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）
第二回利用説明会（ソフトウェア）資料

2014年11月26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

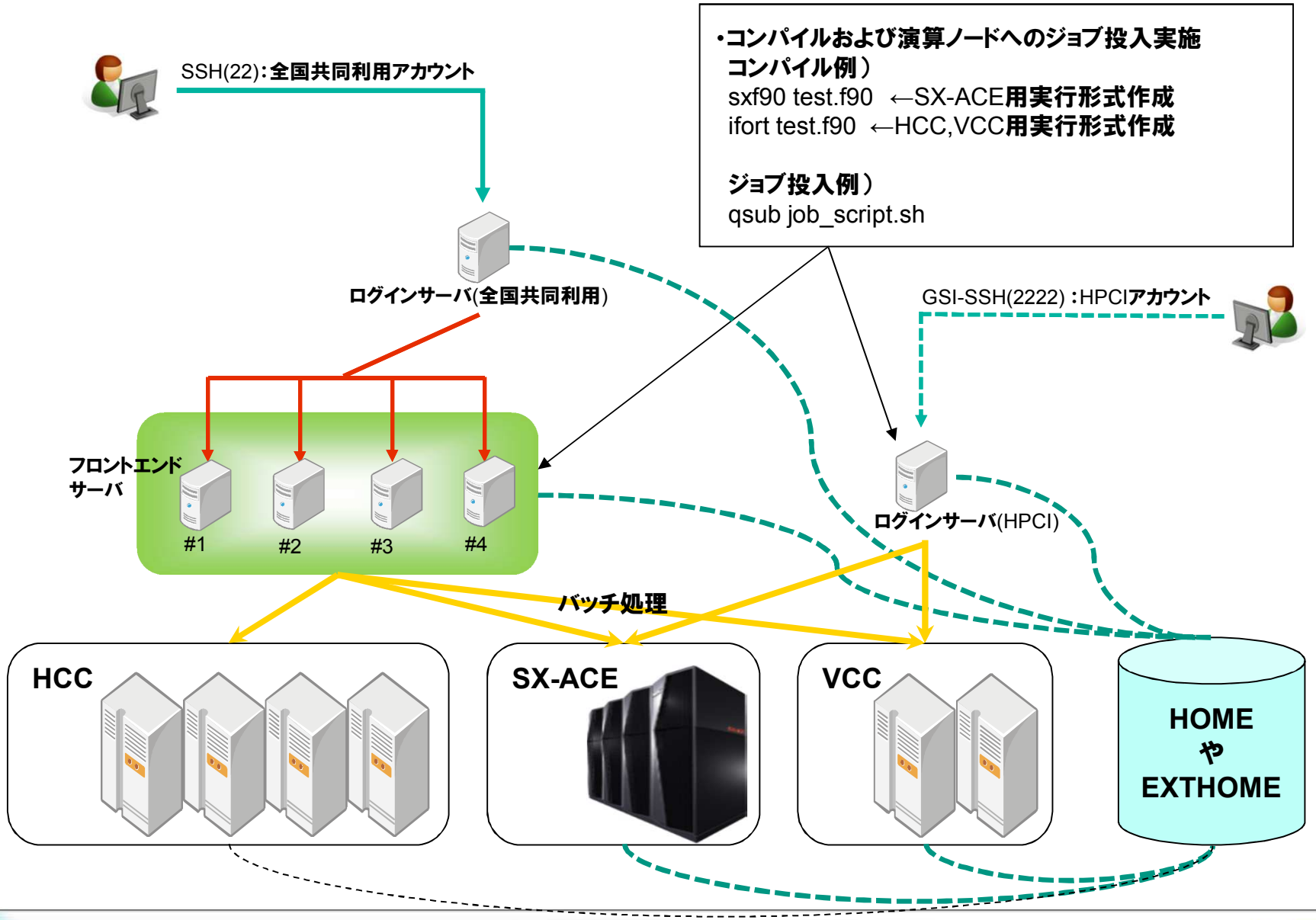
SX-ACE

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



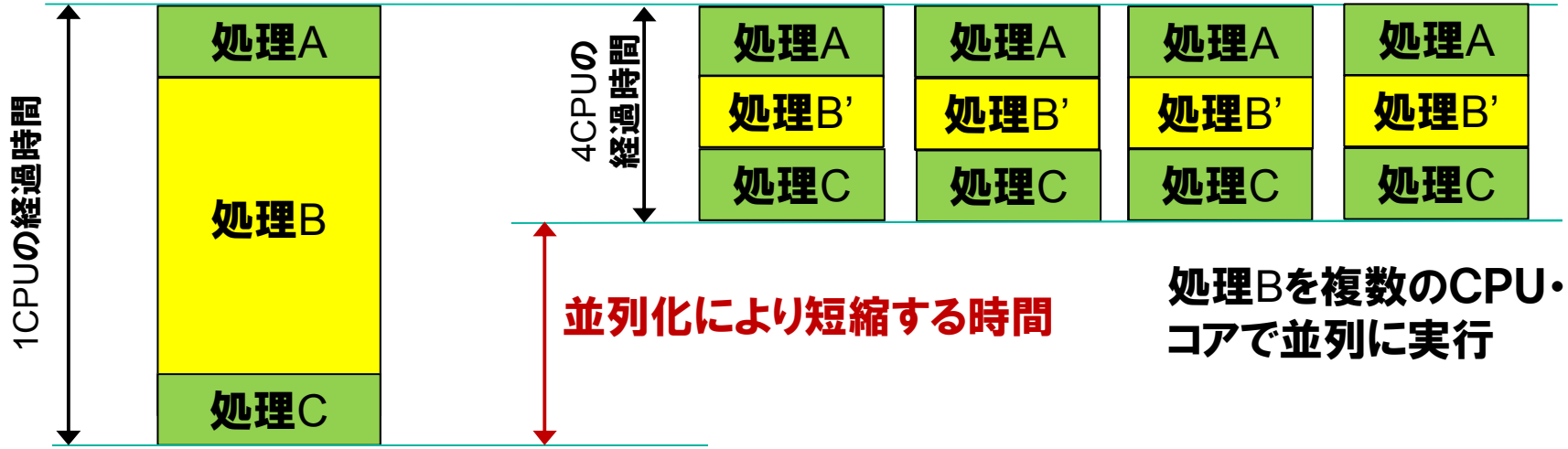
並列化概要

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

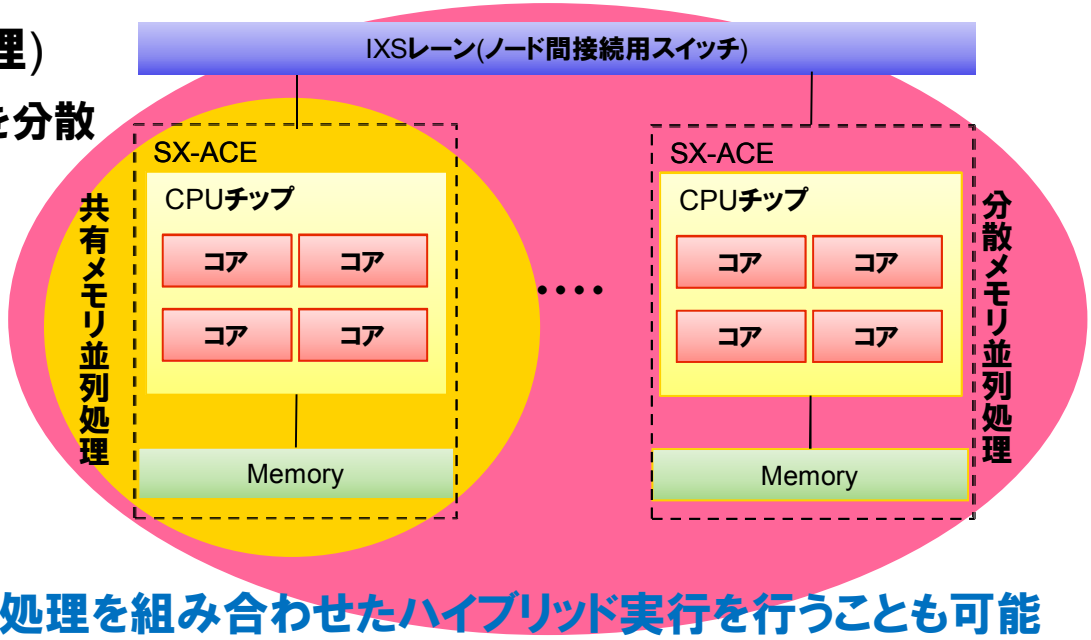


並列処理モデル on SX-ACE

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。

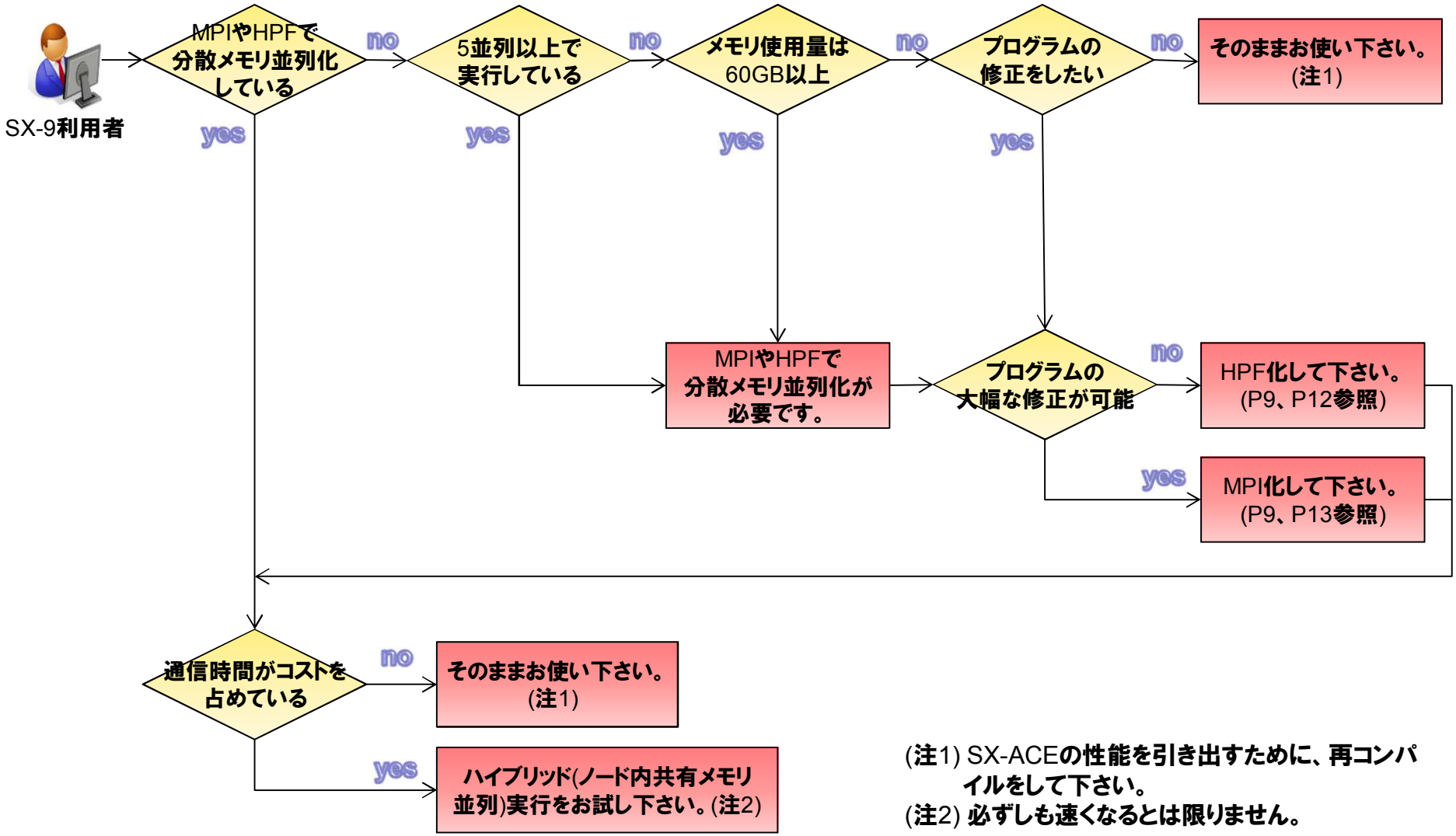
1. ノード内並列(共有メモリ並列処理)
   - コア間で負荷を分散
   - → 自動並列,OpenMP

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

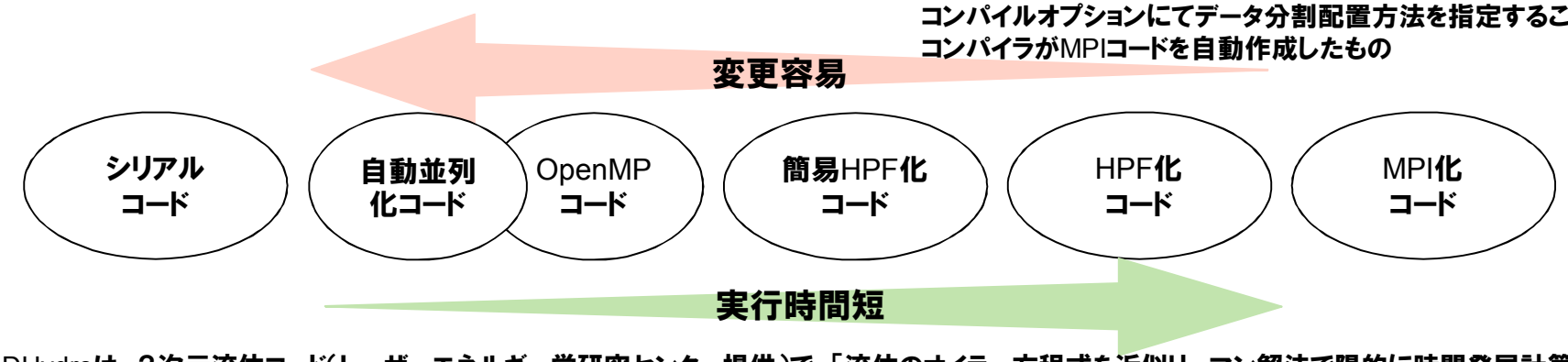
	HPF	MPI
①データの分割配置	手動	手動
②並列化・処理の分担	自動	手動
③通信処理の挿入	自動	手動
プログラム例	P12	P13

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

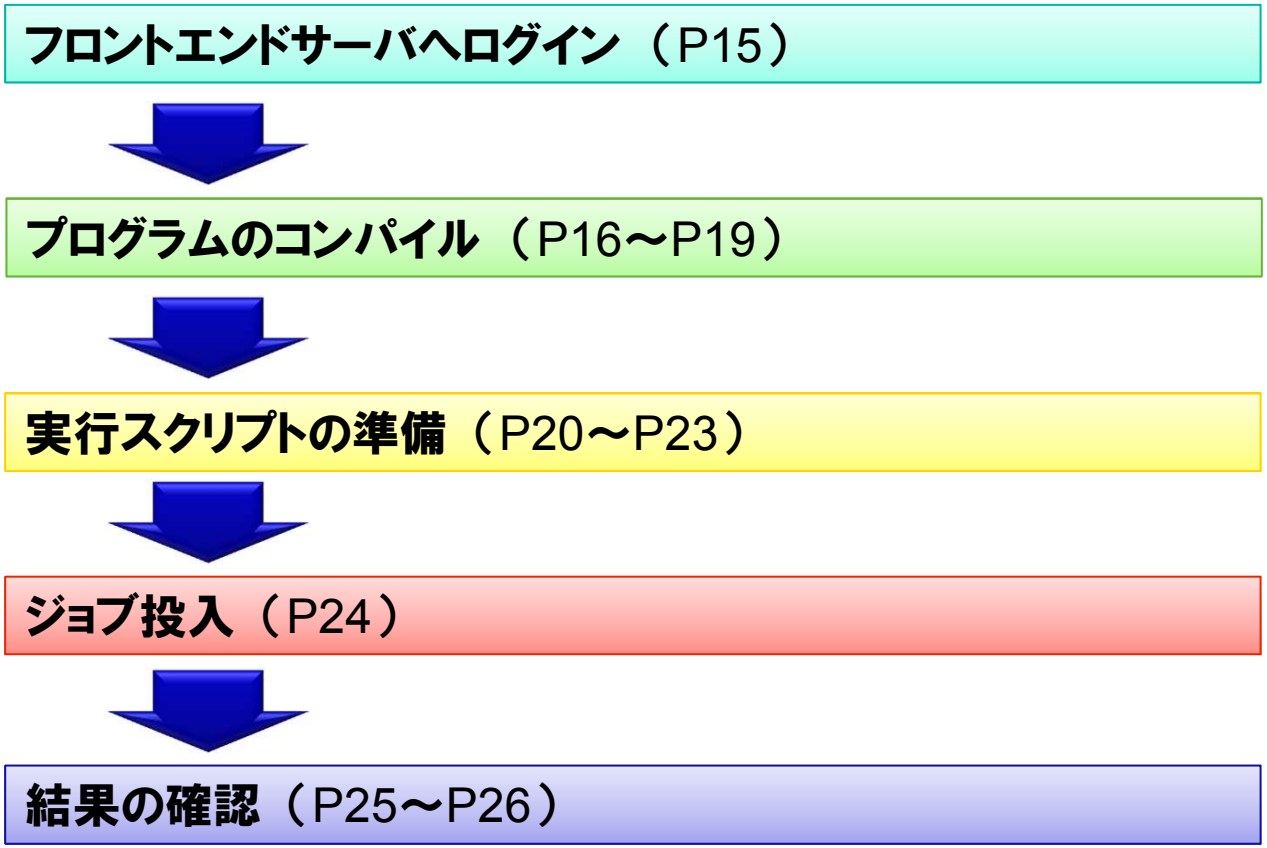
```
subroutine sender3_type1_mstep1

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．

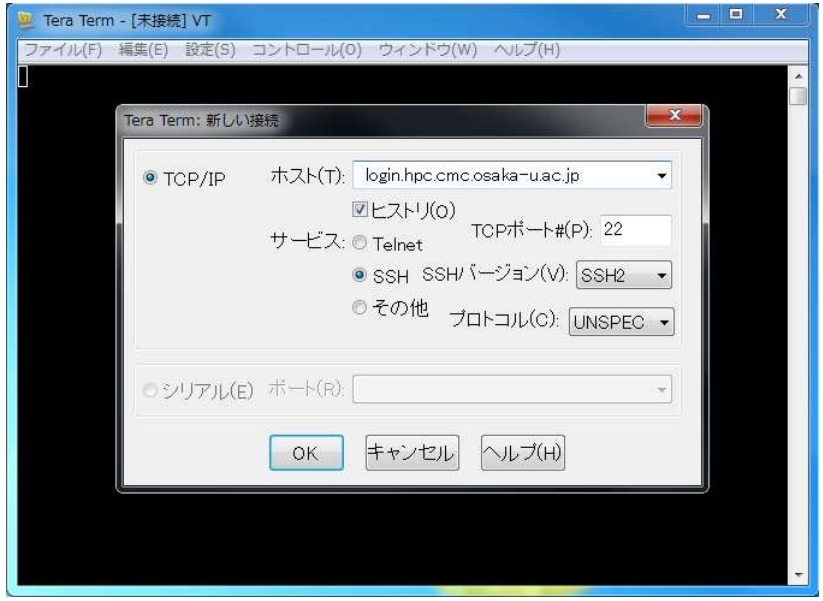
SX-ACEジョブ実行の流れ

SX-ACEを利用するジョブ実行の流れは以下



フロントエンドサーバへログイン(全国共同利用アカウント)

フロントエンドサーバへログインいたします. Teraterm**等**SSH**クライアントソフトを利用して、ログインサーバ「login.hpc.cmc.osaka-u.ac.jp」へログインします(下図の例は**Teratermの**ログイン画面です).**



- **ログインサーバの選択はなく, ログイン数の少ないサーバに自動でログインします.**
- HPCI**アカウントの方は,「**GSI-SSHTerm**」を利用してログインサーバ「**glogin.hpc.cmc.osaka-u.ac.jp**」へログインします.**

プログラムのコンパイル(1)

プログラムのコンパイルはフロントエンドサーバ上で行います.

①　②　③

```
> sxf90 -P□□□ -C△△△ ○○○ ソースファイル名
```

①共有メモリ並列化オプション　-Pauto：自動並列化／-Popenmp：OpenMP並列化
リンク時も同じ指定が必要. -Pautoと-Popenmpの混在不可

②コンパイルモードの選択

-Chopt：最大限の最適化処理およびベクトル化処理を行うことを指定. 副作用を伴う場合あり. エラーチェックを省略
-Cvopt：最大限の最適化処理を行い、既定レベルのベクトル化処理を行うことを指定(既定値)
-Cvsafe：最適化処理およびベクトル化処理は行うが、副作用を伴う可能性のある機能は抑止することを指定
-Csopt：最大限の最適化処理を行い、ベクトル化処理を抑止することを指定
-Cssafe：ベクトル化処理を抑止し、最適化処理も副作用を伴う機能は抑止することを指定

プログラムのコンパイル(2)

③代表的なオプション指定　-ftrace：**簡易性能解析機能対応のオブジェクトを生成**
-pi　：**自動インライン展開を行う(詳細オプションあり)**
-R2　：**編集リストと変形リストの両方を出力**

④詳細オプション指定　-Wf”(**オプション指定**)”
-Wf”-pvctl fullmsg”　：**ベクトル化診断メッセージの出力**
-Wf”-O extenreorder”：**命令の並び替えによる最適化**

※コンパイルオプションの詳細は「FORTRAN90/SX **プログラミングの手引き」**
「FORTRAN90/SX **並列処理機能利用の手引き」に記載しています.**

※サイバーメディアセンター開催の講習会でも詳細を説明します.

プログラムのコンパイル(3)

- MPIで記述されたプログラムのコンパイル方法は以下の通りです(フロントエンドサーバ上で行います).

```
> sxmpif90 ○○○ ソースファイル名
```

オプションの指定. P16に記載のオプションをそのまま使用することが可能. MPIライブラリの指定は必要ありません(-lmpiは不要).

- MPIプログラムと共有メモリ並列のハイブリッド(ノード内は共有メモリ並列, ノード間はMPI並列)並列する場合は, 以下のように指定します.

```
> sxmpif90 { -Pauto    } ○○○ ソースファイル名
           { -Popenmp }
```

- 自動並列化(-Pauto)もしくはOpenMP並列化(-Popenmp)を選択します.

プログラムのコンパイル(4)

HPFで記述されたプログラムのコンパイル方法は以下の通りです(フロントエンドサーバ上で行います).

```
> sxhpf -Mlist2 ○○○ ソースファイル名
```

オプションの指定. P16に記載のオプションをそのまま使用することが可能.

- -Mlist2オプションは並列化情報リストを採取するものです(推奨オプション)

HPFプログラムと共有メモリ並列のハイブリッド(ノード内は共有メモリ並列, ノード間はMPI並列)並列する場合は, 以下のように指定します.

```
> sxhpf -Mlist2  -Pauto  ○○○  ソースファイル名
```

- HPFコンパイル時は, 自動並列化(-Pauto)のみの指定可能です.

NQSスクリプトの準備(1)

ジョブの実行を行うNQSスクリプトを用意します．NQSスクリプトにはジョブクラスなどの情報の記入が必要です．

```
#!/bin/csh
#PBS -q ①
#PBS -l cpunum_job=②,memsz_job=③,elapstim_req=④
```

① ジョブクラスを設定します．共有利用のジョブクラスは「ACE」と指定します．占有利用の方は別途通知します．

② 使用するコア数を設定します．ノード内のコア数である「4」と指定してください．

③ 使用するメモリ容量を設定します．ノード内の利用可能な最大メモリ容量である「60GB」と指定してください．

④ 使用計算時間(経過時間)を設定します．「HH:MM:SS」のように指定します．指定可能な最大時間は24時間です．占有利用の場合は制限値はありません．

NQSスクリプトの準備(2)

共有メモリ並列のシングルノードジョブの場合は以下のように記述します.

```
#!/bin/csh
#PBS -q ①
#PBS -l cpunum_job=②,memsz_job=③,elapstim_req=④
#PBS -v F_RSVTASK=⑤
setenv F_PROGINF DETAIL
cd $PBS_O_WORKDIR
./a.out (実行文)
```

①~④はP20参照

⑤ スレッド数を指定. 最大「4」になります. OpenMP並列の場合は

#PBS -v OMP_NUM_THREADS=⑤

と指定します.

NQSスクリプトの準備(3)

MPIもしくはHPFコンパイラで作成の分散メモリ並列のマルチノードジョブの場合は以下のように記述します(HPFコンパイラで作成されたモジュールもmpirunで実行します). 下記の例はMPI並列のみの例です.

```
#!/bin/csh
#PBS -q ①
#PBS -l cpunum_job=②,memsz_job=③,elapstim_req=④
#PBS -T mpisx
#PBS -b ⑤
setenv MPIPROGINF DETAIL
cd $PBS_O_WORKDIR
mpirun -nn ⑤ -np ⑥ ./a.out
```

①~④はP20参照.

⑤ 使用するノード数を指定.

⑥ プロセス数を指定(全コアで実行する場合は⑥=②×⑤).

NQSスクリプトの準備(4)

- MPIもしくはHPFコンパイラで作成の分散メモリ並列のマルチノードジョブでノード内は共有メモリ並列(自動並列もしくはOpenMP並列．ただしHPFコンパイラはOpenMP並列を利用できません)．

```
#!/bin/csh
#PBS -q ①
#PBS -l cpunum_job=②,memsz_job=③,elapstim_req=④
#PBS -T mpisx
#PBS -b ⑤
#PBS -v F_RSVTASK=⑥
setenv MPIPROGINF DETAIL
cd $PBS_O_WORKDIR
mpirun -nn ⑤ -np ⑦ ./a.out
```

①~④はP20参照．

⑤ 使用するノード数を指定．

⑥ スレッド数を指定(P21を参照)．

⑦ プロセス数を指定(ノード内はすべて共有メモリ並列の場合は⑦＝⑤)

ジョブ投入・状態監視・ジョブ削除

ジョブの投入はqsubコマンドを使用します.

```
> qsub  [NQSスクリプトファイル名]
```

- ジョブが受付けられた後に, リクエストIDが返ります.

ジョブ開始予定時間はsstatコマンドで確認できます.

```
> sstat
```

ジョブの状態監視はqstatコマンドを使用します. qsub時のリクエストIDで確認してください.

```
> qstat
```

ジョブの削除はqdelコマンドを使用します.

```
> qdel  [リクエストID]
```

結果の確認(1)

ジョブが終わりましたら，標準出力ファイルと標準エラー出力ファイルが作成されます．

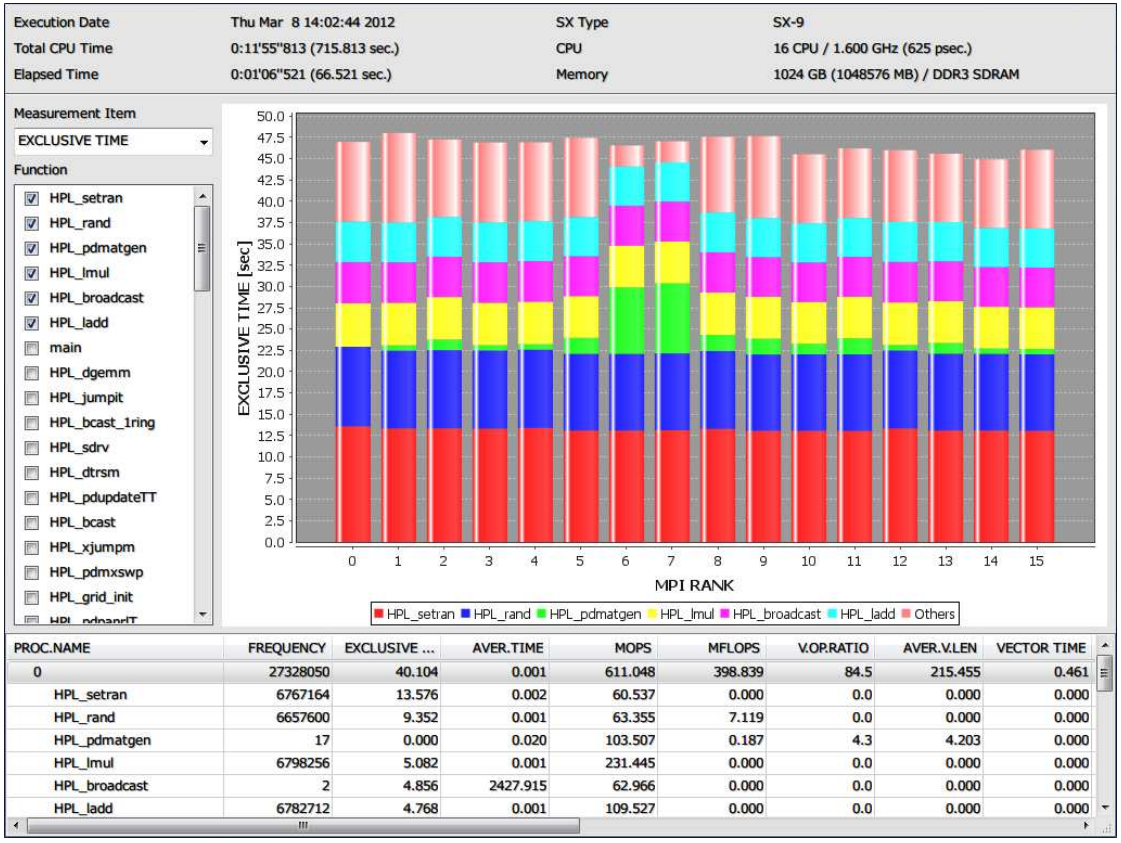
- スクリプト中に “#PBS　-o” ，“#PBS　-e” でファイル名を指定している場合は，それぞれ標準出力，標準エラー出力の内容が，指定した名前のファイルで実行ディレクトリに保存されます（既に存在する場合は上書きされます）．
- 出力ファイル名を指定していない場合は，下記の規則でファイルを自動生成して出力します．

標準出力ファイル　　：[NQSスクリプト名].o[リクエストID]
標準エラー出力ファイル：[NQSスクリプト名].e[リクエストID]

※バッチリクエストに問題が発生した際，調査の為に各出力ファイルとリクエストIDが必要となりますので，できるだけ “#PBS　-o” ，“#PBS　-e” は使用しない様にお願いします．

結果の確認(2)

新システムから簡易解析ツールFtraceの結果を視覚的に確認することが可能です．Ftrace Viewerを使用します．



PROC.NAME	FREQUENCY	EXCLUSIVE ...	AVER.TIME	MOPS	MFLOPS	V.OP.RATIO	AVER.V.LEN	VECTOR TIME
0	27328050	40.104	0.001	611.048	398.839	84.5	215.455	0.461
HPL_setran	6767164	13.576	0.002	60.537	0.000	0.0	0.000	0.000
HPL_rand	6657600	9.352	0.001	63.355	7.119	0.0	0.000	0.000
HPL_pdmatgen	17	0.000	0.020	103.507	0.187	4.3	4.203	0.000
HPL_lmul	6798256	5.082	0.001	231.445	0.000	0.0	0.000	0.000
HPL_broadcast	2	4.856	2427.915	62.966	0.000	0.0	0.000	0.000
HPL_ladd	6782712	4.768	0.001	109.527	0.000	0.0	0.000	0.000

※詳細は講習会で説明します．

インタラクティブ・バッチ

SX-ACEに直接ログインすることはできませんが，インタラクティブ・バッチ機能を使用することで，会話型のように使用することが可能になります．

```
> qlogin -q ① -l “elapstim_req=②,cpunum_job=③"
```

① インタラクティブキュー名を指定します．キュー名は別途通知します．
② インタラクティブ・バッチの開始から終了までの時間を指定します．単位は秒．
③ 使用するコア数を指定します．指定可能なコア数は別途通知します．

- qloginコマンドを実行しますと以下のようなプロンプトが返り，以降会話形式でコマンド実行が可能です．

```
Request 1234.cmc submitted to queue: [インタラクティブキュー名].
Waiting for 1234.cmc to start.
>
```

- 終了する場合は「exit」を入力します．

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

Orchestrating a brighter world

世界の想いを、未来へつなげる。

未来に向かい、人が生きる、豊かに生きるために欠かせないもの。
それは「安全」「安心」「効率」「公平」という価値が実現された社会です。

NECは、ネットワーク技術とコンピューティング技術をあわせ持つ類のないインテグレーターとして
リーダーシップを発揮し、卓越した技術とさまざまな知見やアイデアを融合することで、
世界の国々や地域の人々と協奏しながら、
明るく希望に満ちた暮らしと社会を実現し、未来につなげていきます。

Empowered by Innovation

NEC