2.5　SMPにおけるスレッド並列の台数効果と高速化手法

金澤 正憲[1]　義久 智樹[1]　杉崎 由典[2]　青木 正樹[2]
京都大学[1]　富士通(株)[2]

この報告は、『杉崎由典，青木正樹，義久智樹，金澤正憲：SMP におけるスレッド並列の台数効果と高速化手法について，情報処理学会研究報告「2005 年並列／分散／協調処理に関する「武雄」サマー・ワークショップ（SWoPP 武雄 2005)」，2005-EVA-14，pp.1-6，2005　8 月』をもとに、手を加えたものである。

1. はじめに

京都大学学術情報メディアセンターでは、2004 年 3 月にスーパーコンピュータを、ベクトル並列機（富士通 VPP800/63）から、SMP クラスタ（富士通 PRIMEPOWER　HPC2500/12 ノード）に置き換えた。ここでは、VPP800 導入時に、ベクトル計算機の性能評価に用いられたベンチマーク 3 本を取り上げ、ＳＭＰクラスタで実行し、並列台数効果を測定した。台数効果の十分でないベンチマークに対して、ソースプログラムの簡単な書き換えを行えば、自動並列で、台数効果が著しく改善されることを確認した。また、MPI で書き換えたベンチマークを実行し、MPI による並列処理との比較を行ったが、自動並列によるスレッド並列と大差なかった。

2. ベンチマークについて

ここで用いたベンチマークプログラムは、3 本で、grad、 product5、 shift3 と呼んでいる。概要を表 1 に示す。

表1：ベンチマーク概要

名称	プログラムの概要
grad	近傍との差分
product5	行列積
shift3	値の伝播（他の影響を含む）

それぞれのプログラムの主要部分を附録に記した。どれも 2 次元配列を扱うプログラムで、配列の大きさを増減することにより、いろいろな演算速度のコンピュータで実行できるようになっている。

3. SMP クラスタでの測定結果

3 本のプログラムについて、オリジナルソースのまま自動並列化した場合、高速化のためのチューニングを行った場合、MPI で記述した場合について実測し、3 つの場合の結果をまとめて図示する。測定結果については、特に断らない限り、1CPU 向きの最適化を施したプログラムの実行時間を 1 として表示している。また、実行時間とともにモニタ機能から収集された性能情報をもとに、結果を分析する。

3.1 測定システムおよび環境

ここで使用した測定マシンと条件（パラメータなど）を表 2 に示す。

表 2．測定マシンと条件

実行環境	富士通 PRIMEPOWER HPC2500 （SPARC64V 1.82GHz） Parallelnavi2.4
並列数	128CPU,512GB の 1 ノード上にて 1～120CPU を使用
翻訳時オプション	-Kfast_GP2=3,prefetch=4, parallel=3,V9, largepage=2,hardbarrier -w

3.2 オリジナルソースプログラムの場合

各プログラムに手を加えることなく実行させて得られた結果から判明した性能上の問題を以下に示す。

(1)grad

TLB ミス、L2 キャッシュミスが多発している。

同一ループ内に配列の次元アクセス順序が異なる文が存在するため、連続アクセスとストライドアクセスが混在している。混在によりコンパイラの最適化が効かず、ストライドアクセスによる TLB ミスにより性能が劣化している。

(2)product5

matmul 並列ライブラリ呼出しに展開されている。

(3)shift3

L2 キャッシュミスが多発している。

10 個の do 文があるが、1 つ目の do 文を実行した後の do 文実行ではアクセスするデータがキャッシュに残っていない。そのためミスが多発し、性能が劣化している。

3.3 最適化を行なった場合

オリジナルソースコードの性能検証から、性能影響要因への対処方法を検討・実施し、性能向上を図った。実施した性能向上策と効果を以下に示す。

(1)grad

メモリアクセスを全て連続アクセスにするため、ループ分配及びループ交換を行った。図 a 参照。これにより、データの局所性が高まり、モニタ機能による解析情報から、TLB ミス及び L2 キャッシュミスが削減されたことを確認した。

```
      DO 300 J=1,N-1
        DO 300 I=1,N
          B(I,J)=A(I,J+1)-A(I,J)
300       C(J,I)=A(J+1,I)-A(J,I)
                ↓  変更
      DO 300 J=1,N-1
        DO 300 I=1,N
300       B(I,J)=A(I,J+1)-A(I,J)
      DO 310 I=1,N
        DO 310 J=1,N-1
310       C(J,I)=A(J+1,I)-A(J,I)
```

図 a.　チューニング例（grad　N=50000）

チューニング前後のメモリ性能を表 3 に示す。図 1 に台数効果を示す。この 2 つの結果から、ループ分解と交換が非常に効果の大きいことが判る。

表 3.　メモリ性能(grad)

チューニング	L2 キャッシュミス(%)	TLB ミス (%)	メモリアクセス(%)
前	32.9950	0.5789	98.39
後	3.4604	0.0000	41.98

(N=50000, CPU=4)

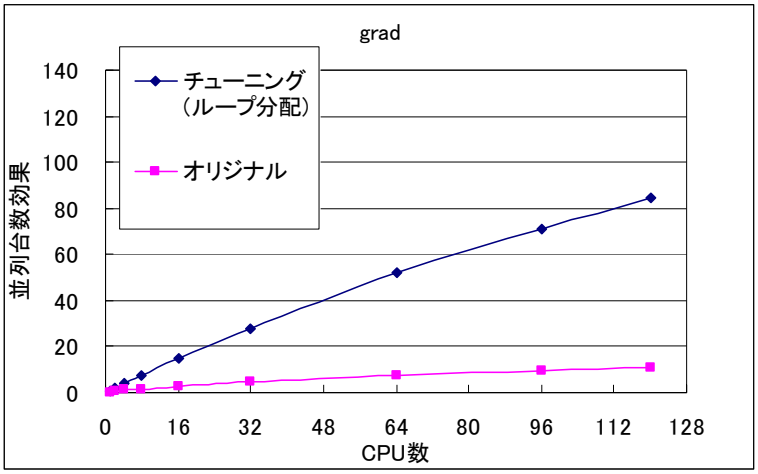

図 1.　grad チューニング効果

(2)product5

matmul 並列ライブラリ呼出しに展開されているため、図 2 に示すとおり、十分高速な結果が得られている。ループ分配最適化を行うことで、配列のゼロクリア処理の部分に、ループ分配最適化を行ったが、余り大きな効果はなかった。なお、64CPU 以上で並列台数効果が低下するのは、行列積(N=12000)を並列処理した場合の計算粒度が十分でないためと思われる。N=100000（プログラムの大きさ約 230GB）とし、2006 年 6 月に測定をしなおすと、図 3 に示すようになり、並列効果は、むしろ、スーパーリニアになった。

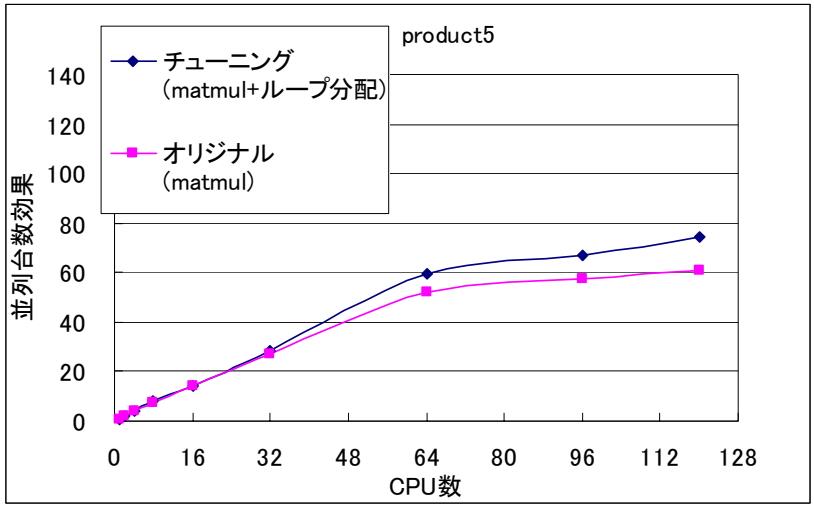

図 2. product5 チューニング効果（N=12000）

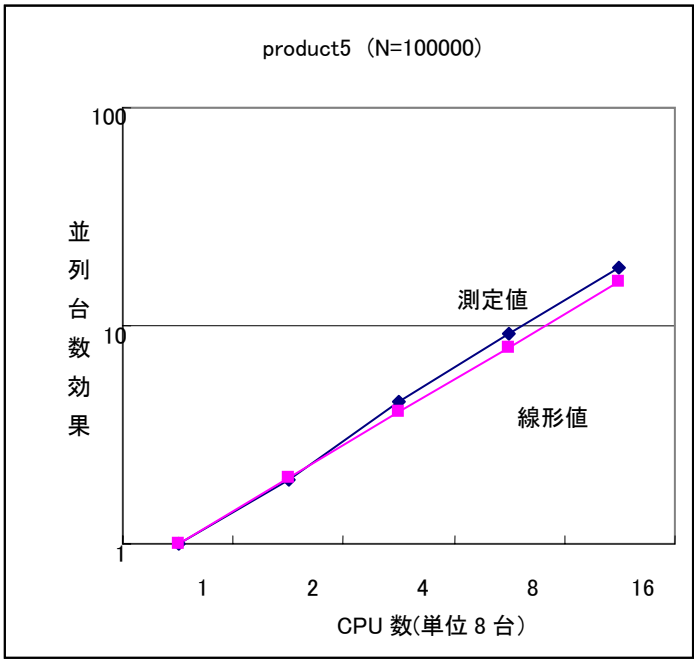

図 3. Product5 自動並列 (N=100000)

(3)shift3

ループ融合を行うことで、内側のループでアクセスする次元方向の配列領域が局所化され、次のループ処理の際にはその次元方向のデータは全てオンキャッシュとなる。チューニングについては、図 b を参照。図 4 に台数効果を示す。

チューニング前後のメモリ性能を表 4 に示す。明らかに、非常に大きな効果があったことが判る。

表 4. メモリ性能(shift3)

チューニング	L2 キャッシュミス(%)	TLB ミス (%)	メモリアクセス(%)
前	0.9596	0.0000	42.89
後	0.0998	0.0000	10.07

(N=50000, CPU=4)

```
      DO 10 J=1,NN
      DO 10 I=1,NN
       A(I,J)=A(I+1,J)+0.48*B(I,J)+0.122*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
10    CONTINUE
       …
      DO 100 J=1,NN
      DO 100 I=1,NN
       A(I,J)=A(I+2,J)+0.46*B(I,J)+0.118*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
100   CONTINUE
                  ↓  変更
      DO J=1,NN
        DO 10 I=1,NN
         A(I,J)=A(I+1,J)+0.48*B(I,J)+0.122*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
10      CONTINUE
       …
        DO 100 I=1,NN
         A(I,J)=A(I+2,J)+0.46*B(I,J)+0.118*(B(I-1,J)+ B(I,J-1)+B(I+1,J)+B(I,J+1))
100     CONTINUE
      ENDDO
```

図 b. チューニング例（shift3　N=100000）

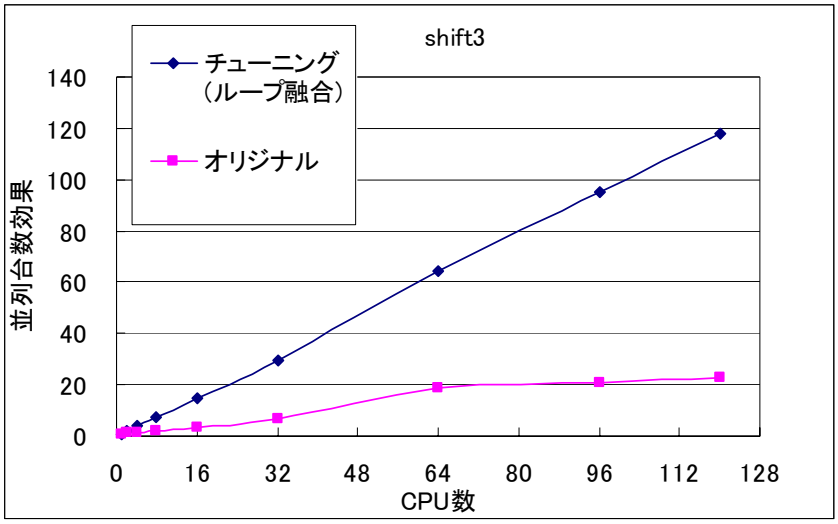

図 4. shift3 チューニング効果

3.4 MPI で書き直した場合

チューニングを施したソースコードを用い、これを MPI へ書き換えを行った。書き換えに際しては、主要ループ内での通信が発生しないような方法にて行っている。MPI への書き換え方法と性能結果を以下に示す。

(1)grad

スレッド並列と同様に、配列の 2 次元目を並列化した。

主要ループ中には通信処理がないこともあり、並列台数効果もスレッド並列と同等の性能を示している。図 5 参照。

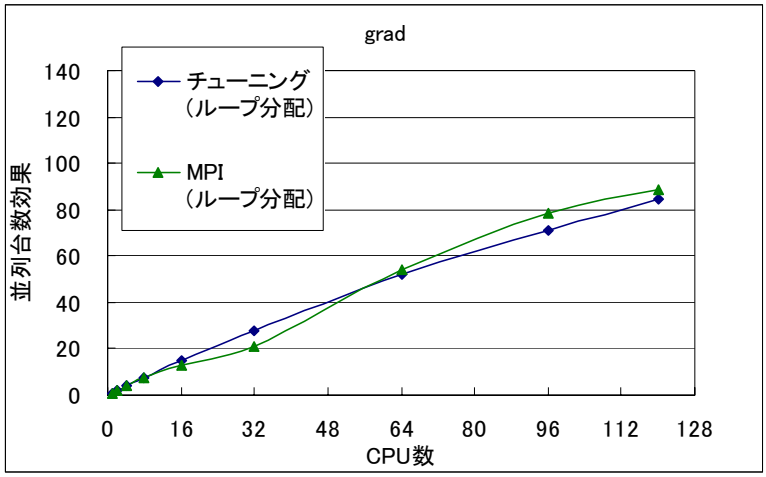

図 5. grad 自動並列と MPI の性能比較

(2)product5

一部の配列を各プロセスで重複して保持することにより、主要ループ中に通信処理が入らないようにしている。その影響で matmul 関数ライブラリ呼出しに展開されなくなった。そのため、性能がスレッド並列と比較して大きく低下している。図 6 参照。

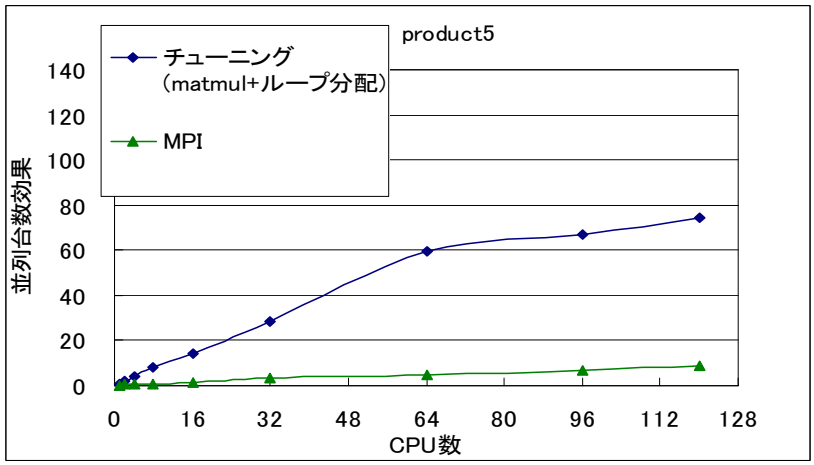

図 6. product5 自動並列と MPI の性能比較

(3)shift3

スレッド並列と同様に、配列の 2 次元目を並列化した。

主要ループ中には通信処理がないこともあり、並列台数効果もスレッド並列と同等の性能を示している。図 7 参照。

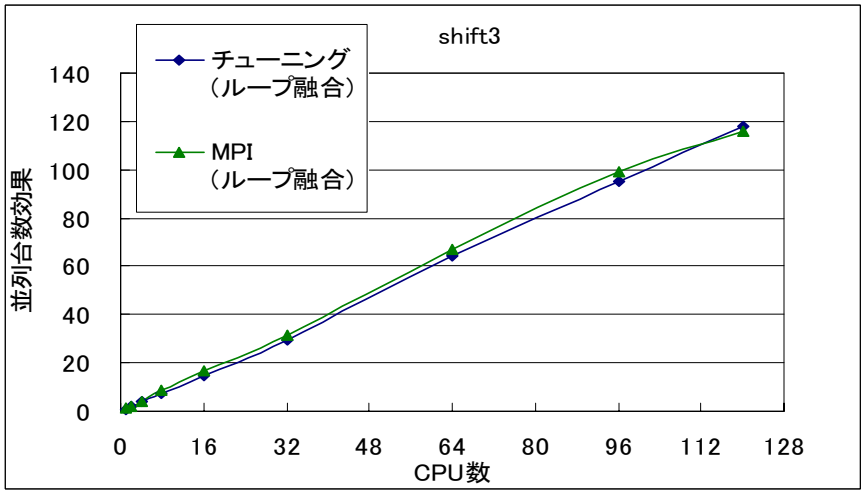

図 7. shift3 自動並列と MPI の性能比較

4. VPP との性能比較

Fujitsu　VPP5000 の 1CPU で同じプログラムを実測し、HPC2500 の実測値と比較した。さらに、スレッド並列で 32CPU の場合も測定した。表 5 に測定値及び比較結果を示す。効果の欄については HPC2500 の 1CPU を 1 とした際の性能効果を表している。

VPP5000 はピーク性能 9.6GFlops、HPC2500 はピーク性能 7.28GFlops である。LINPACK によると、SMP の性能はピークの 50%と換算するのが、妥当と考えられ、VPP5000 と HPC2500 は、約 2.6 倍と推測される。shift3 では、同程度の性能が出ているといえる。

表 5：VPP5000 と HPC2500 の性能比較

Bench mark	Size(N)		VPP5000	HPC2500	
		CPU 台数	1	1	32
grad	500	時間	2,864	25,611	967
		効果	8.94	1.00	26.49
	5000	時間	393,423	2,679,202	101,226
		効果	6.81	1.00	26.47
product5	12000	時間	2,147	903	33
		効果	0.42	1.00	27.36
shift3	1000	時間	14,766	46,520	1,411
		効果	3.15	1.00	32.97
	10000	時間	1,344,125	4,765,837	154,533
		効果	3.55	1.00	30.84

注：時間は grad, shift3 が μ sec, product5 が sec。

5. おわりに

京都大学学術情報メディアセンターで従来から用いてきたベンチマークプログラムによって、SMP による自動並列処理の効果を実測し、評価した。共有メモリ型システムにおいて、自動並列化による効果が、十分多数の台数まで（今回は 120CPU まで)、ほぼ線形的に向上することがわかった。

高速処理をするためには、よく知られた並列化手法であるループ分配、ループ交換、ループ融合が大いに有効であることが実証できた。このことから、コンパイラがループ分配、交換、融合などの典型的な並列化技法を自動的に取り入れる、または、プログラマに候補の技法と場所を推定して、助言を行う必要がある。さらに、会話的に並列化機能とその効果を直ちに判定できる機能が不可欠であろう。

自動ベクトル化コンパイラが種々の機能を順次、取り入れたように、自動並列化コンパイラも今後絶えず新しい機能を盛り込むことが重要である。

product5 の行列積のように、並列化、最適化を駆使するよりも、既存の並列ライブラリを用いたほうが、圧倒的に性能が高いことがわかった。その上、プログラムも

c=matmul(b,c)

と簡潔な形で記述できるので便利である。

したがって、まず、第 1 の解決策として、並列ライブラリの普及にセンターとしては努める必要があると自覚した。一方、コンパイラがソースを読みきって並列ライブラリを組み込むようにすることも非常に有効である。導入当初のコンパイラは、行列積の計算であると読み切る能力が弱かったが、現在は改善されている。

上述したように、自動並列化コンパイラは、ソース解析技術の向上が必要であると考えられる。

SMP マシンは、巨大な主記憶を共有するため、CPU 数が多くなると、メモリアクセスで競合が起こるという問題が指摘されていた。そこで、ベンチマークプログラムを MPI で書き直して、その実行時間を実測したところ、メモリアクセスの競合に関しては、ほとんどないと推測される。

さらに、VPP5000 の 1CPU で実行した結果と比較すると、同じ 1CPU なら VPP5000 が 3 倍から 8 倍程度速いことが確認できた。

これらの結果は、ハードおよびソフトの並列技術の発展に有意義なデータを与えるものと考えている。

今後も利用者プログラムを収集して、ベンチマークを整備していきたい。

最後に、助言をいただいた京都大学学術情報メディアセンター岩下武史助教授、富士通㈱荒川征己氏、並びに、関係者各位に深く感謝します。

参考文献

1）草野義博，新庄直樹；ハイパフォーマンスコンピュータ：PRIMEPOWER HPC，Fujitsu, Vol.53, No.6, pp.444-449，2002.

2）富士通；Parallelnavi Fortran 使用手引書，マニュアル

3）富士通；Parallelnavi Fortran 文法書，マニュアル

附録 1:ベンチマークプログラムの主要ループ部

各プログラムの主要ループ部を以下に示す。

附表1：grad

```
      DO 300 J=1,N-1
        DO 300 I=1,N
          B(I,J)=A(I,J+1)-A(I,J)
 300      C(J,I)=A(J+1,I)-A(J,I)
      DO 400 I=1,N
        B(I,N)=A(I,1)-A(I,N)
 400    C(N,I)=A(1,I)-A(N,I)
      DO 500 J=1,N
        DO 500 I=1,N
  500     A(I,J)=ATAN2(B(I,J),C(I,J))
```

gradでは、配列のｘ軸方向及びｙ軸方向の差分を求め、結果に対してatan2演算を行っている。

附表2：product5

```
      DO 300 I=1,N
        DO 300 J=1,N
          C(I,J)=0
          DO 300 K=1,N
 300        C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

product5では行列積を行っている。

附表3：shift3

```
      DO 10 J=1,NN
      DO 10 I=1,NN
      A(I,J)=A(I+1,J)+0.48*B(I,J)+0.122*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
10    CONTINUE
      DO 20 J=1,NN
      DO 20 I=1,NN
      A(I,J)=A(I+3,J)+0.44*B(I,J)+0.114*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
20    CONTINUE
      DO 30 J=1,NN
      DO 30 I=1,NN
      A(I,J)=A(I+5,J)+0.40*B(I,J)+0.106*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
30    CONTINUE
      DO 40 J=1,NN
      DO 40 I=1,NN
      A(I,J)=A(I+7,J)+0.36*B(I,J)+0.098*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
40    CONTINUE
      DO 50 J=1,NN
      DO 50 I=1,NN
      A(I,J)=A(I+9,J)+0.32*B(I,J)+0.090*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
50    CONTINUE
      DO 60 J=1,NN
      DO 60 I=1,NN
      A(I,J)=A(I+10,J)+0.30*B(I,J)+0.086*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
60    CONTINUE
      DO 70 J=1,NN
      DO 70 I=1,NN
      A(I,J)=A(I+8,J)+0.34*B(I,J)+0.094*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
70    CONTINUE
      DO 80 J=1,NN
      DO 80 I=1,NN
      A(I,J)=A(I+6,J)+0.38*B(I,J)+0.102*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
80    CONTINUE
      DO 90 J=1,NN
      DO 90 I=1,NN
      A(I,J)=A(I+4,J)+0.42*B(I,J)+0.110*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
90    CONTINUE
      DO 100 J=1,NN
      DO 100 I=1,NN
      A(I,J)=A(I+2,J)+0.46*B(I,J)+0.118*(B(I-1,J)+B(I,J-1)+B(I+1,J)+B(I,J+1))
 100   CONTINUE
```

shift3では、配列Aのx軸方向に1～10要素離れた要素に対して、配列B及び配列Bの4近傍の係数付きの加算結果を加えている。

別紙：

SMP におけるスレッド並列の多数台の台数効果

京都大学学術情報メディアセンター
金澤　正憲

前述の『**SMPにおけるスレッド並列の台数効果と高速化手法**』でのベンチマークgradにより、多数のCPUでの台数効果を調べてみた。1回だけの実行で得られた結果を表1に示す。

測定の結果、127台の場合を除いて、台数効果は見られた。複数回実行させて、比較しなければならないが、未だ台数効果があるという傾向は明白になったと判断できる。

表１　台数効果（120～128CPU）

Grad（N=100000）測定日　2006.9.1

台数	経過時間	比率	台数比率
120	12.6848	1.000	1.000
121	12.6360	1.004	1.008
122	12.5131	1.014	1.017
123	12.4507	1.019	1.025
124	12.3664	1.026	1.033
125	12.3001	1.031	1.042
126	12.2559	1.035	1.05
127	12.2646	1.034	1.058
128	12.1948	1.040	1.067

注。京都大学学術情報メディアセンターの HPC2500（128CPU、512GB、1.82GHz）