H28年度
SX-ACE 高速化技法の基礎
（演習用資料）

2016年　6月16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．

無断転載等は，ご遠慮下さい．

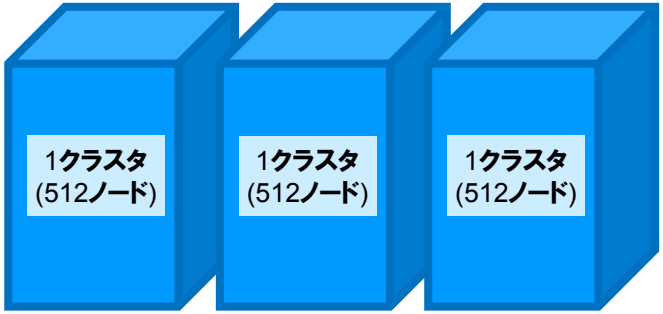
SX-ACEの計算ノード構成

全1536ノード構成

- 1ノードあたり 1CPU(4core)
- メモリ　1ノードあたり64Gバイト(共有)
- ACEキューで最大1024core(256ノード)まで利用可能

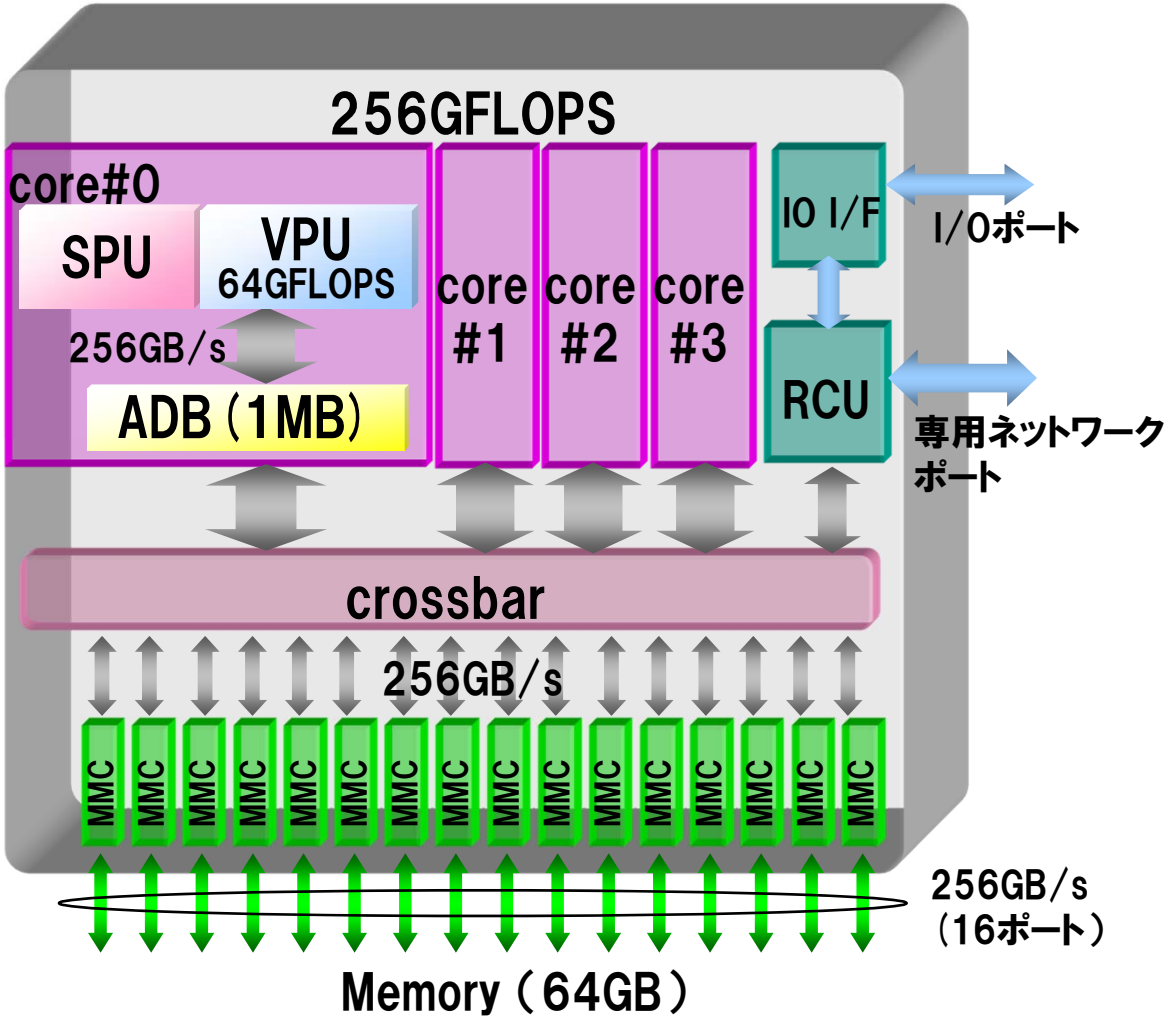
スーパーコンピュータ　SX-ACE
全1536ノード

1コア・・・・・・・・・・・64ギガFLOPS
1ノードあたり
ベクトルプロセッサ・・・・・4core
共有物理メモリ・・・64ギガバイト



SX-ACEのCPU構成

- 演算性能　256GFLOPS（64GFLOPS/コア　×4コア）
- メモリバンド幅 256GB/s　（16GB/s /ポート　×16ポート）



SPU: Scalar Processing Unit
VPU: Vector Processing Unit
ADB: Assignable Data Buffer
RCU: Remote Access Control Unit
MMC: Main Memory Controller

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

行列積のプログラムを使った課題

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

1．演習問題：オリジナルコードのコンパイルと実行

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

```
sxf90  -R2  -Wf,-pvctl fullmsg  mat_tune0.f
```

- -R2
  編集リスト・変形リストを採取
- -Wf, -pvctl fullmsg
  詳細診断メッセージを表示

コンパイル

```
% ./comp.sx.sh
```

1．オリジナルコード：コンパイル（２）

リストの確認

- 変形リスト（mat_tune0.L）

```
25          do j=1,n
26   !cdir nounroll
27            do k=1,n
28              do i=1,n
 .     !cdir nodep
 .     !cdir on_adb(a,b)
 .           do i = 1,  2048
29                a(i,j)=a(i,j)+b(i,k)*c(k,j)
30              end do
31            end do
32          end do
```

- 編集リスト（mat_tune0.L）

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

V：ベクトル化対象ループ

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 ACE
#PBS -l cpunum_job=4,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
- -j o 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

run.sx.sh をジョブ投入 (qsub) してください.

```
% qsub  run.sx.sh
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

1．オリジナルコード：実行結果

結果ファイル(p1-sx-sample.o*****)→ 約25GFLOPS

```
******  Program Information  ******
Real Time (sec)                 :          0.697364
User Time (sec)                 :          0.695252
Sys  Time (sec)                 :          0.000758
Vector Time (sec)               :          0.695070
Inst. Count                     :         537299641.
V. Inst. Count                  :         167962313.
V. Element Count                :       42998350432.
V. Load Element Count           :       17180000302.
FLOP Count                      :       17179869321.
MOPS                            :       62376.933486
MFLOPS                          :       24710.276736
A. V. Length                    :         255.999990
V. Op. Ratio (%)                :          99.148358
Memory Size (MB)                :         256.031250
MIPS                            :         772.812794
I-Cache (sec)                   :           0.000043
O-Cache (sec)                   :           0.000074
Bank Conflict Time
  CPU Port Conf. (sec)          :           0.000051
  Memory Network Conf. (sec)    :           0.065552
ADB Hit Element Ratio (%)       :          49.990669
```

PROGINF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）
- コンパイラオプションの追加（-ftrace）
- 実行（結果，性能の確認）

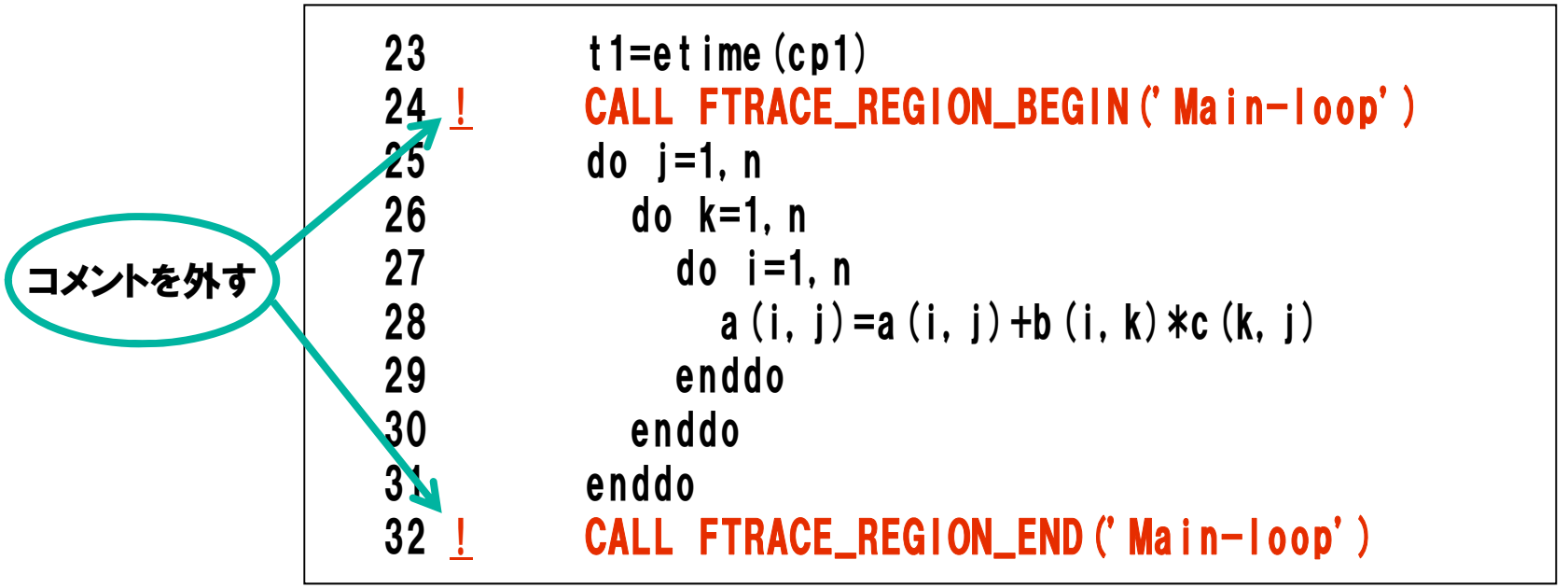
ディレクトリ

- practice_2

2. 性能解析(Ftraceの利用):ソースコード修正

mat_tune0.f へユーザ指定リージョン (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.
    ⇒ Ftrace_Region は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):コンパイラオプションの追加

-ftraceをcomp.sx.shに追記する.

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

- -ftrace

  簡易性能解析機能を利用することを指定する.

※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
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

2. 性能解析(Ftraceの利用):実行結果

結果ファイル(p2-sx-sample.o*****)→ 約40GFLOPS

```
*----------------------*
  FTRACE ANALYSIS LIST
*----------------------*

Execution Date : Mon Jan 19 17:05:07 2015
Total CPU Time : 0:00'00"925 (0.925 sec.)

PROC.NAME   FREQUENCY   EXCLUSIVE TIME[sec](  %)   AVER.TIME [msec]     MOPS     MFLOPS V.OP RATIO  AVER. V.LEN   VECTOR TIME   I-CACHE MISS   O-CACHE MISS   BANK CONFLICT CPU PORT   BANK CONFLICT NETWORK   ADB HIT ELEM.%
main_               1     0.925(100.0)      925.337   46866.9    18566.1   99.15      256.0         0.925          0.000          0.000          0.000                  0.190           49.99
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
total               1     0.925(100.0)      925.337   46866.9    18566.1   99.15      256.0         0.925          0.000          0.000          0.000                  0.190           49.99
Main-loop           1     0.924( 99.9)      923.975   46883.1    18593.4   99.15      256.0         0.924          0.000          0.000          0.000                  0.190           49.99
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 matrix_size = 2048
  p_name        |  user(sec)|   moda        check
  mat_tune0.f_ |      0.924 |      0       0.204800000000000D+04
    18.593 (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.
- ４段アウターアンロールを行う.

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 を修正する．

```
% cp  mat_tune0.f  mat_tune.f
% vi  mat_tune.f
```

4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=4
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：ソースコード修正（２）

コンパイル

```
% ./comp.sx.sh
```

変形リスト（mat_tune0.L）の確認

```
25          do j=1,n
26  !cdir outerunroll=4
27            do k=1,n
28              do i=1,n
29                a(i,j)=a(i,j)+b(i,k)*c(k,j)
30              end do
31            end do
 .          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
32          end do
```

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

実行

```
% qsub  run.sx.sh
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

3. outerunroll指示行:実行結果

結果ファイル(p3-sx-sample.o*****) ⇒ 約40GFLOPS

⇒オリジナル(演習1)の1.6倍の性能向上

```
******  Program Information  ******
Real Time (sec)                :          0.429773
User Time (sec)                :          0.428914
Sys  Time (sec)                :          0.000737
Vector Time (sec)              :          0.428740
Inst. Count                    :        240708305.
V. Inst. Count                 :        117679817.
V. Element Count               :      30126031456.
V. Load Element Count          :      10741743662.
FLOP Count                     :      17179869321.
MOPS                           :      70524.767072
MFLOPS                         :      40054.344976
A. V. Length                   :        255.999986
V. Op. Ratio (%)               :         99.593282
  ・・・
```

PROGINFの出力

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

```
% ./comp.sx.sh
```

編集リスト（mat_tune1.L）の確認

- 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
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

結果ファイル(p4-sx-sample.o*****) ⇒ 約0.034GFLOPS

```
 ******  Program Information  ******
Real Time (sec)                 :          62.914568
User Time (sec)                 :          62.907500
Sys  Time (sec)                 :           0.003401
Vector Time (sec)               :           0.000264
Inst. Count                     :       76243172300
V. Inst. Count                  :             49097
V. Element Count                :          12567136
V. Load Element Count           :             65582
FLOP Count                      :        2147483784
MOPS                            :        1212.187582
MFLOPS                          :          34.137166
A. V. Length                    :         255.965456
V. Op. Ratio (%)                :           0.016480
 ・・・
```

PROGINFの出力

4. 自動インライン展開:コンパイラオプションの追加

-pi expin=mul.f を comp.sx.sh に追記する.

```
sxf90  -R2  -Wf,-pvctl fullmsg  mat_tune1.f  mul.f  -pi  expin=mul.f
```

- -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：コンパイラオプションの追加

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 i = 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 |           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
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

結果ファイル(p4-sx-sample.o*****) ⇒ 約42GFLOPS

インライン展開により, 0.034GFLOPS から 42GFLOPS に性能向上した.

```
 ******  Program Information  ******
Real Time (sec)                 :          0.052125
User Time (sec)                 :          0.051285
Sys  Time (sec)                 :          0.000720
Vector Time (sec)               :          0.051107
Inst. Count                     :          31902926
V. Inst. Count                  :          14741449
V. Element Count                :        3773809248
V. Load Element Count           :        1343291438
FLOP Count                      :        2147483784
MOPS                            :       73919.678756
MFLOPS                          :       41873.526060
A. V. Length                    :         255.999885
V. Op. Ratio (%)                :          99.547307
  ・・・
```

PROGINFの出力

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 を修正する．

```
% 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）の確認

- 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
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

結果ファイル(p5-sx-sample.o*****) ⇒ 約56GFLOPS

行列積ライブラリにより，40GFLOPS から 56GFLOPS に性能向上した．

```
 ******  Program Information  ******
Real Time (sec)                 :            0.307382
User Time (sec)                 :            0.306558
Sys  Time (sec)                 :            0.000721
Vector Time (sec)               :            0.306382
Inst. Count                     :           134048176
V. Inst. Count                  :            85247689
V. Element Count                :         21798240864
V. Load Element Count           :           406978606
FLOP Count                      :         17179869321
MOPS                            :        71265.605044
MFLOPS                          :        56041.171070
A. V. Length                    :          255.704772
V. Op. Ratio (%)                :           99.776627
   ...
```

PROGINFの出力

6. 演習問題：自動並列化

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

-P auto を comp.sx.sh に追記する.

```
sxf90  -R2  -Wf,-pvctl fullmsg  mat_tune0.f  -P auto
```

- -P auto

  自動並列機能を使用することを指定する.

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

ジョブファイル (4タスクでの実行用)

```
#!/bin/csh

#PBS -q ACE
#PBS -l cpunum_job=4,elapstim_req=0:10:00,memsz_job=1GB
#PBS -j o -N p1-sx-sample

cd $PBS_O_WORKDIR

setenv F_RSVTASK=4

timex ./a.out
```

並列タスク数を環境変数F_RSVTASKで
指定することが可能です.
SX-ACEはコア数が4Core/CPUなので,
1~4までの値を指定することができます.
(デフォルトは実行マシンの最大コア/CPU数)

実行

```
% qsub  run.sx.sh
Request ******.cmc submitted to queue: ACE.
```

*****はジョブ番号

6. 自動並列化:実行結果

結果ファイル(p6-sx-sample.o*****)(4タスクで実行)

⇒ 約55GFLOPS(自動並列化前(演習3)の約1.4倍の性能向上)

```
******  Program Information  ******
 Real Time (sec)                :          0.316257
 User Time (sec)                :          1.216064
 Sys  Time (sec)                :          0.008273
 Vector Time (sec)              :          1.164452
 Inst. Count                    :        243435038.
 V. Inst. Count                 :        117679849.
 V. Element Count               :      30126037402.
 V. Load Element Count          :      10741746602.
 FLOP Count                     :      17179869334.
 MOPS                           :      24876.809601
 MFLOPS                         :      14127.438469
 MOPS   (concurrent)            :      97098.118145
 MFLOPS (concurrent)            :      55141.624328
 A. V. Length                   :        255.999967
 V. Op. Ratio (%)               :         99.584305
 Memory Size (MB)               :        512.000000
 Max Concurrent Proc.           :                 4.
   Conc. Time(>= 1) (sec)       :          0.311559
   Conc. Time(>= 2) (sec)       :          0.310523
   Conc. Time(>= 3) (sec)       :          0.310012
   Conc. Time(>= 4) (sec)       :          0.284864
     ・・・
```

PROGINFの出力

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

おすすめコンパイラオプション

プログラムを初めてスーパーコンピュータ(SX-ACE)で実行する場合

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

- -Cvopt
  既定値レベルの最適化・ベクトル化を行う.
- -R2
  ベクトル化の状態を表示する編集リストの採取.
- -pvctl fullmsg
  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.

正常終了した場合, -Choptを使用して-Cvoptの結果と比較

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

- -Chopt
  最大限の最適化・ベクトル化を行う.

デバッグ用コンパイラオプション

正常終了したが，結果がおかしい場合

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

- -Cvsafe

  副作用を伴う可能性のある最適化を抑止する．

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

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

- -eC または -eR

  配列外参照をチェックする．

  ただし，ベクトル化・最適化が抑止されるために実行時間が長くなるので，

  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．

初期化漏れのチェック

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

- -init,stack=nan

  スタックに割り付ける領域をNaNで初期化する．

  これにより，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．