Empowered by Innovation NEC

大阪大学サイバーメディアセンター　御中

次期スパコン（SX-ACE）
利用説明会（ソフトウェア）資料

2014年9月2日

日本電気株式会社

SX-9とSX-ACEの比較

ノードあたりの性能

	SX-9		SX-ACE
CPU数(core数)	16CPU		1CPU(4core)
最大ベクトル性能	1.6TFLOPS	×1/6.4 →	256GFLOPS
主記憶容量	1TB	×1/16 →	64GB

システム全体の性能

	SX-9(10ノード)		SX-ACE(1536ノード)
CPU数(core数)	160CPU		1536CPU(6144core)
最大ベクトル性能	16TFLOPS	×24 →	384TFLOPS
主記憶容量	10TB	×9.6 →	96TB

アプリケーションとライブラリ

利用可能アプリケーションおよびライブラリ

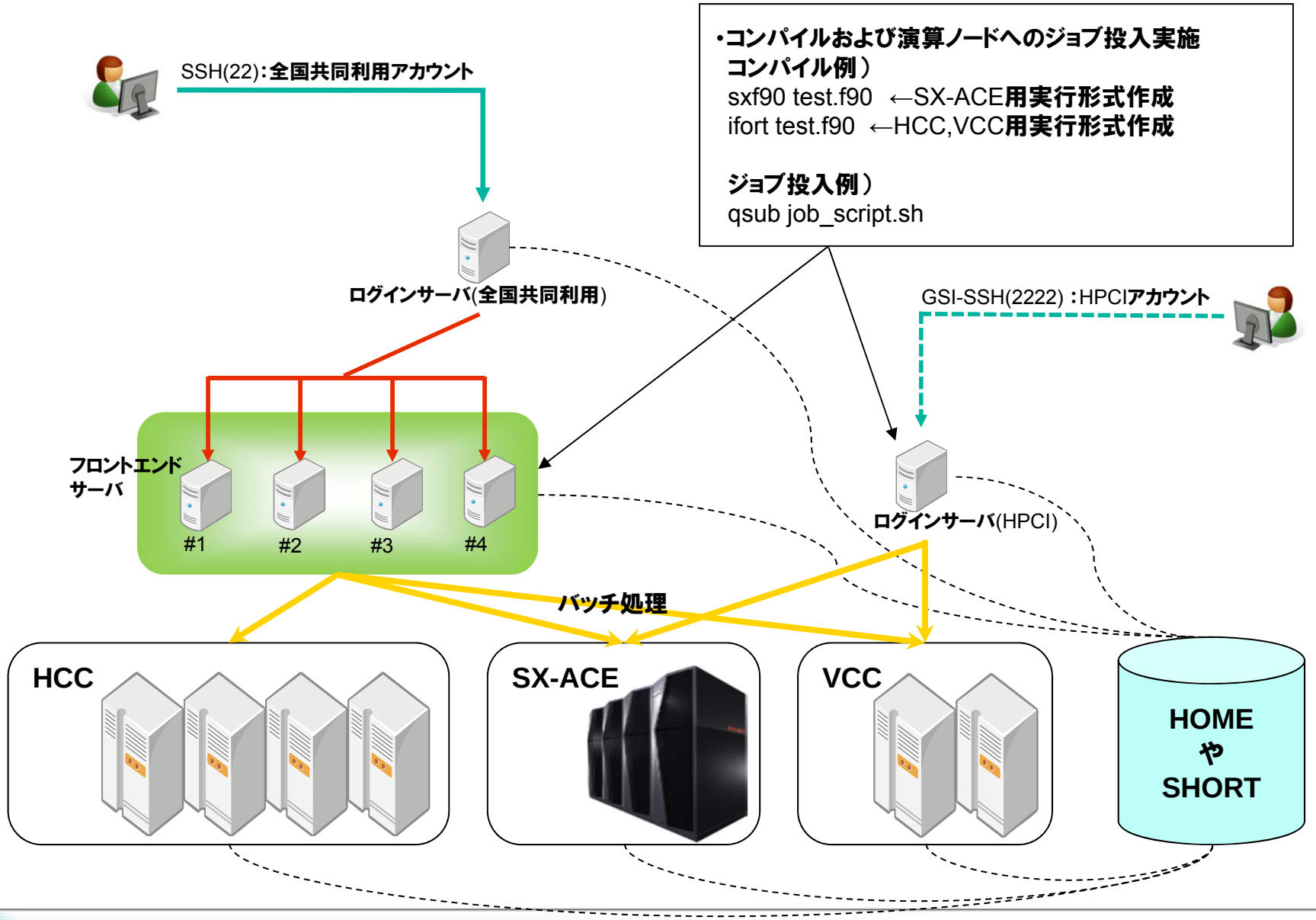
SX-ACE

分類	機能
開発環境用ソフトウェア	Fortran95/2003コンパイラ C/C++コンパイラ
MPIライブラリ	MPI/SX
HPFコンパイラ	HPF/SX V2
デバッガ	dbx/pdbx
性能解析ツール	PROGINF/FILEINF FTRACE prof
数値計算ライブラリ	ASL ASLQUAD
統計計算ライブラリ	ASLSTAT
数学ライブラリ集	MathKeisan

フロントエンド

分類	機能
開発環境用ソフトウェア	Fortran95/2003クロスコンパイラ C/C++クロスコンパイラ Intel Cluster Studio XE
HPFコンパイラ	HPF/SX V2クロスコンパイラ
デバッガ	dbx/pdbx NEC Remote Debugger
性能解析ツール	FTRACE NEC Ftrace Viewer
解析ソフトウェア	MD Nastran MSC Marc MSC Mentat MSC Dytran MSC Patran MSC ADams
汎用可視化ソフトウェア	AVS/ExpressDeveloper
計算科学用ソフトウェア	Gaussian09

ログインおよびコンパイル方法



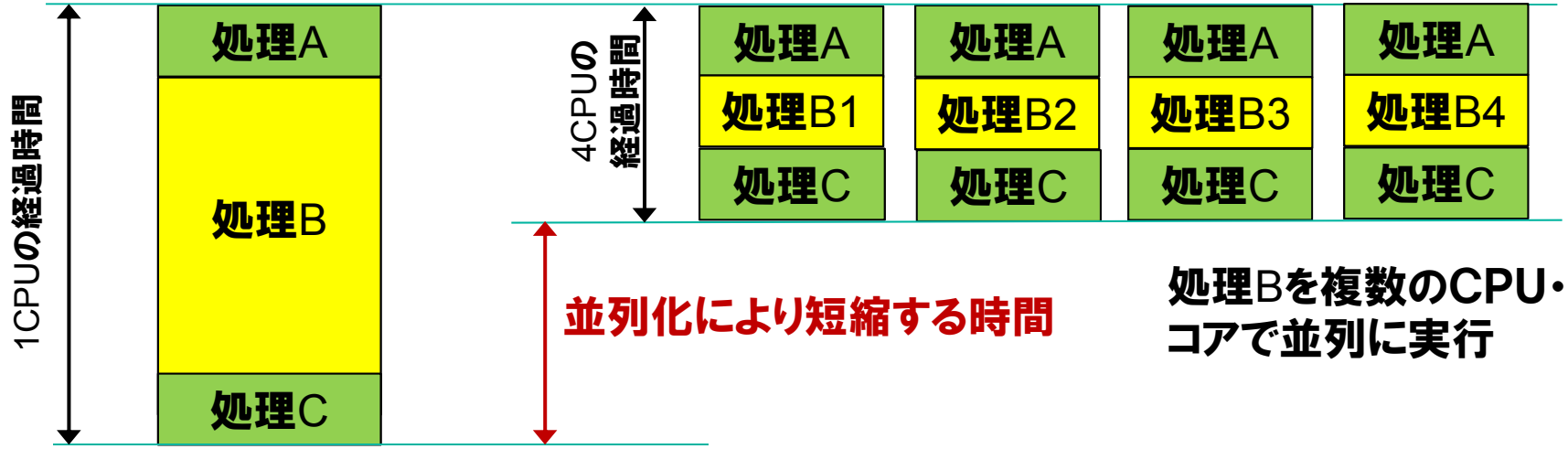
並列化概要

並列処理・並列実行

- 仕事（処理）を複数のCPU・コアに分割し、同時に実行すること

並列化

- 並列処理を可能とするために、処理の分割を行うこと

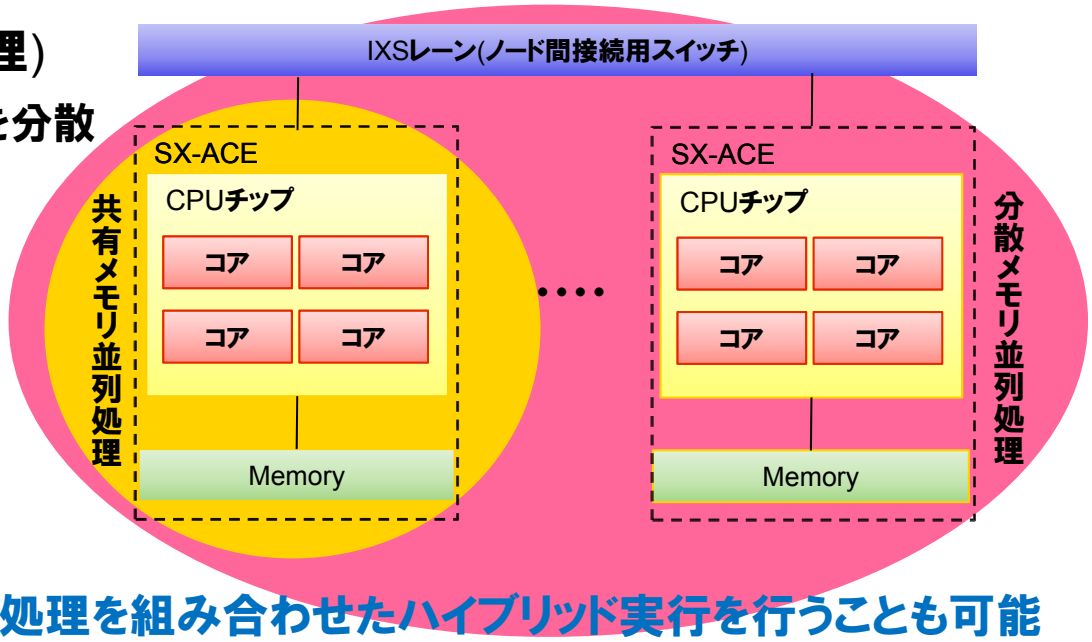


並列処理モデル on SX-ACE

SX-ACEではコンピュータアーキテクチャに応じた処理の分担(分割)方法、領域の分担(分割)方法によって幾つかの並列処理方式がある。

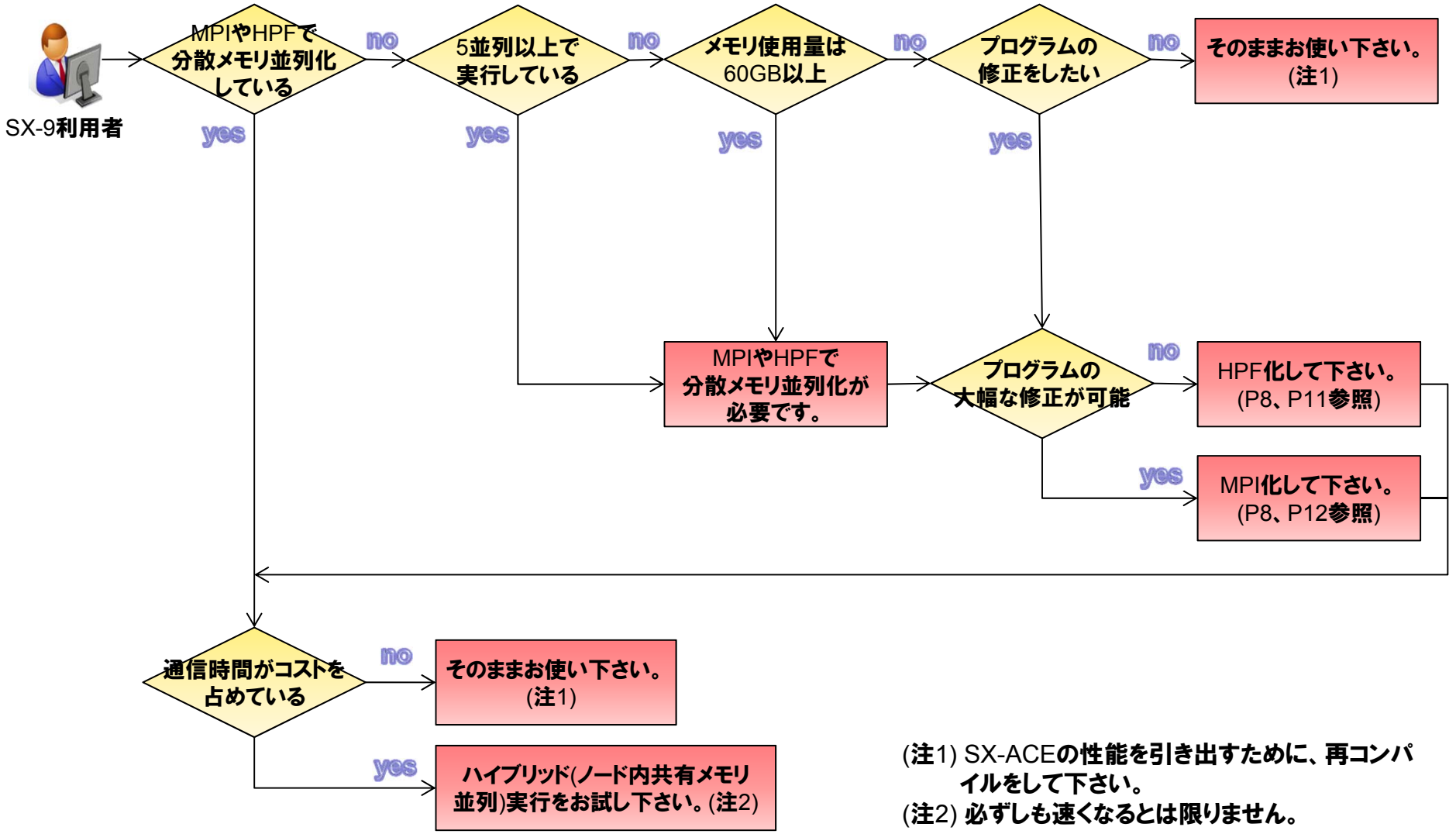
1. **ノード内並列(共有メモリ並列処理)**
   - **コア間で負荷を分散**
   - → **自動並列**,OpenMP

2. **ノード間並列(分散メモリ並列処理)**
   - **メモリ空間は独立。各ノードに負荷を分散**
   - → HPF,MPI **※ノード内でも利用可**



※分散メモリ並列処理と共有メモリ並列処理を組み合わせたハイブリッド実行を行うことも可能

SX-ACEへのプログラム移行



(注1) SX-ACEの性能を引き出すために、再コンパイルをして下さい。
(注2) 必ずしも速くなるとは限りません。

HPFとMPI(1/3)

HPF(High Performance Fortran)とは？

- Fortran95の分散メモリ型並列計算機向け拡張仕様で以下の特徴を有する
  - 国際的な標準仕様(並列処理を主要ベンダ，大学，研究機関で共同で仕様策定)
  - 記述が平易(Fortran＋コメント形式の指示文)
  - 上級ユーザは細かな制御も可能(通信の制御，部分的にMPIを利用したチューニング等)

MPI(Message Passing Interface)とは？

- 分散メモリ並列処理におけるメッセージパッシングの標準規格
  - 複数のプロセス間でのデータをやり取りするために用いるメッセージ通信操作の仕様標準
- FORTRAN，C，C++から呼び出すサブプログラムのライブラリ
- ポータビリティに優れている
  - 標準化されたライブラリインターフェースによって，様々なMPI実装環境で同じソースをコンパイル・実行できる
- プログラマの負担が比較的大きい
  - プログラムを分析して，データ・処理を分割し，通信の内容とタイミングをユーザが自ら記述する必要がある

HPFとMPI(2/3)

分散並列プログラム開発には3つの局面が存在する

① 複数CPU,コアへのデータの分割配置(マッピング)

② 複数CPU,コアへの並列化・処理の分担

③ 複数CPU,コア間の通信処理の挿入

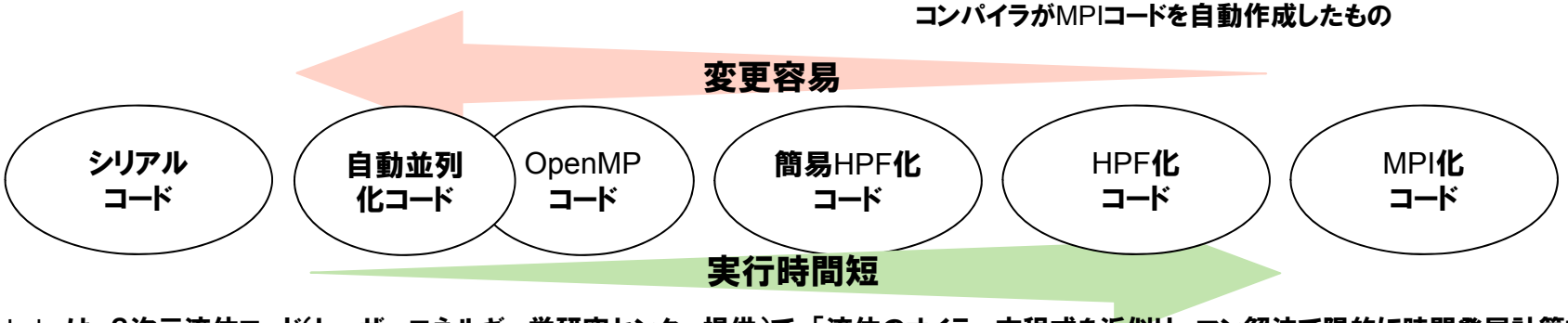
	HPF	MPI
①データの分割配置	手動	手動
②並列化・処理の分担	自動	手動
③通信処理の挿入	自動	手動
プログラム例	P11	P12

HPFとMPI(3/3)

3DHydro(約7,000行)の並列化による比較 on SX-ACE

		1ノード	8ノード
シリアルコード	オリジナルコード	1408.6秒	–
自動並列化コード	コンパイラオプションの追加のみ	511.9秒	–
OpenMP並列化コード	約550行の追加のみ	499.7秒	–
簡易HPF化コード	コンパイラオプションの追加および6行の指示行追加	–	88.0秒
HPF化コード	約150行の指示行追加のみ	–	73.3秒
MPI化コード	すべて手動で並列化. 約1,500行の追加・修正必要	–	67.7秒

※簡易HPF化コード
コンパイルオプションにてデータ分割配置方法を指定することで
コンパイラがMPIコードを自動作成したもの



※3DHydroは、3次元流体コード(レーザーエネルギー学研究センター提供)で、「流体のオイラー方程式を近似リーマン解法で陽的に時間発展計算するコード」です。次の時間ステップの計算において隣り合う格子の情報しか必要でないため、領域分割が容易でHPF、MPI化に適したコードとなります。

HPFによる並列化例

例）3DHydro(一部)

```
 MODULE interp

!HPF$ PROCESSORS P(NUMBER_OF_PROCESSORS())
!HPF$ DISTRIBUTE (*,*,BLOCK) ONTO P ::
!HPF$*           sr3d,  su3d,  sp3d,  sv3d,  sw3d,  se3d
!HPF$*         , dr,    du,    dp,    dv,    dw
!HPF$*         , sr3dr, su3dr, sp3dr, sv3dr, sw3dr
!HPF$*         , sr3dl, su3dl, sp3dl, sv3dl, sw3dl
!HPF$*         , sr3dh, su3dh, sp3dh, sv3dh, sw3dh

 END MODULE inte

 subroutine roe_boundx(mstep)

!HPF$ ON HOME(sm3dh(:,:,iz)),LOCAL BEGIN
        do iy=1,ly
             sm3dh(2,iy,iz) = -sm3dh(3,iy,iz)
             sn3dh(2,iy,iz) =  sn3dh(3,iy,iz)
             sl3dh(2,iy,iz) =  sl3dh(3,iy,iz)
             sr3dh(2,iy,iz) =  sr3dh(3,iy,iz)
c
             sm3dh(1,iy,iz) = -sm3dh(4,iy,iz)
             sn3dh(1,iy,iz) =  sn3dh(4,iy,iz)
             sl3dh(1,iy,iz) =  sl3dh(4,iy,iz)
             sr3dh(1,iy,iz) =  sr3dh(4,iy,iz)
             se3dh(1,iy,iz) =  se3dh(4,iy,iz)
        enddo
!HPF$ END ON

      return
      end
```

CPU,コア構成を宣言する

データの分割配置(マッピング)を指定する

部分配列がマップされているCPU,コアが実行し、通信が不要であることをコンパイラへ教える

```
 subroutine cfl

!HPF$ INDEPENDENT,NEW(ix,iy,iz,wuu,wvv,www,wcc)
!HPF$*           ,REDUCTION(max:sram)
 do iz = 1, lz
       do iy = 1, ly
        do ix = 1, lx
         wuu = su3d(ix,iy,iz)
         wvv = sv3d(ix,iy,iz)
         www = sw3d(ix,iy,iz)
         wcc = sqrt( fixedgamma * sp3d(ix,iy,iz) / sr3d(ix,iy,iz) )
         sram = max( sram,
     &                      (abs(wuu) + wcc)/sdltx ,
     &                      (abs(wvv) + wcc)/sdlty ,
     &                      (abs(www) + wcc)/sdltz )
        end do
       end do
      end do

 end
```

並列化可能であることを指定する。また任意の集計計算を含むためREDUCTION節を指定する。

※HPF並列化による性能向上には限界がありますので、更なる高速化が必要な場合はMPI化の検討をおねがいします。

※3DHydroは、3次元流体コード(レーザーエネルギー学研究センター提供)です。

MPIによる並列化例

例）3DHydro(一部)

```
      Parameter(   isphyx = 1020  )   !X-direction
      Parameter(   isphyy =  256  )   !Y-direction
      Parameter(   isphyz =  256  )   !Z-direction
      Parameter( npartx=1 ,nparty=4, npartz=8 )
      Parameter( lx =  isphyx/npartx + 4  )
      Parameter( ly =  isphyy/nparty + 4  )
      Parameter( lz =  isphyz/npartz + 4  )

 program  apr_main

      call mpi_init(ierr)
      call mpi_comm_size(mpi_comm_world,ncpu,ierr)
      call mpi_comm_rank(mpi_comm_world,id_cpu,ierr)


  call MPI_FINALIZE( ierr )

end program  apr_main

 subroutine zeroclr

 call MPI_ALLREDUCE(zseend,
     >                    seend,
     >                    1,
     >                    MPI_DOUBLE_PRECISION,
     >                    mpi_sum,
     >                    mpi_comm_world,
     >                    ierr)

 end
```

空間分割

MPIの初期化

実行プロセス数の取得

自プロセス番号の取得

MPIの終了

全プロセスと通信

```
subroutine sender3_type1_mstep1

call mpi_isend(sr3dh(1,1,3),
     >                 1,
     >                 ijvec,
     >                 npz(1),
     >                 1,
     >                 mpi_comm_cart,
     >                 ireq1(0),
     >                 ierr)

call mpi_irecv(sr3dh(1,1,1),
     >                 1,
     >                 ijvec,
     >                 npz(1),
     >                 2,
     >                 mpi_comm_cart,
     >                 ireq1(3),
     >                 ierr)

end
```

指定されたプロセス同士で通信

※3DHydroは、3次元流体コード(レーザーエネルギー学研究センター提供)です。

講習会のお知らせ

1. スーパーコンピュータと並列コンピュータの高速化技法の基礎

開催時期：未定（12月～2月開催予定）

場所：サイバーメディアセンター豊中教育研究棟

2. MPIプログラミング入門

開催時期：未定（12月～2月開催予定）

場所：サイバーメディアセンター豊中教育研究棟

3. HPFプログラミング入門

開催時期：未定（12月～2月開催予定）

場所：サイバーメディアセンター豊中教育研究棟