Vol.2021-HPC-179 No.7
2021/5/14

都市気象シミュレーション City-LES における詳細モデル計算の GPU 高速化

菊池 航平[1]　渡邉 孔英[1]　朴 泰祐[2,1]
佐藤 拓人[2]　ドアングアンヴァン[2]　日下博幸[2]

概要：近年の高性能計算システムには GPU (Graphics Processing Unit) などの演算加速装置を搭載し，消費電力当たりの計算能力を向上させることが多くなっている．高いメモリバンド幅と多数の計算コアを持つ GPU はその高い数値演算性能からスーパーコンピュータ向けのアプリケーションの実行に適しており, GPU プログラミングによってアプリケーション高速化が期待できる．そのため筑波大学計算科学研究センターで開発されている都市気象シミュレーションコード City-LES は GPU を活用した高速なシミュレーションのため OpenACC などによる GPU 化が進められており，これまでに建物を考慮しない条件下の実行について高速化が達成されている．本研究では建物や植生を含む詳細モデルを適用した場合のシミュレーションの GPU 化を目的として，詳細モデル計算の実行パス上の関数部の 1 つである移流項計算について OpenACC による GPU 化を行った．実装の結果，2CPU 実行時と比較して 2GPU 実行時の移流項計算の実行性能は約 25.9 倍に高速化された．また移流項計算の高速化により，アプリケーション全体は 2CPU 実行時と比較して 2GPU 実行時の実行性能は約 1.15 倍に高速化されることが見込まれる．

キーワード：アプリケーション GPU 化，LES (Large Eddy Simulation), City-LES, OpenACC

1. はじめに

近年の高性能計算システムでは，消費電力あたりの計算能力を高めるために GPU (Graphics Processing Unit) などの演算加速装置を搭載することが多くなっている[1]．高いメモリバンド幅と多数の計算コアを持つ GPU は大規模な並列計算に向いており，CPU と比較して数値演算性能が非常に高いため，スーパーコンピュータ向けのアプリケーションを実行することに適している. そのため CPU コードで書かれたアプリケーションを GPU 上で実行できるようにすることはアプリケーション高速化への有力な選択肢であり，計算やシミュレーションをより短時間，高精度に実行できるようになることで科学技術計算分野の研究がより進展することが期待できる．

また筑波大学計算科学研究センターでは，池田・日下らによって LES (Large Eddy Simulation)と呼ばれる流体の数値シミュレーション手法を用いた都市気象シミュレーションアプリケーション City-LES が開発されている. LES は空間の格子分割に対して格子間隔以上の乱流を直接シミュレーションし，それ以下の大きさの乱流はパラメータを与えて計算に取り入れるといった手法であり，高精度の計算が可能という特徴がある. また City-LES は都市の建物や街路樹の一本一本を三次元的に考慮するモデルを採用しており，街区内放射計算をラジオシティ法で高解像度に計算し，街区内の熱環境計算を詳細に行えるという特徴を持っている．

City-LES は Fortran で記述された CPU アプリケーションであり, MPI と OpenMP [3]によるハイブリッド並列化がされているため並列計算機による高速実行が可能となっている．その一方で，City-LES の扱う問題にはステンシル計算などの並列性の高い計算が多く，それらは GPU の持つ数値演算性能と高いメモリバンド幅が生かされる計算である．このことから，City-LES は GPU 上での実行で大きな性能向上が見込まれるため GPU 化が進められている．

先行研究[4][5][6]では，City-LES の CPU 実行時の重い処理部分から順次 GPU 化を行いそれぞれの関数で高い性能向上を得るとともに，実行パスのすべての関数をフル GPU 化することでデータ転送を削減しアプリケーション全体の性能向上を達成した．しかし GPU 化された関数は建物なしの入力の場合に通る関数のみであり，City-LES の特色である都市の建物や樹木を含む，詳細モデルを用いる場合にはじめて通る関数はまだ GPU 化されていない．実用的なシミュレーションを GPU 上で実行するためには詳細モデルを適用した場合の実行パスを GPU 化する必要がある．

本研究の目的は City-LES の詳細モデル計算の GPU 化を行い，詳細モデルのシミュレーションにおいて通過する関数部の 1 つである移流項計算の GPU 実装とその性能評価を行うことである．本稿は以下のように構成される．2 章では関連研究について，第 3 章では先行研究についてそれぞれ述べる．4 章では City-LES で用いられている計算モデルについて説明する．5 章では GPU 化に必要な GPU プログラミングの手法について述べる．6 章では移流項計算関数の GPU 化について述べる．7 章では本実装の性能評価について述べる，最後に 8 章でまとめと今後の課題について述べる．

2. 関連研究

気象アプリケーションを GPU で高速化した例として，東京工業大学と気象庁による同庁の開発する気象モデル

1 筑波大学 情報理工学位プログラム
2 筑波大学 計算科学研究センター

ⓒ2021 Information Processing Society of Japan

ASUCA のフル GPU 化の研究がある[7]．この研究でフル GPU 化された ASUCA は TSUBAME2.0[8]上で実行され，単一 GPU 実行では単精度で 49.1GFLOPS の実行性能を，3990GPU を利用した複数 GPU 実行では単精度で 145TFLOPS の実行性能を達成している．

また LES を用いた気象シミュレーションを GPU 高速化した例としては，小野寺らの研究がある[9]．この研究では大規模な並列計算が可能な LES アプリケーションを開発し，都心部の 10km 四方を 1m 格子間隔に解像したモデルのシミュレーションを行っている．TSUBAME2.0[8]上の性能評価では，単体 GPU 実行では単精度で 198GFLOPS の実行性能を達成し，複数 GPU 実行では単精度で，768GPU で 115TFLOPS を，1000GPU で 149TFLOPS の実行性能を達成している．またこのアプリケーションを用いて実際の東京都心部の建物データに対してシミュレーションを行っている．

これらの研究から，本研究の対象とする City-LES も同様に GPU による高速化が可能であると考えられる．

3. 先行研究

これまで辻らによって当時最新版の City-LES を対象に GPU 化が行われている．

[4]では CPU 実行の処理時間の割合が大きく全体の高速化への効果が大きい関数から GPU 実装を行い，合計で CPU 実行の 70%以上を占める関数群に対してそれぞれ GPU 化した．また MPI 袖通信の最適化としてパッキングの実装を行った．性能評価では，各関数部について 2CPU に対する 2GPU 実行で 7.4 倍から 16.6 倍の高速化を達成しており，全体の実行性能で CPU-GPU メモリ間のデータ転送時間がボトルネックになっていることを課題として挙げている．これは GPU 化された関数との間に CPU 関数がある場合，GPU 側で計算されたデータを CPU 側に戻す処理が逐次必要になり，データ転送が蓄積してしまうことによる．

[5]では[4]の課題を受け，さらに関数の GPU 化を進めて一連の処理を GPU 内のデータのみで計算を行えるようにすることで CPU-GPU メモリ間の転送時間を削減した．これにより City-LES 全体の性能評価において，2CPU に対する 2GPU 実行で 2.97 倍，4CPU に対する 4GPU 実行で 2.78 倍の高速化を達成している．また同様に時間発展計算全体がフル GPU 化されれば，計算の高速化とデータ転送時間の削減により 2CPU に対する 2GPU 実行で 3.8 倍，4CPU に対する 4GPU 実行で 3.5 倍の性能が達成されると予測している．

[6]では[5]に引き続き，時間発展計算のフル GPU 化のため残りすべての関数の GPU 化を行っている．これにより CPU-GPU メモリ間のデータ転送がさらに削減され，2CPU 実行に対する 2GPU 実行では 6.2 倍の性能を達成し，Strong Scaling では CPU に対して GPU でノード当たり 10 倍から 4.7 倍の性能を，Weak Scaling では 1 ノードあたり 9 倍以上の性能を達成している．

表 1　City-LES の概要 [2]

基礎方程式	非静力ブシネスク近似方程式系
座標系と離散化	直交座標系，Arakawa-C，有限差分法
時間スキーム	3 段階 Runge-Kutta 法 (Wicker and Skamarock 2002)
空間スキーム	二次，四次，六次精度中央差分 三次，五次精度風上差分
SGS モデル	TKE-1 方程式モデル (Deadoroff 1980), Smagorinsky モデル
数値解法	SMAC 法
ポアソン方程式解法	マルチグリッド前処理付き Orthomin(m)法 共役残差法
単波放射	近藤 (1994)，Dudhia simple (Dudhia 1989), 放射固定
長波放射	近藤 (1994)，RRTM (Mlawer et rl, 1997), 放射固定
街区内放射	ラジオシティ法
地表面モデル	Mascart (1995)，フラックス固定
雲物理	warm rain
コード	Fortran90
並列化	MPI＋OpenMP

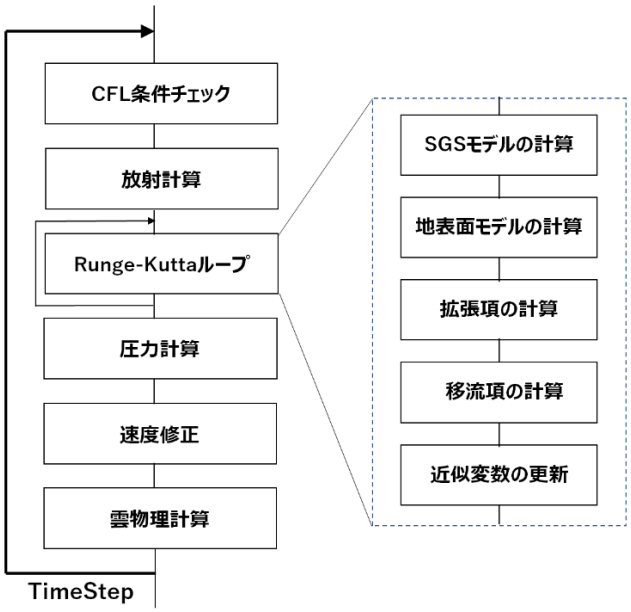

図 1　City-LES の実行過程

4. City-LES の計算モデル

City-LES の概要を表 1 に，その実行過程を図 1 に示す．City-LES は格子分割された都市モデル上で地表面の気温や建物間の乱流についての時間発展計算を行う．土地モデル

の問題格子にはビルや街路樹といった建物の種類や地表面の高さなどのパラメータを与えて都市を描写する．

City-LES は Fortran で記述される CPU アプリケーションで MPI と OpenMP によるハイブリッド並列化が行われている．問題領域は MPI プロセスによって地表面に水平な X-Y 方向が分割され，OpenMP スレッドによって地表面に鉛直な Z 方向に分割されて並列実行される．

先行研究では入力にビルや樹木などの建物を配置しない土地モデルを用いて，その実行パスにある関数のフル GPU 化が行われた．一方， 街区内の放射計算など建物モデルを適用することで新たに実行される関数が存在することがわかっており，その実行パスのみに現れる関数は GPU 化がされていない．本研究ではそのような関数を対象に GPU 化を行う．

5. GPU プログラミング

5.1 CUDA

CUDA[10]は NVIDIA 社が提供している同社製 GPU 向けのプログラミング環境である．C/C++言語からの利用のほか，PGI コンパイラが提供している拡張言語の CUDA Fortran[11]を使用することで Fortran の GPU プログラミングも可能となっている．本研究で使用する City-LES も Fortran で記述されている．

CUDA プログラムでは GPU 上で実行されるプログラムはカーネルと呼ばれ，実行は CPU 側の main プログラムが GPU を操作しカーネルを実行させる形で行われる．

CUDA は NVIDIA 社のハードウェアに最適化されており，ハードウェアの性能を最大限発揮するよう GPU 化できれば高い性能が期待できる一方で，その性能を引き出すためには独自の機能や仕様を十分に理解する必要がある．

5.2 OpenACC

OpenACC[12]はディレクティブと呼ばれるコンパイラへの指示文を用いて GPU などの演算加速装置を簡単に利用することができるように作られた指示文ベースのプログラミングモデルであり，GPU 向けの利用が広がっている．共有メモリ計算機の並列化で用いられる OpenMP のように，既存の CPU コードの要所にディレクティブを挿入することでデータ転送や並列化領域を指定でき，コンパイラが演算加速装置用のコードを生成する．

ディレクティブベースプログラミングの性質上，モジュールや関数単位で段階的に GPU 化を進められるため生産性・移植性が高い．また，OpenACC のコンパイルオプションを切ることで元の CPU コードを保持することができる．これらの利点を鑑みて，本研究においては OpenACC を用いた実装を行った．

5.2.1 OpenACC プログラム例

OpenACC プログラムの例をリスト 1 に示す．この例ではサイズ num の配列 arr を入力として受け取りその総和

リスト 1　配列の総和を求める関数の OpenACC 適用例

```
1  subroutine  aggarr(arr, num, sum)
2    implicit none
3    integer, dimension(num), intent(in)     :: arr
4    integer,                 intent(in)     :: num
5    integer,                 intent(in out) :: sum
6
7    integer :: i
8
9    !$acc data copy(sum) copyin(arr, num)
10
11   !$acc parallel
12     !$acc loop reduction(+:sum)
13     do i = 0, num-1
14       sum = sum + arr(i)
15     end do
16   !$acc end parallel
17
18   !$acc end data
19
20 end subroutine aggarr
```

sum を求めるような関数 addarr に OpenACC を適用して GPU 化している．プログラムリストのうち，!$acc で始まる行が OpenACC ディレクティブである．

9 行目と 18 行目の data ディレクティブは CPU-GPU 間のデータ転送を指示する．各変数に対して，CPU から GPU に送られる copyin，GPU から CPU に送られる copyout，その両方を実行する copy を指示できる．

11 行目の parallel ディレクティブは GPU で実行される並列化領域を指示する．この例では 11 行目から 16 行目が指定している．

12 行目の loop ディレクティブはループの並列化を指示し，この例では 13 行目の do ループをスレッドによって並列に実行することを指示している．また reduction 句で変数 sum の総和計算に対してリダクション処理を行うことを指示している．

その他の代表的な OpenACC ディレクティブに kernels ディレクティブがある．parallel ディレクティブと同様に並列化領域の指定ができるが，loop ディレクティブなどによる具体的な並列化指示を必要とせず，コンパイラに並列化を委ねることができる．より少ないディレクティブ数で実装できるため生産性が高く，細かい最適化が可能な parallel ディレクティブとの使い分けが有効である．

6. 移流項計算の GPU 化

6.1 概要

本研究で GPU 化の対象とする関数は，建物を考慮しない場合には実行されず詳細モデル計算の実行パスではじめて動作する関数である．そのような関数は，街区内の放射計算と，高次精度の差分法を用いた移流項計算がある．

放射計算では建物がある場合，地表面と同様に建物の表

ⓒ2021 Information Processing Society of Japan

表 2　実行環境

CPU	Intel Xeon E5-2660 v4 (14 cores) x2
GPU	NVIDIA Tesla V100 PCIe x2
ホスト OS	CentOS 7.3
コンパイラ	PGI 17.10
ライブラリ	OpenMPI 3.0.0, CUDA 9.1.85

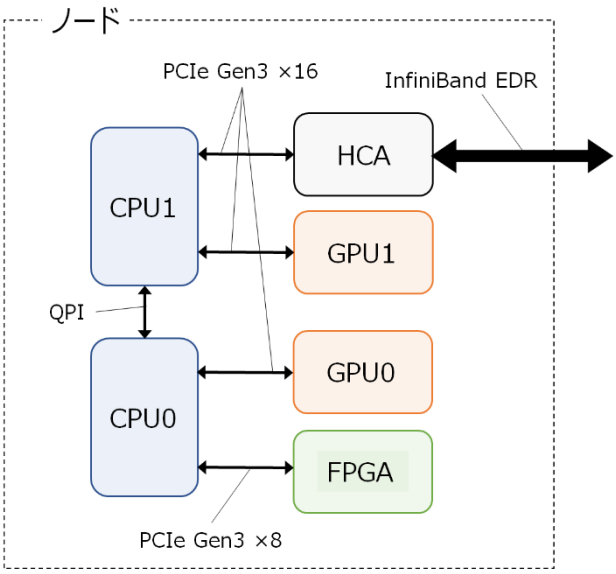

図 2　PPX のノード構成

面等を考慮する必要がある．また移流項計算ではステンシル計算が行われるが，その差分の選択と計算手法について，建物なしの場合二次中央差分を用いていた一方，詳細モデル計算ではそれより高次精度の差分法を用いることが多い．そのためこの差分計算に対応する必要がある．

どちらも詳細モデル計算のフル GPU 化による CPU-GPU 間のデータ転送削減の観点で GPU 化が求められるが，移流項計算がステンシル計算を行っており並列性が高く，街区内の放射計算の並列性が高くないと考えられたため，今回移流項計算関数を対象とした．

6.2 CPU を用いた予備調査

アムダールの法則より，元々の処理時間の割合が小さい関数を高速化しても，全体から見た高速化率に与える影響は小さい．そのため高速化にあたっては元々の処理時間の把握が有効である．よって予備調査として詳細モデル計算の CPU 実行時間の測定を行い，移流項計算関数の実行時間の把握を行う．

実行は筑波大学計算科学研究センターが運用する PPX (Pre-PACS-X) の 1 ノード上で行った．実行環境を表 2 に，PPX のノード構成を図 2 に示す．また，表 3 に今回の実行パラメータを示す．ここで MPI プロセスは計算領域の X-Y 方向を分割するが，本実験では 2 プロセスで実行されるため X 方向を分割している．また建物は計算安定化のため，境界付近の 10 格子には配置していない．

表 3　City-LES の実行パラメータ

問題サイズ	256×256×256 1 プロセスあたり 128×256×256
MPI プロセス数	2
解像度	1 格子あたり 5[m]×5[m]×5[m]
時間発展数	200
境界条件	周期境界条件
数値解法	SMAC 法 (Runge-Kutta3 回目のみ圧力補正)
ポアソン方程式解法	マルチグリッド前処理付き Orthomin(m)法
時間スキーム	3 段階 Runge-Kutta 法
空間スキーム	五次精度風上差分
建物	高さ 20[m]，幅 20[m]，建物列の幅 50[m] コンクリート
土地	アスファルト

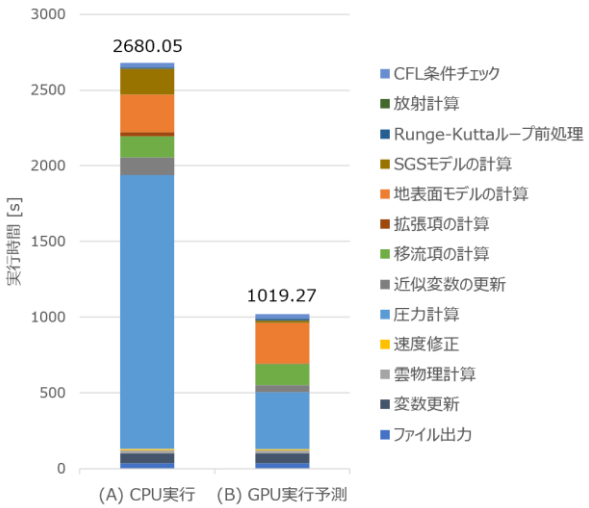

図 3　詳細モデル適用時の時間発展計算の実行時間

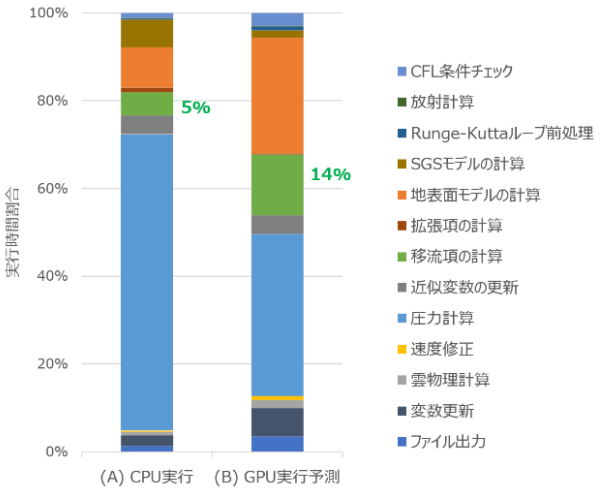

図 4　詳細モデル適用時の時間発展計算の実行時間

ⓒ2021 Information Processing Society of Japan

実行を行った結果，City-LES 全体の実行時間の約 90%が時間発展計算であった．2 プロセス 14 スレッドの 2CPU で実行した時間発展計算中の各関数の実行時間を図 3 (A)，実行時間割合を図 4(A)に示す．また(A)の実行時間に対し，先行研究[6]で GPU 化された関数に性能評価で得られた高速化率を適用し，GPU 実行時間の予測を示したものを図 3 (B)，図 4(B)に示す．高速化率を適用した部分は，圧力計算，SGS モデル計算，拡張項計算，近似変数更新，そして地表面モデル計算のうち建物なしと詳細モデルの両方で実行される部分である．

図 3 より移流項計算の実行時間割合は，(A) CPU 実行で約 5%，(B) GPU 実行予測で約 14%となっており，(B)では 3 番目に大きい．地表面モデル計算はより大きな割合となっているが，それぞれの並列性から予想される高速化率を鑑みると移流項計算を対象として問題ないと考えられる．

6.3 移流項計算の GPU 実装

移流項計算における計算ループに対する OpenACC 実装をリスト 2 に示す．ステンシル計算は多数の 3 重ループで構成されており，それぞれに parallel ディレクティブと loop ディレクティブを用いて並列化の指示を行った．ここで is, ie といったループの始終端の変数は各プロセスの持つ問題領域を表す変数である．また 1 行目の parallel ディレクティブの private 句は指定した変数を各スレッド独自で扱うことを表し，2 行目の loop ディレクティブの collapse 句はループの融合の指示を行っている．ここでは 3 重ループ対して collapse 句を適用し，1 つのループに展開している．

PGI コンパイラにおける OpenACC プログラムのコンパイルは，オプションとして `-acc -ta=tesla:cc70` 等を加えることで可能である．`ta` オプションは使用する GPU を示し，この例では NVIDIA Tesla V100 を使用することを表している．

7. 性能評価

7.1 評価結果

移流項計算の実行時間を図 5 に示す．(A)は 2 プロセス 14 スレッドの 2CPU 実行，(B)はリスト 2 の実装から collapse 句を外したコードを 2GPU 実行，(C)はリスト 2 の実装を 2GPU 実行したものである．

リスト 2　計算ループの OpenACC 実装

```
1  !$acc parallel private(i, j, k)
2  !$acc loop collapse(3)
3  do k = ks, ke
4    do j = js, je
5      do i = is, ie
6        . . .
7      enddo
8    enddo
9  enddo
10 !$acc end parallel
```

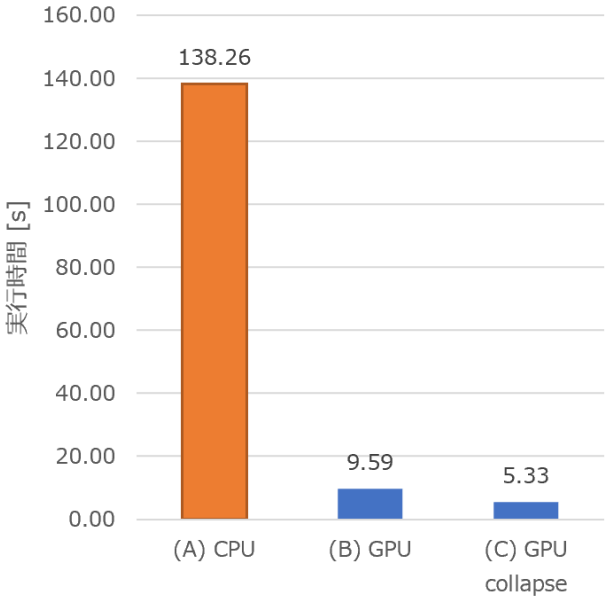

図 5　移流項計算の性能評価

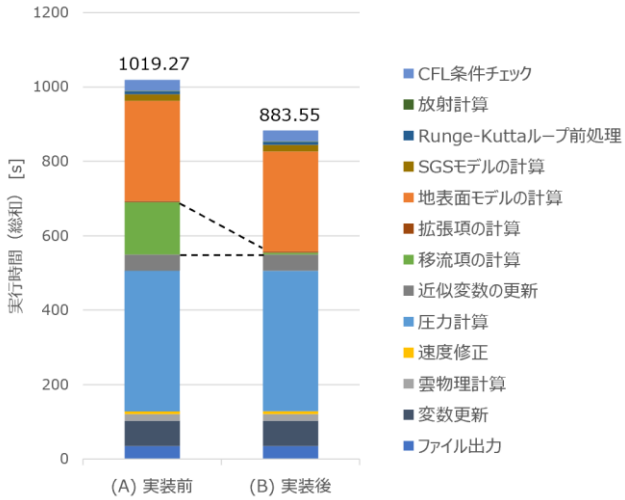

図 6　移流項計算の GPU 実装による時間発展計算の実行時間予測

(A) 2CPU 実行と比較して，(B) collapse 句なしの 2GPU 実行は約 14.4 倍の高速化，(C) collapse 句ありの 2GPU 実行は約 25.9 倍の高速化となっている．ステンシル計算の並列性の高さから高い高速化率が得られたほか，(B)と(C)を比較すると collapse 句の適用によって約 1.8 倍高速化されていることがわかる．これはループ回数に関わる各プロセスの計算領域のサイズを袖領域の計算のために単純な分割よりも大きく確保していて OpenACC の 128 スレッド実行で(B)では負荷の不均衡が生じていたものが，(C)で解消されたと考えられる．

7.2 性能予測

移流項計算の実装による時間発展計算全体の実行時間予

ⓒ2021 Information Processing Society of Japan

測を図 6 に示した．(A)は図 3(B)と同一で詳細モデル計算の GPU 実行予測，(B)は(A)に対して図 5(C)の高速化を適用したものである．(A)と(B)を比較すると，移流項計算の GPU 実装によって時間発展計算の実行性能は 1.15 倍になると予測できる．

8. まとめ

本研究では都市気象シミュレーションアプリケーション City-LES における建物や植生を考慮した詳細モデルを扱う計算の GPU 化を目的として，移流項計算関数の GPU 化および性能評価を行った．実装の結果として移流項計算の計算性能は約 25.9 倍に高速化され，これにより時間発展計算全体の性能は約 1.15 倍の性能向上が期待できると考えられる．

今後の課題として，本稿で実行時間予測を行った GPU 実行を実際に行うことがあげられる．詳細モデルを適用した実行パスを GPU 上で実行するにあたり，詳細モデル計算のみで実行される関数は GPU 化されていないためその計算の前後で CPU-GPU 間のデータ転送を行う必要がある．City-LES のデータ転送管理には CUDA Fortran と OpenACC の両方が用いられており，そのデータ転送の記述が複雑になっていることが今回実測に至ることができなかった要因の 1 つである．また，今回実装した移流項計算関数をその GPU 実行に組み込み，本稿の移流項計算を組み込んだ実行時間予測の検証を行っていく．

謝辞

本稿の執筆にあたりご協力頂いた筑波大学計算科学研究センターの藤田典久助教，小林諒平助教に深く感謝を申し上げます．

本研究成果は筑波大学計算科学研究センターの学際共同利用プログラム (Cygnus) における 2020 年度課題「都市を対象とした気象学 LES モデルの開発と応用」を利用して得られたものである．また，本研究の一部は「高性能汎用計算機高度利用事業」における課題「次世代演算通信融合型スーパーコンピュータの開発」による．

参考文献

[1] “November 2020 | TOP500,” https://www.top500.org/lists/top500/2020/11/ .
[2] R. Ikeda, H. Kusaka, S. Iizuka, T. Boku: Development of Urban Meteorological LES model for thermal environment at city scale. 9th International Conference for Urban Climate, Toulouse, France, (July, 2015).
[3] “Home – OpenMP,” https://www.openmp.org/
[4] 辻 大亮，多田野 寛人，朴 泰祐，池田 亮作，佐藤 拓人，日下 博幸：都市気象 LES コードの並列 GPU 環境における高速化．研究報告ハイパフォーマンスコンピューティング (HPC)，2019-HPC-168, No.8, (February, 2019).
[5] 辻 大亮，多田野 寛人，朴 泰祐，池田 亮作，佐藤 拓人，日下 博幸：都市気象コード City-LES の並列 GPU 実装の最適化と性能評価．研究報告ハイパフォーマンスコンピューティング (HPC)，2019-HPC-170, No.39, (July, 2019).
[6] D. Tsuji, T. Boku, R. Ikeda, T. Sato, H. Tadano, H. Kusaka: Parallelized GPU Code of City-Level Large Eddy Simulation. 19th International Symposium on Parallel and Distributed Computing, Warsaw, Poland, (July, 2020).
[7] 下川辺隆史，青木尊之，石田純一，河野耕平，室井ちあし：メソスケール気象モデル ASUCA の TSUBAME2.0 での実行．ながれ：日本流体力学会誌 30(2), 75-78, 2021-04-25.
[8] “TSUBAME2 | [GSIC]東京工業大学学術国際情報センター,” https://www.gsic.titech.ac.jp/tsubame2 .
[9]小野寺 直幸，青木 尊之，下川辺 隆史，小林 宏充：格子ボルツマン法による 1m 格子を用いた都市部 10km 四方の大規模 LES 気流シミュレーション，ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集 2013，123-131，2013-01-08.
[10] “CUDA Zone | NVIDIA Developer,” https://developer.nvidia.com/cuda-zone .
[11] “CUDA FORTRAN | NVIDIA Developer,” https://developer.nvidia.com/cuda-fortran.
[12] “Homepage | OpenACC,” https://www.openacc.org/ .

ⓒ2021 Information Processing Society of Japan