Vol.2018-HPC-165 No.37
2018/8/1

OpenMP の対象ループを変更する自動チューニング手法の評価

櫻井 刀麻[†1]　片桐 孝洋[†2]　永井 亨[†2]　荻野 正雄[†2]

近年の計算機はアーキテクチャが多様化しており，数値計算ソフトウェアが高い性能を発揮するためには，対象の計算機に合わせたソフトウェアのチューニングが必要となる．しかし，各環境に合わせた最適化はハードウェアを知る専門家でなければ難しく，手間と時間がかかる．自動チューニング技術を適用することにより，チューニングの労力を減らすことができ，専門家でなくても計算機の性能を引き出す実装が利用できる．本研究では，並列プログラミングインターフェースの 1 つである OpenMP を用いたプログラムの自動チューニングを対象とする．OpenMP の対象ループを変える自動チューニング手法の提案を行い，その有効性を検証する．名古屋大学情報基盤センターに設置されているスーパーコンピュータ CX400 を利用して性能評価を行った結果，核融合プラズマシミュレーションコード GKV のカーネルのオリジナルコードに対して，提案手法による自動チューニングを適用することで，最大で約 7.72 倍の速度向上を得た．また，自動チューニング対象となる場所の実行毎に適するスレッド数に実行時に切り替える，新しい自動チューニング方式の提案と予備評価を行った．FX100 を利用して地震波シミュレーションコード Seism3D を利用した予備評価の結果，自動チューニングによる動的なスレッド切り替えのオーバーヘッドは小さいことが明らかとなった．

1. はじめに

近年の計算機はマルチコア化が進み，メモリやキャッシュの構造，GPU の有無など，アーキテクチャが多様化している．数値計算ソフトウェアが高い性能を発揮するためにはソフトウェアのチューニングが重要となるが，計算機環境に合わせた最適化はハードウェアの専門知識も必要であり，手間と時間がかかる．さらに，ある環境に特化したチューニングを行ったプログラムは他の環境では性能が低下する可能性があり，再びチューニングが必要となる．また，広く利用されているコンパイラ最適化はソフトウェア開発者にとって十分でないことが多い．

数値計算におけるソフトウェア自動チューニング [1] 技術とは，プログラムの性能を向上させる性能チューニングを自動化させる技術をいう．この技術を使用することで，異なる環境においても同じプログラムで高い性能を発揮することが期待でき，ソフトウェアの性能可搬性を高めることができる． そのため，近年，自動チューニング機能が付いた数値計算ソフトウェアが多く開発されるようになった [2] [3] [4] ．また，自動チューニング付き数値計算ソフトウェア作成のための自動チューニングフレームワークが提案されている．たとえば，OpenTuner [5] ，Xevolver [6] ，および FIBER [7] などが代表的である．

本報告の構成は以下のとおりである．2 章に本研究で使用した自動チューニングのフレームワークと自動チューニング言語を説明する．3 章で OpenMP の対象ループを変更するループ変換の性能評価を行った結果を示す．4 章で OpenMP のスレッド並列数をプログラム実行中に変更して実行した性能評価の結果を示す．最後に本研究で得られた知見と今後の課題について述べる．

2. 自動チューニングのフレームワーク FIBER と自動チューニング言語 ppOpen-AT

2.1 FIBER

FIBER (Framework of Install-time, Before Execute-time, and Run-time optimization layers) [7] は，数値計算ソフトウェアへの適用を目的とした自動チューニングの枠組みである．このフレームワークは，以下に示す 3 つのタイミング (階層) で自動チューニングを行うことができるソフトウェア構成方式となっている．

1. ライブラリをインストールする際のインストール時最適化階層
2. ユーザーによりパラメタ (問題サイズ，MPI プロセス数と OpenMP スレッド数) が確定した実行起動前最適化階層
3. 実際にライブラリが実行された時点の実行時最適化階層

以上の階層により，自動チューニングの適用対象のアプリケーションを増やし，パラメタ推定の精度を向上させることができる．

また，FIBER では，次に示す 2 つの機能を定めている．

1. 専用言語 (ディレクティブ) などによってユーザーのプログラムに指示を与えることで，自動チューニングを実行するコードの生成，コードのパラメタ化，パラメタの登録を行う機能
2. それぞれの階層におけるパラメタの最適化機能．

FIBER における自動チューニングは，基本パラメタ集合，性能パラメタ集合，コスト定義関数によって定められる．

基本パラメタ集合 (BP: Basic Parameters) は，数値計算を行う場合での基本情報 (行列サイズなど)， 計算機環境

†1 名古屋大学 情報学研究科 情報システム学専攻
†2 名古屋大学 情報基盤センター 大規模計算支援環境研究部門

(プロセッサ台数，計算機構成方式など) に関連するパラメタの集合である．

性能パラメタ集合 (PP: Performance Parameters) は， BP を固定した際にプログラムの性能を決定するパラメタの集合である．

コスト定義関数は BP を固定した場合に，各タイミングで PP により決まるコストを求める関数である．このコストはプログラムの実行時間，メモリ使用量，消費電力量などが考えられる．

FIBER での自動チューニングは，各タイミングで定まる基本パラメタ集合 BP を固定したとき，コスト定義関数を最小化する性能パラメタ集合 PP を求める処理であると定義されている．

2.2 ppOpen-AT

ppOpen-AT [8] は，次世代の科学技術アプリケーションの開発・実行環境 ppOpen-HPC [9] の自動チューニング機構として開発された自動チューニング言語である．並列数値計算ライブラリの開発効率を向上させることを目的に設計された，FIBER 方式による自動チューニング機能の付加を支援する自動チューニング言語（ディレクティブ）ABCLibScript [10] の機能を引き継いでいる．

ppOpen-AT は，Fortran90 と C 言語のプログラムに FIBER 方式の自動チューニング機能を付加する． 自動チューニングについての記述は専用のディレクティブ形式で行うので，ppOpen-AT を使用しない環境での実行を阻害せず，効率的にソフトウェアを開発できる．ppOpen-AT 専用のプリプロセッサによってディレクティブが解釈され，最適化候補のコードと，最適なコードを探索する自動チューニング機能を含むプログラムのコードが生成される．

ppOpen-AT による自動チューニングの機能はプログラムを書き換えることで実装されており，エンドユーザーにソフトウェアが渡る前にチューニング候補となるコードが生成されている．そのため，エンドユーザーは自分の環境での実行時に動的にコード生成を行わなくてよい [11] ．これにより，例えばスーパーコンピュータを使用する際にログインノードにコード生成の負荷を与えることなく自動チューニングを使用できる． ただし，事前に生成した候補となるコード全てがプログラムに含まれるので，コードの量が増大する恐れがある．これを防ぐためには，ソフトウェア開発者が自動チューニング対象の処理についての知識を利用して，チューニング候補を削減しなければならない [12] ．

ppOpen-AT は最適化候補のコード生成機能として，ループアンローリング，コード選択機能，ループ分割 (split)，ループ融合 (collapse)，および，演算順序変更機能を提供している．ここで生成したコードは，チューニング候補ライブラリとしてソフトウェアに含まれる．

3. OpenMP の並列対象ループの変更

3.1 提案手法

2 重以上の多重ループを OpenMP でスレッド並列化する場合において，並列化の対象ループを変更するループ変換を ppOpen-AT の新機能として提案する．

ppOpen-AT のディレクティブの記述は， !oat$ region start と !oat$ region end で囲むことで，対象ループを指定する．本提案では，OpenMP のディレクティブの移動を伴うコード変換を提案する．具体的には，Fortran90 における do ループを

!oat$ install Exchange(対象ループ番号，対象ループ番号,...**) region start**
～
!oat$ install Exchange(対象ループ番号，対象ループ番号,...**) region end**

で囲むことで対象ループを指定する．ただし対象ループには OpenMP の並列化指定が 1 つのみ記述されているとする．対象ループ番号は，最外ループから数えて，何番目のループを OpenMP の並列化対象とするかを指定する．この数字をコンマで区切り指定することで，複数のチューニング候補を生成可能である．

3.2 実験

提案手法の評価を行うため， ppOpen-AT のループ変換と提案する手法で同じループを変換し，それぞれの性能を比較する．

使用するプログラムはプラズマ乱流解析コード GKV [13] のサブルーチン exb_realspcal の 4 重ループ（図 1）である．このループを ppOpen-AT の collapse 機能により変換したループが図 2，図 3 の 2 つのループである．提案する手法では図 1 のループを 3 つのループに変換できる．そのループを図 4，図 5，図 6 に示す．提案する ppOpen-AT の記述では，図 1 のオリジナルループの全体を， 3.1 提案手法で記したディレクティブで囲むとする．

```
do iv = 1,   2*nv
!$OMP parallel do private(mx,  my)
   do iz = (-nz),   nz-1
      do mx = ist_xw,   iend_xw
         do my = 0,   nyw
               !計算部分
   enddo enddo enddo
!$OMP end parallel do
enddo
```

図 1　オリジナルの 4 重ループ(チューニング対象)

```
do iv = 1,  2*nv
!$OMP parallel do private(mx, my, mx_my)
 do iz = (-temp_nz),  temp_nz-1
  do mx_my = 1 ,  (iend_xw-ist_xw+1)*(nyw-0+1)
   mx=mod((mx_my-1)/(nyw-0+1), (iend_xw- &
          &ist_xw+1))+ist_xw
   my = mod((mx_my-1), (nyw-0+1)) + 0
   !計算部分
 enddo enddo
!$OMP end parallel do
enddo
```

図 2　ppOpen-AT で生成した 3 重ループ

```
do iv = 1,  2*nv
!$OMP parallel do private(mx, my, iz)
 do iz_mx_my = 1,  (temp_nz-1-(-temp_nz)+1)* &
                   &(iend_xw-ist_xw+1)*(nyw-0+1)
  iz = mod( (iz_mx_my-1)/((iend_xw-ist_xw+1)* &
            &(nyw-0+1)),  (temp_nz-1- &
            &(-temp_nz)+1)) + (-temp_nz)
  mx = mod((iz_mx_my-1)/(nyw-0+1),  &
       &(iend_xw-ist_xw+1)) + ist_xw
  my = mod((iz_mx_my-1), (nyw-0+1)) + 0
  !計算部分
 enddo
!$OMP end parallel do
enddo
```

図 3　ppOpen-AT で生成した 2 重ループ

```
!$OMP parallel do private(iz, mx, my)
do iv = 1,  2*nv
  do iz = (-nz),  nz-1
    do mx = ist_xw,  iend_xw
      do my = 0,  nyw
        !計算部分
enddo enddo enddo enddo
!$OMP end parallel do
```

図 4　OpenMP の対象を変更したループ（提案手法）

```
do iv = 1,  2*nv
  do iz = (-nz),  nz-1
!$OMP parallel do private(my)
    do mx = ist_xw,  iend_xw
      do my = 0,  nyw
        !計算部分
    enddo enddo
!$OMP end parallel do
enddo enddo
```

図 5　OpenMP の対象を変更したループ（提案手法）

```
do iv = 1,  2*nv
  do iz = (-nz),  nz-1
    do mx = ist_xw,  iend_xw
!$OMP parallel do private
      do my = 0,  nyw
        !計算部分
  enddo enddo enddo
!$OMP end parallel do
enddo
```

図 6　OpenMP の対象を変更したループ（提案手法）

計算機環境として，名古屋大学情報基盤センターのスーパーコンピュータ Fujitsu PRIMEHPC FX100（FX100）及び Fujitsu PRIMERGY CX400/2550（CX400），東京大学情報基盤センターの Oakforest-PACS を使用した. 各計算機の構成を次に示す.

- FX100
  - ・計算ノード　Fujitsu PRIMEHPC FX100
  - ・プロセッサ名　Fujitsu SPARC64 XIfx
  - ・1 ノード当たりのプロセッサ数　1
  - ・1 ノード当たりのコア数　32
  - ・1 ノード当たりの最大スレッド数　32
  - ・周波数　2.2 GHz
  - ・コンパイラ　frtpx: Fujitsu Fortran Driver Version 2.0.0 P-id: T01815-01
  - ・コンパイルオプション　-Kfast -Qt -Cpp -X9 -fs -fw -Kopenmp
- CX400
  - ・計算ノード　Fujitsu PRIMERGY CX2550 M1
  - ・プロセッサ名　Intel Xeon E5-2600 v3 processor family
  - ・1 ノード当たりのプロセッサ数　2
  - ・1 ノード当たりのコア数　28
  - ・1 ノード当たりの最大スレッド数　28
  - ・周波数　2.6 GHz
  - ・コンパイラ　frt: Fujitsu Fortran Driver Version 1.2.0 P-id: T01816-01
  - ・コンパイルオプション　-Kfast -Qt -Cpp -X9 -fs -fw -Kopenmp
- Oakforest-PACS
  - ・計算ノード　Fujitsu PRIMERGY CX1640 M1
  - ・プロセッサ名　Intel Xeon Phi　7250
  - ・1 ノード当たりのプロセッサ数　1
  - ・1 ノード当たりのコア数　68
  - ・1 ノード当たりの最大スレッド数　272 (HT4)
  - ・周波数　1.4 GHz
  - ・コンパイラ　ifort: Intel(R) Fortran Intel(R)64 Compiler for applications running on Intel(R) 64, Version 18.0.1.163 Build 20171018
  - ・コンパイルオプション　-axMIC-AVX512 -O2 -mcmodel=medium -shared-intel –qopenmp

実験では次の条件で図 1 から図 6 の各ループを実行した.

- 行列のサイズ nz を 1 から 8 に変化させる.
- OpenMP 並列数を 1 から最大 (FX100 は 32,　CX400 は 28,　Oakforest-PACS は 272) まで変える.
- ループを 1000 回繰り返す.

3.3 実験結果

実行時間を計測し，各ループで最速となったスレッド数での時間を基準として，ppOpen-AT による変換と提案手法による変換がオリジナルのループに比べて何倍速くなったかのグラフを作成した．FX100 の結果を図 7，CX400 の結果を図 8，Oakforest-PACS の結果を図 9 に示す．

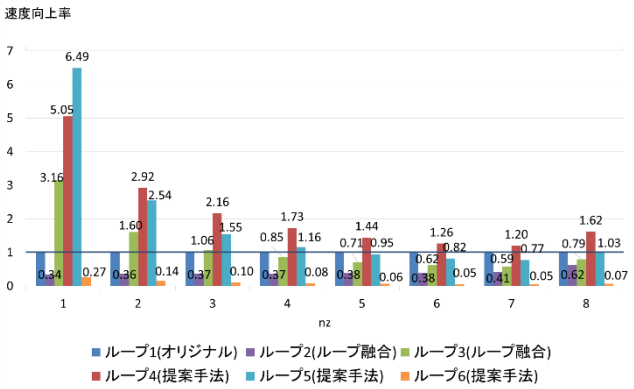

図 7　オリジナルと比較した速度向上率 (FX100)

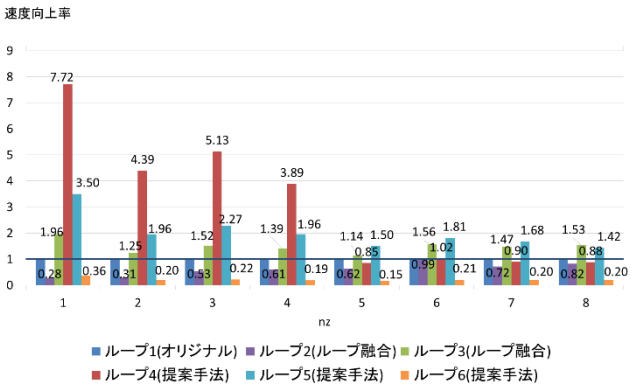

図 8　オリジナルと比較した速度向上率 (CX400)

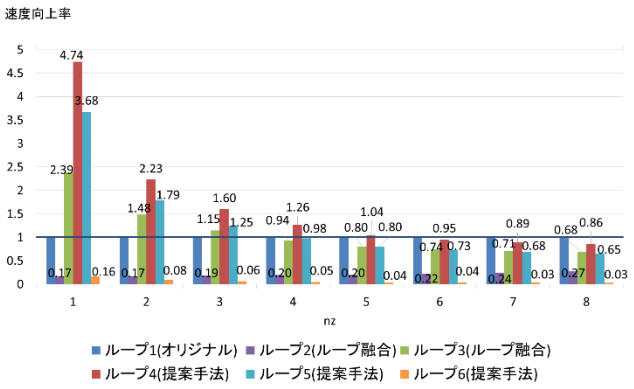

図 9　オリジナルと比較した速度向上率 (Oakforest-PACS)

3.4 性能プロファイラによる解析

提案手法によるループがなぜオリジナルループより高速になったかを解析するため，FX100 で利用できる性能プロファイルツールを用いて，性能解析を行った．

FX100 で行列サイズ nz=8 の場合に詳細プロファイラ(精密 PA 可視化機能)[14] を実行し，各ループ実行時のハードウェア情報を取得した．ここでは，オリジナルループと提案手法によるループ 4 について，それぞれ最速となったスレッド数 21 と 19 の情報で比較した．メモリ・キャッシュヒジー率を図 10 と図 11 に，実行時間の内訳を各スレッドの平均をとりグラフを図 12 に， 内訳の表を表 1 に示した．ただし平均はスレッド番号 0 から 15 のデータを使用している．

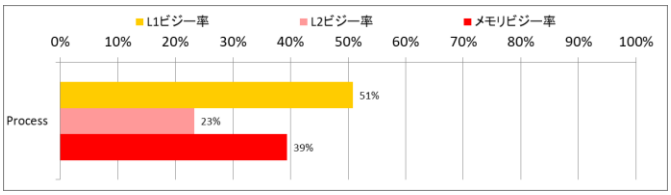

図 10　オリジナルループのメモリ・キャッシュヒジー率

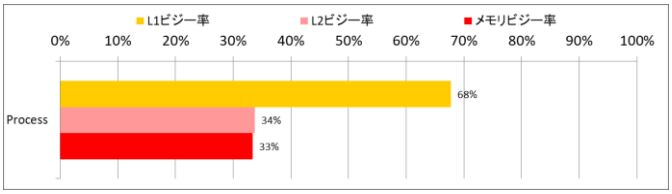

図 11　提案手法(ループ 4)のメモリ・キャッシュヒジー率

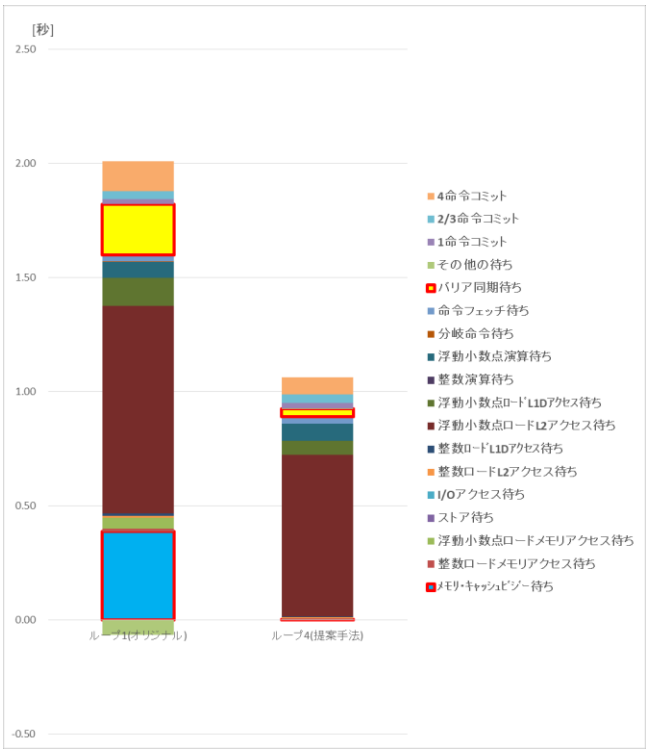

図 12　実行時間内訳のスレッド平均のグラフ

表 1　実行時間内訳のスレッド平均

（単位は秒）	ループ1(オリジナル)	ループ4(提案手法)
4命令コミット	0.13	0.08
2/3命令コミット	0.03	0.04
1命令コミット	0.02	0.02
その他の待ち	0.07	0.00
バリア同期待ち	0.22	0.03
命令フェッチ待ち	0.03	0.03
分岐命令待ち	0.00	0.00
浮動小数点演算待ち	0.07	0.08
整数演算待ち	0.00	0.00
浮動小数点ﾛｰﾄﾞL1Dｱｸｾｽ待ち	0.12	0.06
浮動小数点ロードL2アクセス待ち	0.91	0.71
整数ﾛｰﾄﾞL1Dｱｸｾｽ待ち	0.01	0.00
整数ロードL2アクセス待ち	0.01	0.00
I/Oアクセス待ち	0.00	0.00
ストア待ち	0.00	0.00
浮動小数点ロードメモリアクセス待ち	0.05	0.00
整数ロードメモリアクセス待ち	0.01	0.01
ﾒﾓﾘ･ｷｬｯｼｭﾋﾞｼﾞｰ待ち	0.39	0.00

図 10，図 11 から提案手法により変形したループではオリジナルループと比べると L1 ビジー率が 51%から 68%に，L2 ビジー率は 23%から 34%に上がりキャッシュを効率的に使えるようになった．図 12 のグラフからは，オリジナルループと比べて，提案手法で変形したループは赤線で囲って示したバリア同期待ちとメモリ・キャッシュビジー待ちが減少しており，表 1 を見るとバリア同期待ちは 0.22 秒から 0.03 秒に，メモリ・キャッシュビジー待ちは 0.39 秒から 0.00 秒になったことが確認できる．プロファイラの結果から，提案手法による変形がこのループに有効であることがわかる．

3.5 考察

変換前のオリジナルループと ppOpen-AT により自動生成されたループ融合を行ったループを比較対象として実行時間を計測した結果，FX100 では，実験したすべての行列サイズで，提案手法がほかのループの実行時間より高速となった．また，詳細プロファイラの結果から，提案手法によるコードでは，キャッシュの利用効率が上がり，バリア同期待ちが減少したことで高速化されたことが明らかになった．

CX400 では，GKV の主演算ループのループ長を示す nz が 1 から 7 の場合は，提案手法が最高速であり，nz=1 のとき，オリジナルループに比べて約 7.72 倍の速度向上を得た．Oakforest-PACS では，nz が 1 から 5 の場合に提案手法が最速となり，提案手法の自動チューニング方式は，複数の計算機環境で有効であることが示された．

4. OpenMP のスレッド数変更

4.1 概要

ppOpen-AT による自動チューニングでは性能パラメタを設定し，プログラムをチューニングする際に性能パラメタを変化させて性能を測定し，プログラムの実行時に測定結果を参照する．現在は生成された最適化候補のうちどれを使用するかを性能パラメタとしており，時間などの計測結果で実際にどの候補を利用するかを決定する．

本研究では 3 節の OpenMP の並列対象ループの変更機能に加えて，性能パラメタとして OpenMP のスレッド数をとり自動チューニングを行う枠組みを提案する．

OpenMP のスレッド数変更にはサブルーチン **omp_set_num_threads** を使用した．

4.2 実験

チューニングの対象となる部分のスレッド並列数を変更し，性能向上に効果があるかを検証する．

使用するプログラムは有限差分法による地震波シミュレーションコード Seism3D [15] を原型とする ppOpen-APPL/FDM である．ppOpen-APPL/FDM は ppOpen-HPC により提供されており，OpenMP と MPI のハイブリッド実行に対応したライブラリとなっている．ppOpen-APPL/FDM は自動チューニング言語 ppOpen-AT を使用してコード選択による自動チューニング機能を実装されているが，本実験では自動チューニング機能を使用せず，デフォルトの実装で性能評価を行う．

計算機環境は名古屋大学情報基盤センター設置のスーパーコンピュータ Fujitsu PRIMEHPC FX100 を使用した．FX100 の計算機構成は 3.2 節と同じである．コンパイルオプションには-Kfast,openmp を使用した．

ppOpen-APPL/FDM にはいくつか計算が集中するカーネルがあるが，今回は実行時間の約 35%を占める計算を行う update_stress [15] をチューニングの対象とする．

実験は ppOpen-AT で自動生成される AT 領域を手続き化する手続き OAT_InstallRoutines.f90（実行起動前時の自動チューニングを指定した場合）のコードにおいて，チューニング対象の update_stress のカーネル部分の計算を行う直前で，サブルーチン omp_set_num_threads を使用して OpenMP のスレッド数を変更し，計算後に再びサブルーチン omp_set_num_threads を使用してスレッド数をもとに戻し，時間を計測する方法で行なった．このとき，スレッド数を変更しない実行（従来の ppOpen-AT の自動チューニング方式，従来手法と記載する）と実行時間を比較した．

実験は次の条件で行った．

- 8 ノードを使用する
- 2000 時間ステップの計算を行う（全体時間は，2000 時間ステップの実行時間である）
- プロセス数は 8，16，32，64，128，256 で実行する
- スレッド数の変更の実験は，事前に調べた一番速いスレッド数を使用して行う
- スレッド数を変更しない実行（従来手法）は，MPI/OpenMP ハイブリッド実行時の各 MPI プロセス数で取れる最大の OpenMP スレッド数とする

4.3 実験結果

スレッド数変更しない従来手法と，実行時に最速なスレッド数に対象となる手続きの実行前に切り替える提案手法の実行時間を並べた結果を図 13 に示す．また，スレッド数変更を行ったことによる速度向上率のグラフを図 14 に示す．

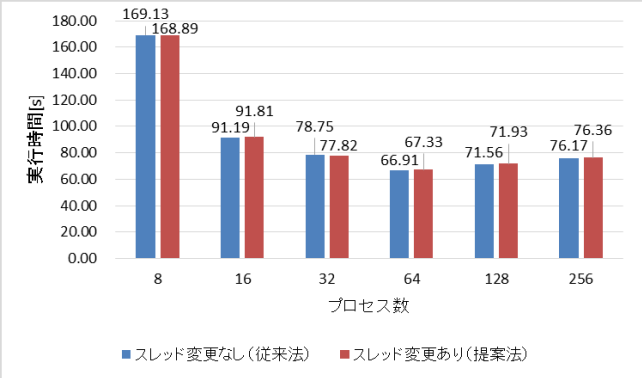

図 13　全体時間の比較．8 ノードを使うため，各プロセス数の実行における最大のスレッド数は以下になる：8 プロセス：32 スレッド、16 プロセス：16 スレッド、32 プロセス：8 スレッド、64 プロセス：4 スレッド、128 プロセス：2 スレッド、256 プロセス：1 スレッド（ピュア MPI 実行）．

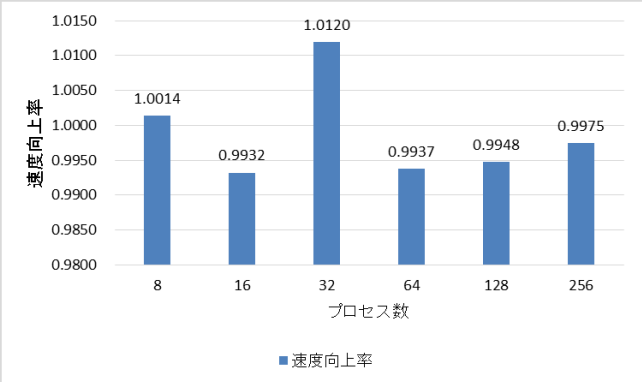

図 14　スレッド数変更による全体の速度向上率．8 ノードを使うため，各プロセス数の実行における最大のスレッド数は以下になる：8 プロセス：32 スレッド、16 プロセス：16 スレッド、32 プロセス：8 スレッド、64 プロセス：4 スレッド、128 プロセス：2 スレッド、256 プロセス：1 スレッド（ピュア MPI 実行）．

4.4 考察

本実験ではスレッド数変更により，変更しない従来法と比較して，8 プロセスで約 1.0014 倍，32 プロセスで約 1.012 倍の速度となった．対象の手続きのみの実行時間は 128 プロセスで約 1.0055 倍の速度向上となった．しかし，それ以外のプロセス数ではスレッド数の変更により速度向上が得られず、速度が上がったプロセスでも上昇幅は僅かであった．

この一方で，omp_set_num_threads によりスレッド数を変更しても大きく実行時間が落ちることはなかった．このことは，FX100 では，実行時に頻繁にスレッド数を変更しても，スレッド数変更のオーバーヘッドの影響は小さいことを意味している．そのため，提案手法の対象場所（AT 領域）ごとに適するスレッド数に変更する AT 方式が，AT 領域の数を増やすと，さらに有効になる可能性があり，今後新しい AT 方式として研究開発と性能評価を進める必要がある．

5. おわりに

本研究では複雑化する計算機環境に合わせた自動チューニングを行うため，従来の ppOpen-AT に実装されていない自動チューニング機能を検討し，予備評価を行った．ppOpen-AT の新しい自動チューニング機能として 2 つの方法を提案し，それらの予備評価を行った．

OpenMP の並列対象を変えるループ変換手法（自動チューニング方式）は FX100，CX400，Oakforest-PACS の 3 つの計算機環境を用いて実験を行い，変換前のループや，ppOpen-AT の collapse により変換したループより高速となる場合があることを確認した．そのため，新しい自動チューニングによる最適化方式として提案したループ変換手法を適用するのは効果的であると考えられる．

OpenMP のスレッド数をプログラム中で変更するチューニングの予備評価では，プロセス数を変えて測定を行った中のいくつかの場合で，従来の固定スレッド数での実行より高速となった．今回は ppOpen-APPL/FDM 中の多数あるカーネルのうち update_stress のみを対象カーネルとしてスレッド数を変更した．しかし，その他のカーネルごとに最適となるスレッド並列数があるはずである．そのため，提案手法が有効となる可能性がある．

今後の課題として，上記 2 種の新しい自動チューニング手法が，今回扱ったベンチマーク中のその他のループで効果あるか，また，今回扱ったベンチマーク以外のプログラムで効果があるかの検証が必要である．また，OpenMP のスレッド数変更については，FX100 でスレッド数の動的切り替え時のオーバーヘッドが少ないことが明らかとなった．そのため，提案手法が有効となる場合があると予想されるため，Seism3D 内の複数のカーネルをチューニング対象として，再評価を行う予定である．

謝辞

本研究の一部は，科学研究費補助金，基盤研究（B）「通信回避・削減アルゴリズムのための自動チューニング技術の新展開」（課題番号：16H02823）の支援による．

参考文献

[1] 弓場敏嗣, “数値計算応用を対象とする自動性能チューニング技術,” 電気通信大学紀要 20 巻 1・2 合併号, pp.1-14, (2007)

[2] Frigo, M. and S. G. Johnson, “FFTW: An Adaptive Software Architecture for the FFT,” Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, pp.1381 -1384 (1998)

[3] R. Clint Whaley, Antoine Petitet, Jack J. Dongarra, “Automated empirical optimizations of software and the ATLAS Project,” Parallel Computing, Vol. 27, Issue 1-2, pp.3-35, 2001.

[4] 櫻井隆雄，片桐孝洋，直野健，黒田久泰，中島研吾，猪貝光祥，大島聡史，伊藤祥司，“自動チューニングインターフェース OpenATLib における自動チューニング機能の評価,” 研究報告ハイパフォーマンスコンピューティング（HPC） 2011-HPC-130(43), pp.1-6 (2011)

[5] Jason Ansel, Shoaib Kamil, Kalyan Veeramachaneni, Una-May O'Reilly, and Saman Amarasinghe, “OpenTuner: An Extensible Framework for Program Autotuning,” in 23rd International Conference on Parallel Architectures and Compilation Techniques (PACT 2014), pp.303-315 (2014).

[6] Hiroyuki Takizawa, Shoichi Hirasawa, Yasuharu Hayashi, Ryusuke Egawa, Hiroaki Kobayashi,, “Xevolver: An XML-based Code Translation Framework for Supporting HPC Application Migration,” IEEE International Conference on High Performance Computing (HiPC),, pp.1-11 (2014)

[7] 片桐孝洋，吉瀬謙二，本多弘樹，弓場敏嗣，“FIBER: 汎用的な自動チューニング機能の付加を支援するソフトウェア構築方式,” 情報処理学会研究報告 2003-HPC-94, pp.1-6, 2003.

[8] 片桐孝洋, “ppOpen-AT: ポストペタスケール時代の数値シミュレーション基盤ソフト,” 数理解析研究所講究録 第1791 巻, pp.107-111 (2012)

[9] K. Nakajima, M. Satoh, T. Furumura, H. Okuda, T. Iwashita, H. Sakaguchi, T. Katagiri, M. Matsumoto, S. Ohshima, H. Jitsumoto, T. Arakawa, F. Mori, T. Kitayama, A. Ida and M. Y. Matsuo, “ppOpen-HPC: Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT),” Optimization in the Real World, Mathematics for Industry 13, K. Fujisawa et al. (eds.), Springer, pp.15-35 (2016)

[10] T. Katagiri, K. Kise, H. Honda, T. Yuba, “ABCLibScript: A directive to support specification of an auto-tuning facility for numerical software,” Parallel Computing, Vol. 32, Issue 1, pp.92–112 (2006)

[11] 片桐孝洋，松本正晴，大島聡史，“SCG-AT：静的コード生成のみによる自動チューニング実現方式,” 情報処理学会研究報告 Vol.2015-HPC-150 No.32, pp.1-11 (2015)

[12] 片桐孝洋，吉瀬謙二，本多弘樹，弓場敏嗣，“ユーザ知識を活用するソフトウエア自動チューニングについて,” 情報処理学会研究報告 2004-EVA-10, pp.19-24 (2004)

[13] Watanabe, T-H., and H. Sugama., “Velocity-space structures of distribution function in toroidal ion temperature gradient turbulence.,” Nuclear Fusion 46.1, (2005):24.

[14] FUJITSU LIMITED, “FUJITSU Software Technical Computing Suite V2.0 プロファイラ使用手引書 J2UL-1891-03Z0(01),” (2017)

[15] F. Mori, M. Matsumoto, T. Furumura, Performance of FDM Simulation of Seismic Wave Propagation using the ppOpen-APPL / FDM Library on the Intel Xeon Phi Coprocessor, Springer LNCS (2014)