

並列プログラミング入門

OpenMP・自動並列

（演習用資料）

大阪大学D3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D3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。**

演習問題の構成

■ ディレクトリ構成

```
parallel/  
|-- practice_1   オリジナルコード実行環境  
|-- practice_2   OpenMP並列演習問題  
`-- practice_3   自動並列演習問題
```

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
- 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

```
nfort -report-all mat_tune0.f -ftrace
```

- -report-all

コード生成リスト、診断メッセージリスト、編集リスト、インラインリスト、オプションリスト、ベクトルリストを出力する

- -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。

コンパイル

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

1. オリジナルコード:実行

ジョブファイル (run.sx.sh)

```
#!/bin/bash

#PBS -q LECTUREV
#PBS --group=kosyuXXX
#PBS --venode=1
#PBS -l cpunum_job=2, elapstim_req=0:05:00
#PBS -j o -N p1-sx-sample

cd $PBS_O_WORKDIR

time ./a.out
```

NQSVオプション

-q ジョブクラス名を指定
--group 課金グループを指定
--venode 使用するVE数を指定
-l 使用CPU数, 経過時間
-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。

投入したジョブのステータスは、qstatコマンドで確認して下さい。

```
$ qsub run.sx.sh
Request *****.sqd submitted to queue: LECTUREV.
$ qstat
```

RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs	
*****.sqd	p1-sx-sa	kosyuXXX	LECTUREV	0	QUE	-	0.00B	0.00		0	Y	Y	Y	1

*****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33GFLOPS

***** Program Information *****

PROGINFの出力

Real Time (sec)	:	0.511899
User Time (sec)	:	0.510496
Vector Time (sec)	:	0.509786
Inst. Count	:	583677718
V. Inst. Count	:	134402095
V. Element Count	:	34406933536
V. Load Element Count	:	17179869585
FLOP Count	:	17179869376
MOPS	:	85160.400972
MOPS (Real)	:	84872.547647
MFLOPS	:	33674.900426
MFLOPS (Real)	:	33561.074846
A. V. Length	:	255.999979
V. Op. Ratio (%)	:	98.965902
L1 Cache Miss (sec)	:	0.000235
CPU Port Conf. (sec)	:	0.000000
V. Arith. Exec. (sec)	:	0.222687
V. Load Exec. (sec)	:	0.287098
VLD LLC Hit Element Ratio (%)	:	49.975928
FMA Element Count	:	8589934592
Power Throttling (sec)	:	0.000000
Thermal Throttling (sec)	:	0.000000
Memory Size Used (MB)	:	560.000000
Non Swappable Memory Size Used (MB)	:	90.000000

2. 演習問題: OpenMP並列

目的

- OpenMP機能を利用する.

手順

- ソースコードの修正
- 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_2

2. OpenMP並列:ソースコードの修正

mat_tune0.f にOpenMP指示行を挿入.

```
% vi mat_tune0.f
```

- parallel do/end parallel do指示行の挿入例

```
25 !$omp parallel do
26     do j=1, n
27         do k=1, n
28             do i=1, n
29                 a(i, j)=a(i, j)+b(i, k)*c(k, j)
30             end do
31         end do
32     end do
33 !$omp end parallel do
```

2. OpenMP並列:コンパイル

comp.sx.sh に-fopenmpを追記する.

```
nfort -report-all mat_tune0.f -ftrace -fopenmp
```

- -fopenmp

OpenMP機能を使用する。-pthreadは暗黙的に有効となる。

コンパイル

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

2. OpenMP並列:編集リストの確認

OpenMP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...)

LINE	DIAGNOSTIC MESSAGE
25:	par (1801): Parallel routine generated.: MAIN_\$1
26:	par (1803): Parallelized by "do".
25:	!\$omp parallel do
26:	P-----> do j=1, n
27:	+-----> do k=1, n
28:	V-----> do i=1, n
29:	F a (i, j)=a (i, j)+b (i, k)*c (k, j)
30:	V----- end do
31:	+----- end do
32:	P----- end do
33:	!\$omp end parallel do

P:ループの並列化

2. OpenMP並列:実行

ジョブファイル (10タスクでの実行用)

```
#!/bin/bash

#PBS -q LECTUREV
#PBS --group=kosyuXXX
#PBS --venode=1
#PBS -l cpunum_job=2, elapstim_req=0:05:00
#PBS -j o -N p2-sx-sample
#PBS -v VE_PROGINF=DETAIL
#PBS -v OMP_NUM_THREADS=10

cd $PBS_O_WORKDIR

time ./a.out
```

並列タスク数を環境変数

OMP_NUM_THREADSで指定することが可能です。

SX-Aurora TSUBASAはコア数が10Core/VEなので、**1~10**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/VE数)

実行

```
$ qsub run.sx.sh
Request *****.sqd submitted to queue: LECTUREV.
```

*****はジョブ番号

2. OpenMP並列:実行結果

結果ファイル(p2-sx-sample.o*****) (10タスクで実行)

⇒ 約150GFLOPS(オリジナル(演習1)の約4.5倍の性能向上)

***** Program Information *****

Real Time (sec)	:	0.113899
User Time (sec)	:	0.170061
Vector Time (sec)	:	0.903272
Inst. Count	:	666608579
V. Inst. Count	:	134402349
V. Element Count	:	34406939338
V. Load Element Count	:	17179872215
FLOP Count	:	17179869425
MOPS	:	40068.016997
MOPS (Real)	:	382173.768857
MFLOPS	:	15813.871959
MFLOPS (Real)	:	150834.693095
A. V. Length	:	255.999539
V. Op. Ratio (%)	:	98.777355
. . .		
Max Active Threads	:	10
Available CPU Cores	:	10
Average CPU Cores Used	:	1.493085
Memory Size Used (MB)	:	1802.000000
Non Swappable Memory Size Used (MB)	:	100.000000

PROGINFの出力

3. 演習問題：自動並列

目的

- 自動機能を利用する.

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_3

3. 自動並列:コンパイル

comp.sx.sh に-mparallelを追記する.

```
nfort -report-all mat_tune0.f -ftrace -mparallel
```

- -mparallel

自動並列化を適用する。-pthreadは暗黙的に有効となる。

コンパイル

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

3. 自動並列:編集リストの確認

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...)

LINE	DIAGNOSTIC MESSAGE
25:	par (1801): Parallel routine generated. : MAIN_\$2
25:	par (1803): Parallelized by "do".
25:	P-----> do j=1, n
26:	+-----> do k=1, n
27:	V-----> do i=1, n
28:	F a (i, j)=a (i, j)+b (i, k)*c (k, j)
29:	V----- end do
30:	+----- end do
31:	P----- end do

P:ループの並列化

3. 自動並列:実行

ジョブファイル (10タスクでの実行用)

```
#!/bin/bash

#PBS -q LECTUREV
#PBS --group=kosyuXXX
#PBS --venode=1
#PBS -l cpunum_job=2, elapstim_req=0:05:00
#PBS -j o -N p2-sx-sample
#PBS -v VE_PROGINF=DETAIL
#PBS -v OMP_NUM_THREADS=10

cd $PBS_O_WORKDIR

time ./a.out
```

並列タスク数を環境変数

OMP_NUM_THREADSで指定することが可能です。

SX-Aurora TSUBASAはコア数が10Core/VEなので、**1~10**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/VE数)

**OpenMP並列・自動並列のスレッド
並列実行を行う際は
同一の環境変数
(OMP_NUM_THREADS) を指定します。**

実行

```
$ qsub run.sx.sh
Request *****.sqd submitted to queue: LECTUREV.
```

*******はジョブ番号**

3. 自動並列:実行結果

■ 結果ファイル(p2-sx-sample.o*****) (10タスクで実行)

⇒ 約138GFLOPS(オリジナル(演習1)の約4.1倍の性能向上)

Real Time (sec)	:	0.123873
User Time (sec)	:	0.245144
Vector Time (sec)	:	0.919688
Inst. Count	:	698017446
V. Inst. Count	:	134402415
V. Element Count	:	34406941412
V. Load Element Count	:	17179873242
FLOP Count	:	17179869422
MOPS	:	36974.129797
MOPS (Real)	:	351658.075656
MFLOPS	:	14582.267252
MFLOPS (Real)	:	138690.810811
A. V. Length	:	255.999428
V. Op. Ratio (%)	:	98.706133
. . .		
Max Active Threads	:	10
Available CPU Cores	:	10
Average CPU Cores Used	:	1.979000
Memory Size Used (MB)	:	1802.000000
Non Swappable Memory Size Used (MB)	:	100.000000

PROGINFの出力