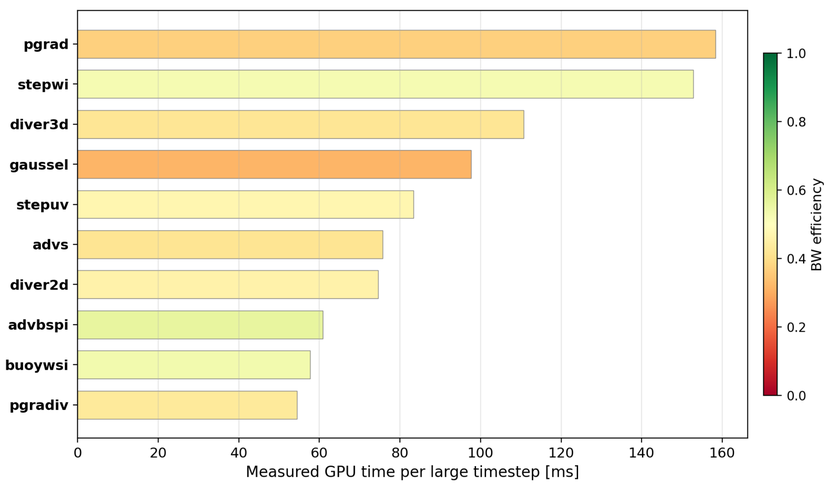

図 1：実行時間 Top 10 カーネルの実行時間とメモリバンド幅効率

- 得られた知見
  - AIは「実時間コスト」をあまり考慮してくれず，データダンプのための実行（1時間程かかる）を何度も繰り返した
  - 各カーネルの入出力ダンプで500 GB程のデータ出力があり，ストレージに高負荷
  - Miyabiのインタラクティブ（2時間制限）で実行のため，頻繁なセッション終了に対応するための仕様書化は必須
  - ロールバックできるようgitは必須

今後の課題

- R2D2への適用
  - アプリに寄らない部分のGPU化プロセス共通化
- カーネル間依存を伴う最適化
  - 現時点では，AIによるコード生成処理をカーネル内に閉じ込めるため，OpenACCのasync節適用やMPIの通信隠蔽など未適応
  - 検証可能な適用手法の確立