Orchestrating a brighter world

NEC

Introduction to NEC HPF

2020/9

NEC

Table of Contents

Introduction to HPF

High Performance Fortran (HPF) とは

■Fortran 95の分散メモリ型並列計算機向け拡張仕様

- 特徴

国際的な標準仕様

●並列処理の主要ベンダ，大学，研究機関が協同で仕様策定

記述が平易

●Fortran ＋ コメント形式の指示文

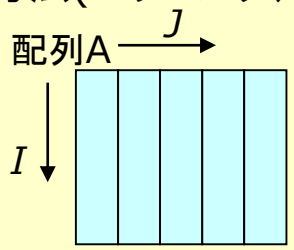
細かな制御も可能

●通信の制御，処理のプロセスへの割当て

- 例: 逐次FortranコードにHPF指示文を挿入する

```
      DIMENSION A(10000,10000)
!HPF$ DISTRIBUTE A( *, BLOCK)
!HPF$ INDEPENDENT, NEW(I)
      DO J = 1, 10000
        DO I = 1, 10000
          A(I,J) = 0.0
        END DO
      END DO
```

配列データの
分散メモリ上への
配置方法(マッピング)を指定

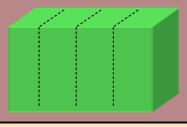


直後のループが
並列実行可能であることを明示

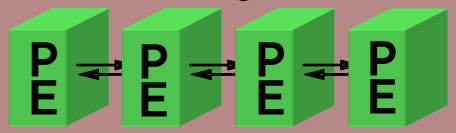
分散メモリ並列プログラミングのモデル

グローバルモデル

ユーザーは，プログラム全体の動作を記述し，データを*n*個に分割指示する



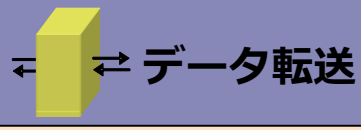
処理系が，*n*個に分割する



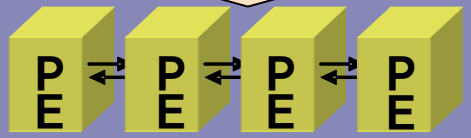
HPF

MPMD/SPMDモデル (ローカルモデル)

ユーザーは，プログラムの一部(1プロセス分)の動作を記述する



ユーザーは，*n*個同時に実行する

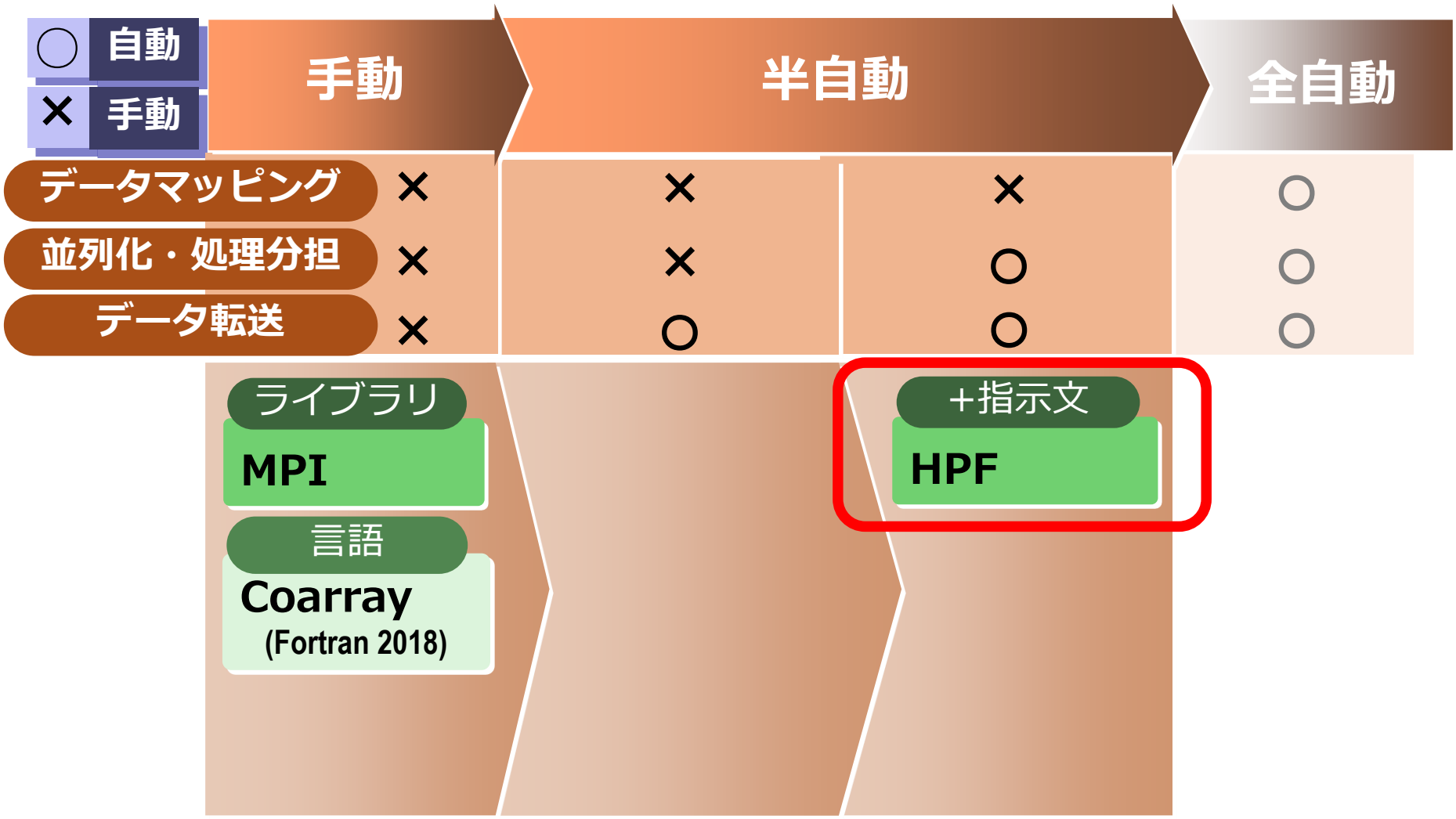


MPI, Coarray

- 規則的な問題なら，グローバルモデルのほうが並列化・保守が容易

・既存の逐次プログラムはグローバルモデル
・全体としての動作はいずれにせよ理解している必要がある

分散メモリ並列の自動化



コード例

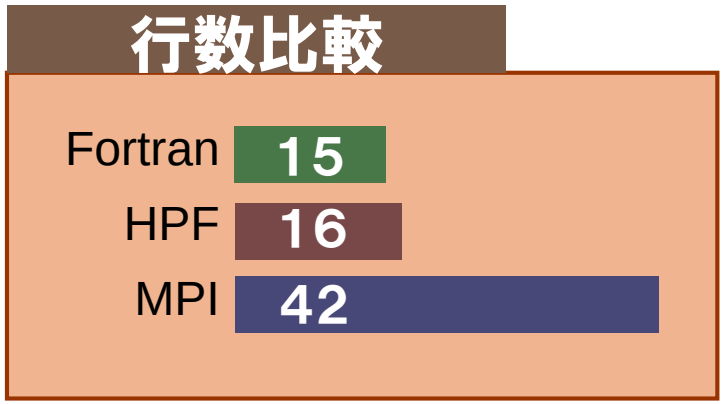
Fortran

```
parameter(n=100)
real*4 x(n),y(n)
do i=1,n
  x(i) = 1.e0 / float(i)
enddo
y = 0.e0
do i=2,n-1
   y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
enddo
dot=0.e0
do i=1,n
   dot = dot+ y(i)*y(i)
enddo
write(6,*)'dot =', dot
end
```

HPF

```
      parameter(n=100)
      real*4 x(n),y(n)
!hpf$ distribute (block) :: x,y ！データマッピング
      do i=1,n
        x(i) = 1.e0 / float(i)
      enddo
      y = 0.e0
      do i=2,n-1
         y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
      enddo
      dot=0.e0
      do i=1,n
         dot = dot+ y(i)*y(i)
      enddo
      write(6,*)'dot =', dot
      end
```

行数比較



MPI（4並列）

```
 include 'mpif.h'
 parameter(n=25)
 real*4 x(0:n+1),y(n),dotr        ！ 宣言の変更，袖の付加
 integer ist(mpi_status_size)
 call mpi_init(ierr)                                    ！ 初期化
 call mpi_comm_size(mpi_comm_world,np,ierr)
 call mpi_comm_rank(mpi_comm_world,id,ierr)
 do i=1,n
   x(i) = 1.e0 / float(i+50*id)
 enddo
 y = 0.e0
 lb = 1                                                 ！データ転送
 ub = n
 isnd1=id+1
 ircv1=id-1
 isnd2=id-1
 ircv2=id+1
 if(id.eq.0)then
    ircv1=mpi_proc_null
    isnd2=mpi_proc_null
    lb=2
 else if(id.eq.np-1)then
    isnd1=mpi_proc_null
    ircv2=mpi_proc_null
    ub=n-1
 endif
 call mpi_sendrecv(x(n),1,mpi_real,isnd1,0,
&     x,1,mpi_real,ircv1,0, mpi_comm_world,ist,ierr)
 call mpi_sendrecv(x(1),1,mpi_real,isnd2,1,
&     x(n+1),1,mpi_real,ircv2,1, mpi_comm_world,ist,ierr)
 do i=lb,ub
    y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
 enddo
 dot=0.e0
 do i=1,n
    dot = dot+ y(i)*y(i)
 enddo
 call mpi_reduce(dot,dotr,1,mpi_real,mpi_sum,0,！データ転送
&  mpi_comm_world,ierr)
 if(np.eq.0)write(6,*)'dot =', dotr
 call mpi_finalize(ierr)                                ！  終了処理
 end
```

HPFの利点

コードの開発・改変が容易	Fortran(逐次処理)のレベルで可能 物理モデル・アルゴリズムの変更など
コードの並列化が容易	煩雑なデータ転送・同期を記述しなくて良い 必要に応じ，指示文・代入文で記述可能
並列化方法の変更が容易	1次元分割→2次元分割など アルゴリズムと並列化が分離
デバッグが比較的容易	Fortran(逐次処理)のレベルで可能 MPIの場合，問題がアルゴリズムにあるか 通信・並列化にあるかの切り分けが難しい

Introduction to NEC HPF

NEC HPFの特徴

自動分散並列化機能
- 集計計算(SUM, MAXLOC等)を含むループの自動並列化・通信生成
- 袖領域の自動割付け，データ転送情報再利用による高速化
- データ分散指定翻訳時オプション

HPF2.0仕様に加え，さまざまな拡張仕様をサポート
- 主要なHPF公認拡張仕様，HPF/JA拡張仕様
- 部分的な袖通信などの独自拡張機能

HPFプログラム向けデバッグ，チューニング機能
- 分散並列化情報リスト，診断メッセージ
- HPF向けデバッグ用オプション

上級者向け機能
- 部分的にMPIを利用したチューニングが可能
- SX-Aurora TSUBASAの持つ3階層の並列性を利用可能
  - ベクトル化，共有並列化(VE内)，分散並列化(VE間/VE内)

チューニング支援機能：　分散並列化情報リスト

■ソースレベルで，分散並列化状況・データ転送発生状況を確認可能

- “COMM:”等のマークをgrepすることで，チューニングすべき箇所をピックアップ可能

並列化できないと判定されたループ(<S>)

並列化可能な場合，**INDEPENDENT指示文**を指定すれば並列化される可能性あり

通信発生箇所(COMM:)

多くの場合 性能低下の主因
ON-HOME指示文等の指定で通信を削減できる可能性あり

```
(  1  )                      subroutine sub(a,inew,iold)
(  2  )                      real a(100,100,2)
(  3  )                !HPF$ DISTRIBUTE a(*,BLOCK,*)
(  4  ) <S>---------         do j=1,100
        COMM: SFT [a] [LINO:  5  in sample.F]
(  5  ) <N>---------           do i=1,100
(  6  )   |                      a(i,j,inew)=a(i,j-1,iold)+a(i,j,iold)
(  7  )   +---------           enddo
(  8  )                      enddo
(  9  )                      end
```

並列化可能だが並列化されなかったループ(<N>)

データマッピングを変えれば，並列化される可能性あり

分散並列化支援機能：　データ分散指定オプション

データ分散を指定できる翻訳時オプション

書式　-Mautodist[=rank?[:*n*]]

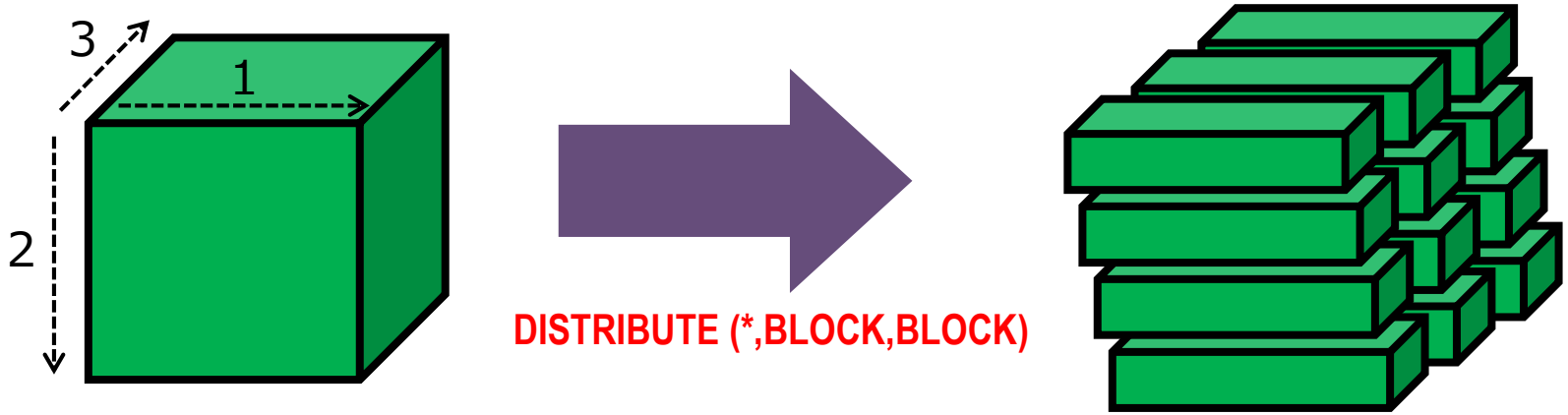
- 既定値では，全配列を最後の次元でBLOCK分散する (データマッピングが指示文で指定された配列を除く)
- =rank? を指定した場合，?次元配列を最後の次元でBLOCK分散する
- =rank?:*n* を指定した場合，2進整数*n*の1に対応する次元をBLOCK分散する

- 規則的なデータ構造のコードなら，オプションのみで並列化できる場合も

例)

3次元配列の2次元目・3次元目をBLOCK分散する場合

```
%> ve-hpf -Mautodist=rank3:011  sample.hpf
```



分散並列化支援機能：　HPFソース生成オプション

HPFソースプログラムを自動生成する翻訳時オプション

書式　-Mhpfout

●-Mautodistオプションで指定したデータ分散入りのソースを出力する

●出力されたソースを出発点に，チューニングを行うことができる

●データ分散の指定し忘れ などのミス防止にも有効

例)

配列の最後の次元を分割するHPF指示文を挿入したソースを生成

%> ve-hpf　-Mautodist　-Mhpfout　sample.hpf

```
      real, dimension(100,100)  :: a,b,c
!hpf$ distribute (*,*) :: a  ！非分散
      do j=1,100
        do k=1,100
          do i=1,100
            c(i,j) = c(i,j) + a(i,k)*b(k,j)
          enddo
        enddo
      enddo
```

sample.hpf

```
      real, dimension(100,100)  :: a,b,c
!hpf$ distribute a(*,*)  ！非分散
!hpf$ distribute b(*,block)
!hpf$ distribute c(*,block)
      do j=1,100
        do k=1,100
          do i=1,100
            c(i,j) = c(i,j) + a(i,k)*b(k,j)
          enddo
        enddo
      enddo
```

sample.hpf.src