Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

OSCAR自動並列化コンパイラと NECベクトル化コンパイラの協調による ベクトル・パーソナルスパコン上での自動ベクトル並列化

田處 雄大[1]　見神 広紀[1]　細見 岳生[2]　木村 啓二[1]　笠原 博徳[1]

概要：科学技術計算や画像処理，機械学習等の分野において，プログラムの高速化が求められている．これらのプログラムが持つ高い並列性に着目した高速化手法として，ベクトル化と並列化が挙げられる．ベクトルプロセッサや GPGPU，SIMD 命令を用いたベクトル化，マルチコア CPU を用いた並列化はこれまでも多く研究されてきた．本稿では OSCAR 自動並列化コンパイラと NEC のベクトルプロセッサ SX-Aurora TSUBASA 用コンパイラを組み合わせた自動ベクトル並列化を提案する．本コンパイルフローの特徴は，OSCAR コンパイラによるマルチグレイン並列化及びキャッシュ最適化と，NEC コンパイラによる自動ベクトル化を組み合わせて利用することにより，ターゲットとなる SX-Aurora TSUBASA ベクトルプロセッサ上のプログラム実行をさらに加速可能とすることにある．提案手法を，SPEC2000 の swim 及び NAS Parallel Benchmarks（NPB） の BT・CG を用いて SX-Aurora TSUBASA 上で評価を行った．swim に対しては OSCAR 自動並列化コンパイラの機能であるデータローカライゼーション手法を用いたキャッシュ最適化を適用した．OSCAR コンパイラによる自動並列化と NEC コンパイラによる自動ベクトル化の協調により，1 コア上での NEC ベクトル化単体に比べ．8 コア使用時に SPEC2000 swim で 3.52 倍の速度向上が得られた．また，NPB BT でも 6.75 倍，NPB CG で 56.92 倍の速度向上が確認できた

1. はじめに

科学技術計算や画像処理，機械学習等の分野において，プログラムの高速化に対する需要は高い．プログラム高速化の手段としてマルチコアやマルチプロセッサ上の並列処理が従来から利用されているが，手動並列化はプログラマの高い専門知識が必要である．

この問題を解決するため，筆者等は OSCAR 自動並列化コンパイラを開発している．コンパイラが自動的並列化を行うことによって高い専門知識を持たないプログラマでも容易に高い性能と高い生産性を得ることができる．OSCAR コンパイラはマルチグレイン並列化に加えて，データローカライゼーションによるキャッシュ最適化など様々な機能を有している．

高速化対象となるアプリケーションの特性に着目すると，科学技術計算などのプログラムは高いデータレベル並列性を有している場合が多い．そのためデータレベル並列性を利用した NEC の SX-Aurora TSUBASA ようなベクトルプロセッサや GPGPU[1]，Intel AVX のような SIMD 命令拡張などが広く利用されている．本稿で用いた SX-Aurora TSUBASA は，エッジモデルからデータセンターモデルまで用意されているスケーラブルなベクタースーパーコンピュータである．SX-Aurora TSUBASA はベクトルパイプラインを持つプロセッサコアを複数搭載したアクセラレータであり，ベクトル並列化に加えてコア間の並列性を利用可能である．SX-Aurora TSUBASA は高い性能を持っており，Intel Xeon CPU をベースとした FOCUS スパコンシステムより科学技術計算アプリケーションにおいて優位であることも報告されている [2][3]．また，姫野ベンチマークにおいては，SX-ACE に対しては最大 3.4 倍，Skylake に対しては最大 7.96 倍の性能を出していることも報告されている [4]．

NEC コンパイラは C/C++，Fortran プログラムを SX-Aurora TSUBASA 用に自動でベクトル化することができる．また，OSCAR 自動並列化コンパイラは逐次 C あるいは Fortran プログラムを入力とし，OpenMP ディレクティブを挿入された並列化 C・Fortran コードを出力する．NEC コンパイラは OpenMP に対応しているため，OSCAR 自動並列化コンパイラによって生成された並列化プログラ

[1] 早稲田大学
Waseda University.
[2] NEC データサイエンス研究所

Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

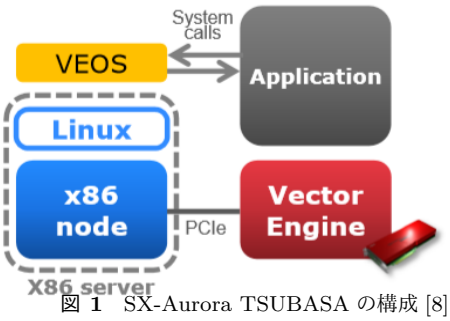

図 1　SX-Aurora TSUBASA の構成 [8]

ムを入力として受けることができる．そこで，本稿では，OSCAR 自動並列化コンパイラによって並列化したコードを，NEC コンパイラによってベクトル化するというコンパイルフローを提案する．本コンパイルフローにより，ネストされたループの外側ループを OSCAR コンパイラにより分割及び複数コアへ割り当て，内側ループを NEC コンパイラにより自動ベクトル化するといった階層的な並列処理が可能となる．さらに，OSCAR コンパイラが複数ループにまたがるデータ再利用性を利用して最適化するキャッシュ最適化も適用可能となる．本稿では上記提案手法を SPEC2000[5] の swim 及び NAS Parallel Benchmarks（NPB）[6], [7] の BT・CG を用いて評価を行った結果について報告する．

以下本稿では，第 2 章では評価環境として用いた NEC SX-Aurora TSUBASA について述べる．第 3 章では OSCAR 自動並列化コンパイラについて述べる．第 4 章では評価結果について述べる．

2. SX-Aurora TSUBASA

SX-Aurora TSUBASA は NEC によって開発されたベクトルコンピュータである．SX-Aurora TSUBASA は Vector Host（VH）と Vector Engine（VE）から構成されている．VH は x86 ノードであり，主に OS の処理を行う．VE は NEC のベクトルマルチコアプロセッサであり，アプリケーションの実行を行う．両者は PCIe で接続されている．

2.1 SX-Aurora TSUBASA の実行モデル

SX-Aurora TSUBASA のプログラムは，OS の処理を行う VH 上で起動され，VE 上で実行される．VH 上で VE の実行を制御するソフトウェアとして VEOS が提供されている．VE 上で動いているプログラムがファイルアクセスなどによりシステムコールを呼び出した際は，それらの処理を VH に委譲するため VH 上の VEOS とコミュニケーションをとる．**図 1** に実行モデルを示す．図の X86 Server が VH にあたる．

2.2 Vector Engine のアーキテクチャ

Vector Engine は 8 つのベクターコアを持つプロセッサである．メモリは HBM2 であり，それによって高いバンド幅を実現している．本稿で用いた VE type 10C のプロセッサの理論演算性能（倍精度）は 2.15TFLOPS であり，メモリ帯域は 0.75TB/s である．

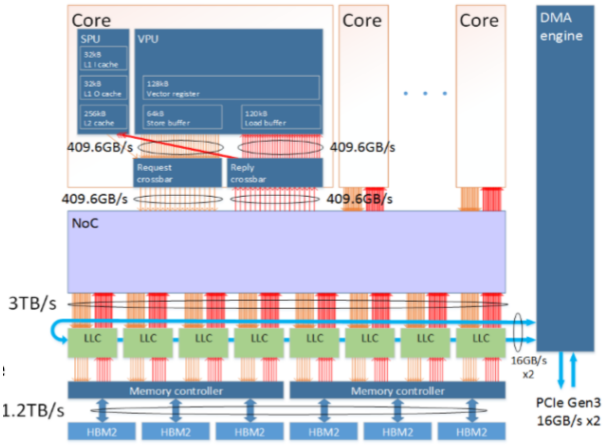

図 2　SX-Aurora TSUBASA Vector Engine のアーキテクチャ図 [8]

それぞれのコアが Scalar Processing Unit（SPU）と Vector Processing Unit（VPU）を持っている．VPU のベクトル長は最大 256 要素である．また，キャッシュはそれぞれのコアに L1・L2 キャッシュを持っており，LLC は 8 つのコアから共有されるキャッシュである．1 つのプロセッサは 8 つの LLC スライスを持ち，一つの LLC スライスは 2MB である．そして，データは 8 つの LLC スライスにインターリーブされて置かれる．また，ウェイ数は 4way で，ラインサイズは 128B である．SX-Aurora TSUBASA の VE のアーキテクチャ図を**図 2** に示す．

2.3 NEC コンパイラ

NEC が提供する SX-Aurora TSUBASA 用コンパイラは Fortran/C/C++に対応しており，自動ベクトル化・自動並列化の機能を有している．Vector Engine 上で動かすことのできるバイナリを出力する．

2.3.1 自動ベクトル化

この機能を用いると，コンパイラがプログラムを解析して，ベクトル化が可能と判定された部分に対してベクトル化をする [9] ことができる．

また，総和や累積，漸化式等の特殊なパターンにもベクトル命令が用意されていて，そのようなパターンも自動でベクトル化することが可能である．

2.3.2 自動並列化

NEC コンパイラは自動並列化の機能も持つ．この機能により，コンパイラがプログラム内を解析して，並列実行できるループや文の集まりを検出し，並列化のためにプログラムを変形および並列処理制御のための処理の挿入を自動的に行う [9]．

Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

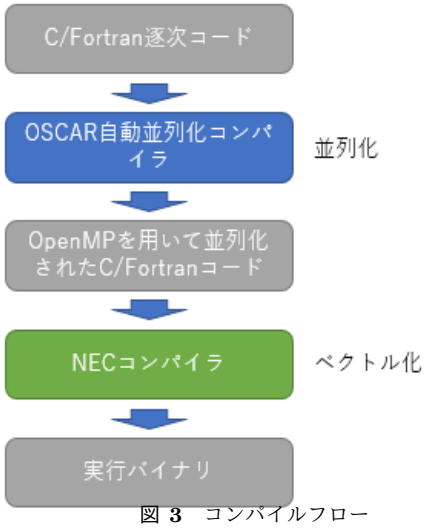

図 3　コンパイルフロー

2.4　ftrace

ftrace は Vecotr Engine 上で動かしたプログラムのプロファイルをとることができる．LLC キャッシュのヒット率や平均ベクトル長，ベクトル OP の割合などが取得することができる．

3.　OSCAR 自動並列化コンパイラ

OSCAR 自動並列化コンパイラは C/Fortran の逐次ソースコードを受け取り，OpenMP を用い並列化された C/Fortran コードを出力する source to source コンパイラである．本稿では，C/Fortran 逐次ソースコードを OSCAR コンパイラに通し，生成された並列 C/Fortran コードを NEC コンパイラに通しベクトル化を行うという**図 3** のようなコンパイルフローをとった．

本節では OSCAR コンパイラによる並列化とキャッシュ最適化，及び OSCAR コンパイラと NEC コンパイラの協調動作についてその概要を説明する．

3.1　OSCAR 自動並列化コンパイラの概要

OSCAR コンパイラは逐次プログラムから並列性を自動的に抽出し，階層的な並列化を行う．OSCAR コンパイラの特徴としてマルチグレイン並列処理 [10][11] が挙げられる．マルチグレイン並列処理とは，粗粒度並列化と中粒度並列化，近細粒度並列化を組み合わせることによりプログラム全体から並列性を抽出することができる．

3.1.1　粗粒度並列化

粗粒度並列化では，ソースプログラムを基本ブロック（BB），繰り返しブロック（RB），サブルーチンブロック（SB）の三種類のマクロタスクに分割し，これらのマクロタスク間にある並列性を利用する．並列化できない RB や SB の内部に対しては，階層的にマクロタスクを生成していく．

```
do i = 1, n
  a(i) = b(i)
enddo
do i = 1, n
  d(i) = a(i)
enddo
do i = 1, n
  f(i) = d(i)
enddo
```

図 4　ストリップマイニング前のコード例

```
do ii = i, n, step
  do i = ii, ii + step
      a(i) = b(i)
  enddo
  do i = ii, ii + step
      a(i) = b(i)
  enddo
  do i = ii, ii + step
      a(i) = b(i)
  enddo
enddo
```

図 5　ストリップマイニング後のコード例

3.1.2　中粒度並列処理

中粒度並列化は，ループイタレーション間の並列性を利用する並列処理である．OSCAR 自動並列化コンパイラでは，ループに対し，Doall や Dosum や Doacross といった判定を自動で行い，中粒度並列性を利用できる場合には，ループの種別に応じて並列化を行う．

3.2　キャッシュ最適化

ここでは，本稿で用いたループ整合分割 [12] によるキャッシュ最適化 [13] について説明する．

ループ内でアクセスする配列の合計サイズがキャッシュサイズより大きな場合，キャッシュヒット率が悪化する．そのようなキャッシュヒット率の悪化を防ぐために，ループ整合分割を用いて同一配列を使うループ群をそれぞれ分割する．

整合分割によるループ分割の際に，OSCAR コンパイラは分割によるコードの増大を防ぐため，ループの分割と同じ効果を持つ，ストリップマイニングによるコードコンパクションを行う．**図 4** のような，3 つのループに対してストリップマイニングを適用して融合を行うと，**図 5** のような 1 つのループになる．これにより，外側ループのステップ step の分だけ，内側の 3 つのループがそれぞれ回転するので分割しているのと同様の効果が得られる．

3.3　ベクトル化との協調

本稿で評価対象とした科学技術計算のプログラムでは，多重ループを含む場合が多く，そのような多重ループがプログラム実行時間の大半を占める．OSCAR コンパイラは並列化階層を自動決定する [14]．その際に，同期オーバーヘッドを避けるため，なるべく外側の階層で並列化を行う．すなわちループを並列化する際には，なるべく外側のループで並列化を試みる．そのため，内側ループを並列化することによって，ベクトル長が短くなってしまうことを防ぎ，並列化とベクトル化を両立することができる．

Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

表 1　Vector Engine Type 10C の諸元

Vector Core	8 cores
動作周波数	1.4 GHz
L1 I Cache	32KB
L1 D Cache	32KB
L2 Cache	256KB
LLC	16MB for 8 cores

4. NEC SX-Aurora TSUBASA 上での評価

SPEC2000 の swim および NAS Parallel Benchmark（NPB）の BT・CG を用いて，OSCAR コンパイラの自動並列化機能と NEC コンパイラの自動ベクトル化機能の協調による高速化手法を評価した．swim に対しては 3.2 節で述べたキャッシュ最適化を適用した．NAS Parallel Benchmark は NASA によって開発されたベンチマークであり，問題サイズを S・W・A・B・C から選ぶことができる．今回は評価にあたって，サイズの一番大きなクラス C を使用した．また，NAS Parallel Benchmarks のバージョンは 2.3 であり，C 言語に書き換えられたものを使用した．

4.1 評価環境

提案手法は第 2 節で述べた SX-Aurora TSUBASA 上で評価を行った．使用した VE のモデルは Type 10C である．VE Type 10C の諸元を**表 1** に示す．Type 10C の動作周波数は 1.4GHz で，8 つのベクトルプロセッサコアを持ち，LLC のサイズは 16MB である．また，使用したコンパイラは nfort/ncc であり，バージョンは 2.2.2 である．

4.2 swim

オリジナルのデータサイズは 1335 × 1335 だが，キャッシュ最適化の評価のためサイズを 8005 × 8005 に拡大した．その際に必要とされる配列のサイズの合計は 6.68GB である．そして，キャッシュ最適化は LLC をターゲットとして行った．swim の評価結果を**図 6** に示す．図は，NEC 自動ベクトル化により 1 コアのみ使用した場合の実行時間を基準として，OSCAR コンパイラと NEC コンパイラを組み合わせてコア数を 1 コアから 8 コアまで変化させたときの速度向上率，及び NEC コンパイラのみによる自動ベクトル化と自動並列化で 8 コア使用した場合の速度向上率を示す．図より，1 コア上での NEC コンパイラによるベクトル化単体に比べ，8 コア上で OSCAR コンパイラによる並列化と NEC コンパイラによるベクトル化の協調により，3.52 倍の速度向上が得られたことがわかる．

また，ftrace によって取得した LLC ヒット率を**図 7** に示す．キャッシュ最適化によって 1 コアで 33.83 %から 76.37 %に改善され，1.05 倍の速度向上が得られた．通常 LLC のヒット率が向上すると大幅な速度向上が得られることが多いが，ここで 1.05 倍の速度向上となったのは，

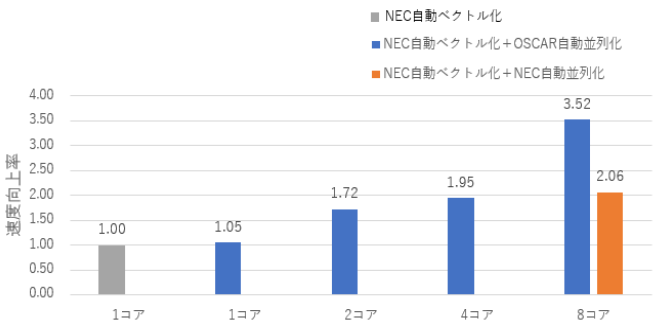

図 6　SPEC2000 swim 評価結果

表 2　キャッシュ最適化の前後での平均ベクトル長の変化

キャッシュ最適化前	キャッシュ最適化後
250.1	247.3

SX-Aurora TSUBASA の場合，LLC へのラインの割り当てがハッシュを用いた最適化方式となっているため，LLC ヒット率が低い場合でも高い処理性能を得られているためであると考えられる．8 コアを使用した場合も，NEC コンパイラのみによるベクトル化及び並列化の場合のヒット率が 33.90%だったものが 43.00%に改善されていることがわかる．

OSCAR コンパイラによって，キャッシュ最適化されたコードを省略および簡略化したものを**図 8** に示す．複数個の 2 重ループと 1 重ループから成るループ群に対して，キャッシュ最適化によってストリップマイニングされている．OSCAR コンパイラは外側の jj ループで並列化され，最内側で NEC コンパイラによってそれぞれベクトル化されている．図 8 では，内側の一つ目のループと三つ目のループでは，ストリップマイニング前と同じベクトル長が確保できており，並列化とベクトル化が両立できている．また，二つ目のループにおいてはストリップマイニング前よりベクトル長が短くなってしまっている．しかし，ベクトル長が短くなってしまったループは他のループよりも計算量が少ないためキャッシュ最適化によって，速度向上が得られたと考えられる．ftrace の結果より，平均ベクトル長は**表 2** のようにキャッシュ最適構えの 250.1 と最適化後の 247.3 と，ほとんど落ちていないことが分かる．

4.3 BT

BT の評価結果を**図 9** に示す．図は図 6 と同様に NEC 自動ベクトル化による 1 コアに対する速度向上率を表している．

図より 1 コア上での NEC コンパイラによるベクトル化単体に比べ，8 コア上で OSCAR コンパイラによる並列化と NEC コンパイラによるベクトル化の協調により，6.75 倍の速度向上が得られたことがわかる．また，NEC コンパイラによる並列化では，1.06 倍の速度向上しか得られな

Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

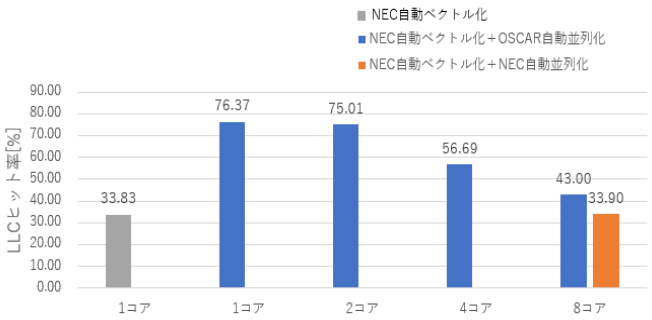

図 7　SPEC2000 swim LLC ヒット率

```
do jj = 1, N, step   並列化
    do j = jj , jj + step
        do i = 1, M   ベクトル化
            PNEW(I,J) = P(I,J) + ・・・
        enddo
    enddo
    do j = jj, jj + step   ベクトル化
            PNEW(M + 1,J) = P(1,J) + ・・・
    enddo
    do j = jj , jj + step
        do i = 1, M   ベクトル化
            P(I,J) = PNEW(I,J) + ・・・
        enddo
    enddo
enddo
```

図 8　キャッシュ最適化された swim コードのイメージ

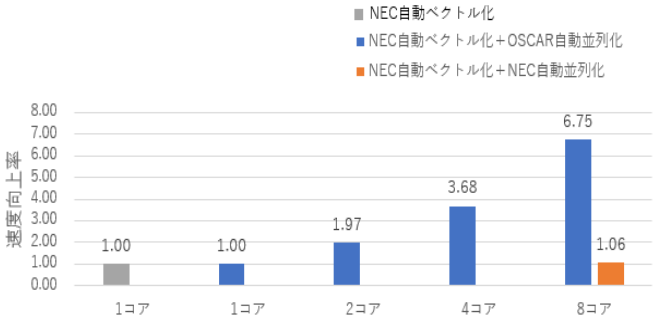

図 9　Nas Parallel Benchmarks BT 評価結果

かった．これは，実行時間の大半を占める，**図 10** のような関数呼び出しを含むループが並列化できていないためである．OSCAR コンパイラでは，解析によって外側の j ループでの並列化を実現している．このようなループが NEC コンパイラによって並列化されない原因として，NEC コンパイラではループ内に関数呼び出しがあった場合には cncall というディレクティブなしには並列化しないためである．そして cncall はユーザがその関数呼び出しを含むループが並列化できることを保証しなければならない． ftrace の結果より，NEC ベクトル化単体の平均ベクトル長は 16.8 で，OSCAR コンパイラによる 8 コア並列化後の平均ベクトル長は 16.8 と変化していないため，OSCAR コンパイラによる並列化は NEC コンパイラのベクトル化を阻害してないことが分かる．

```
for (j = 1; j < grid_points[1]-1; j++) {
  for (k = 1; k < grid_points[2]-1; k++) {
    binvcrhs( lhs[0][j][k][BB], lhs[0][j][k][CC],
        rhs[0][j][k] );
  }
}
```

図 10　NEC コンパイラで並列化されないコード例

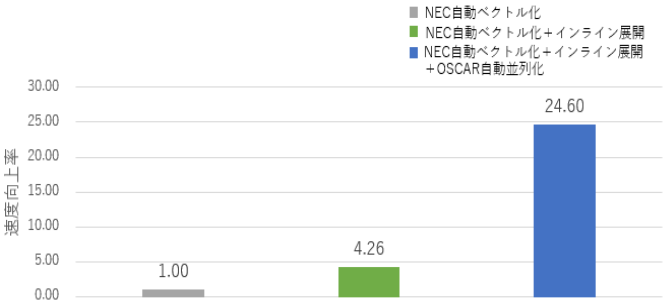

図 11　Nas Parallel Benchmarks BT インライン展開 評価結果

図 10 の binvcrhs はループを含まない関数である．そのため，binvcrhs 関数内ではベクトル化がされない．そして，呼び出し元である図 10 の内側 k ループは関数呼び出しを含むためベクトル化されない．関数呼び出しは NEC コンパイラによる並列化を阻害すると同様に，ベクトル化も阻害してしまう．そのため，インライン展開した評価も行った．その結果を**図 11** に示す．

インライン展開によって，図 10 の内側の k ループでベクトル化が行われた．外側の j ループでは OSCAR コンパイラによる並列化が行われるので，並列化とベクトル化が両立していると言える．

4.4　CG

CG は疎行列を扱うプログラムであり，間接参照を行うループが実行時間の 90 %以上を占めている．CG の評価結果を**図 12** に示す．図は図 6 と同様に NEC 自動ベクトル化による 1 コアに対する速度向上率を表している．

図より 1 コア上での NEC コンパイラによるベクトル化単体に比べ，8 コア上で OSCAR コンパイラによる並列化と NEC コンパイラによるベクトル化の協調により，56.92 倍の速度向上が得られたことがわかる．1 コアにおいて，NEC コンパイラによるベクトル化に単体に比べ，OSCAR コンパイラ 1 コアでは 11.28 倍の速度向上が得られた．**図 13** は CG のオリジナルのコードの一部である．NEC コンパイラはこのコードの内側ループはベクトル化している．しかし，OSCAR コンパイラが生成したコードに対しては内側ループのベクトル化に加え，図 13 の外側

Vol.2019-ARC-240 No.26
Vol.2019-SLDM-191 No.26
Vol.2019-EMB-53 No.26
2019/2/28

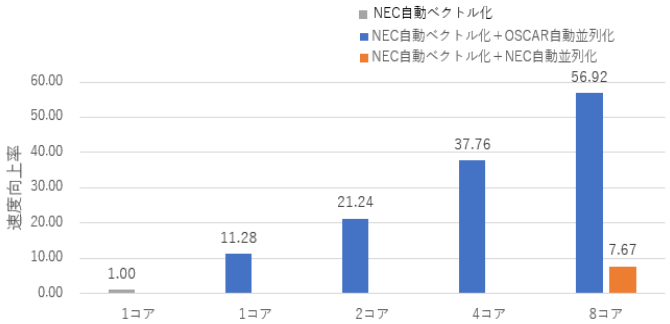

図 12　Nas Parallel Benchmarks CG 評価結果

```
for (j = 1; j <= lastrow-firstrow+1; j++) {
  sum = 0.0;
  for (k = rowstr[j]; k < rowstr[j+1]; k++) {
    sum = sum + a[k]*p[colidx[k]];
  }
  w[j] = sum;
}
```

図 13　cg の間接参照を行うコード部分

ループに相当するコード箇所の部分ベクトル化が NEC コンパイラによって行われたため，11.28 倍の速度向上が得られた．

図 13 において，OSCAR コンパイラが外側の j ループを並列化を行い，NEC コンパイラが内側の i ループをベクトル化をすることで，並列化とベクトル化を両立できている．

5. まとめ

本稿では，OSCAR 自動並列化コンパイラによる自動並列化と NEC コンパイラによる自動ベクトル化の協調による SX-Aurora TSUBASA 上での高速化手法を提案した．本手法を 8 個のベクターコアを搭載した SX-Aurora TUSBASA の Vector Engine Type 10C を用いて評価を行った．評価に用いたアプリケーションは，SPEC2000 swim と Nas Parallel Benchmark（NPB）の BT と CG である．swim に対しては，OSCAR コンパイラによる自動並列化に加え，データローカライゼーションによるキャッシュ最適化を適用した．1 コア上での NEC ベクトル化単体に比べ．8 コア使用時に swim で 3.52 倍，BT で 6.75 倍，CG で 56.92 倍の速度向上を得ることができ，提案する OSCAR コンパイラによるマルチコア並列化及びキャッシュ最適化と NEC コンパイラによる自動ベクトル化の協調動作が有効であることを確認した．

NEC SX-Aurora TSUBASA 上で，各種アプリケーションの大幅な速度向上が図れる可能性があることが確かめられたため，今後より広い範囲のアプリケーションに本手法を適用し，性能向上を確認していく予定である．

参考文献

[1] NVIDIA Tesla. https://www.nvidia.com/ja-jp/data-center/tesla/.
[2] 西川武志. 各種計算科学アプリケーションにおける nec sx-aurora tsubasa システムの性能評価 (1). 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2018, No. 17, pp. 1–4, 2018.
[3] 西川武志. 各種計算科学アプリケーションにおける nec sx-aurora tsubasa システムの性能評価 (2). 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2019, No. 13, pp. 1–4, 2019.
[4] Kazuhiko Komatsu, Shintaro Momose, Yoko Isobe, Osamu Watanabe, Akihiro Musa, Mitsuo Yokokawa, Toshikazu Aoyama, Masayuki Sato, and Hiroaki Kobayashi. Performance evaluation of a vector supercomputer sx-aurora tsubasa. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 685–696. IEEE, 2018.
[5] SPEC 2000 CPU. https://www.spec.org/cpu2000/.
[6] NAS Parallel Benchmarks. http://www.nas.nasa.gov/Software/NPB/.
[7] David H Bailey and Paul O Frederickson. Performance results for two of the nas parallel benchmarks. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pp. 166–173, 1991.
[8] Yohei Yamada and Shintaro Momose. Vector engine processor of nec's brand-new supercomputer sx-aurora tsubasa. In *Proceedings of A Symposium on High Performance Chips, Hot Chips*, Vol. 30, pp. 19–21, 2018.
[9] NEC SX-Aurora TSUBASA Documentation. https://www.hpc.nec/documents/.
[10] Hironori Kasahara, Hiroki Honda, A Mogi, A Ogura, K Fujiwara, and Seinosuke Narita. A multi-grain parallelizing compilation scheme for oscar (optimally scheduled advanced multiprocessor). In *International Workshop on Languages and Compilers for Parallel Computing*, pp. 283–297. Springer, 1991.
[11] 木村啓二, 小高剛, 小幡元樹, 笠原博徳. Oscar チップマルチプロセッサ上でのマルチグレイン並列処理. 情報処理学会研究報告. ARC, 計算機アーキテクチャ研究会報告, Vol. 150, pp. 29–34, nov 2002.
[12] 吉田明正, 越塚健一, 岡本雅巳, 笠原博徳ほか. 階層型粗粒度並列処理における同一階層内ループ間データローカライゼーション手法. 情報処理学会論文誌, Vol. 40, No. 5, pp. 2054–2063, 1999.
[13] 石坂一久, 中野啓史, 八木哲志, 小幡元樹, 笠原博徳ほか. 共有メモリマルチプロセッサ上でのキャッシュ最適化を考慮した粗粒度タスク並列処理. 情報処理学会論文誌, Vol. 43, No. 4, pp. 958–970, 2002.
[14] 小幡元樹, 白子準, 神長浩気, 石坂一久, 笠原博徳ほか. マルチグレイン並列処理のための階層的並列性制御手法. 情報処理学会論文誌, Vol. 44, No. 4, pp. 1044–1055, 2003.