H26年度
スーパーコンピュータの高速化技法入門
演習用資料

2014年　6月17日

大阪大学サイバーメディアセンター
日本電気株式会社

本資料は，東北大学サイバーサイエンスセンターとＮＥＣの
共同により作成され，大阪大学サイバーメディアセンターの
環境で実行確認を行い，修正を加えたものです．

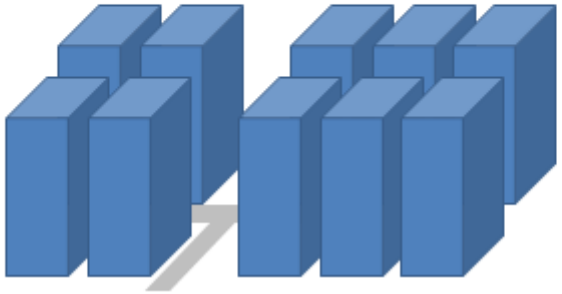
SX-9の計算ノード構成

全１０ノード構成

- 1ノードあたり 16CPU
- メモリ　１ノードあたり1Tバイト（共有）
- SX9キューで最大128CPU（8ノード）まで利用可能(8ノード利用は届出制)

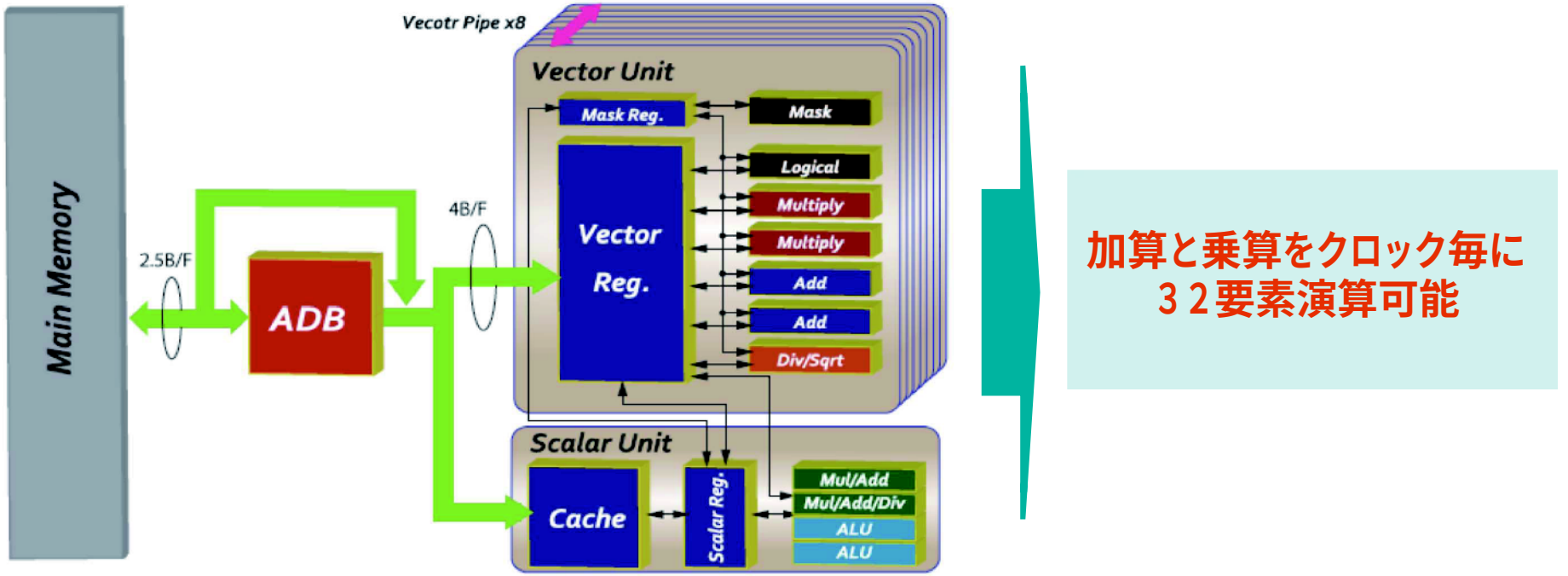
スーパーコンピュータ SX-9
全10ノード

1コア・・・・・・・・102.4ギガFLOPS
1ノードあたり
ベクトルプロセッサ・・・・・16CPU
共有物理メモリ・・・・・1テラバイト



SX-9のCPU（ベクトル演算部分）

- 乗算パイプライン　２ｘ８ベクトルユニット（＝１６）
- 加算パイプライン　２ｘ８ベクトルユニット（＝１６）
- ピーク性能　（１６＋１６）ｘ３．２GHz＝１０２．４GFLOPS/CPU



行列積のプログラムを使った課題

- オリジナルコードのコンパイルと実行
- 性能解析 (ftraceの利用)
- アンローリング
  - outerunroll指示行 (指示行による最適化)
- 自動インライン展開 (コンパイルオプションによる最適化)
- 行列積ライブラリ (コンパイラによる最適化)
- 自動並列化 (コンパイルオプションによる最適化)

演習問題の構成

ディレクトリ構成

```
super/
├-- practice_1    オリジナルコード実行環境
├-- practice_2    性能解析（ftrace）演習問題
├-- practice_3    outerunroll指示行演習問題
├-- practice_4    自動インライン展開演習問題
├-- practice_5    行列積ライブラリ
└-- practice_6    自動並列化
```

プログラム最適化の流れ

MFLOPS　～１０GF
VLEN　～１００
V.Op.Ratio　～９８%
は、改善の余地あり

指示行とは。。。
コンパイラは、最適化を行う上でソースコード上からは判断できない条件があった場合、最適化を抑止します。ユーザーが明示的に指示行で条件を与えてあげることにより、最適化を促進させることが可能になります

1. 演習問題：オリジナルコード

目的

- 現状のプログラムの性能を把握する

手順

- コンパイル（リストの確認）
- 実行（結果、性能の確認）

ディレクトリ

- practice_1

1．オリジナルコード：コンパイル（1）

コンパイラオプション

- コンパイラバージョン表示（-V）
- SX-9向けの命令列を生成することを指定(-cfsx9)
- 編集リスト、変形リストを採取（-R2）
- 詳細診断メッセージを表示（-Wf, -pvctl fullmsg）

```
sxf90 -V -Cvopt –cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune0.f
```

コンパイル方法
```
% ./comp.sx.sh
```

1．オリジナルコード：コンパイル（2）

変形リスト（mat_tune0.L）

```
      25          do j=1,n
      26            do k=1,n
      27              do i=1,n
      28                a(i,j)=a(i,j)+b(i,k)*c(k,j)
      29              enddo
      30            enddo
       .          do k = 1, 2048, 4
       .  !cdir     nodep
       .  !cdir     on_adb(a,b)
       .            do i = 1, 2048
       .               a(i,j) = a(i,j) + b(i,k)*dble(c(k,j)) + b(i,k+1)*dble(c(k+1,
       .         1        j)) + b(i,k+2)*dble(c(k+2,j)) + b(i,k+3)*dble(c(k+3,j))
       .            enddo
       .          enddo
      31          enddo
```

4段アウターアンロールが行われる
⇒配列aのメモリアクセスの回数が
1/4になるため高速化される

編集リスト（mat_tune0.L）

V：ベクトル化対象ループ

```
      25: +------>          do j=1,n
      26: |V----->            do k=1,n
      27: ||V---->              do i=1,n
      28: |||     A               a(i,j)=a(i,j)+b(i,k)*c(k,j)
      29: ||V----               enddo
      30: |V-----             enddo
      31: +------           enddo
```

1. オリジナルコード：実行

ジョブファイル(run.sx.sh)

```
#!/bin/csh
#PBS -q PCC
#PBS -l  cpunum_job=1,elapstim_req=0:10:00,memsz_job=1GB
#PBS -j o -N p1-sx-sample

cd $PBS_O_WORKDIR

timex ./a.out
```

NQSⅡオプション
- -q ジョブクラス名を指定
- -l 使用CPU数、経過時間、メモリ容量の申告
- -jo 標準エラー出力を標準出力と同じファイルへ出力する
- -N ジョブ名を指定

実行

```
% qsub run.sx.sh
Request 14620.sx9 submitted to queue: DBG9.
```

1．オリジナルコード：実行結果

結果ファイル（p1-sx-sample.o*****）→ 約33GFLOPS

```
******  Program Information  ******
Real Time (sec)                :          0.520426
User Time (sec)                :          0.516751
Sys  Time (sec)                :          0.003175
Vector Time (sec)              :          0.516180
Inst. Count                    :        232350334.
V. Inst. Count                 :        117679771.
V. Element Count               :      30126016250.
FLOP Count                     :      17179869353.
MOPS                           :      58520.809467
MFLOPS                         :      33245.933444
A. V. Length                   :        255.999956
V. Op. Ratio (%)               :         99.620807
Memory Size (MB)               :        256.031250
MIPS                           :        449.636932
I-Cache (sec)                  :          0.000162
O-Cache (sec)                  :          0.000182
Bank Conflict Time
  CPU Port Conf. (sec)         :          0.003185
  Memory Network Conf. (sec)   :          0.126729
```

プログラムインフォメーションの出力

2．演習問題：性能解析（ftraceの利用）

目的

- 性能解析ツールftraceを使い、性能情報を採取する

手順

- ソースコードの修正（ftrace_regionの挿入）
- コンパイルオプションの追加（-ftrace）
- 実行（結果、性能の確認）

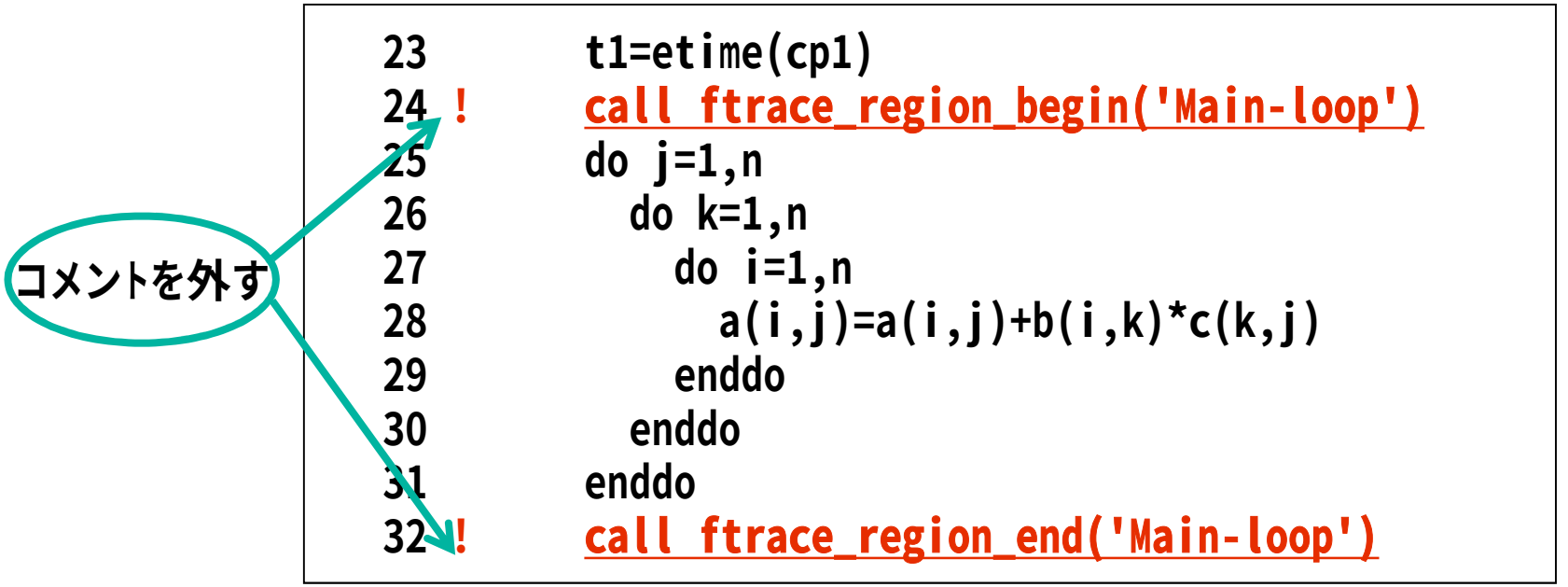
ディレクトリ

- practice_2

2．性能解析（ftraceの利用）：ソースコード修正

mat_tune0.fへftrace_regionを挿入

- プログラムの局所的な部分の性能を知りたい場合に使用する
  - 通常のftraceはサブルーチン単位での情報を表示
    ⇒ループ単位で細かく情報採取が可能
- call ftrace_region_begin/endで測定したい区間をはさむ



2．性能解析（ftraceの利用）：コンパイルオプションの追加

-ftraceをcomp.sx.shに追加

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune0.f -ftrace
```

コンパイルオプションの意味

- -ftrace　簡易性能解析機能を利用する

※注意
-ftraceオプションは測定オーバーヘッドが生じるため、実行回数の多いサブルーチンがある場合には、実行時間が延びます。そのため、常に使用することはお勧めしません。

2．性能解析（ftraceの利用）：コンパイルと実行

- コンパイル
  - % ./comp.sx.sh
- 実行
  - % qsub run.sx.sh

2．性能解析（ftraceの利用）：実行結果

結果ファイル（p2-sx-sample.o*****）→ 約33GFLOPS

```
Execution Date : Thu May 10 16:57:42 2012
Total CPU Time : 0:00'00"526 (0.526 sec.)

PROC.NAME  FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS V.OP  AVER.    VECTOR I-CACHE O-CACHE   BANK CONFLICT
                      TIME[sec](  % )    [msec]                    RATIO V.LEN     TIME    MISS    MISS CPU PORT  NETWORK

main_             1     0.526(100.0)   525.576  57538.1  32687.7 99.62 256.0     0.525   0.000   0.000    0.004    0.154
-------------------------------------------------------------------------------------------------------------------------
total             1     0.526(100.0)   525.576  57538.1  32687.7 99.62 256.0     0.525   0.000   0.000    0.004    0.154

Main-loop         1     0.525( 99.8)   524.600  57552.0  32748.5 99.62 256.0     0.525   0.000   0.000    0.004    0.154
-------------------------------------------------------------------------------------------------------------------------
 matrix_size = 2048
  p_name       |  user(sec)|   moda       check
  mat_tune0.f_ |     0.525 |     0       0.204800000000000D+04
    32.744 (GFlops)
```

ftraceの情報

ftrace_regionの情報

プログラムの出力

3．演習問題：outerunroll指示行

目的

- アウターアンローリング指示行の使い方を理解する
- コンパイラは４段アウターアンロールを行っていたので、指示行で８段にしてみる

手順

- ソースコードの修正
- コンパイル（リストの確認）
- 実行（結果、性能の確認）

ディレクトリ

- practice_3

3．outerunroll指示行：ソースコード修正

- mat_tune0.fをmat_tune.fにコピーしてから、mat_tune.fを修正
  - % cp mat_tune0.f mat_tune.f
  - % vi mat_tune.f
- 8段outerunroll指示行の挿入例

段数は2のべき乗の値のみ有効

```
25        do j=1,n
26 !cdir outerunroll=8
27          do k=1,n
28            do i=1,n
29              a(i,j)=a(i,j)+b(i,k)*c(k,j)
30            enddo
31          enddo
32        enddo
```

3．outerunroll指示行：コンパイルと実行

コンパイル

- % ./comp.sx.sh

変形リスト（mat_tune.L） ⇒ アンロールの段数が４から８になる

```
      26  !cdir outerunroll=8
      27           do k=1,n
      28             do i=1,n
      29               a(i,j)=a(i,j)+b(i,k)*c(k,j)
      30             enddo
      31           enddo
       .         do k = 1, 2048, 8
       .  !cdir    nodep
       .  !cdir    on_adb(a,b)
       .           do i = 1, 2048
       .              a(i,j) = a(i,j) + b(i,k)*dble(c(k,j)) + b(i,k+1)*dble(c(k+1,
       .     1           j)) + b(i,k+2)*dble(c(k+2,j)) + b(i,k+3)*dble(c(k+3,j))
       .     2            + b(i,k+4)*dble(c(k+4,j)) + b(i,k+5)*dble(c(k+5,j)) + b(
       .     3           i,k+6)*dble(c(k+6,j)) + b(i,k+7)*dble(c(k+7,j))
       .           enddo
       .         enddo
      32         enddo
```

実行

- % qsub run.sx.sh

3．outerunroll指示行：実行結果

結果ファイル（p3-sx-sample.o*****）⇒ 約３６GFLOPS

⇒約１０％の性能向上

```
******  Program Information  ******
 Real Time (sec)              :            0.473364
 User Time (sec)              :            0.469779
 Sys  Time (sec)              :            0.003154
 Vector Time (sec)            :            0.469206
 Inst. Count                  :          187710097.
 V. Inst. Count               :          109291163.
 V. Element Count             :        27978532602.
 FLOP Count                   :        17179869353.
 MOPS                         :        59723.724424
 MFLOPS                       :        36570.109249
 A. V. Length                 :          255.999953
 V. Op. Ratio (%)             :           99.720501
  . . .
```

4．演習問題：自動インライン展開

目的

- 自動インライン展開のオプションの使い方を理解する

手順

- インライン展開前の性能の確認
  - コンパイル（リストの確認）
  - 実行（結果、性能の確認）
- インライン展開後の性能の確認
  - コンパイルスクリプトへオプション追加、再コンパイル（リストの確認）
  - 再実行（結果、性能の確認）

ディレクトリ

- practice_4

4．自動インライン展開：インライン展開前のコンパイル

コンパイル

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune1.f mul.f
```

- % ./comp.sx.sh

編集リスト（mat_tune1.L）

- サブルーチン呼び出しがあり、ベクトル化ができていない

```
LINE  LEVEL( NO.): DIAGNOSTIC MESSAGE
   27  vec  (   3): Unvectorized loop.
   28  opt  (1017): Subroutine call prevents optimization.
   28  vec  (  10): Vectorization obstructive procedure reference.:mul

   25: +------>          do j=1,n
   26: |+----->            do k=1,n
   27: ||+---->              do i=1,n
   28: |||                     call mul(n, moda, i, j, k, a, b, c)
   29: ||+----               enddo
   30: |+-----             enddo
   31: +------           enddo
```

4．自動インライン展開：インライン展開前の実行結果

実行

- % qsub run.sx.sh

結果ファイル（p4-sx-sample.o*****）⇒ 約0．013GFLOPS

```
******  Program Information  ******
 Real Time (sec)                :          159.258982
 User Time (sec)                :          159.199437
 Sys  Time (sec)                :            0.027252
 Vector Time (sec)              :            0.000201
 Inst. Count                    :       76243183724.
 V. Inst. Count                 :              49051.
 V. Element Count               :            12551930.
 FLOP Count                     :         2147483815.
 MOPS                           :          478.994700
 MFLOPS                         :           13.489268
 A. V. Length                   :          255.895497
 V. Op. Ratio (%)               :            0.016460
・・・
```

4．自動インライン展開：コンパイルオプションの追加

-pi expin=mul.fをcomp.sx.shに追加

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune1.f mul.f –pi  expin=mul.f
```

コンパイルオプションの意味

- -pi 自動インライン展開を有効にする
- expin=filename.f　展開元のサブルーチンが含まれるファイル (filename.f) を指定する

4．自動インライン展開：コンパイルオプションの追加

コンパイル

- % ./comp.sx.sh

⇒ インライン展開され、ベクトル化できた

```
27  vec  (   1): Vectorized loop.
27  vec  (  29): ADB is used for array.: a
27  vec  (  29): ADB is used for array.: b
28  opt  (1222): Procedure “mul” expanded inline.

      .        do k = 1, 1024, 4
      .  !cdir    nodep
      .  !cdir    on_adb(a,b)
      .           do j = 1, 1024
      .             a(i,j) = a(i,j) + b(i,k)*dble(c(k,j)) + b(i,k+1)*dble(c(k+1,
      .       1           j)) + b(i,k+2)*dble(c(k+2,j)) + b(i,k+3)*dble(c(k+3,j))
      .           enddo
      .         enddo
     31         enddo

     25: +------>          do j=1,n
     26: |V----->            do k=1,n
     27: ||V---->              do i=1,n
     28: |||     A I             call mul(n, moda, i, j, k, a, b, c)
     29: ||V----               enddo
     30: |V-----             enddo
     31: +------           enddo
```

4段アウターアンローリングも行われている

4．自動インライン展開：インライン展開後の実行結果

実行

- % qsub run.sx.sh

結果ファイル（p4-sx-sample.o*****）⇒ 約33GFLOPS

```
******  Program Information  ******
 Real Time (sec)                :            0.068940
 User Time (sec)                :            0.065116
 Sys  Time (sec)                :            0.003401
 Vector Time (sec)              :            0.064488
 Inst. Count                    :            30866545.
 V. Inst. Count                 :            14741403.
 V. Element Count               :          3773794042.
 FLOP Count                     :          2147483816.
 MOPS                           :         58202.579765
 MFLOPS                         :         32979.357086
 A. V. Length                   :           255.999652
 V. Op. Ratio (%)               :            99.574525
・・・
```

5．演習問題：行列積ライブラリの利用

目的

- 行列積ライブラリの性能を確認する

手順

- コンパイルスクリプトの修正
- コンパイル（リストの確認）
- 実行（結果、性能の確認）

ディレクトリ

- practice_5

5．行列積ライブラリの利用：プログラム修正

mat_tune0.fをmat_tune.fにコピーしてから、mat_tune.fを修正

- % cp mat_tune0.f mat_tune.f
- % vi mat_tune.f　（配列Cの型をreal(4)からreal(8)に変更する）

```
4       implicit real(8)(a-h,o-z)
5       parameter ( n=2048 , moda=0 )
6       real(8) a(n+moda,n),b(n+moda,n)
7       real(4) c(n+moda,n)
```

```
4       implicit real(8)(a-h,o-z)
5       parameter ( n=2048 , moda=0 )
6       real(8) a(n+moda,n),b(n+moda,n)
7       real(8) c(n+moda,n)
```

コンパイル

- % ./comp.sx.sh

5．行列積ライブラリの利用：リストの確認

メッセージ、編集、変形リスト（mat_tune.L）

- 行列積ライブラリへ変換
  - コンパイラが認識できる演算パターンでは、ライブラリへの置換が行われる

```
28  opt  (1800): Idiom detected (matrix multiply).

25          do j=1,n
26            do k=1,n
27              do i=1,n
28                a(i,j)=a(i,j)+b(i,k)*c(k,j)
29              enddo
30            enddo
31          enddo
 .          call vdmxqa (b, 1, 2048, c, 1, 2048, a, 1, 2048, 2048, 2048, 2048)
    :

25: *------>            do j=1,n
26: |*----->              do k=1,n
27: ||V---->                do i=1,n
28: |||                       a(i,j)=a(i,j)+b(i,k)*c(k,j)
29: ||V----                 enddo
30: |*-----               enddo
31: *------              enddo
```

5．行列積ライブラリの利用：実行結果

実行

- % qsub run.sx.sh

結果ファイル（p5-sx-sample.o*****）⇒ 約96GFLOPS

```
******  Program Information  ******
Real Time (sec)                 :            0.181327
User Time (sec)                 :            0.177855
Sys  Time (sec)                 :            0.003060
Vector Time (sec)               :            0.177299
Inst. Count                     :          134070619.
V. Inst. Count                  :           85247643.
V. Element Count                :        21798225658.
FLOP Count                      :        17179869353.
MOPS                            :       122836.291552
MFLOPS                          :        96594.806741
A. V. Length                    :          255.704731
V. Op. Ratio (%)                :           99.776524
 ...
```

6．演習問題：自動並列化（1）

目的

- 自動並列化機能を利用する

手順

- コンパイルスクリプトの修正
- コンパイル（リストの確認）
- 実行（結果、性能の確認）

ディレクトリ

- practice_6

6．自動並列化：コンパイル

コンパイルスクリプトの修正

- 修正前

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune0.f
```

- 修正後（-Pautoを追加）

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg mat_tune0.f -Pauto
```

コンパイル

- % ./comp.sx.sh

6．自動並列化：編集、変形リストの確認

自動並列化

- DOループをサブルーチンに抜き出して、並列化を行う。
- サブルーチン名_＄ｎ（ｎは1,2,3…）

```
   25        do j=1,n
   26          do k=1,n
   27            do i=1,n
   28              a(i,j)=a(i,j)+b(i,k)*c(k,j)
   29            enddo
   30          enddo
   31        enddo
    .        call main_$2 (a, b, c)

LINE                    FORTRAN STATEMENT
.         subroutine main_$2
    .   !cdir pardo for, nobarr = (entry,exit)
    .   !cdir nodep
    .         do j = 1, 2048
    .            do k = 1, 512
    .   !cdir       nodep
    .   !cdir       on_adb(a,b)
    .                do i = 1, 2048
    .                   a(i,j) = a(i,j) + b(i,(k-1)*4+1)*dble(c((k-1)*4+1,j)) + b
    .        1             (i,(k-1)*4+2)*dble(c((k-1)*4+2,j)) + b(i,(k-1)*4+3)*
    .        2             dble(c((k-1)*4+3,j)) + b(i,(k-1)*4+4)*dble(c((k-1)*4+4
    .        3             ,j))
    .                enddo
    .            enddo
    .         enddo
    .         end
```

```
   25  mul  (  10): Parallel routine generated : main_$2
   25  mul  (   1): Parallelized by PARDO.

   25. P------>          do j=1,n
   26: |V----->            do k=1,n
   27: ||V---->              do i=1,n
   28: |||     A               a(i,j)=a(i,j)+b(i,k)*c(k,j)
   29: ||V----               enddo
   30: |V-----             enddo
   31. P------           enddo
```

P：自動並列化対象ループ

6．自動並列化：実行結果

実行

- % qsub run.sx.sh

結果ファイル（p6-sx-sample.o*****）（4タスクで実行）

⇒ 約122GFLOPS（オリジナルコードの約3．7倍の性能）

```
******  Program Information  ******
 Real Time (sec)                :          0.150634
 User Time (sec)                :          0.547443
 Sys  Time (sec)                :          0.015307
 Vector Time (sec)              :          0.538167
 Inst. Count                    :         232565343.
 V. Inst. Count                 :         117679803.
 V. Element Count               :       30126022196.
 FLOP Count                     :       17179869389.
 MOPS                           :       55240.285721
 MFLOPS                         :       31382.024045
 MOPS   (concurrent)            :      215354.268045
 MFLOPS (concurrent)            :      122342.828783
 A. V. Length                   :         255.999937
 V. Op. Ratio (%)               :          99.620099
 Memory Size (MB)               :         512.000000
 Max Concurrent Proc.           :                  4.
   Conc. Time(>= 1) (sec)       :          0.140424
   Conc. Time(>= 2) (sec)       :          0.137214
   Conc. Time(>= 3) (sec)       :          0.136968
   Conc. Time(>= 4) (sec)       :          0.135452
・・・
```

よく使うコンパイラオプション

	オプション名	サブオプション	内容
リスト制御	-V		コンパイラのバージョン情報を表示する。
	-R	2	コンパイラによる変形リスト、編集リストを出力する。
		5	コンパイラによる編集リストを出力する。
	-Wf, -L [*list*]	fmtlist summary objlist	コンパイラによる最適化処理およびベクトル化処理に関する各種レポート、リストを出力することを指定する。
	-Wf, -pvctl fullmsg		詳細な診断メッセージを出力することを指定する。
最適化レベル	-C	vopt（規定値）	最大限の最適化処理と規定レベルのベクトル化処理を行うことを指定する。
		hopt	最大限の最適化処理およびベクトル化処理を行うことを指定する。
		vsafe	最適化処理およびベクトル化処理を行うが、副作用を伴う可能性のある機能は抑止することを指定する
		ssafe	ベクトル化処理を抑止し、副作用を伴う可能性のある最適を行わないことを指定する。
	-O	extendreorder	命令の並べ換えを行う範囲を広くして、より強力な命令並べ替えの最適化を行う。
	-pi	auto（規定値）	手続きの自動インライン展開を行うことを指定する。
		noauto	明示的なインライン展開を行うことを指定する。
		line=α	自動インライン展開の対象となる手続きの最大行数を指定する。
		nest=β	自動インライン展開の対象となる手続きのネストの深さを指定する。
		exp=手続き名	指定された手続きがインライン展開の対象となることを指定する。
		expin=ファイル名	指定されたファイルにインライン展開の対象となる手続きがあることを指定する。
	-Wf		きめ細かなオプションを指定する。
		-pvctl chgpwr	べき算 R1**R2 を EXP(R2*LOG(R1)) に置き換えることを指定する。
		-pvctl expand=n	ループ長がn以下のループを展開することを指定する
		-pvctl noloopchg	ループ入れ換えによるベクトル化を行わないことを指定する。
並列化	-P auto		自動並列化機能を使用することを指定する。
	-P openmp		OpenMP機能を使用することを指定する。（並列化対象サブルーチンのみ）
デバッグ	-e	C	実行時に配列要素参照の添字の値が、その配列に対して許される範囲内にあるかどうかチェックを行う。
		R	配列要素参照および配列部分参照において、添字あるいは部分配列添字の値が許される範囲内にあるかどうかチェックを行う。
	-Wf	-init stack=zero	スタックに割付ける領域を、0 で初期化することを指定する。
		-init heap=zero	ヒープに割付ける領域を、0 で初期化することを指定する。
性能解析	-ftrace		SXの性能分析ツールFTRACE対応の実行ファイルを作成することを指定する。

よく使う指示行

指示行	内容
vector/novector	直後のDOループをベクトル化する/しないことを指定する。
nodep	DOループ内データ依存関係が不明な場合、ベクトル化不可の依存がないものとしてベクトル化/最適化を行う。
outerunroll[=n]	外側ループのアンローリングを許可する
loopchg/noloopchg	ループ入れ換えによるベクトル化を行なうことを指定する。
expand[=n]	直後のDOループをベクトル化する/しない展開することを指定する。
shortloop	直後のDOループのループ長が、レジスタ長(256)以下であることを指定する。
select(vector¦concur)	直後のDOループに対して、ベクトル化を優先させるか、並列化を優先させるかを指定する。
on_adb[(識別子)]	直後のループ中の配列のベクトルロード、ストアにおいて、配列をADBにバッファリングする。

ライブラリ

- コンパイラが自動的に置き換えるもの
  - 行列積パターン
- ソースコード修正により使用できるもの
  - ASL / SX (科学技術計算ライブラリ)
    - 行列演算、FFTなど
  - Mathkeisan
    - BLASライブラリ

おすすめコンパイルオプション

プログラムを初めてスーパーコンピュータ（SX-9）で実行する場合

- 既定値レベルの最適化、ベクトル化（-Cvopt）
- SX-9向けの命令列を生成することを指定（-cfsx9）
- ベクトル化の状態を表示する編集リストの採取（-R2）
- ベクトル化が行われなかった場合の詳細メッセージ出力（-pvctl fullmsg）

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg “プログラムファイル名”
```

正常終了した場合、-Choptを使用して-Cvoptの結果と比較

```
sxf90 -V -Chopt -cfsx9 -R2 -Wf,-pvctl fullmsg “プログラムファイル名”
```

デバッグ用コンパイルオプション

正常終了したが、結果がおかしい場合

- 副作用を伴う可能性のある最適化を抑止（-Cvsafe）

```
sxf90 -V -Cvsafe -cfsx9 -R2 -Wf,-pvctl fullmsg “プログラムファイル名”
```

異常終了（Segmentation fault）した場合、デバッグ用オプションで分析

- 配列外参照をチェック（-eC、-eRオプション）
- ベクトル化、最適化が抑止されるため、実行時間が長くなる

  ⇒エラー終了したファイル（サブルーチン）のみにオプションをつけた方がよい

```
sxf90 -V -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg -eC “プログラムファイル名”
```

初期化漏れのチェック

- スタックに割り付ける領域をNaNで初期化する

  ⇒初期化漏れの変数をアクセスするとアボートさせることができる

```
sxf90 -V ー-Pmulti -Cvopt -cfsx9 -R2 -Wf,-pvctl fullmsg,-init stack=nan “プログラムファイル名”
```