



---

# HPF実習II

2009年7月

NEC 第一コンピュータソフトウェア事業部

林 康晴

# 内容

---

- 練習用プログラムsample.F
- 3次元流体コードimpact3d
- 2次元静電粒子コードespac2
- 発展1 実行プロセッサ数を増やすには
- 発展2 プロセッサ数を可変にするには

# 練習用プログラムsample.F

---

- Fortranソースを取得して、HPFで並列化してください。

```
%> tar -xvf /tmp/hpf/sample.tgz
%> cd sample
%> vi sample.F
```

- 3つのプログラム単位: 全86行(空白行とコメントを含む)
  - ◆ モジュールparam
    - 問題サイズと時間発展ループの繰返し数を大域定数として宣言
  - ◆ 主プログラムsample
    - 主要配列は2次元:  $a(n,n), b(n,n), c(n,n)$
    - 初期化後、時間発展ループ中で計算を行い、実行時間とチェックsumを出力
  - ◆ サブルーチンbound
    - 境界処理ループだけを含む

**備考:** 頻出パタンの練習です。計算内容に意味はありません。

# まずは、並列化対象ループを選択する

- コストの高い手続、ループを調べる
  - ◆ 「main loop」とコメントのあるループ
    - 本来は、Fortranコンパイラのプロファイル機能などで調べる
- どのループを並列化するかを決める
  - もっともコストの高いループネストの最外側ループ(37行目)

```
12  double precision a(n,n),b(n,n),c(n,n),sum,ap
      :
! main loop
37  do j=2, n-1
38      do i=2, n-1
39          ix = idxx(i)
40          iy = idxy(j)
41          a(i,j) = (b(i,j)+b(i-1,j)+b(i+1,j)+b(i,j-1)+b(i,j+1))*0.2d0*c(ix,iy)*ap
42      enddo
43  enddo
```

# 完成したら

## ■ 完成したら翻訳/実行し、納得いくまでチューニングする

### ◆ hpfコマンドの場合

```
%> hpf -Minfo -Mlist2 sample.F -o sample ← 実行ファイルsampleを生成する
```

### ◆ fhpfコマンドの場合

```
%> fhpf sample.F ← MPIプログラムsample.F.mpi.fを生成する  
%> mpif90 sample.F.mpi.f90 -o sample ← 実行ファイルsampleを生成する
```

### ◆ 実行方法

```
%> cat run.sh ← ジョブスクリプトの例  
#PBS -l cputim_job=00:10:00 ← 使用CPU時間を指定する  
#PBS -l memsz_job=1gb ← 使用メモリ量を指定する  
#PBS -T vltmpi ← Voltaire MPIを指定する  
#PBS -l cpunum_job=1 ← 各ノードの使用CPU(コア)数を指定する  
#PBS -b 4 ← 使用ノード数を指定する  
#PBS -q PCS-B ← 使用するキューを指定する  
cd sample ← 実行ディレクトリへ移動する  
mpirun_rsh -np 4 ${NQSII_MPIOPTS} ./sample ← 実行する(4並列実行の場合)  
%>qsub run.sh ← ジョブを投入する
```

# 3次元流体コードimpact3d (坂上先生)

---

## ■ ソースを取得する

```
%>tar -xzvf /tmp/hpf/impact3d.tgz  
%>cd impact3d
```

## ■ fhpfによる翻訳と実行

◆ 翻訳: `%> ./gof`      •1, 2, 4並列用の実行ファイルimpact3df-1, impact3df-2, impact3df-4が生成される

◆ 実行: `%> qsub runf.sh`

## ■ hpfコマンドによる翻訳と実行

◆ 翻訳: `%> ./gon`      •1, 2, 4並列用の実行ファイルimpact3dn-1, impact3dn-2, impact3dn-4が生成される

◆ 実行: `%> qsub runn.sh`

# 2次元静電粒子コード`espac2` (坂上先生)

---

## ■ ソースを取得する

```
%>tar -xzvf /tmp/hpf/espac2.tgz
%>cd espac2
```

## ■ fhpfによる翻訳と実行

◆ 翻訳: `%> ./gof`

•1, 2, 4並列用の実行ファイルespac2f-1, espac2f-2, espac2f-4が生成される

◆ 実行: `%> qsub runf.sh`

## ■ hpfコマンドによる翻訳と実行

◆ 翻訳: `%> ./gon`

•1, 2, 4並列用の実行ファイルespac2n-1, espac2n-2, espac2n-4が生成される

◆ 実行: `%> qsub runn.sh`

◆ 備考:

- 本プログラムは、HPFプログラムからFortranのサブルーチンを呼び出している(FFTの計算部分)。hpfコマンドを使う場合、Fortranのサブルーチンの翻訳に、-Mf90 -Mlocalオプションが必要。詳しくはMakefile.necを参照。
- fhpfの場合は、FortranのサブルーチンはFortranコンパイラで翻訳する

# espac2のポイント

•2次元FFTを並列実行するために、FortranのFFTサブルーチンを並列ループ中から呼び出している。並列化するためには、以下の4点に注意が必要。

```
dimension fsx1(kx),fsx2(kx),ksx3(15), fdat(kx,ky)
!HPF$ DISTRIBUTE (*,BLOCK) ONTO proc :: fdat
interface!
  extrinsic('FORTRAN','LOCAL') subroutine rfftf(k, ftmp, f1, f2, k3)
    dimension ftmp(k),f1(k),f2(k), k3(15)
    intent(inout) :: ftmp,f1
    intent(in) :: k,f2,k3
  end subroutine
end interface
!HPF$ INDEPENDENT, NEW(iy,fsx1)
do iy = 1, ky
  call rfftf ( kx,fdat(:,iy),fsx1,fsx2,ksx3 )
end do
```

•呼び出されるのがFortranの手続であることを示す  
➢ 引数をFortran用に変形

•非分散の引数は、INTENT(IN)を指定するかNEW節を指定する  
➢ さもないと、終値保障が必要なため、効率が大幅に下がる

•INDEPENDENT指示文が必要  
➢ 自動では並列化できない

•分散配列は、各呼出しで処理する部分だけを渡す。  
➢ コンパイラは、分散に合わせて並列化する



# 発展1 実行プロセッサ数を増やすには

- 翻訳スクリプトgof、gonを以下の様に修正して、翻訳する

- ◆ 8並列用の実行プログラムを生成する場合

```
set proc = '1 2 4'
```



```
set proc = '8'
```

・8並列用の実行ファイルespac2n-8が生成される

- ジョブスクリプトrunf.sh、runn.shを以下の様に修正して実行する

- ◆ 実行ノード数を修正する

```
#PBS -b 4
```



```
#PBS -b 8
```

- ◆ mpirun\_rshコマンドの実行プロセス数と実行ファイル名を修正する

```
mpirun_rsh -np 4 ${NQSII_MPIOPTS} ./espac2f-4
```



```
mpirun_rsh -np 8 ${NQSII_MPIOPTS} ./espac2f-8
```

## 発展2 プロセッサ数を可変にするには(1)

### ■ impact3dの場合

- ◆ インクルードファイルproc.hのPROCESSORS指示文をコメントアウトする

```
!HPF$ PROCESSORS proc(lpara)
```



```
!!HPF$ PROCESSORS proc(lpara)
```

- ◆ インクルードファイルphys.h、work.hのDISTRIBUTE指示文から、ONTO節を削除する

```
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO proc :: .....
```



```
!HPF$ DISTRIBUTE (*,*,BLOCK) :: .....
```

### ■ espac2の場合 (impact3dと同様に、全てのDISTRIBUTE指示文を修正してもよい)

- ◆ インクルードファイルproc.hのPROCESSORS指示文を、以下のように修正する (number\_of\_processors()は、実行プロセッサ数を返す組み込み関数)

```
!HPF$ PROCESSORS proc(lpara)
```



```
!HPF$ PROCESSORS proc(number_of_processors())
```

## 発展2 プロセッサ数を可変にするには(2)

### ■ makeする

- ◆ fhpfコマンドの場合

```
%> make -f Makefile.fhpf ← 実行ファイルimpact3d / espac2が生成される
```

- ◆ hpfコマンドの場合

```
%> make -f Makefile.nec ← 実行ファイルimpact3d / espac2が生成される
```

### ■ 実行する

- ◆ ジョブスクリプトの実行ファイル名を修正する

```
mpirun_rsh -np 4 ${NQSII_MPIOPTS} ./impact3df-4
```



```
mpirun_rsh -np 4 ${NQSII_MPIOPTS} ./impact3d
```

- ◆ 実行プロセス数を変える場合は、発展1と同様に、実行ノード数と実行プロセス数の指定も修正する
- ◆ 実行ノード数に合わせて、使用するキューも変更する