1H-03

自動チューニング言語 ppOpen-AT における新ループ変換手法の提案

櫻井 刀麻† 片桐 孝洋‡ 永井 亨‡ 荻野 正雄‡

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

近年の計算機は，メモリアクセスの均一性，階層化されたメモリ，CPU が備えるコア数の違いなど，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必要となるが，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．数値計算における自動チューニングとは，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ことをいう．本研究で想定する自動チューニングは，同じ動作をするプログラムの実装を複数用意し，実際の測定結果から使用する実装を環境に応じて選択する．用意する実装の種類を増やすことは，多くの計算機環境で高い性能を出すためには効果的である．

ppOpen-AT[1]は自動チューニング言語で，プログラムに自動チューニング機能を付けるディレクティブを提供する．本研究では OpenMP の並列領域を変える実装を試し， ppOpen-AT における新機能として提案することを目的とする．

2. 自動チューニング言語 ppOpen-AT

ppOpen-AT は Fortran と C 言語のコードにディレクティブ(プラグマ)の形で指示を与える自動チューニング言語である． Fortran のプログラムコードでは **!oat$** で始まるディレクティブでコードの変換方法を指定し， ppOpen-AT のプリプロセッサにかけることで指定したループ変換を行い，自動チューニングを実行する候補を生成する．使用できる変換手法は，現在ループアンローリング，ループ融合(collapse)，ループ分解(split)が実装されている．

Proposal of a new loop transformation method for auto-tuning language ppOpen-AT

† Toma Sakurai, Electrical and Electronic Engineering and Information Engineering, School of engineering, Nagoya University

‡Takahiro Katagiri, Toru Nagai, Masao Ogino, Information Technology Center, Nagoya University

3. 提案手法

ループ変換の一手法として， OpenMP の並列領域を指定しているディレクティブ(**!$OMP** の行)の位置を変え，並列化の対象ループを変更する機能を提案する．

ppOpen-AT の記述としてはループを ***!oat$ install exchange(1,2) region start*** と ***!oat$ install exchange(1,2) region end*** で囲むものとする．()内の数字は，外側から何番目のループを OpenMP の並列化対象とするかを示している．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する．

4. 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[2]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 4 つのループに変換できる．ここでは OpenMP のディレクティブを一番外側に置いた，図 4 のループを使用する．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.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 の並列領域を変更したループ

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(コンパイラ frtpx Version 2. 0. 0, オプション -Kfast -Qt -Cpp -X9 -fs -fw -Kopenmp)及び Fujitsu PRIMERGY CX400/2550(コンパイラ frt Version 1. 2. 0, オプション -Kfast -Qt -Cpp -X9 -fs -fw -Kopenmp)，東京大学情報基盤センターの Oakforest-PACS(コンパイラ ifort Version 18. 0. 1. 163，オプション -axMIC-AVX512 -O2 -mcmodel=medium -shared-intel -qopenmp)を使用した．それぞれ 1 ノードを使用し，スレッド並列数は FX100 で 1 から 32，CX400 は 1 から 28，Oakforest-PACS は 1 から 272 に変え，それぞれ実行時間を測定した．

5. 結果

図 1 から図 4 のループを nv=8, nz=4, ist_xw=0, iend_xw=127, nyw=64 の条件で実行した．それぞれの計算機環境で最速となったスレッド数と実行時間，オリジナルのループからの速度向上率を表 1，表 2，表 3 に示す．

表 1 FX100 の実行結果

	図1のコード	図2のコード	図3のコード	図4のコード(提案手法)
スレッド数	10	29	32	30
実行時間(秒)	0.944756	2.633558	1.131273	0.535411
速度向上率		0.358737	0.835126	1.764543

表 2 CX400 の実行結果

	図1のコード	図2のコード	図3のコード	図4のコード(提案手法)
スレッド数	19	27	28	24
実行時間(秒)	1.239136	2.08015	0.906768	0.315256
速度向上率		0.595696	1.366541	3.930572

表 3 Oakforest-PACS の実行結果

	図1のコード	図2のコード	図3のコード	図4のコード(提案手法)
スレッド数	19	19	255	18
実行時間(秒)	1.259473	6.224855	1.331034	0.99455
速度向上率		0.20233	0.946237	1.266375

表 1，表 2，表 3 の結果から実験した環境では，提案手法によりチューニングしたループが，オリジナルのループ，ppOpen-AT により変換したループよりも高い性能を出していることが分かる．特に，CX400 では提案手法のループが，オリジナルのループより速度が約 3．9 倍向上しており，自動チューニングによる高い速度向上を示している．

6. まとめ

実験結果から，提案手法である OpenMP の対象ループを変更するチューニングが，ppOpen-AT の従来手法でのループ変換よりも高速となる場合が明らかとなった．したがって，この変換を自動チューニング言語に組み込むことは有効であると考えられる．ただし，今回の実験結果は実行時間だけを対象としており，かつ 1 種類のループでしか評価をしていない．今後の課題として，異なるループでの評価，および OpenMP の対象ループをさらに変えて評価し，自動チューニングの機能として効果があるか検証することが必要である．

謝辞

本研究の一部は，JSPS 科研費 16H02823，および JSPS 二国間交流事業オープンパートナーシップ共同研究「国際交流による自動チューニングのための性能モデルの深化」の助成による．

参考文献

[1] T. Katagiri, S. Ohshima, M. Matsumoto, “Auto-Tuning on NUMA and Many-Core Environments with an FDM Code,” in Proc. IEEE Parallel and Distributed Processing Symposium Workshops (IPDPSW)2017, pp. 1399-1407, 2017.

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