HPF言語と今後のプログラミングインタフェース

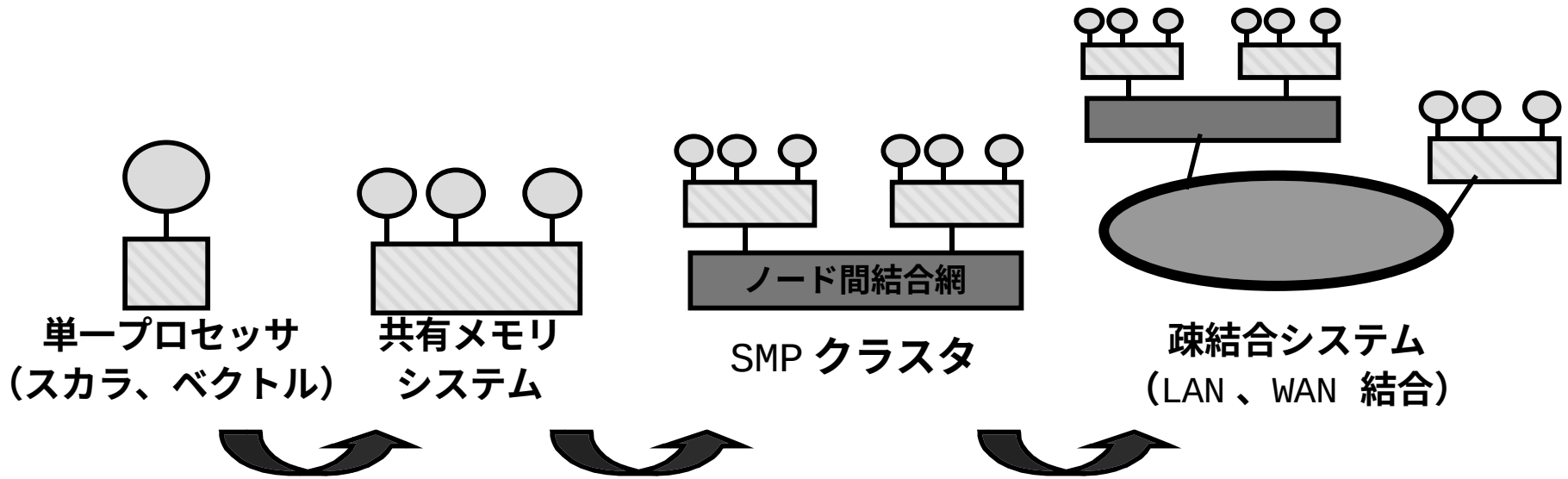
200年3月22日

NECソリューションズ
第一コンピュータソフトウェア事業部
妹尾義樹、村井均、末広謙二

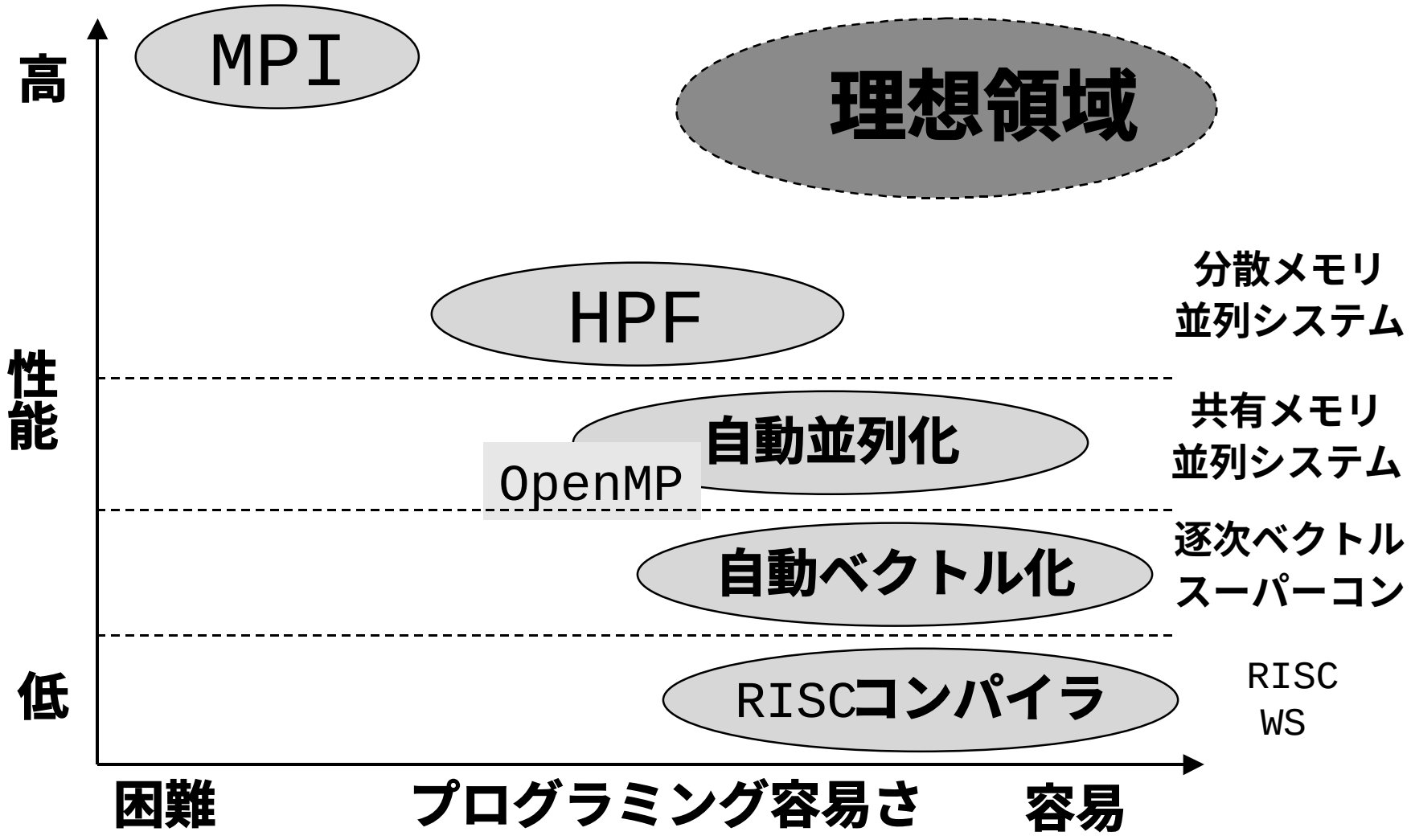
内容

- HPF言語の概要
- JAHPFの活動について
- 拡張言語仕様HPF/JA1.0
- NECのHPFに対する取り組みと現状
- 今後のプログラミングインタフェースについて
- まとめ

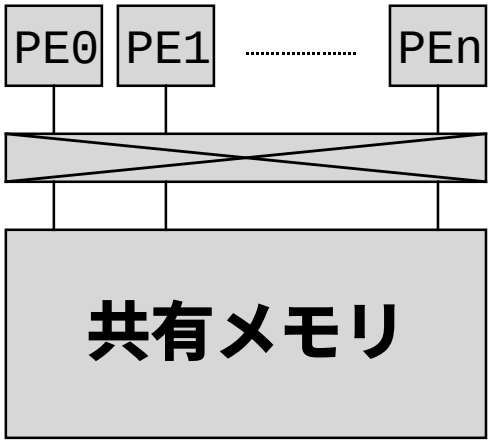
HPCシステムアーキテクチャの動向と並列化インタフェース



コンパイラ技術　VS　性能

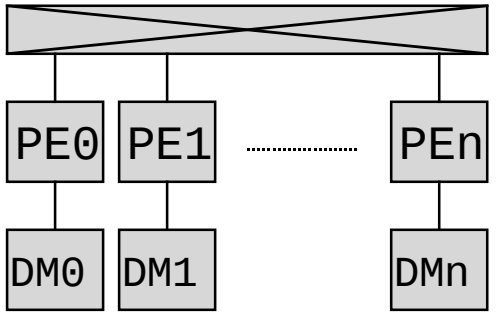


共有メモリと分散メモリ



共有メモリシステム

- 接続台数に限界
- メモリが遠い
- プログラミングが簡単



分散メモリシステム

- 高並列システムの構築が容易
- 各プロセッサの自メモリへのアクセスが高速
- プログラミングが大変

不要

データアクセス局所性の抽出

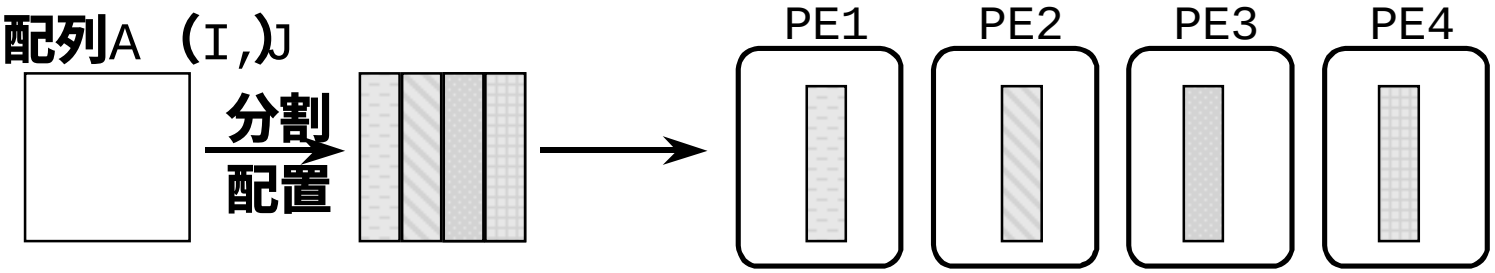
必要

High Performance Fortra

- 分散メモリ用データパラレル言語
- Supercomputing'91で最初に提案
- HPFフォーラム（Rice大学Ken Kennedy主導）で仕様決定作業
  - HPF1.0 Spring 1993
  - HPF1.1 Nov. 1994（バグ修正）
  - HPF2.0 Jan. 1997（公認拡張仕様として機能拡張）
- Basic Concept:
  - データの分散メモリへの配置を明示指定
  - シングルストリーム
  - 通信と計算のマッピング（並列化）はコンパイラが自動処理

HPF言語の基本的な考え方

- 配列の分散配置をユーザが指定



データ転送の生成
ループ並列化
} コンパイラが行う

- Owner Computes Rule
  - 更新データを保持するプロセッサが処理を行う

ＨＰＦプログラムの例（ＭＰＩとの比較）

Fortran(逐次)

```
program sample
parameter( n = 100 )
real*4 x(n)

do i = 1 , n
   x(i) = 1.e0 / float(i)
end do
dot = 0.e0

do i = 1 , n
   dot = dot + x(i) * x(i)
end do

write(6,*) 'dot = ', dot
stop
end
```

MPI並列プログラム

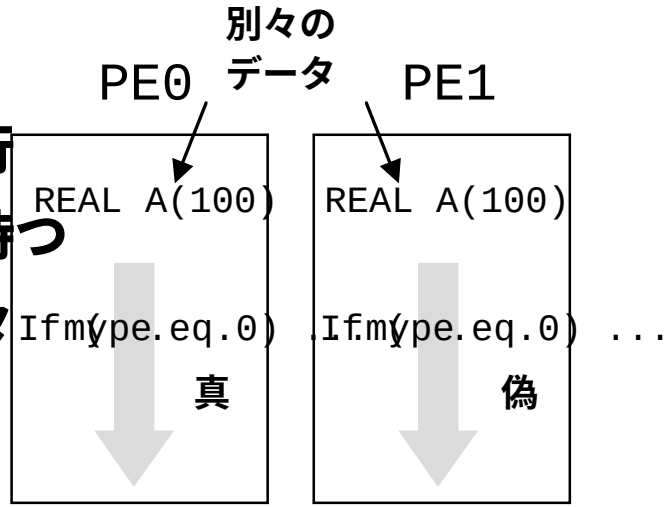
```
program sample
include 'mpif.h'
parameter(n = 100)
real*4 x(n)
call mpi_init(ierr)
call mpi_comm_size(mpi_comm_world,npe,ierr)
call mpi_comm_rank(mpi_comm_world,np,ierr)
if ( np .eq. 0 ) then
  do i = 1 , n
    x(i) = 1.e0 / float(i)
  end do
  do i=1,npe-1
    call mpi_send( x , n, mpi_real, i , 0
     &          mpi_comm_world, ierr )
  end do
else
  call mpi_recv( x , n, mpi_real, 0 , 0
     &          mpi_comm_world, istat, ierr )
end if
is = ( n / (npe-1) ) * np + 1
ie = min( n , ( n / (npe-1) ) * ( np + 1 ) )
dot = 0.e0
do i = is , ie
  dot = dot + x(i) * x(i)
end do
call mpi_allreduce( dot , mpi_dot, 1, mpi_real, mpi_sum
     &              mpi_comm_world, ierr )
if ( np .eq. 0 ) write(6,*) 'dot = ', dot
call mpi_finalize(ierr)
stop
end
```

HPFプログラム

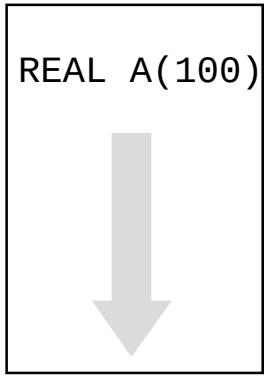
```
program sample
parameter( n = 100 )
real*4 x(n)
!HPF$ PROCESSORS P(4)
!HPF$ DISTRIBUTE X(BLOCK) ONTO P

do i = 1 , n
   x(i) = 1.e0 / float(i)
end do
dot = 0.e0

do i = 1 , n
   dot = dot + x(i) * x(i)
end do

write(6,*) 'dot = ', dot
stop
end
```

プログラミングインタフェースの比較

- SPMDプログラミング(MPIの例)
  - 一つのプログラムを全員にコピーして実行
  - プロセッサごとにそれぞれ制御の流れを持つ
  - データ空間はそれぞれのプロセッサで別々
  - 並列処理制御、通信はexplicit



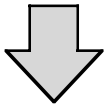
- HPFプログラミング
  - 通常の逐次プログラミングと同一
  - 制御の流れはただ一つ
  - データ空間は全プロセッサで共有
  - 並列処理制御、通信はimplicit



HPF言語の基本機能

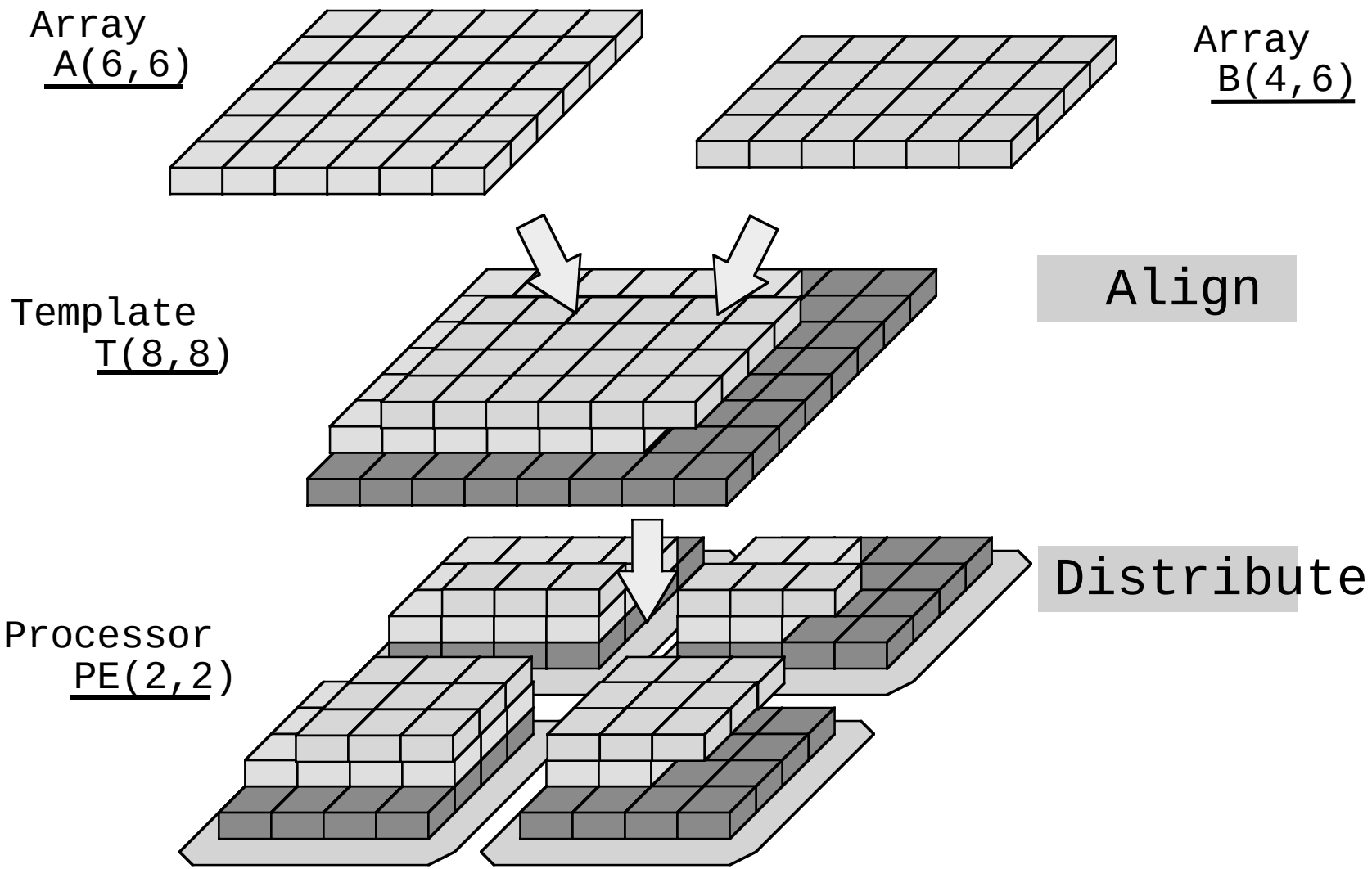
- プロセッサ配列の宣言：ＰＲＯＣＥＳＳＯＲＳ指示
- 配列の分散配置：ＤＩＳＴＲＩＢＵＴＥ指示
- ループ並列化可能性指示：ＩＮＤＥＰＥＮＤＥＮＴ指示
  - REDUCTION 演算の指定、テンポラリ変数指定（NEW）を 含む
- 利用可能プロセッサ数獲得関数

NUMBER_OF_PROCESSORS()



定型問題なら、大半のプログラムがこれだけで並列化可能

2-Level Data Mapping in HPF

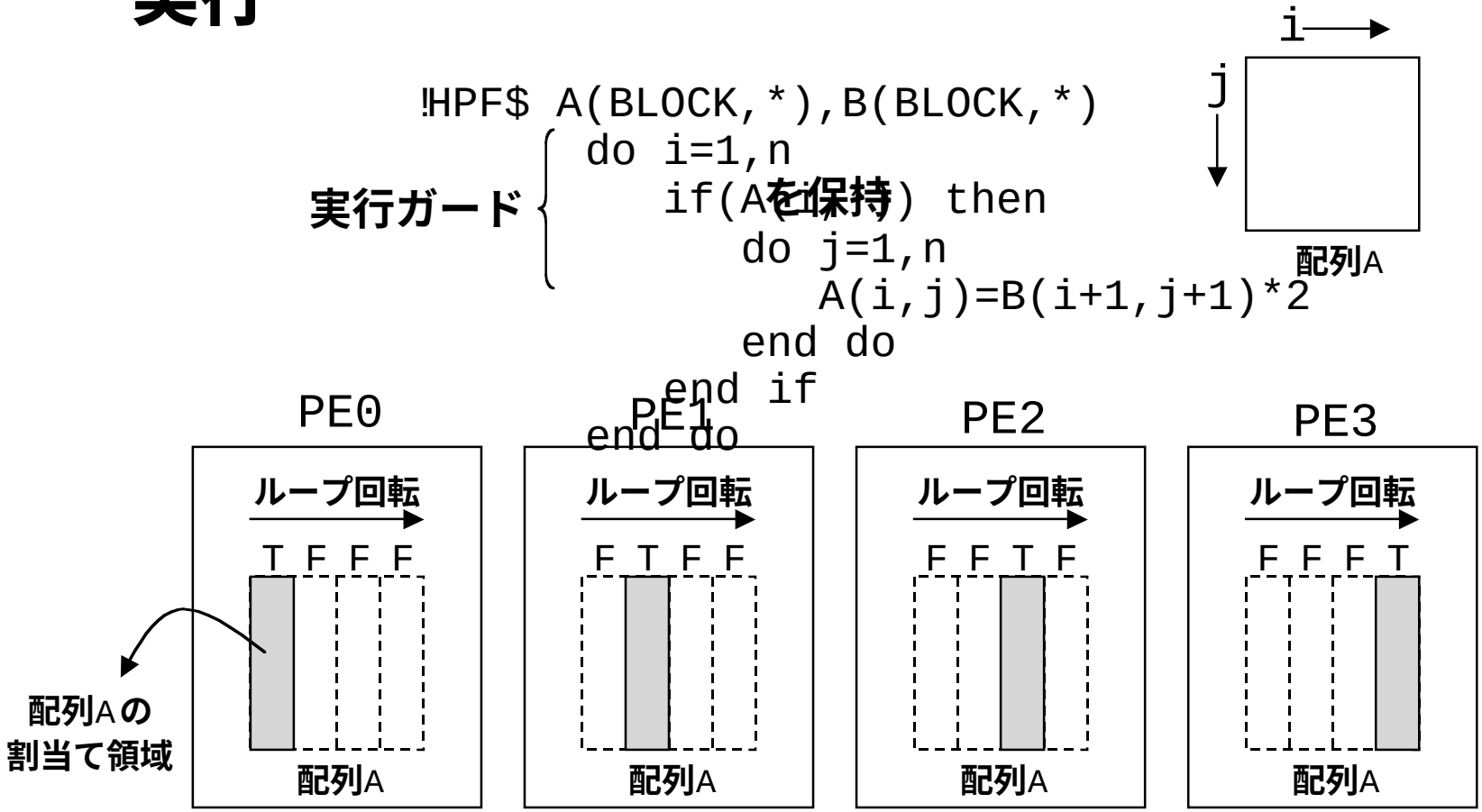


HPF2.0公認拡張機能

- GEN_BLOCK 分散：
  - （ブロック幅がプロセッサ毎に指定可能）
- REDISTRIBUTE：
  - 動的分散配置の変更
- ON 節：
  - 計算マッピングを特定の配列のマッピングに合わせる
- SHADOW ：
  - SHIFT転送のためにあらかじめ袖領域を確保
- TASK REGION ：
  - タスク並列（e.g. 連続する２つのループを別プロセッサで）

Owner Computes Rule

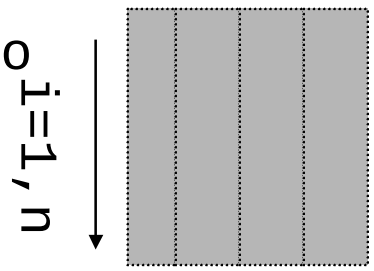
（更新）データを保持するプロセッサが計算を実行



計算マッピングの決定

データ転送コストが最小になるように計算空間をプロセッサに割り当て

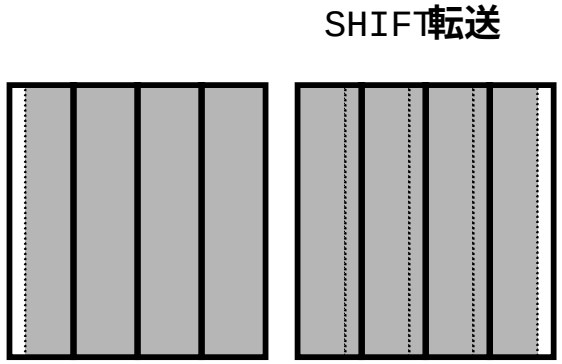
```
REAL*8 A(n,n),b(n,n),c(n,n)
!HPF$ DISTRIBUTE (*,block) :: A,C
!HPF$ DISTRIBUTE (block,*) :: B
          do j=2,n
                  do i=1,n
            A(i,j) = B(i,j) + C(i,j-1)
               enddo
        end do
               enddo
```

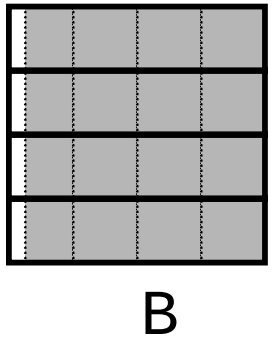


Aに合わせて分割

計算空間

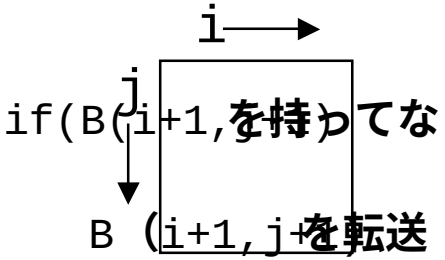
(iteration space)





HPFコンパイラによる通信生成

```
!HPF$ A(BLOCK,*),B(BLOCK,*)
      do i=1,n
        if(A(iを保持) then
          do j=1,n
```

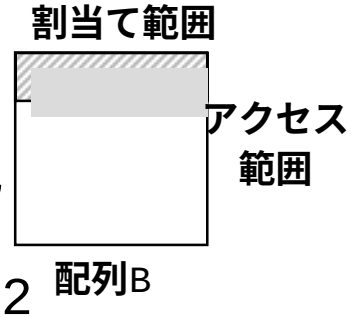


1要素ずつの通信

```
            A(i,j)=B(i+1,j+1)*2
        end do
        end if
      end do
```

Bに対する一括通信生成

```
do i =iis,← 実行ガードの効率化
        do j=1,n
           A(i,j)=B(i+1,j+1)*2
        end do
end do
```



HPF言語の問題点

- コンパイラが未成熟
  - この数年で格段に進歩
  - まだ十分とは言えない
- 適用範囲に制限
  - 非定型処理の扱い、非同期処理の扱い
- プログラマの最適化についての細部の制御が不能
  - 処理系の実装に依存する部分が多い
  - 処理系の実装を知らないと使えない
- プログラミング技法が確立されていない
  - 例　分散配列を手続き間で共有する方法
    手続き呼び出しを含むループの並列化

' 1999JAHPF

JAHPFの活動について

- HPF言語の普及促進活動
  - HPCユーザへの宣伝、機能説明など
  - 日本語マニュアルの整備

コンパイラ開発者とユーザの密な連携

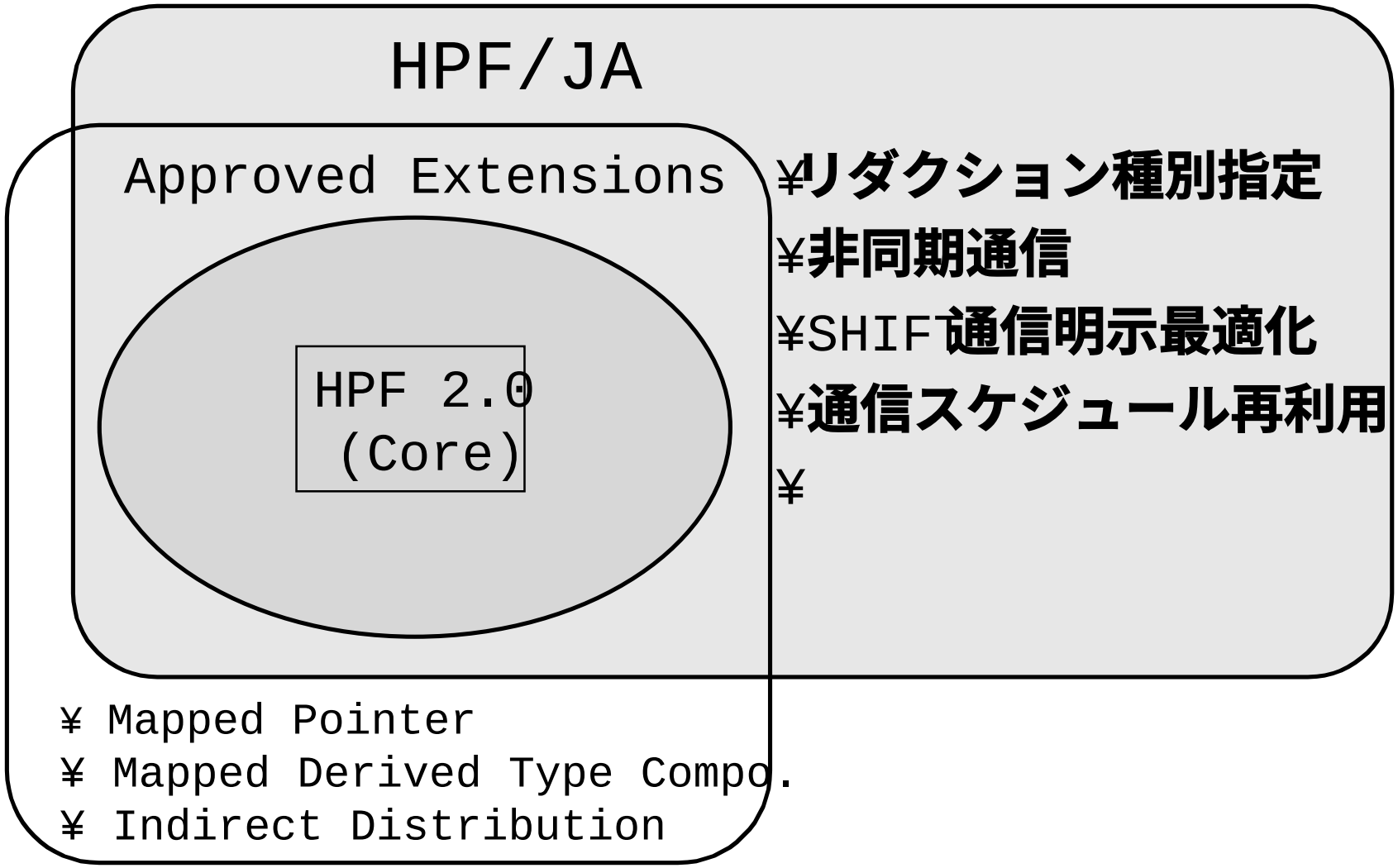
- 独自拡張仕様HPF/JAの策定
  - ベンダによる処理系開発、ユーザによる評価

1999年6月、Springer-Verlagより
日本語訳＋JA拡張仕様を出版

- HPF プログラミング経験の蓄積
  - 実コードの並列化作業（15本程度）

' 1999JAHPF

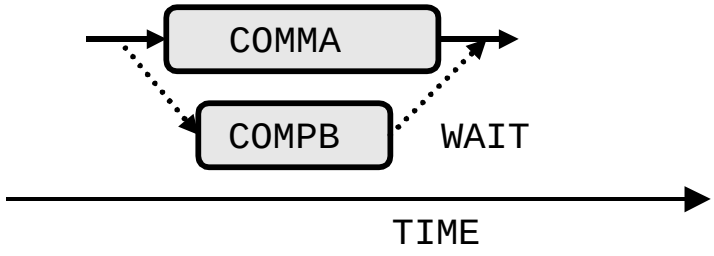
HPF/JA 1.0言語仕様



非同期通信指定

- 計算とデータ転送のオーバーラップ
  - 代入文で示されたデータ転送 (Array assignments, WHERE constructs, FORALL constructs)と計算がオーバラップ可能
  - 非同期通信 +WAIT
- 例

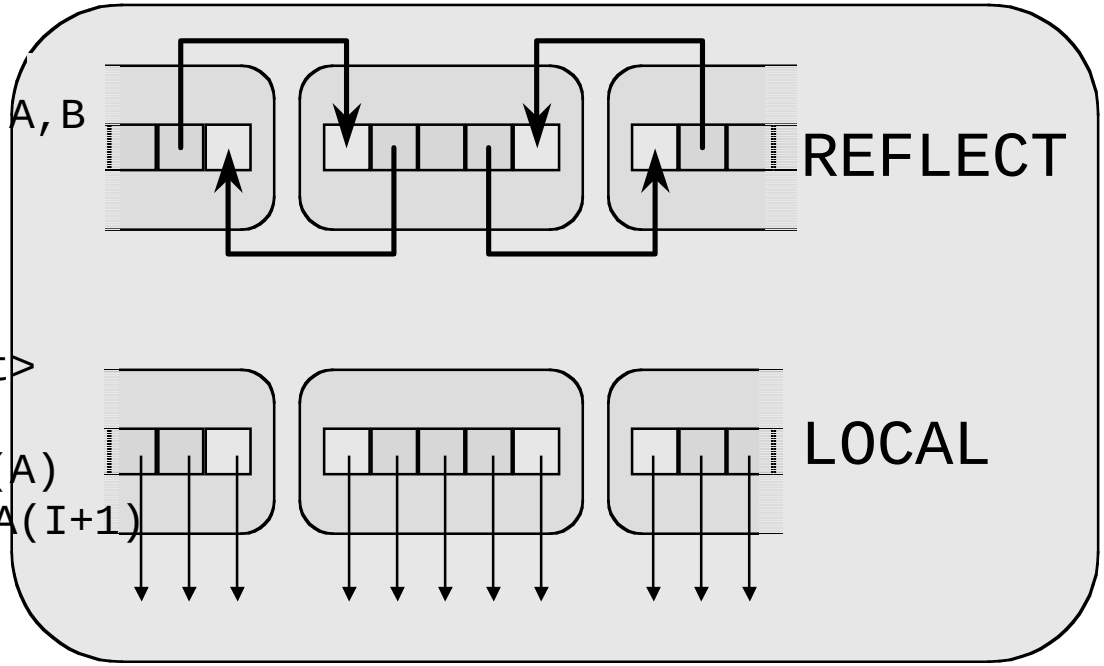
```
!HPFJ  ASYNCHRONOUS (ID=X) BEGIN
       S(1:N) = T(1:N)
!HPFJ  END ASYNCHRONOUS
    Computation independent of the above communication
!HPFJ  ASYNCWAIT (ID=X)
```



SHIFT型通信の明示的最適化

- REFLECT: Set values into the SHADOW area from its original.
- LOCAL: Assert no communication required
  - RESIDENT : Communication among active processors may

```
 REAL A(N),B(N)
!HPF$ DISTRIBUTE (BLOCK)::A,B
!HPF$ SHADOW(1)::A
       DO I=1,N
         A(I) = ...
       ENDDO
!HPF$ REFLECT <object-list>
       DO I=2,N-1
!HPF$ ON HOME(B(I)),LOCAL(A)
         B(I)=A(I-1)+A(I)+A(I+1)
       ENDDO
```



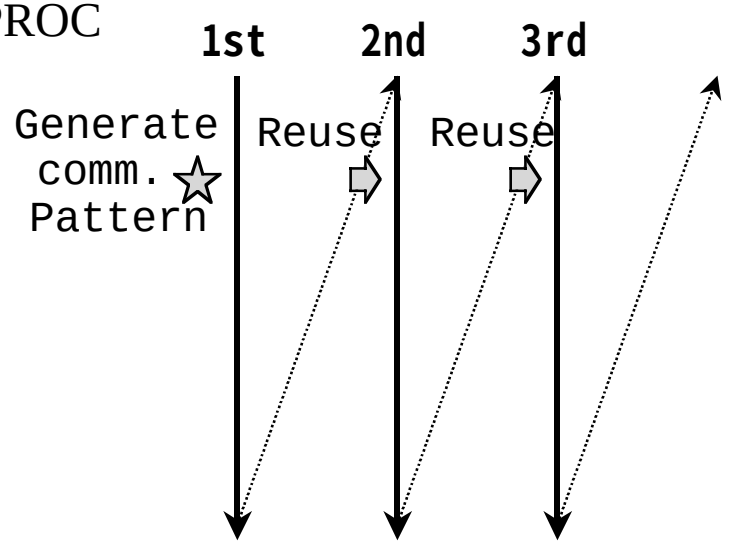
通信スケジュール再利用

- 目的
  - ¥ **非定型通信において、通信パターンを再利用することで高速通信を実現（非線形添え字からの通信パターン計算を再利用）**
  - ¥ **スイスCSCSやVienn太学で有用性実証**
- Example

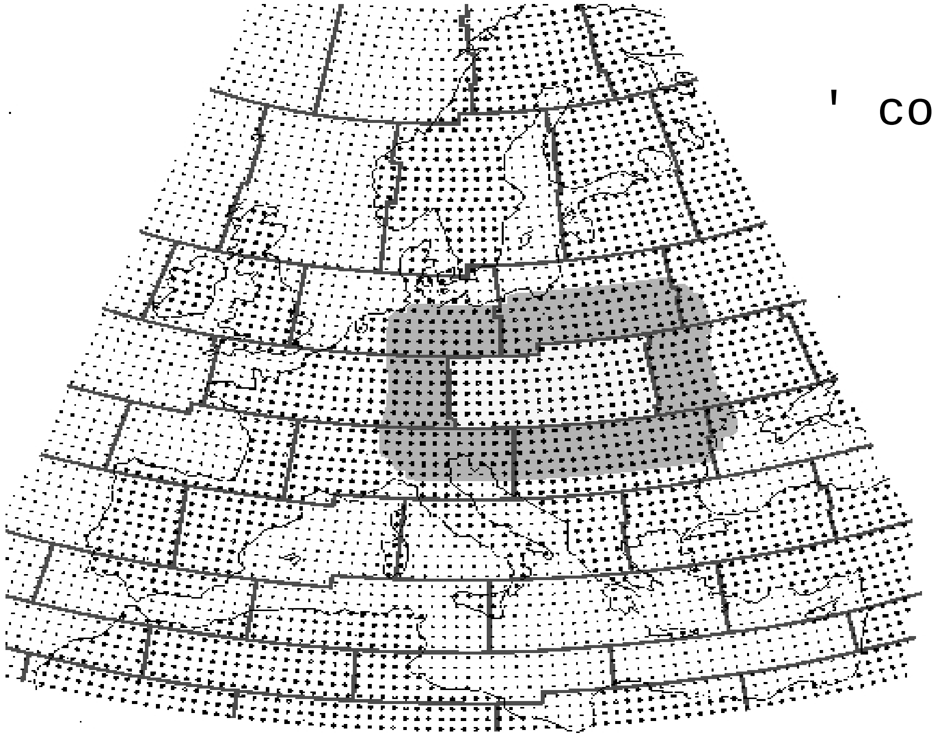
```
!HPF$ DISTRIBUTE A(BLOCK) ONTO PROC
   DO ...
    ...
!HPF$ INDEX_REUSE (.true.) A
     DO I=1,N
       ... = A(IDX(I))
     ENDDO
    ...
   ENDDO
```

Outer Loop

Indirect Access



非定型処理の例



' courtesy of ECMWF

計算量が同等になるように、また転送量が少なく済む
ように計算領域を分割。

非定型の隣接通信の高速化が鍵

' 1999JAHPF

ベンチマーク活動

昨年までに並列化されたコード

(1) Molecular Code
- Car-Parrinello Method

(2) Particle in Cell
- Electro-Static Plasma Simulation

(3) IMPACT3D
- 3-Dimensional CFD code with TVD Scheme

(4) MICCG
- 3-Dimensional Poisson Solver
- Incomplete Cholesky Factorization

(5) CIP(Cubic Interpolated Pseudo-particle Code)
- 2-Dimensional CFD with upwind scheme

(6) NJR(SAGCM)
- Atmospheric Global Circulation Model using Spectral metho

' 1999JAHPF

Benchmarking Effort (Con

_ 今年並列化されたコード

(1) FEM kernel (Yokohama National Univ., RIST)
  – Sparse matrix solver
(2) NAL-LES (National Aerospace Lab.)
  – CFD with Large-Eddy simulation method
(3) Shallow water equation model (Univ. of Tokyo)
  – FDM
(4) Plasma MHD Code (National Inst. for Fusion Science)
  – Particle code with open boundary condition
(5) ヘリコプタ周り流体計算（MHI）
  – Cartesian CFD
(6) Astrophysical Rotating Plasma Simulator(千葉大)
(7) CFDコード～3次元FFT～(ESRDC)

_ 上記は、HUG2000にて発表

  – 京大、原研、NEC 並列処理センターなどのマシン利用環境の提供
  – ベンダとの密な協調作業

' 1999JAHPF

JAHPF今後の計画

- 活動5年を契機に組織形態をリニューアル
  - 「HPF推進協議会」新規発足へ
  - 技術論と同時に普及促進活動により注力
  - オープン化：一般参加が可能な組織に転換
  - ユーザに評価環境を積極的に提供
- ベンチマーク活動の活性化→ benchmark suiteの開発
- HPF/JA 2.0拡張仕様の検討 → 標準仕様へ
- 標準プログラミング手引書の作成
  - 典型的並列化手法を網羅
  - 処理系ごとの実装依存部分をできるだけ見えない形に
  - 共通ツールインタフェース（どこまでできるか）

HUG2000 (HPF User Group Meeting

- 第一回 (Santa Fe, New Mexico, USA, 1997)、第二回 (Porto, Portugal, 1998)、第三回 (Redondo Beach, California, USA, 1999)につづく第四回のHPFに関する国際会議
- 日時： 2000年10月19-20日
- 場所：ホテルインターコンチネンタル東京ベイ
- Keynote：Prof. Ken Kennedy (Rice大学)
- 招待講演：Barbara Chapman (Houston大学)
- パネル：HPF, MPI, OpenMPのインプリメンター、ユーザの第一人者が集まり、将来の並列プログラミングインタフェースについて討論

http://tokyo.rist.or.jp/jahpf/hug2000/index.html

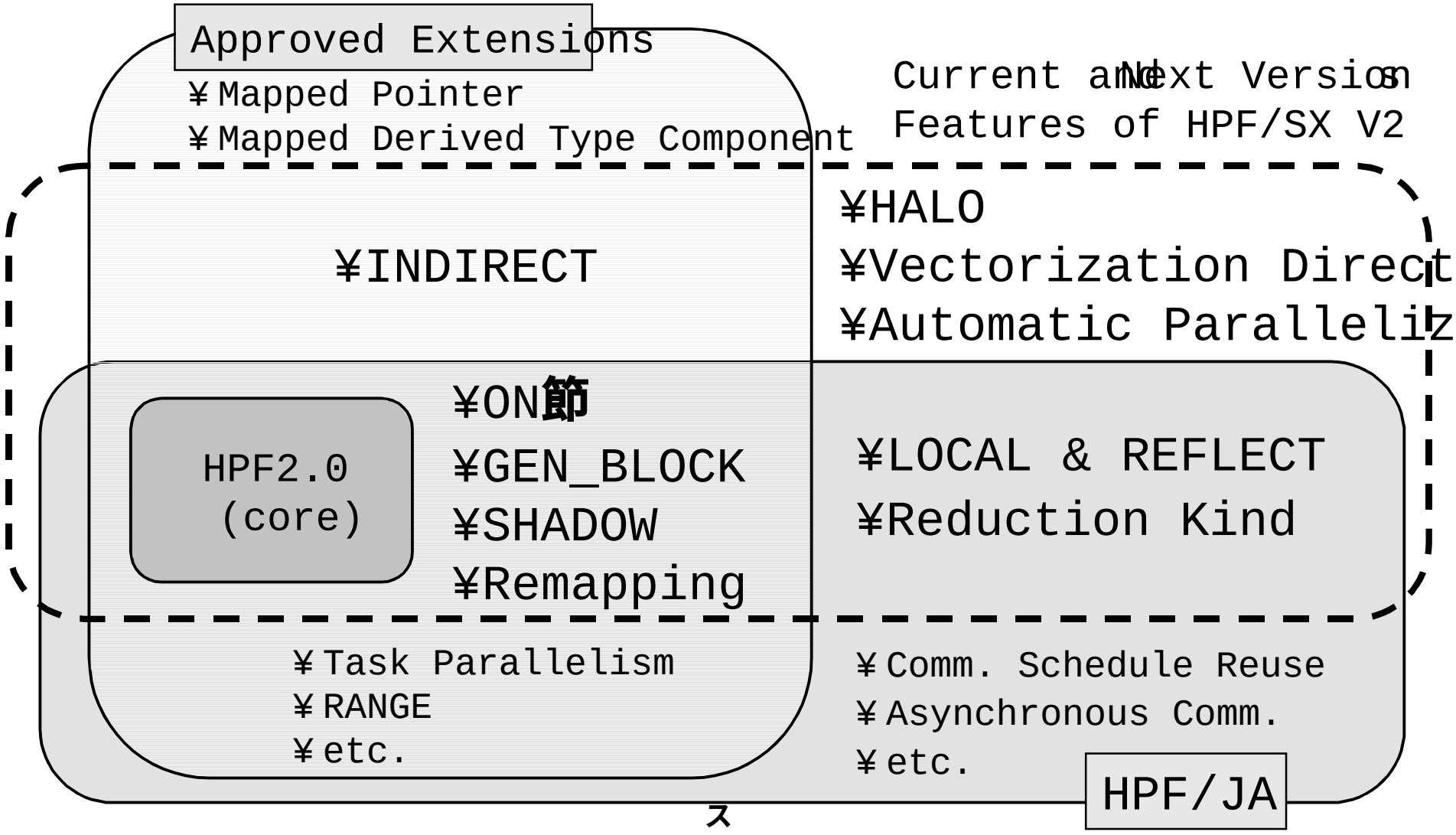
HPFの現状（HUG2000から）

- 欧米：
  - 一部の研究者で使われるのみ
  - OpenMP（共有メモリ並列化I/F）のC-NUMA マシン用拡張としてHPF機能を取り込む動き
- 日本：
  - まだまだHPFに対する期待は大きい
    - ￥VPPユーザ、潜在的地球ユーザにHPFへの期待大
  - HPFを用いた並列化成功例の蓄積が少しずつではあるが進行
    - ￥名大荻野先生：プラズマMHD コードで

HPF/SX V2 の開発状況

- 2006年3月より製品化
- 2006年6月に機能/性能強化
  - アドレス変換高速化
  - 重要ベクトル化指示行認識
  - 自動並列化機能拡充
  - 通信最適化強化
  - 配列の動的再配置性能強化
- Almost ready to use for APS regular

Current Status of HPF/SX

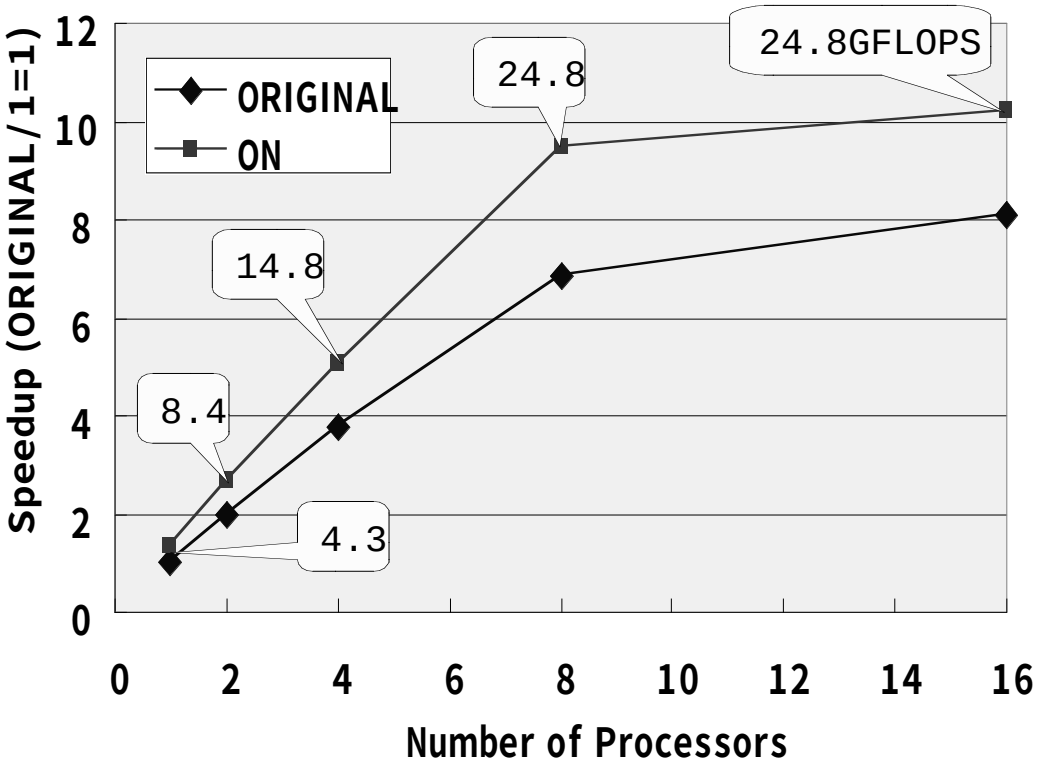


ON節の指定による効果

Target : tomcatv (4097x4097) of APR benchmarks

```
!HPF$ independent
      DO 250 i = i1p , i2m
!HPF$ on home( aa(:,i) ) begin
        ip = i + 1
        im = i - 1
        m = 0
        DO 310 j = j1p , j2m
          ...
          xx = x(j,i+1) - x(j,i-1)
          ...
          aa(m,i) = -b
          dd(m,i) = b + b + a * rel
          pxx = x(j,i+1) - 2. * x(j,i) + x(j,i-1)
          qxx = y(j,i+1) - 2. * y(j,i) + y(j,i-1)
          ...
          rx(m,i) = a * pxx + b * pyy - c * pxy + xx * qi + xy * qj
          ry(m,i) = a * qxx + b * qyy - c * qxy + yx * qi + yy * qj
  310   CONTINUE
!HPF$ end on
  250 CONTINUE
```

ON 節の指定による効果（2）



The ON directives im the performance at a 30%.

Evaluation environment

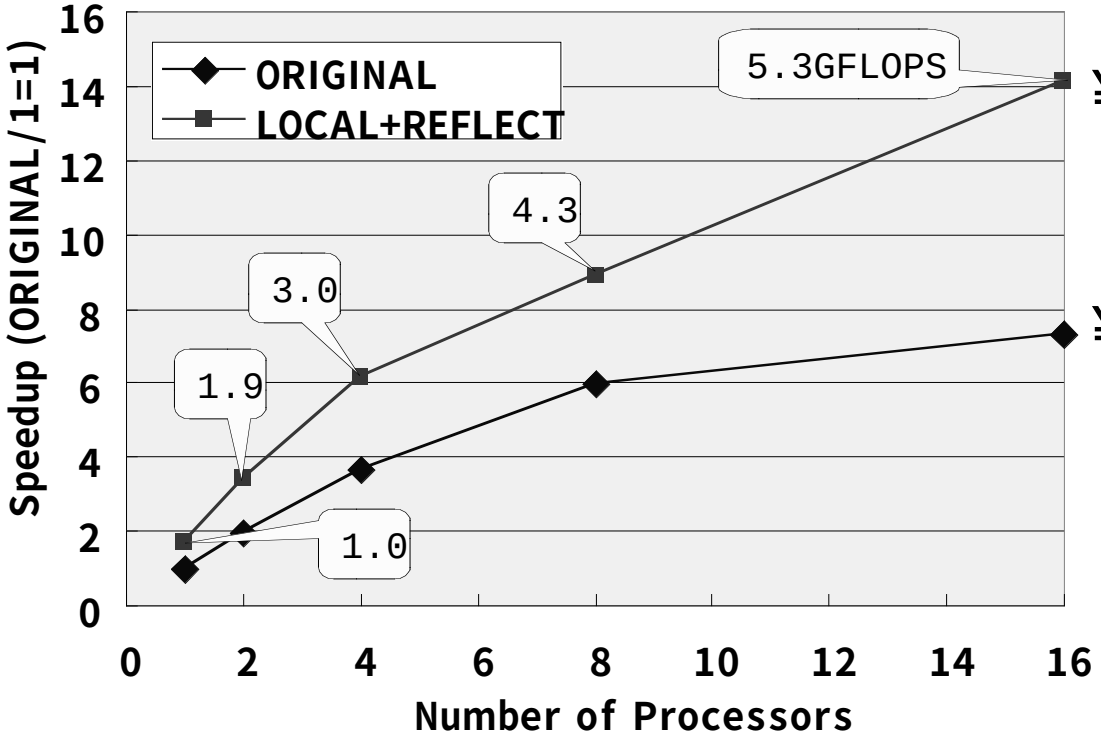
- **NEC SX-5**
- **HPF/SX V2 Rev.1.1.1**
- **default compiler options**

LOCAL/REFLECT指定の効果

Target: a benchmark program of the Red-Black S
iterating 100 times four five po
computations for a 4097x4097 arra

```
            DO K=1, 100

!HPFJ reflect a

!HPF$ independent
         DO J=1, (N-1)/2
!HPF$ on home( a(:,2*J+1) ), local
            DO I=1, (N-1)/2
               A(2*I+1, 2*J+1) = (W/4)*(A(2*I, 2*J+1)+A(2*I+2, 2*J+1) +
     1         A(2*I+1, 2*J) +A(2*I+1, 2*J+2)) + A(2*I+1, 2*J+1)*(1-W)
            END DO
         ENDDO
         ...

      END DO
```

LOCAL/REFLECT指定の効果（2）



- LOCAL and REFLECT reduce the number comms. by half.
- **LOCAL+REFLECT** is 1.5-2 times faster **ORIGNAL**.

Vectoriza指示行認識

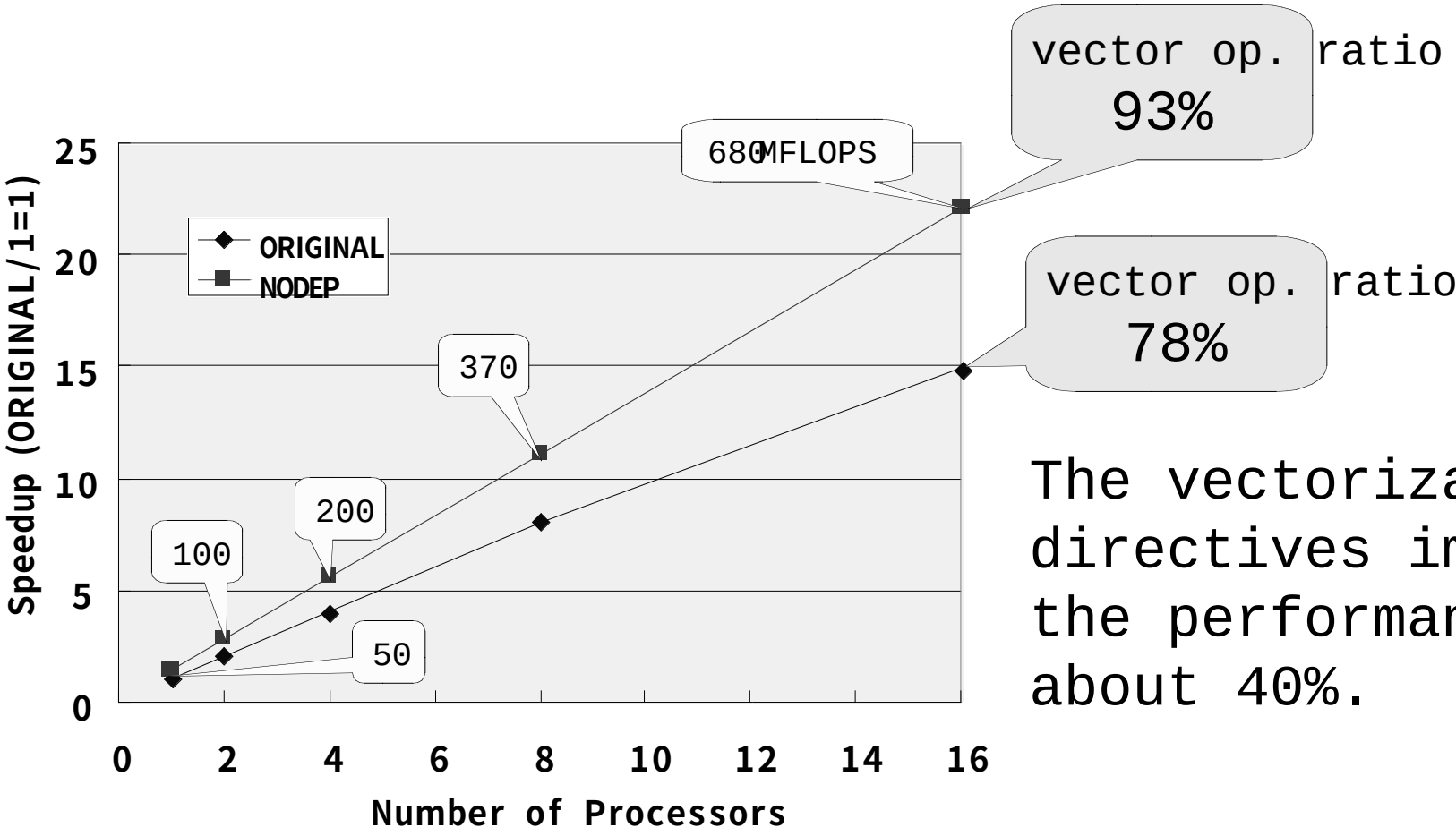
- HPF/SX V2は下記の指示行を認識し、変形後の対応するループに指示を挿入
  - shortloop:
    asserts that loop length does not exceed the vector r
  - vector:
    instructs the loop is to be vectorized.
  - novector:
    instructs the loop is not to be vectorized.
  - select(vector|concur):
    instructs whether the loop is to be vectorized or SMP
  - nodep:
    asserts no data dependence.

ベクトル化指示行挿入の効果

Target: mo of HPFBench

```
      ...
!hpf$ independent
      do iv=1,nvus
!CDIR NODEP
         do j=1,lvec
            trout(ksamp(j,iv),iv) = trout(ksamp(j,iv),iv)
     &  + wt(j + jbeg,iv)*(trin(isamp(j,iv),iv)
     &    + del(j,iv)*(trin(isamp(j,iv) + 1,iv) - trin(isamp(j,iv),iv)))
         enddo
      enddo
      ...
```

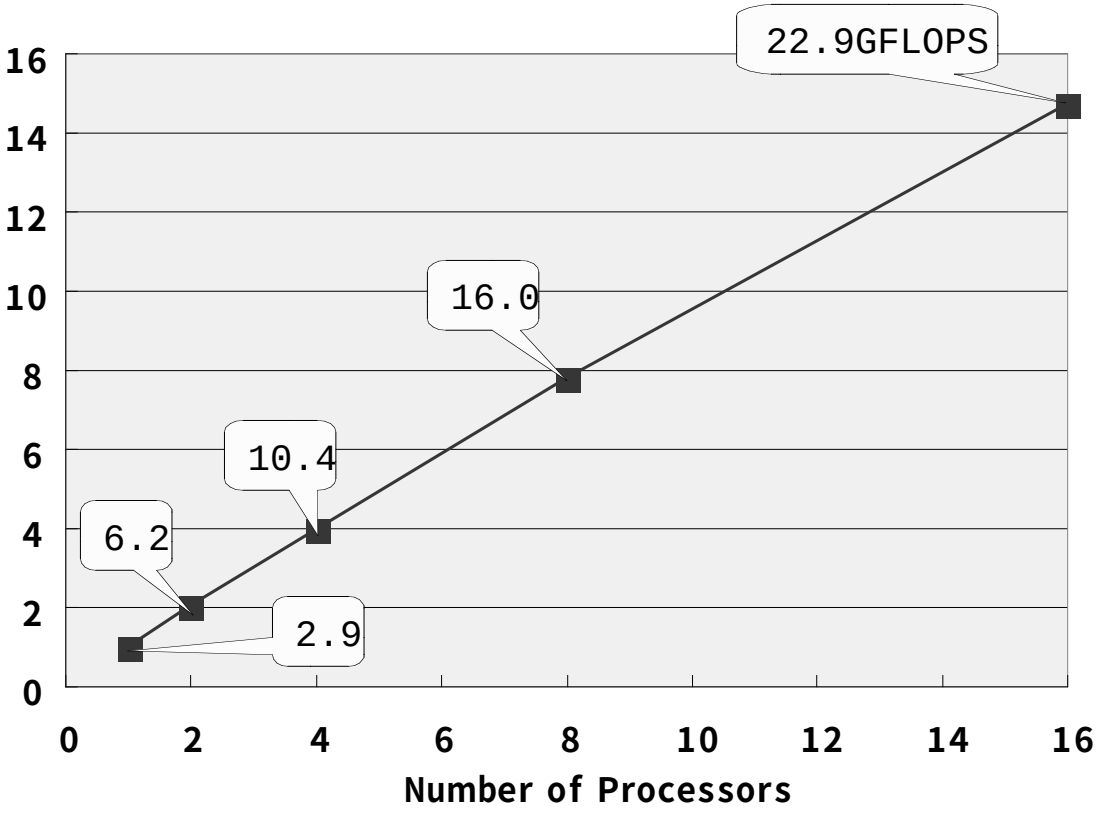
Evaluation of vectorization directive



The vectorization directives improve the performance at about 40%.

自動並列化の効果

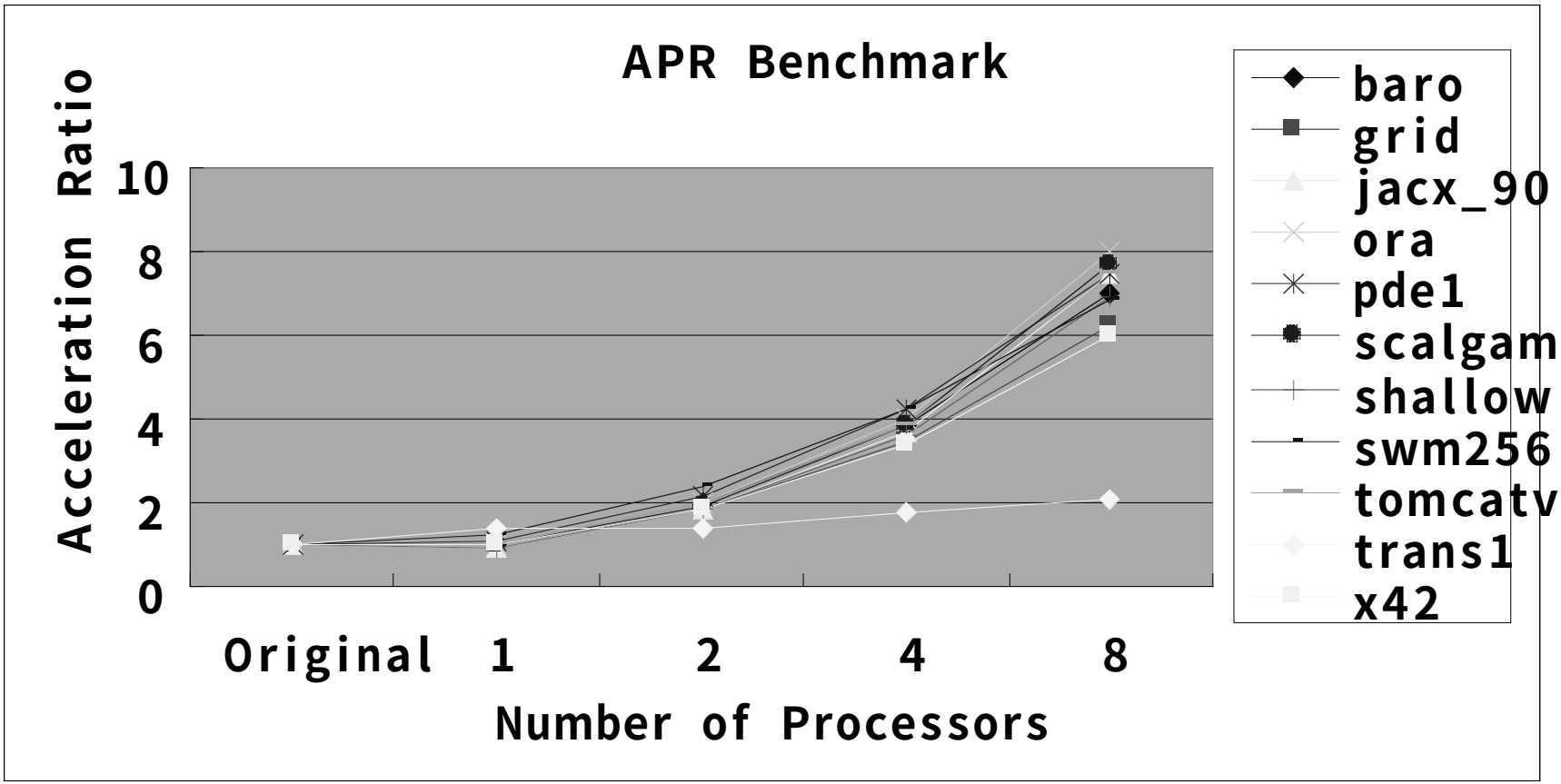
Target: shallow (4097x4097) benchmarks of 4PR



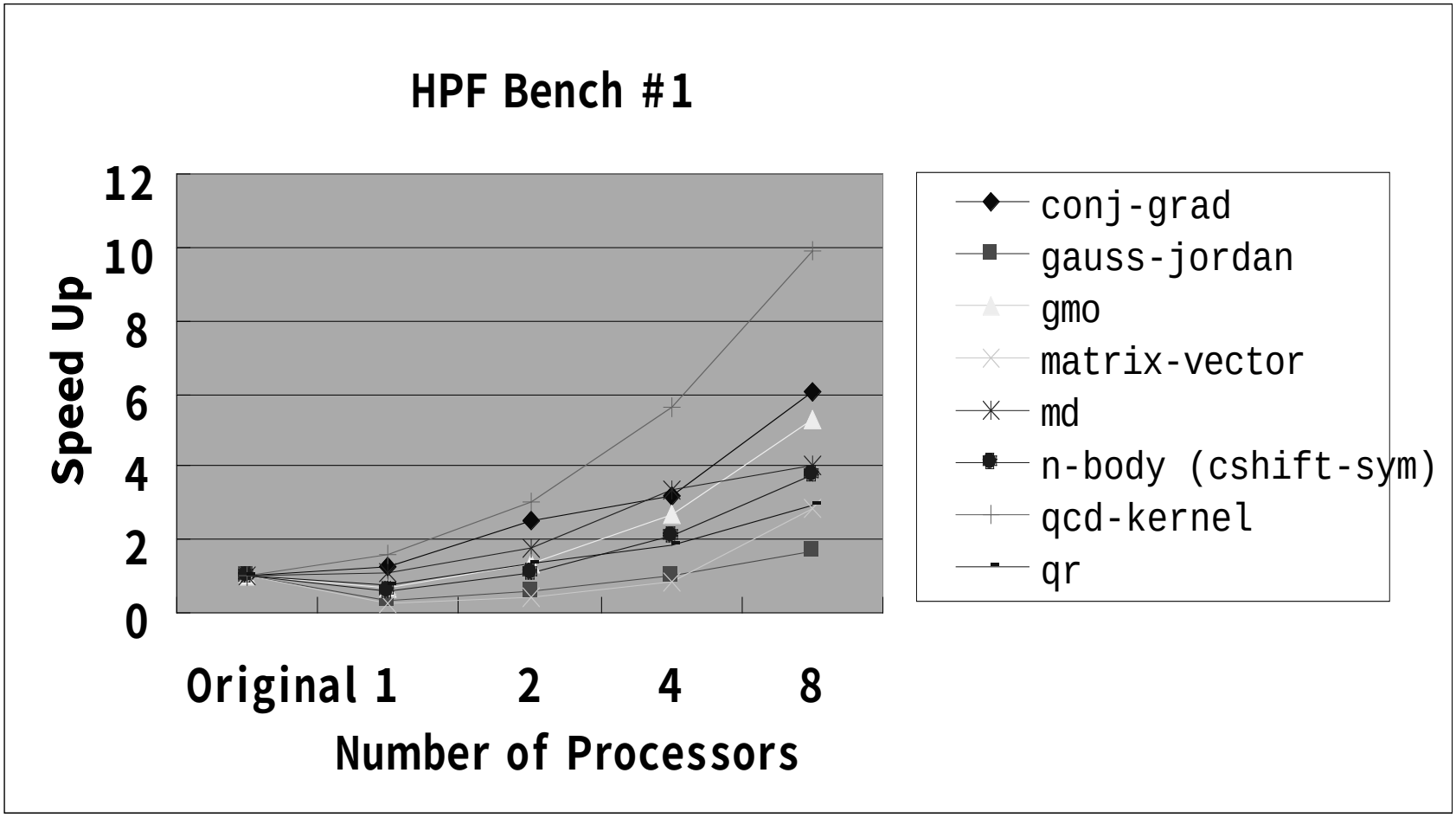
With only mapping directives and no others, good scala is achieved.

Lines: 658
Directive Lines: 16 (

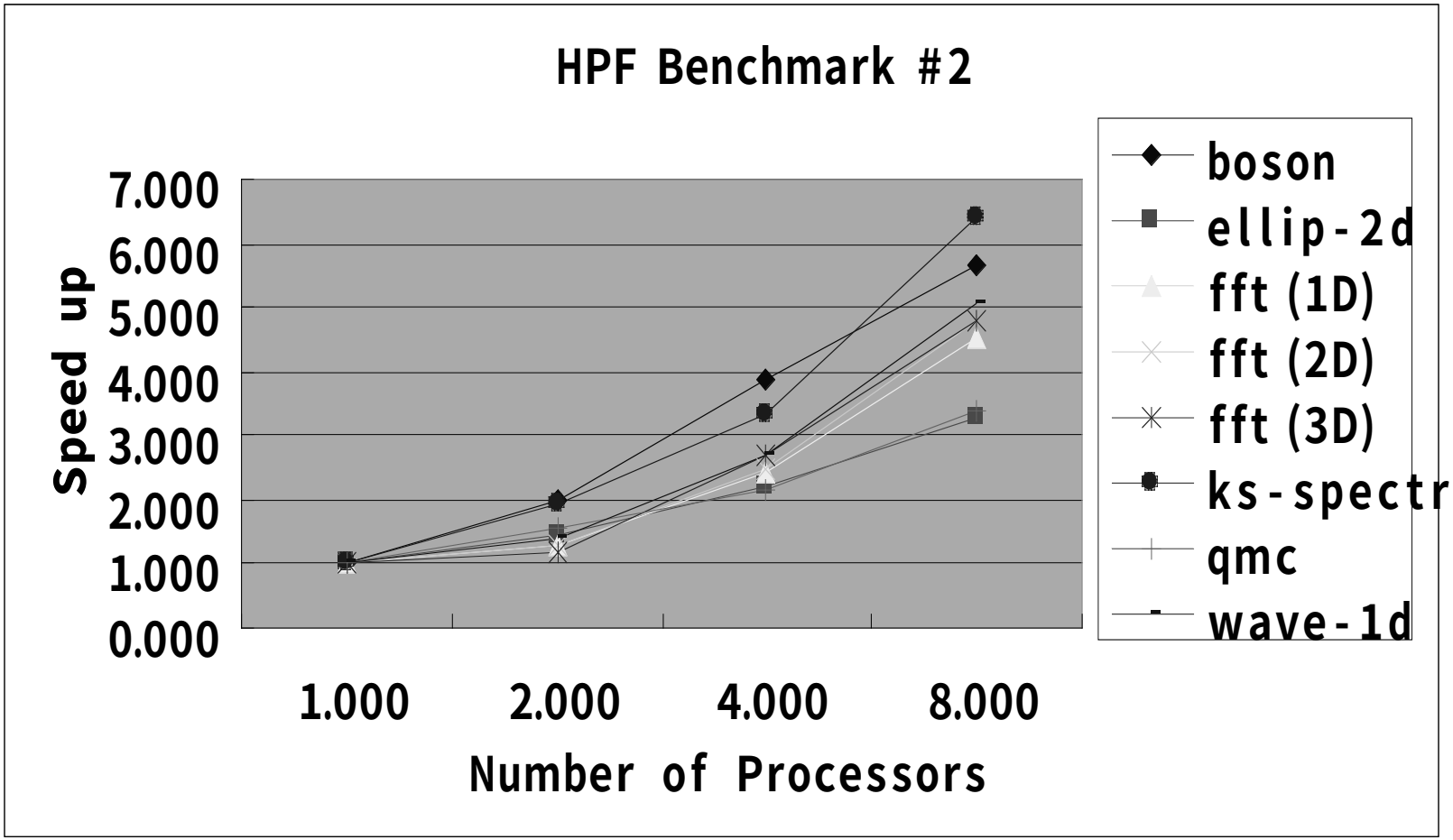
APR Benchmark Results



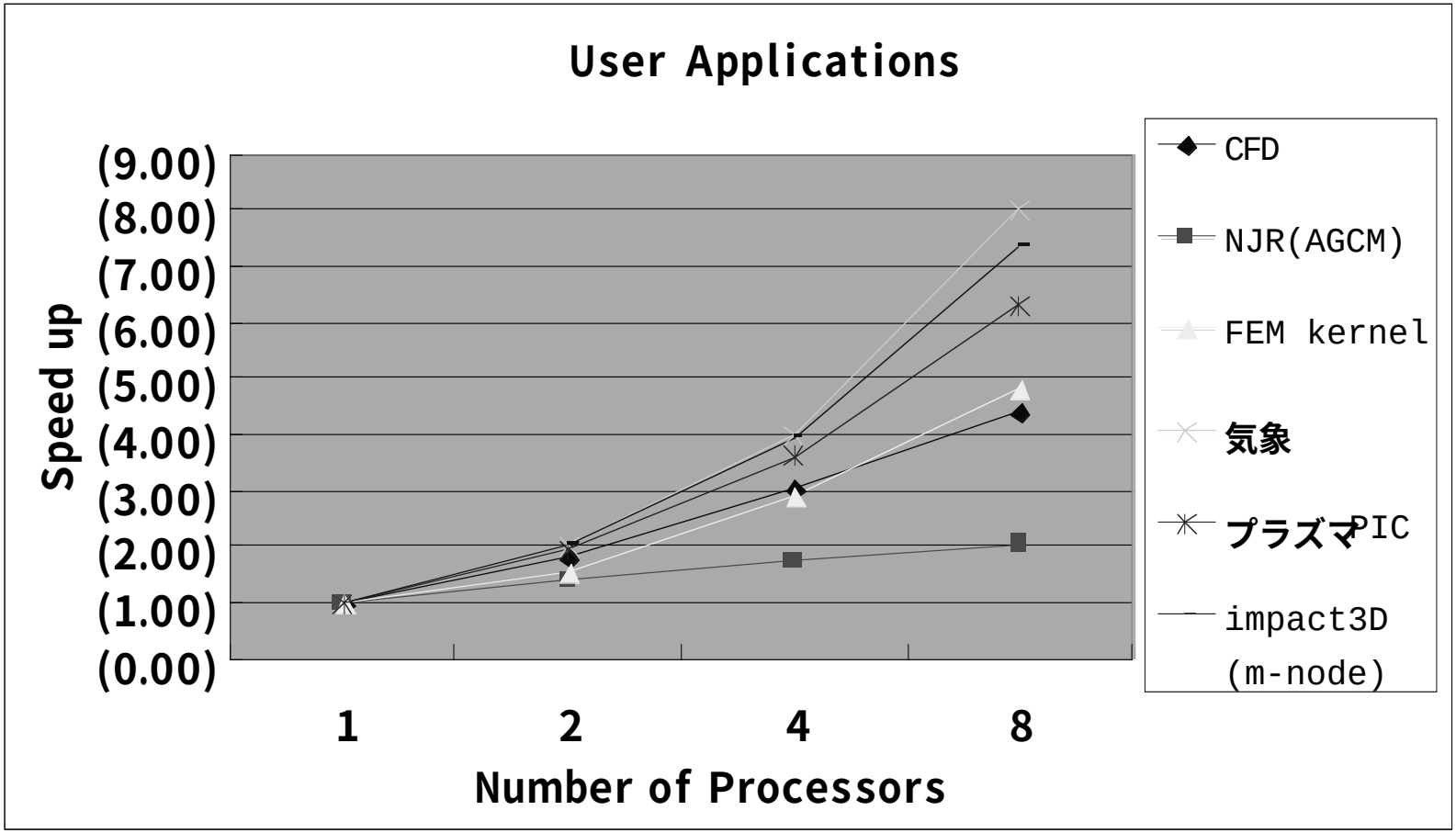
HPF Benchmark Result #1



HPF Benchmark Result #2



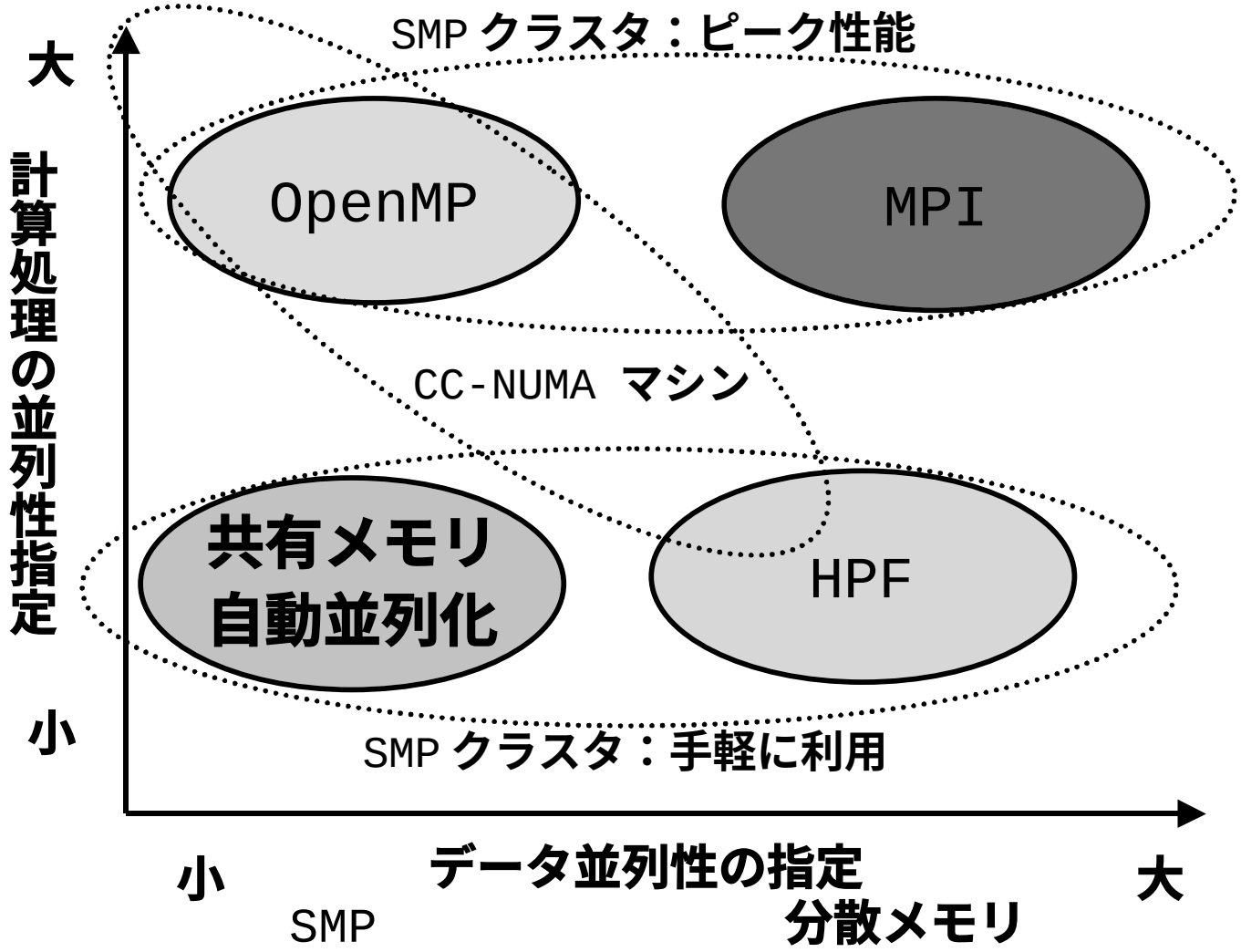
User Applications



HPF/SX V2強化予定

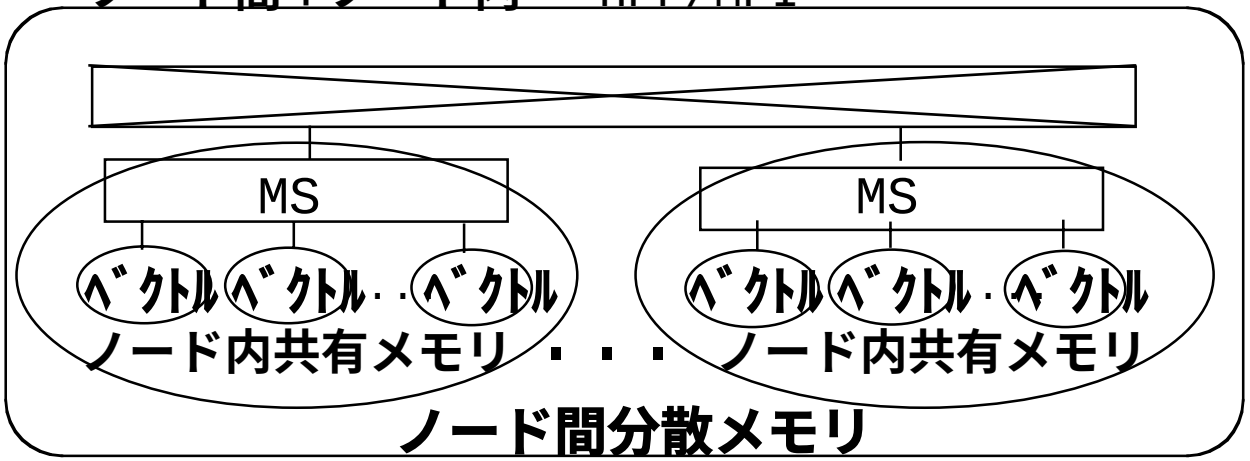
- 現在の機能拡張（今年度末実現予定）
  - 非定型通信の高速化
    - ¥ Indirect data distribution
    - ¥ Communication schedule reuse
    - ¥ HALO (Irregular shadow)
  - 共有メモリ自動並列化指示行の認識
- 来年度の機能拡張予定
  - 並列 I/O
  - タスク並列化（e.g.大気モデルと海洋モデルの結合）
  - 非同期通信
  - 自動負荷分散機能（ブロックサイズ自動調整）

今後の並列化インタフェース



SMPクラスタの利用

- プログラミングモデル
  - ノード間→HPF/MPI　ノード内→ 共有メモリ自動並列化
  - ノード間→HPF/MPI　ノード内→（共有メモリOpenMP）
  - ノード間＋ノード内　→HPF/MPI



階層モデルの成功には、
ノード内自動並列化
の有効性が鍵

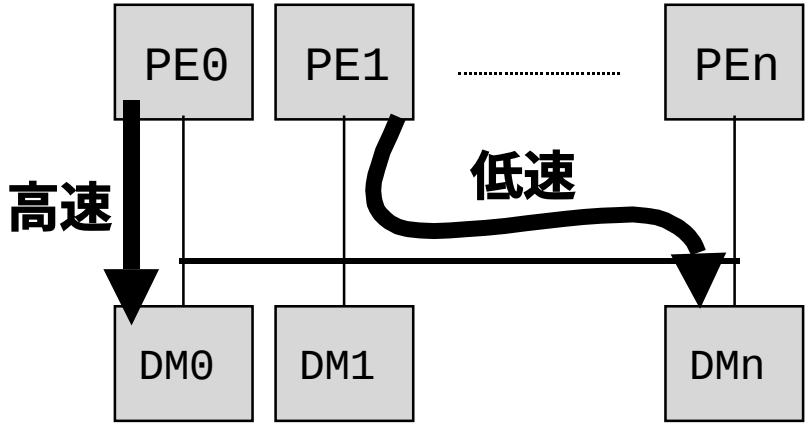
- 階層モデル：
  - 2段階の並列化制御オーバヘッド　×
  - ノード内ローカリティー抽出可能　○
- 単一モデル
  - 1段階並列化制御オーバヘッド、並列化楽　○
  - 多重ループの並列化、ローカリティー抽出　×

DSM用 OpenMPによる並列化

- Houston大学Barbara ChapmanのHUG2000発表
  - http://www.tokyo.rist.or.jp/jahpf/hug2000/presen/Chapman.pdf
- 分散共有メモリ（DSM）用の 並列化には、主要配列のデータへのマッピング指定が重要
  - SGI/Origin、Compaq並列システム
  - First Touchによるデータマッピング指定
  - 明示的データマッピング指定（HPFライク）
- SPMDスタイルのコーディングをしないと性能が出ないケースがある

CC - NUMA マシン向け並列化I/F

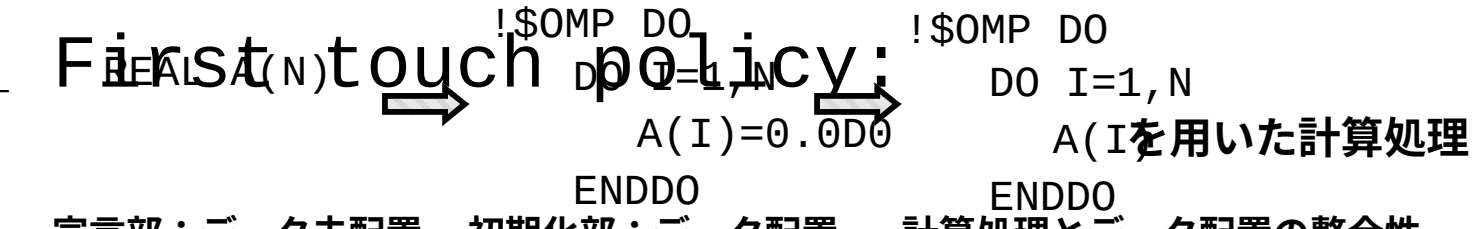
- CC-NUMA(Cache-Coherent Non-Uniform Memory Access)マシン



- **データアクセスローカリティーを抽出しないと性能が劣化**
- **ただし、データのマッピングはページ単位**
- **1次元目でのブロック分割はメモリの連続性を放棄しないと無理**

OpenMP on CC-NUMAs

- OpenMPには、CC - NUMA 対応機能はない。
- ベンダ（特にSGI,Compaq）はデータ配置をユーザがコントロールするための独自拡張を個別にサポート
  - first touch allocation policy
  - automatic page migration
  - page-based mappings
  - HPF-style element mappings
  - association of work with location of data
- First touch policy:



宣言部：データ未配置　初期化部：データ配置　計算処理とデータ配置の整合性

CC-NUMA マシン用I/Fの課題

- 現状では、Jacobのようなシンプルなコードでも、SPMD コーディングスタイルへの書き換えがないと、スケーラブルな性能向上がむずかしい。
- HPF技術とOpenMP技術の融合による、データアクセスローカリティーのコンパイラによる自動抽出が必要
- 先日のSC2000にて、HPFFメンバが集まり、OpenMPにHPF技術（部分仕様）を提案することに

まとめ

まとめ

- HPFを日本発の並列化言語として発展・普及させたい
  - このためのインフラは着々と整いつつある！
- 鍵は、実用コードの並列化でどこまで成果を蓄積できるか
- HPFを用いた並列プログラミングノウハウの蓄積も重要

今後の課題

- HPFの適用範囲の拡大
  - 非定型処理での実用性の実証
  - メモリへのデータ配置の連続性を仮定したプログラムの扱い
- OpenMPへのHPFデータマッピングの取り込み
  - CC-NUMAマシンへのデータ配置制御
- HPF普及促進活動
  - HPF推進委員会(仮称)