

粒子コードのHPF化

2003年6月11日(水)

NEC 林 康晴

pic3desloopのHPF化

- 3次元静電粒子シミュレーションコード
- マッピングの指定
 - 粒子と関連する作業配列はBLOCK分散、場の量は非分散

```
REAL*8,DIMENSION(NPM,NPE)::x,y,z,vx,vy,vz
```

```
!HPF$ DISTRIBUTE (*,BLOCK) :: X,Y,Z,VX,VY,VZ
```

```
REAL*8,DIMENSION(0:NGXMX,0:NGYMX,0:NGZMX,NPE)::RHON
```

```
!HPF$ ALIGN RHON(*,*,*,I) WITH X(*,I)
```

```
REAL*8,DIMENSION(0:NGXMX,0:NGYMX,0:NGZMX,LR,NPE)::WORKP
```

```
!HPF$ ALIGN WORKP(*,*,*,*,I) WITH X(*,I)
```

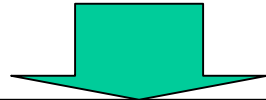
```
REAL*8,DIMENSION(0:NGXMX,0:NGYMX,0:NGZMX) :: RHO,PHI,EX,EY,EZ
```

- -Minfoオプションにより、通信が発生していることが分かったループに、independent指示文を指定
 - communication is generated: array copy
 - expensive communication:

修正が必要なループ

- 配列構文を含むリダクションをDOループに

```
do ipe = 1,npe  
  rho = rho + rhon(:, :, ipe)  
enddo
```



HPF/SX V2による自動並列化可能

```
do ipe = 1,npe  
  do k=0,ngzmx  
    do j=0,ngymx  
      do i=0,ngxmx  
        rho(i,j,k) = rho(i,j,k) + rhon(i,j,k,ipe)  
      enddo  
    enddo  
  enddo  
enddo
```

性能と問題点

- 実行性能は、共有メモリ並列とほぼ同等

	Fortran	HPF
1CPU	20.94(s)	
2CPU	11.25(s)	11.60(s)

- ところが、メモリ消費量が共有メモリ並列の2倍以上

	Fortran	HPF
1CPU	512.0MB	
2CPU	640.0MB	1488.1MB

2つの原因

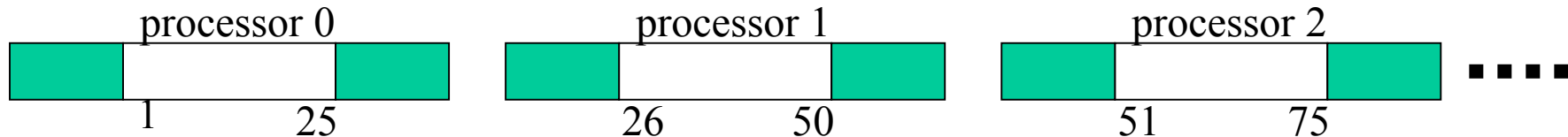
- 処理系が暗黙的に確保するシャドウ領域と入出力バッファ

シャドウ領域(1)

- 最適化用通信バッファ (shift通信用)

```
REAL A(100),B(100)
```

```
!HPF$ DISTRIBUTE (BLOCK(25)) :: A,B
```



```
DO I = ....
```

```
B(I) = A(I-1) + A(I) + A(I+1) + A(I+2)....
```

- 大域変数、引数となる分散配列には、デフォルトで、分散次元の両端に幅4のシャドウ領域が確保される。
- このプログラムの場合、隣接計算のパターンはないので、シャドウ領域は必要ない。

シャドウ領域(2)

- SHADOW指示文で指定可能

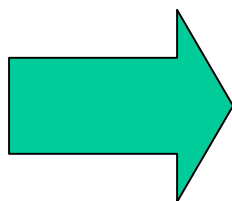
```
REAL A(100),B(100)
```

```
!HPF$ DISTRIBUTE (BLOCK) :: A,B
```

```
!HPF$ SHADOW (0) :: A,B
```

- 翻訳時オプション-Moverlap=size:nで翻訳時に指定可能

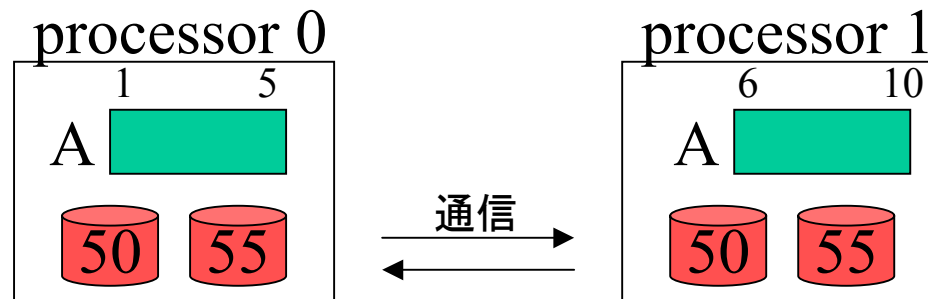
%> sxhpf -Moverlap=size:0 ! 全ての配列のシャドウ領域を0に



	Fortran	HPF
1CPU	512.0MB	
2CPU	640.0MB	816.1MB

入出力バッファ(1)

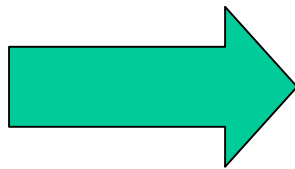
```
REAL A(10)  
!HPF$ DISTRIBUTE A(BLOCK)  
READ(50,*)A  
WRITE(55,*)A
```



- 非同期入出力/SFS並列入出力用バッファとして、デフォルトでは、各入出力装置に対し、各プロセス上で、16MB確保される。
- 非同期入出力/SFS並列入出力を利用しない場合は必要ない。

入出力バッファ(2)

- 実行時の環境変数F_FFSETBUFnn
 - 非同期入出力用バッファ領域を指定
setenv F_FFSETBUF50 1 ! 装置番号50に対し1MB
setenv F_FFSETBUF 1 ! 全て1MB
(詳細は、HPF並列入出力利用の手引 1.4.3参照)



	Fortran	HPF
1CPU	512.0MB	
2CPU	640.0MB	544.1MB

PTSUM

- 粒子コードの中核部分
- マッピングの指定
 - 粒子と関連する作業配列はBLOCK分散

```
COMMON /COMV1/X(ipt2,2),Y(ipt2,2),Z(ipt2,2),XM(ipt2,2),YM(ipt2,2),ZM(ipt2,2)
!HPF$ DISTRIBUTE (*,BLOCK) :: X,Y,Z,XM,YM,ZM
COMMON /WORK2/WW1(0:ISTP,LI,LJ,LK,2),WW2(0:ISTP,LI,LJ,LK,2),
&WW3(0:ISTP,LI,LJ,LK,2)
!HPF$ DISTRIBUTE (*,*,*,*,BLOCK) :: WW1,WW2,WW3
```
- -Minfoオプションにより、通信が発生していることが分かったループに、independent指示文を指定
 - expensive communication:

問題点と調査(1)

- そのままでは、HPFでは動作しない。
 - メモリ不足のため
- 翻訳時オプション-Moverlap=size:0を指定
 - 実行性能が極端に悪い。

	Fortran	HPF
1CPU	49.77(s)	
2CPU	25.20(s)	689.11(s)

- ベクトル化状況を調査
 - 翻訳時オプション、-R1及び-Wf'-pvctl fullmsg'
(-R1:変形リスト出力、fullmsg:詳細な診断メッセージ出力)
f90: opt(1589): ptsum.f, line 248: 外側ループを内側ループと入れ換えた
f90: opt(1036): ptsum.f, line 294: 異なる繰り返しで定義された値を参照...

問題点と調査(2)

- 手続PTSUM2で、最外側ループ(ループ長1)のループがベクトル化されているのが原因

```
PARAMETER(IPT=60000000,IPT2=IPT/2,ISTP=500)
PARAMETER(nprc=2)
.....
!HPF$ INDEPENDENT
246      DO 30 n=1,nprc  ! HPFによる並列化によりdo n= n$indl,n$indu
247          DO 30 J=1,ISTP  ! 依存無し
248          DO 20 I=1,MPT  ! 依存は不明
```

 ベクトル化

```
247      do j=1,500
248          do i=1,60000
!CDIR      NODEP
246          do n= 1, n$indu + 1 - n$indl  ! 最外側ループがベクトル化
```

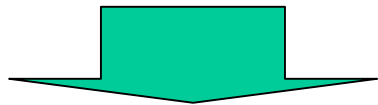
チューニング

- 最外側ループにNOVECTOR指示行を挿入
- ベクトル化を抑止

```
!HPF$ INDEPENDENT
```

```
!CDIR NOVECOTR
```

```
247      DO 30 n=1,nprc  ! HPFによる並列化によりdo n= n$indl,n$indu  
248          DO 30 J=1,ISTP  ! 依存無し  
249          DO 20 I=1,MPT  ! 依存は不明
```



ベクトル化

```
249      do i=1,60000  
      !CDIR  NODEP  
248      do j=1,500      ! Jのループ(248行目)がベクトル化
```

結果

•実行性能

	Fortran	HPF
1CPU	49.77(s)	
2CPU	25.20(s)	27.76(s)

•メモリ量

	Fortran	HPF
1CPU	35.9GB	
2CPU	35.9GB	38.9GB

ただし

• 手続 PTSUM2 (PTSUM)

```
DO n=1,nprc    ! nprc = 2
  DO i=1,ISTP
    DO j=1,MPT
      K=(i-1)*ISTP + J
      JX = IDNINT(XM(K,n)*DXI+1.5D0) ! nによりXMの前半/後半を制御
      WW1(J,JX,JY,JZ,n) = WW1(J,JX,JY,JZ,n) + .....
```

• オリジナルのFORTRAN77コード対応部分

```
DO ksp=1,2
  DO i=1,IPNE-1,ISTP
    DO j=1,min(ISTP,IPINE-i)
      K=i + J - 1 + (ksp - 1)*IPT2 ! kspによりXMの前半/後半を制御
      JX = KBPX(IDNINT(XM(K)*DXI+1.5D0))
      WW1(JX,JY,JZ,J) = WW1(JX,JY,JZ,J) + .....
```

まとめ

- メモリ量が過大にとられる場合
 - `SHADOW`指示文、又は翻訳時オプション`-Moverlap=size:n`により、必要最小限のシャドウを指定
 - 実行時の環境変数`F_FFSETBUFnn`により、入出力用バッファサイズを縮小(非同期入出力を利用しない場合は、1でよい)
- 極端に性能が悪い場合
 1. `-Minfo`オプションにより、HPFによる並列化が適切か、無駄な通信がでていないかをチェック
 2. `-R1、-Wf"-pvctl fullmsg"`オプションによりFORTRANコンパイラによるベクトル化をチェック