XcalableMP: a directive-based language extension for scalable and performance-aware parallel programming

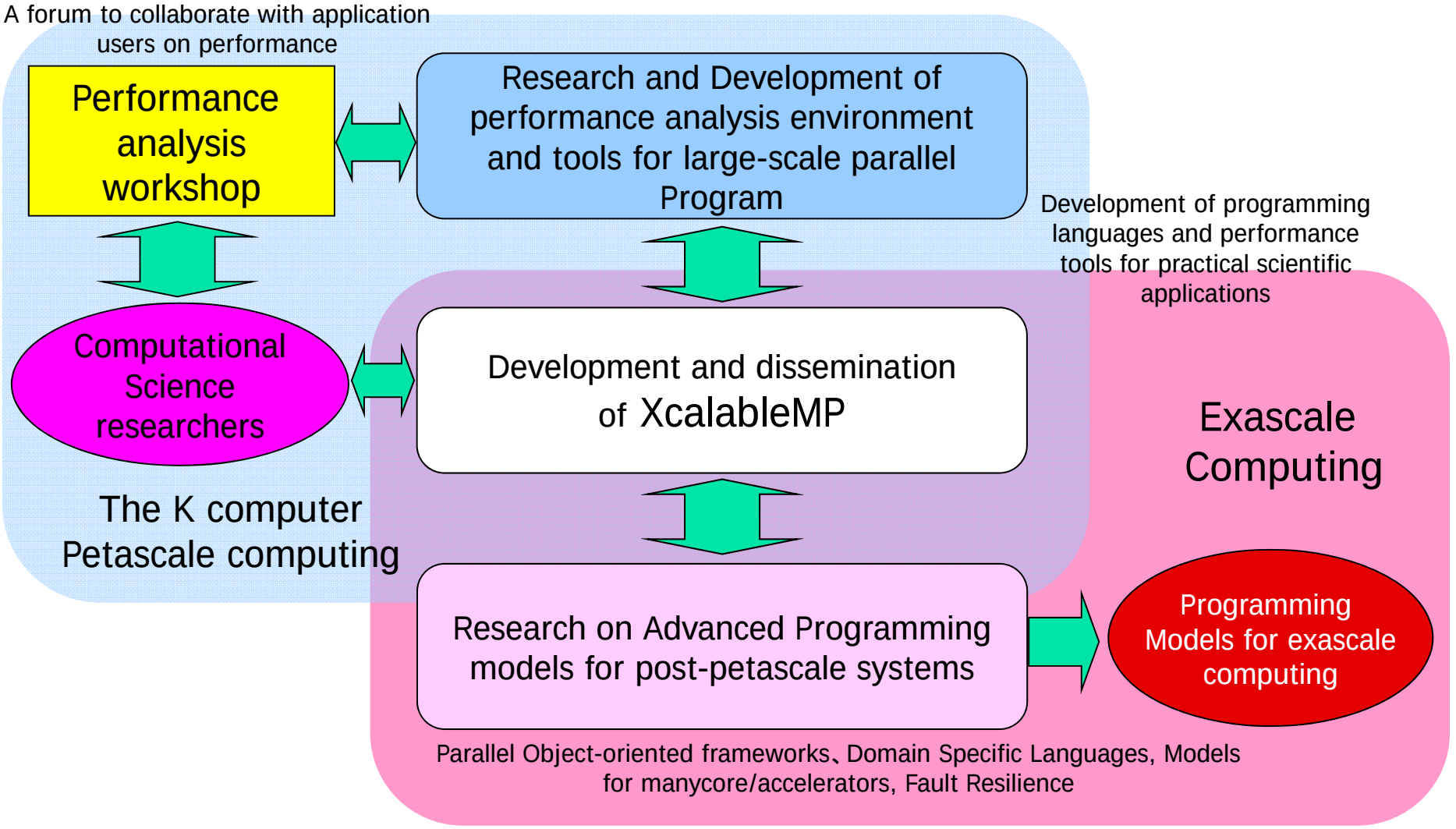
Mitsuhisa Sato

Programming Environment Research Team

RIKEN AICS

Research Topics
in AICS Programming Environment Research Team

The technologies of programming models/languages and environment play an important role to bridge between programmers and systems. Our team conducts researches of programming languages and performance tools to exploit full potentials of large-scale parallelism of the K computer and explore programming technologies towards the next generation “exascale” computing.



もくじ

- なぜ、並列化は必要なのか
- 並列化と並列プログラミング

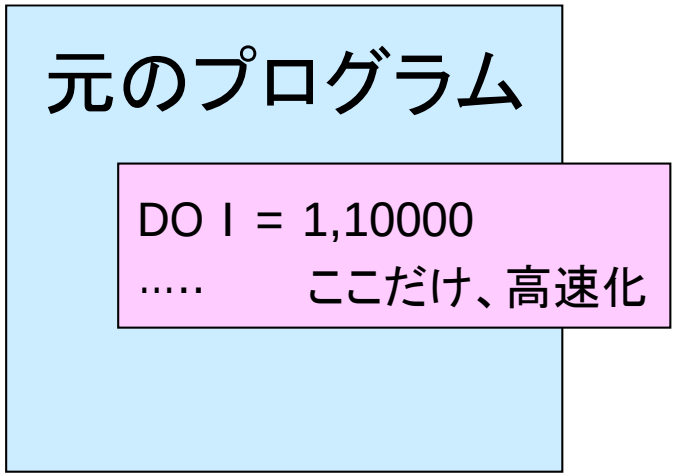
- これまでの並列プログラミング言語について
  - （OpenMP), UPC, CAF, HPF, XPF, …

- XcalableMP
  - 動機、経緯
    - 並列プログラミング言語検討会（e-scienceプロジェクト）
  - 概要
  - 現状

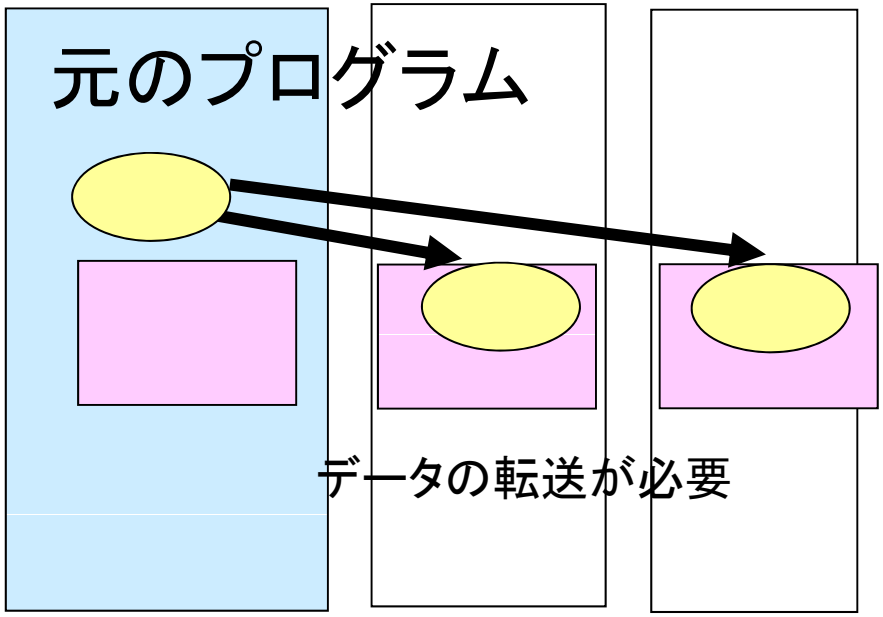
並列処理の問題点：並列化はなぜ大変か

XcalableMP

- ベクトルプロセッサ
  - あるループを依存関係がなくなるように記述
  - ローカルですむ
  - 高速化は数倍



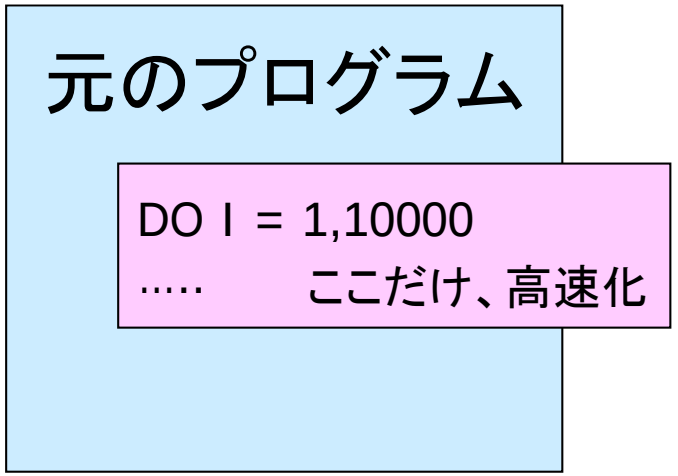
- 並列化
  - 計算の分割だけでなく、通信（データの配置）が本質的
  - データの移動が少なくなるようにプログラムを配置
  - ライブラリ的なアプローチが取りにくい
  - 高速化は数千倍ー数万



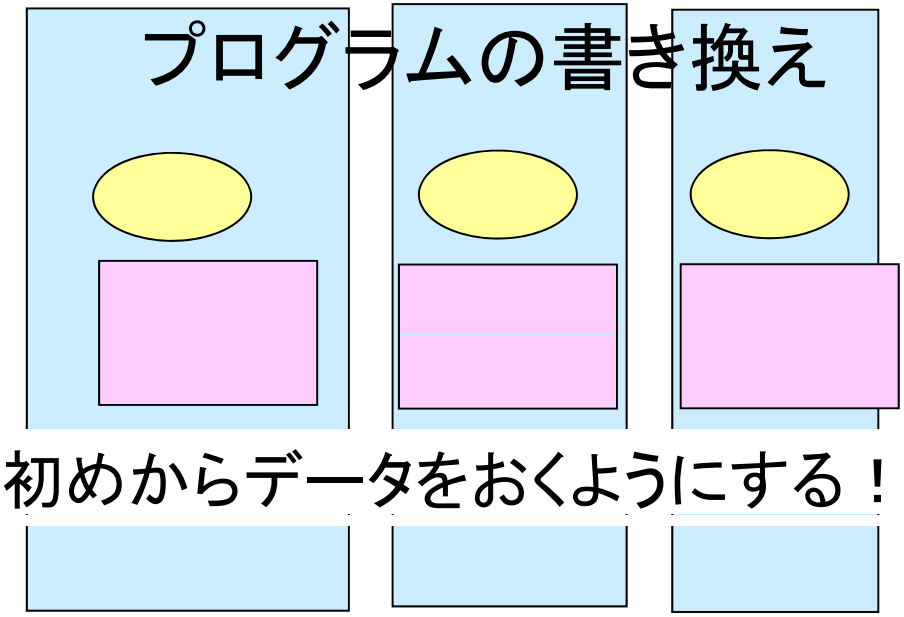
並列処理の問題点：並列化はなぜ大変か

XcalableMP

- ベクトルプロセッサ
  - あるループを依存関係がなくなるように記述
  - ローカルですむ
  - 高速化は数倍



- 並列化
  - 計算の分割だけでなく、通信（データの配置）が本質的
  - データの移動が少なくなるようにプログラムを配置
  - ライブラリ的なアプローチが取りにくい
  - 高速化は数千倍ー数万

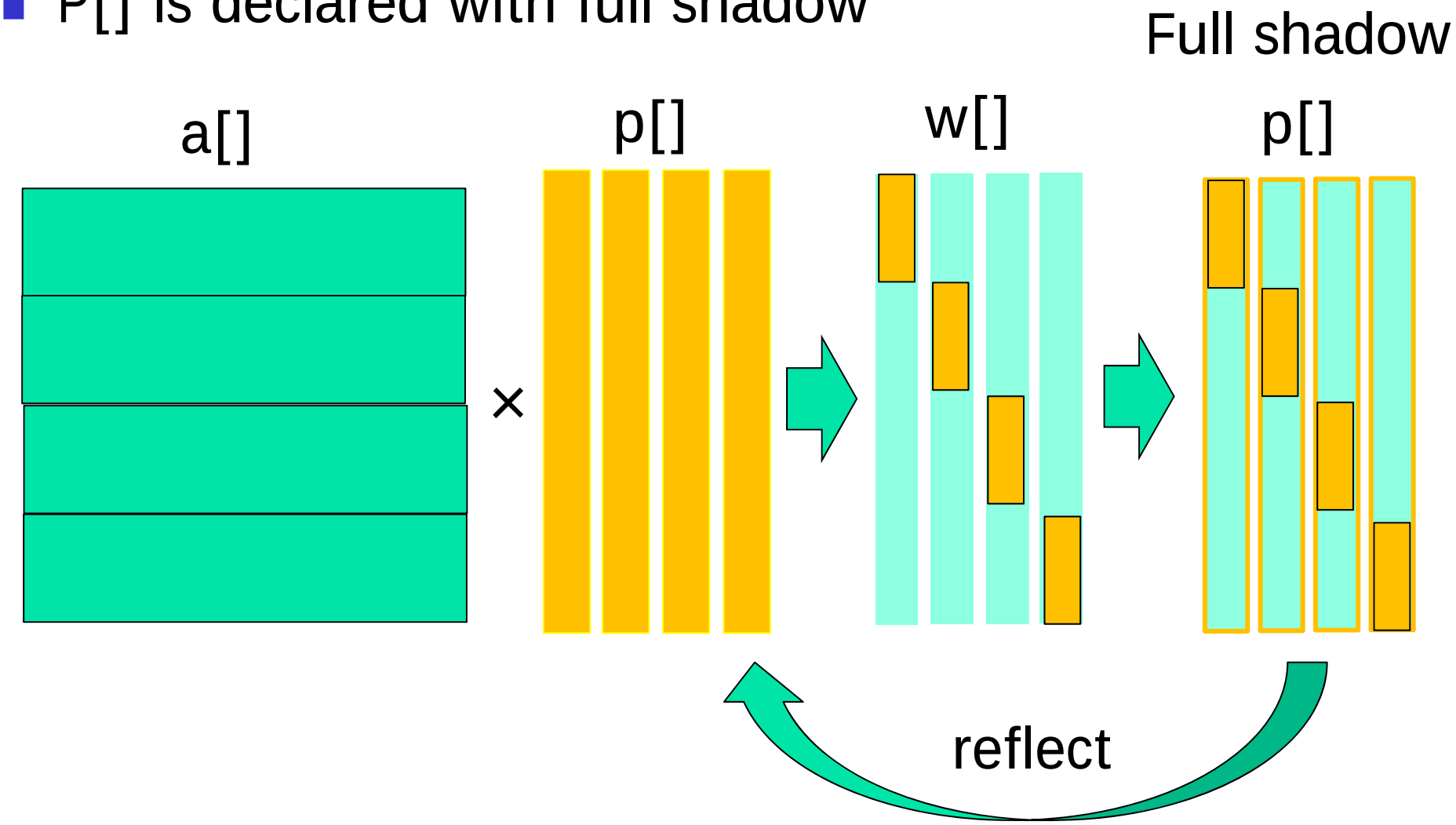


並列化と並列プログラミング

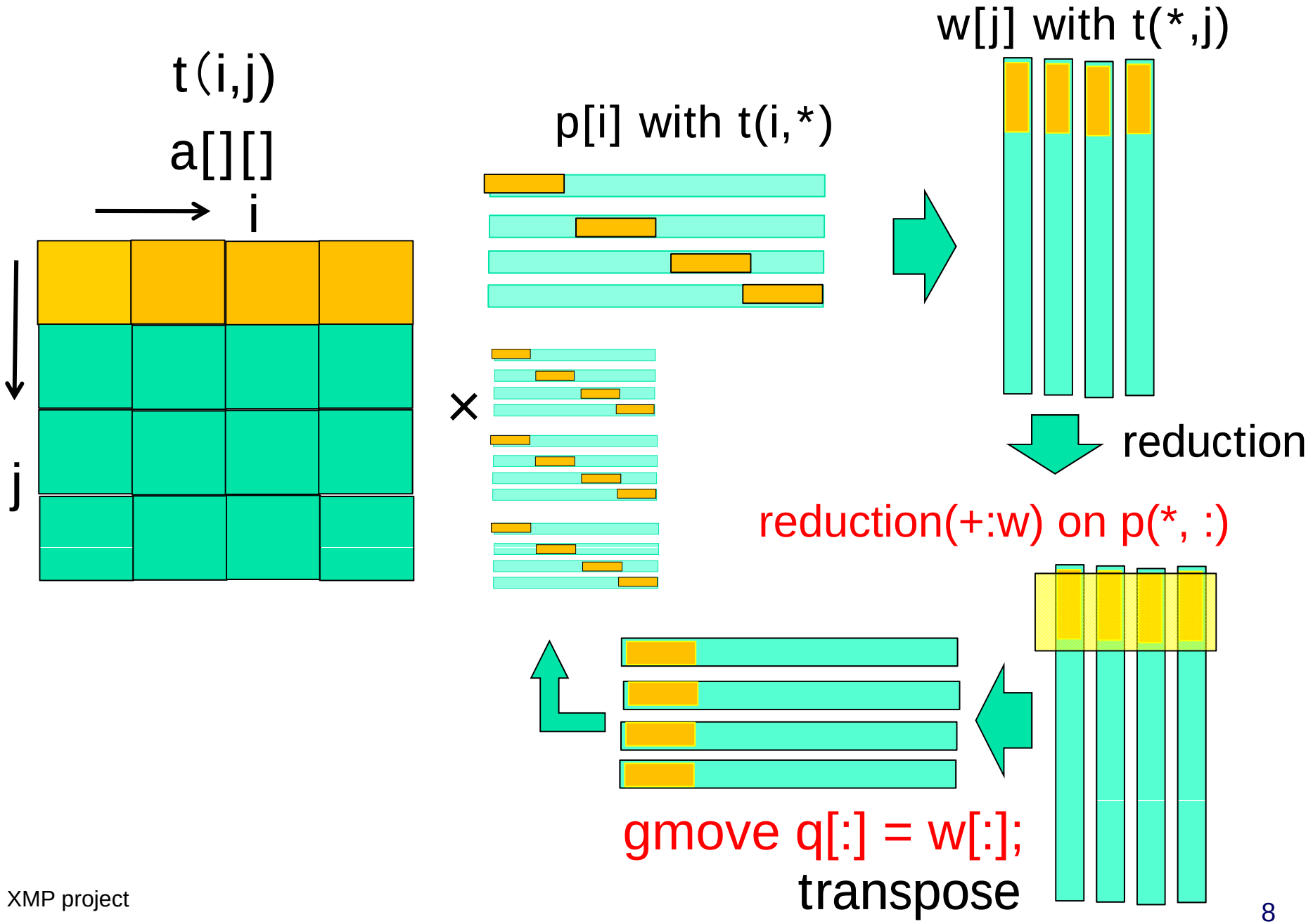
- 理想：自動並列コンパイラがあればいいのだが、…
- 「並列化」と並列プログラミングは違う！
- なぜ、並列プログラミングが必要か
  - ベクトル行列積を例に、…

1次元並列化

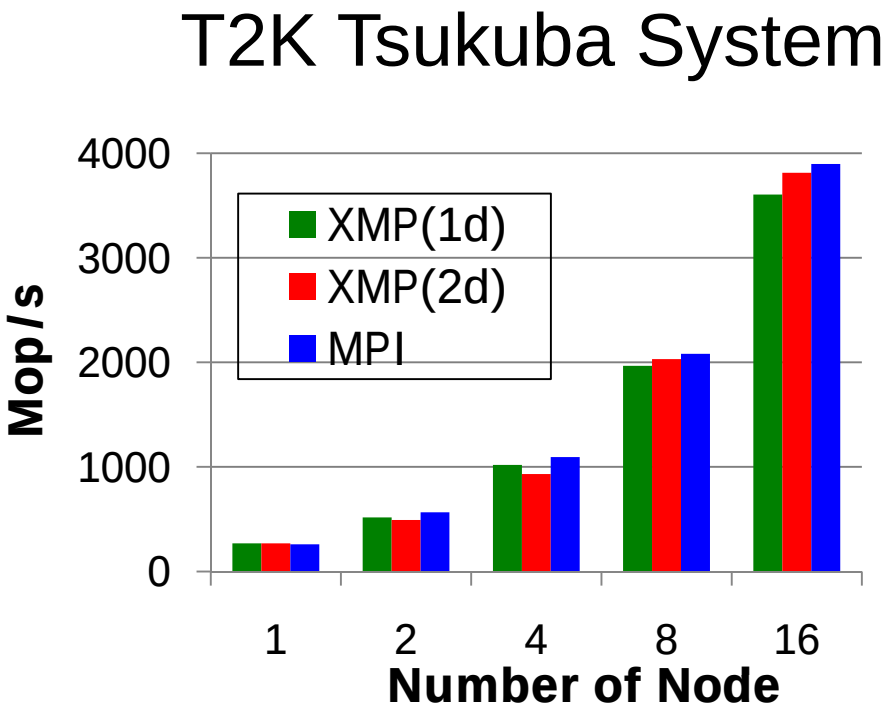
- P[] is declared with full shadow

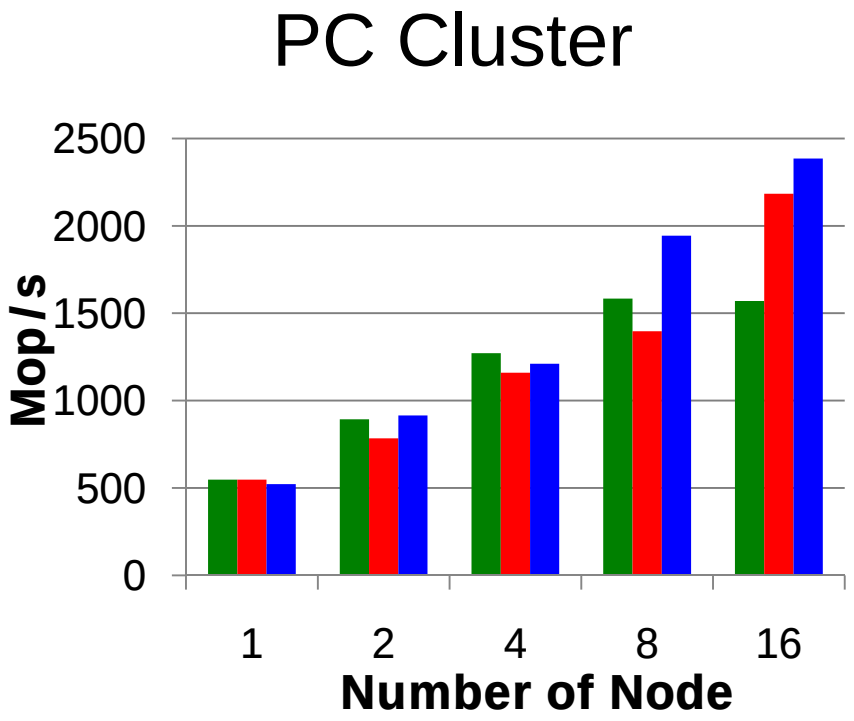


2次元並列化


t(i,j)
a[][]
i
j
p[i] with t(i,*)
×
w[j] with t(*,j)
reduction
reduction(+:w) on p(*, :)
gmove q[:] = w[:];
transpose

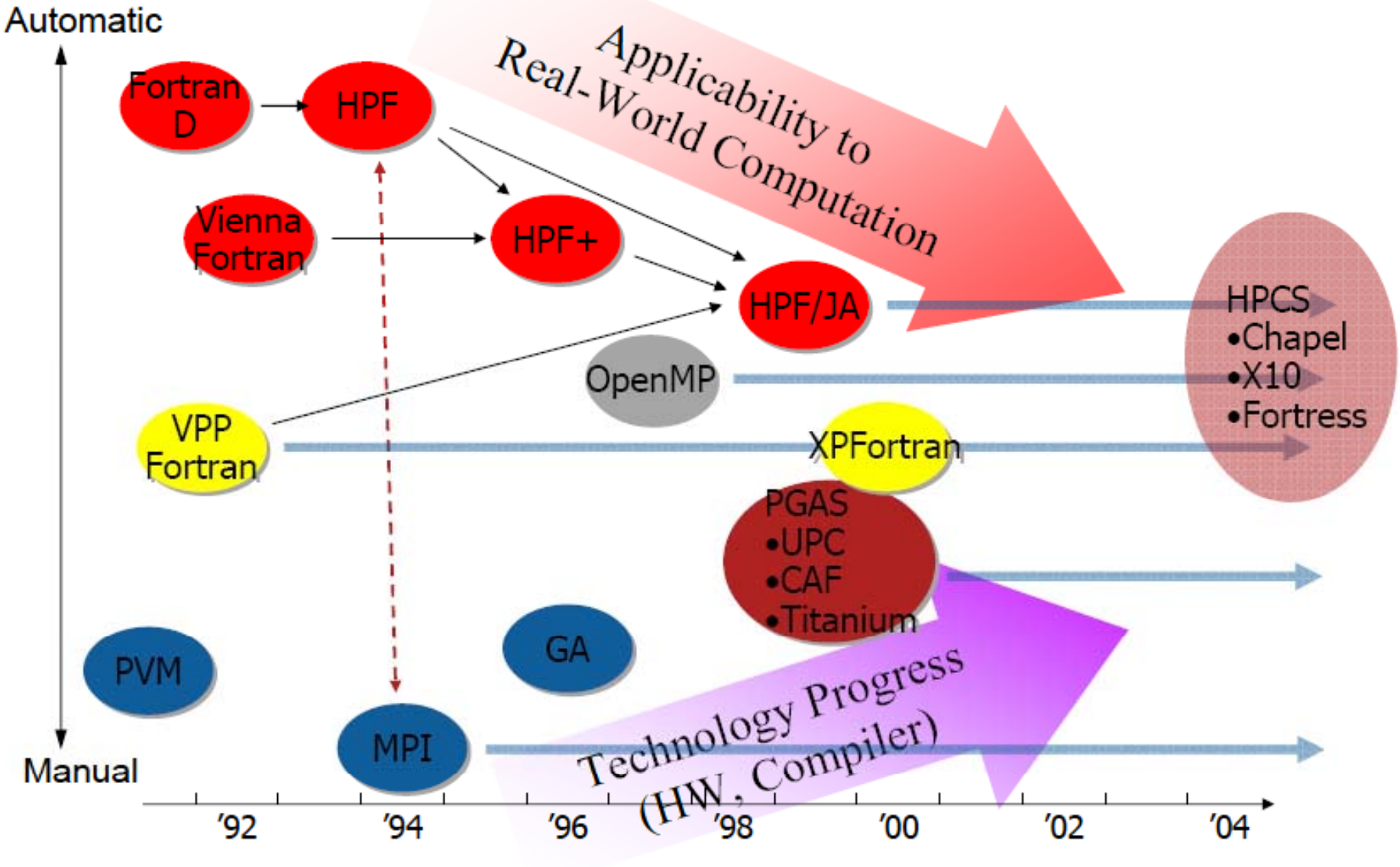
Performance Results : NPB-CG





The results for CG indicate that the performance of 2D. parallelization in XMP is comparable to that of MPI.

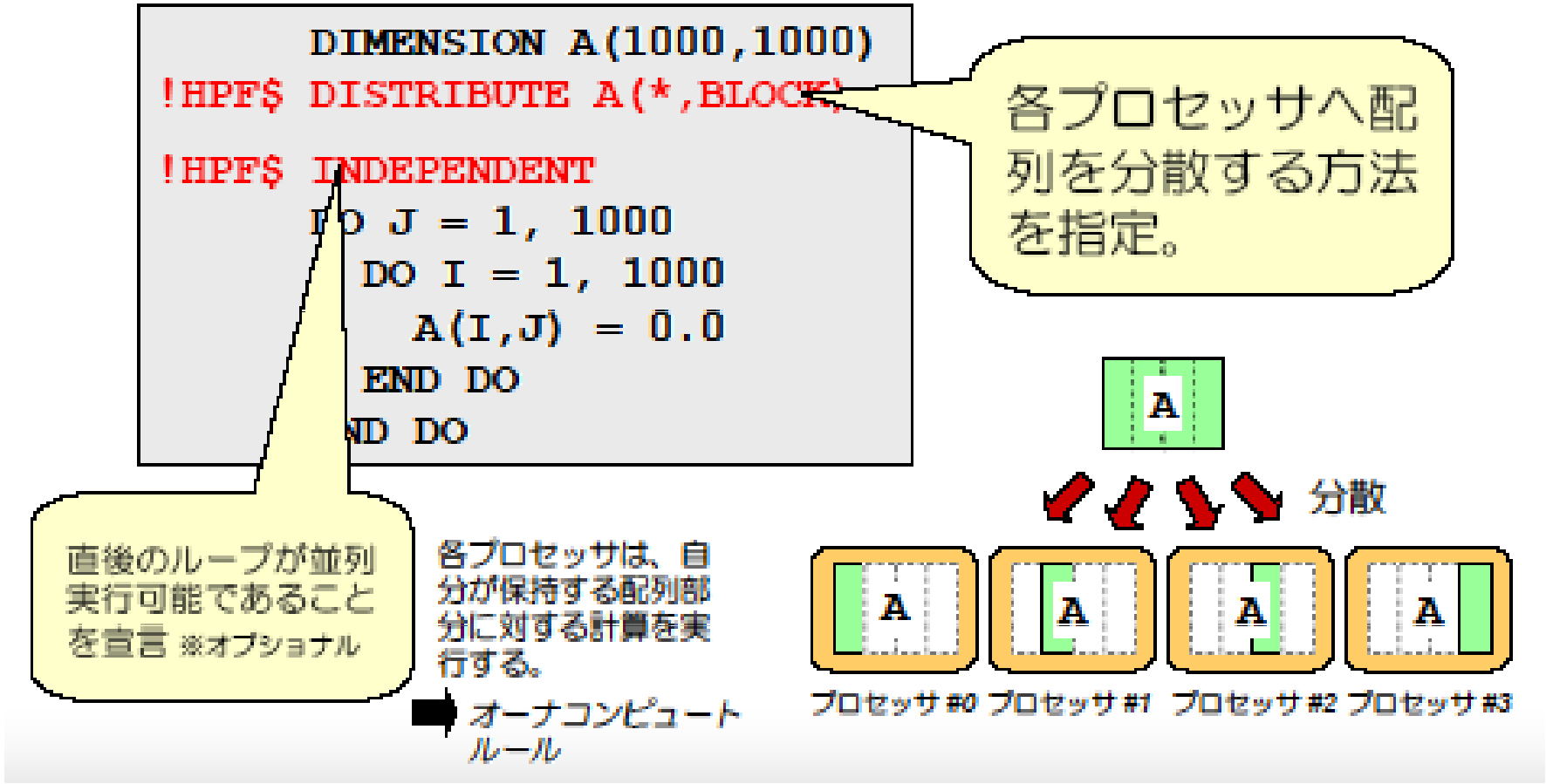
History and Trends for Parallel Programming Languages



courtesy of K. Hotta@fujitsu、Seo@NEC

HPF: High Performance Fortran

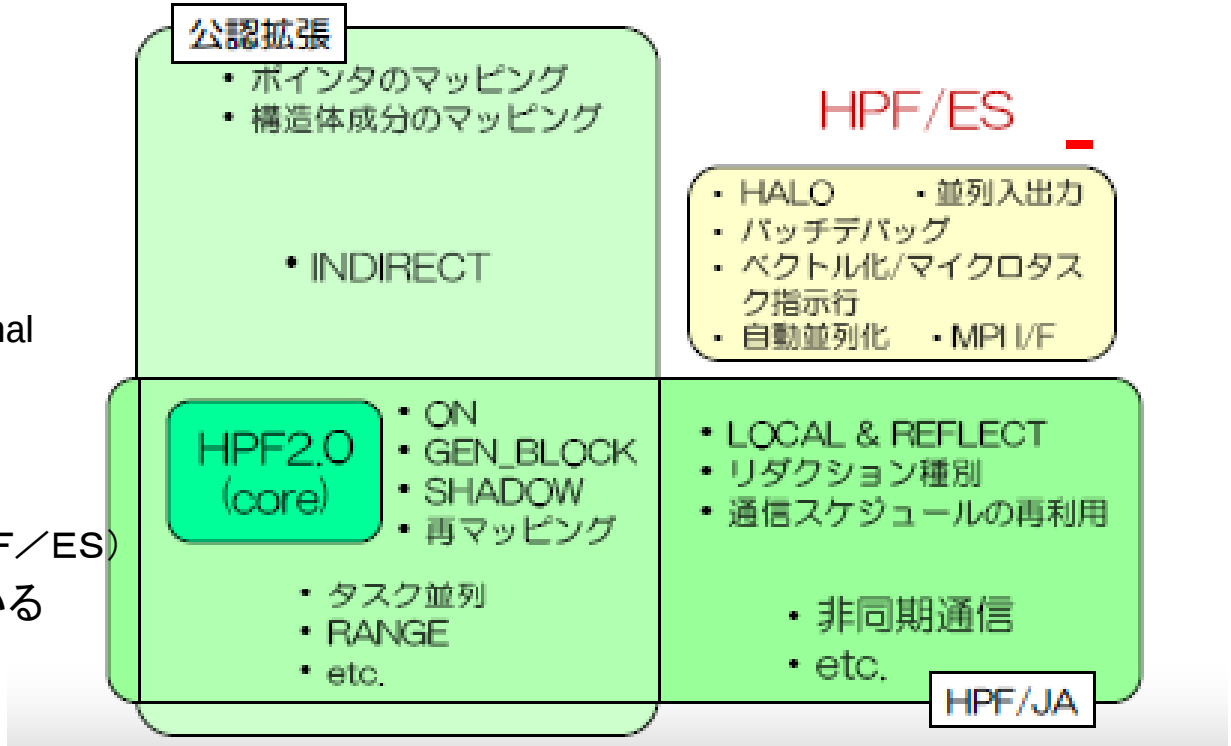
- Data Mapping: ユーザが分散を指示
- 計算は、owner-compute rule
- データ転送や並列実行制御はコンパイラが生成



HPF/JA、HPF/ES

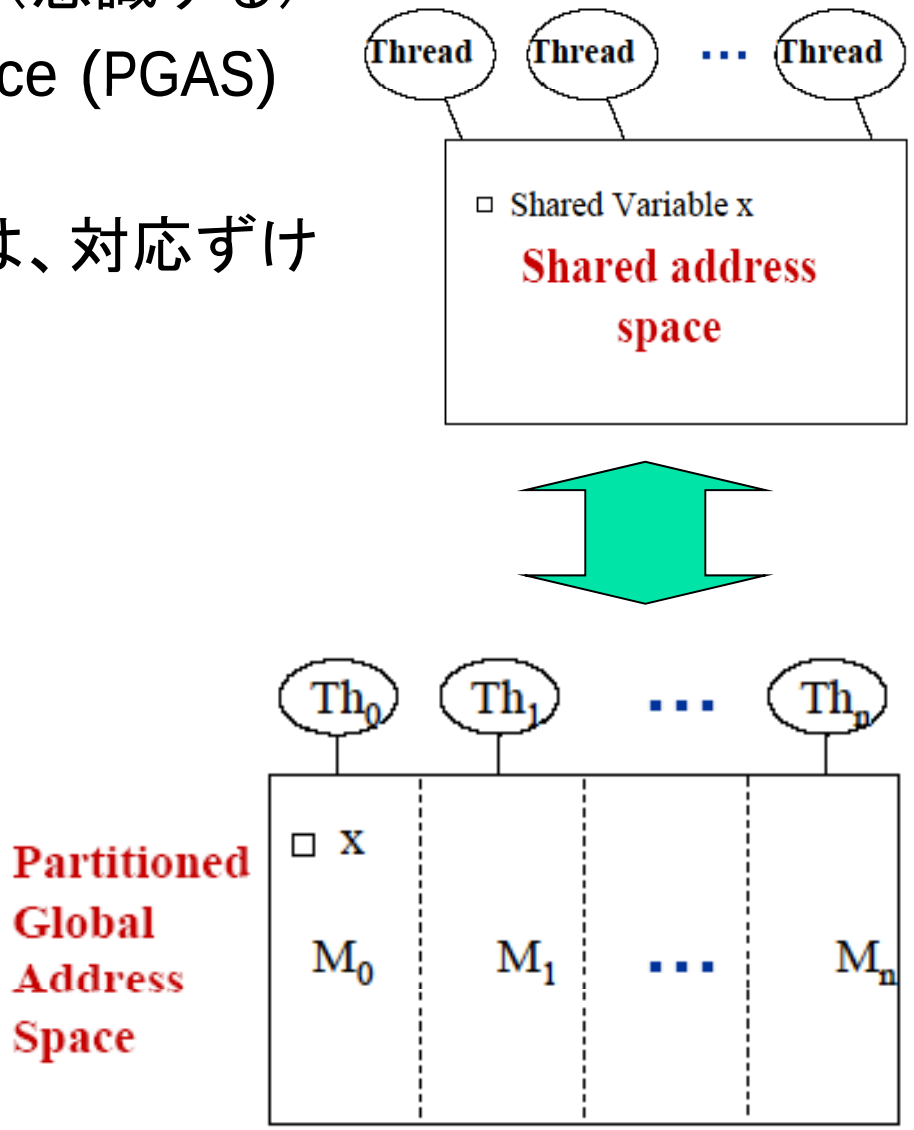
- HPF/JA
  - データ転送制御directiveの拡張
    - Asynchronous Communication, Shift optimization, Communication schedule reuse
  - 並列化支援の強化（reduction等）
- HPF/ES
  - HALO, Vectorization/Parallelization handling, Parallel I/O

- 現状
  - HPFは、日本（HPFPC（HPF推進協議会））でサポートされている
  - SC2002 Gordon Bell Award
    - 14.9 Tflops Three-dimensional Fluid Simulation for Fusion Science with HPF on the Earth Simulator
      - HPFそのままではない （HPF／ES）
  - 国内ベンダーはサポートしている
  - 米国dHPF at Rice U.



Global Address Space Model Programming

- ユーザがlocal/globalを宣言する（意識する）
- Partitioned Global Address Space (PGAS) model
- スレッドと分割されたメモリ空間は、対応ずけられている（affinity)
  - 分散メモリモデルに対応
- 「shared/global」の発想は、いろいろなところから同時に出てきた。
  - Split-C
  - PC++
  - UPC
  - CAF: Co-Array Fortran
  - （EM-C for EM-4/EM-X)
  - (Global Array)

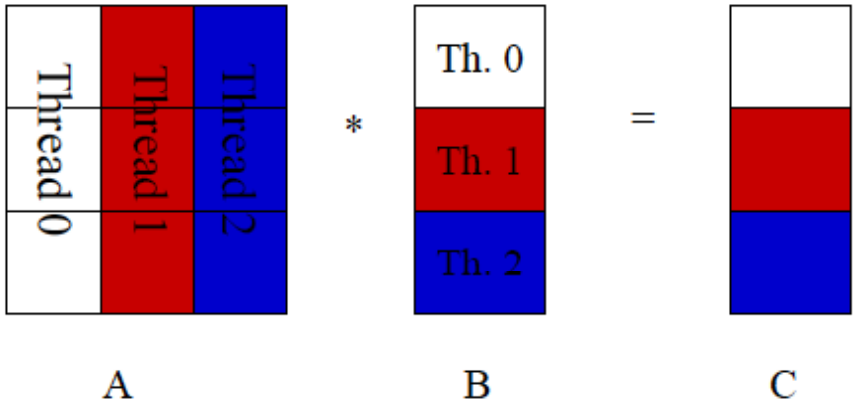


UPC： Unified Parallel C

- Unified Parallel C
  - Lawrence Berkeley National Lab.を中心に設計開発
- Private/Shared を宣言
- SPMD
  - MYTHREADが自分のスレッド番号
  - 同期機構
    - **Barriers**
    - **Locks**
    - **Memory consistency control**
- User’s view
  - 分割されたshared space について、複数のスレッドが動作する。
  - ただし、分割されたshared space はスレッドに対してaffinityを持つ。

行列積の例

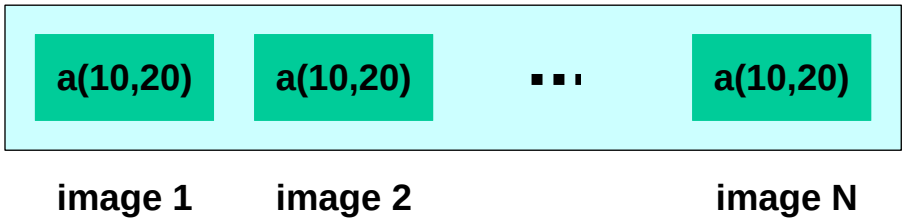
```
#include <upc_relaxed.h>
shared int a[THREADS][THREADS];
shared int b[THREADS], c[THREADS];

void main (void) {
    int i, j;
    upc_forall(i=0;i<THREADS;i++;i){
      c[i] = 0;
      for (j=0; j<THREADS; j++)
          c[i] += a[i][j]*b[j];
    }
}
```



CAF: Co-Array Fortran

- Global address space programming model
  - one-sided communication (GET/PUT)
- SPMD 実行を前提
- Co-array extension
  - 各プロセッサで動くプログラムは、異なる”**image**”を持つ。

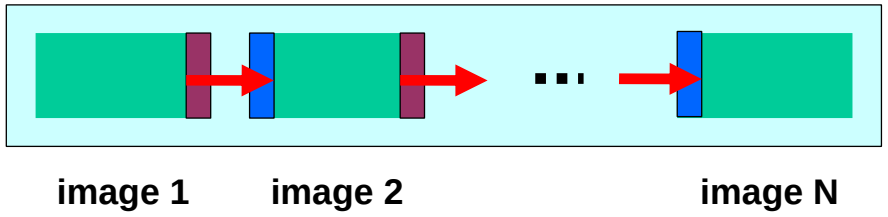
```
integer a(10,20)[*]
```




```
real, dimension(n)[*] :: x,y
x(:) = y(:)[q]
```

qのimageで動くyのデータをローカルなxにコピする(get)

- プログラマは、パフォーマンス影響を与える要素に対して制御する。
  - データの分散配置
  - 計算の分割
  - 通信をする箇所



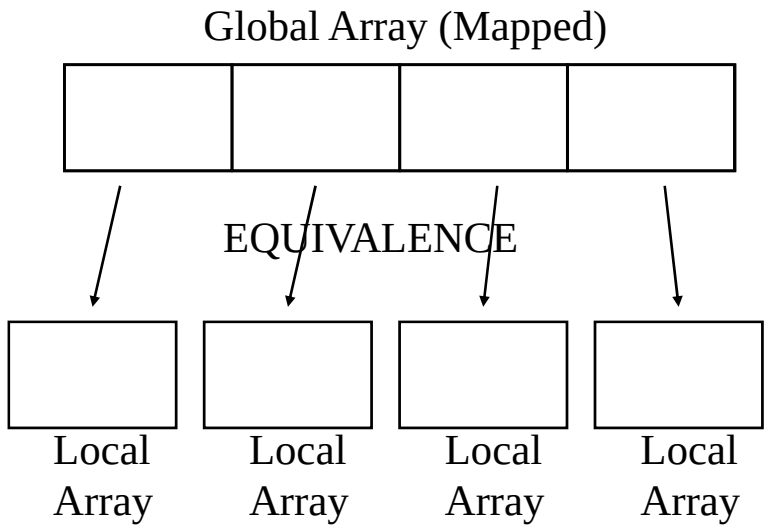
- データの転送と同期の言語プリミティブをもっている。
  - amenable to compiler-based communication optimization

```
if (this_image() > 1)
   a(1:10,1:2) =
     a(1:10,19:20)[this_image()-1]
```

XPFortran (VPP Fortran)

- NWT (Numerical Wind Tunnel)向けに開発された言語、実績あり
- localとglobalの区別をする。
- インデックスのpartitionを指定、それを用いてデータ、計算ループの分割を指示
- 逐次との整合性をある程度保つことができる。言語拡張はない。

```
!XOCL PROCESSOR P(4)
      dimension a(400),b(400)
!XOCL INDEX PARTITION D=
     (P,INDEX=1:1000)
!XOCL GLOBAL a(/D(overlap=(1,1))), b(/D)
!XOCL PARALLEL REGION
!XOCL SPREAD DO REGIDENT(a,b) /D
      do i = 2, 399
        dif(i) = u(i+1) - 2*u(i) + u(i-1)
      end do
!XOCL END SPREAD
!XOCL END PARALLEL
```



Partitioned Global Address Space言語の利点・欠点

- MPIとHPFの中間に位置する
  - わかり易いモデル
  - 比較的、プログラミングが簡単、MPIほど面倒ではない。
  - ユーザから見えるプログラミングモデル。通信、データの配置、計算の割り当てを制御できる
  - MPIなみのtuningもできる。
    - プログラムとしてpack/unpackをかいてもいい。

- 欠点
  - 並列のために言語を拡張しており、逐次には戻れない。(OpenMPのようにincrementalではない）
  - やはり、制御しなくてはならないか。
  - 性能は？

現状のまとめ

XcalableMP

- MPI
  - これが残念ながら、現状！
  - これでいいのか…!?
- OpenMP:
  - 簡単。incrementalに並列化できる。
  - 共有メモリ向け、100プロセッサまで
  - incrementalでいいのだが、そもそも分散メモリには対応していない。
  - MPIコードがすでにある場合は、Mixed OpenMP-MPI はあまり必要ないことが多い
- HPF:
  - 使えるようになってきた（HPF for PC cluster)
  - が、実用的なプログラムは難しいし、問題点もある
  - コンパイラに頼りすぎ。実行のイメージが見えない
  - そもそも、…
- PGAS (Patitioned Global Address Space) 言語
  - 米国では、だんだん広まりつつある。
  - MPIよりはまし。そこそこの性能もでる… が、まだ、one-sidedはむずかしい
  - 基本的に、プログラムを書き換える必要がある。
  - そもそも、このくらいで手をうってもいいのか…
- 自動並列化コンパイラ
  - 究極。共有メモリにはそこそこ使えるようになってきている。が、分散メモリは、むずかしい。

- あまり、分散メモリ向けの言語の話はない。PGASぐらいか。
- プログラミング言語の研究は、マルチコアで盛り上がりを見せているが、相変わらずMPIとのハイブリッドだけ。
- MPIで満足しているのか？

- そもそも、もうすでに大方のプログラムはMPIで書かれてしまっているのか。
- 日本のユーザは、自分でプログラムを書いているケースがすくないので、新しい言語をつくっても役に立たない？

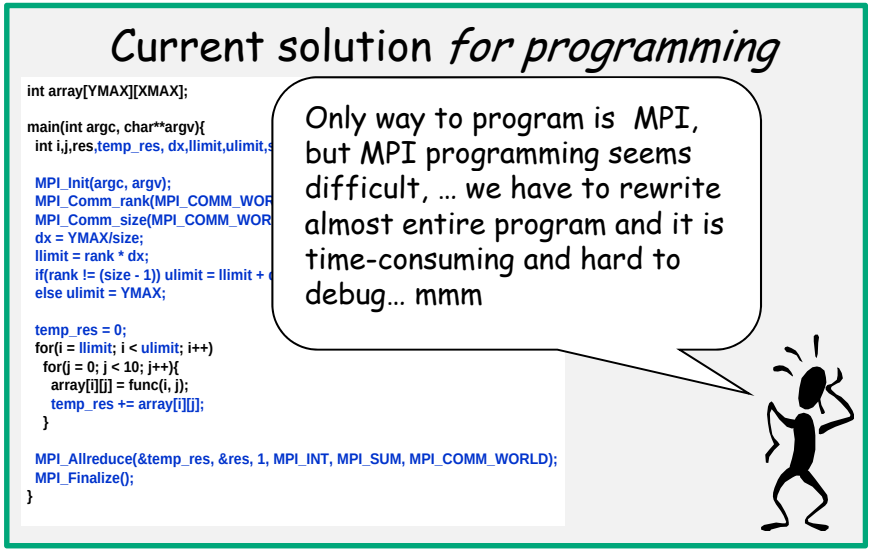
- でも、やっぱりMPIは問題だ！（と、私はおもう）
- 日本ではHPFがあったじゃないか？

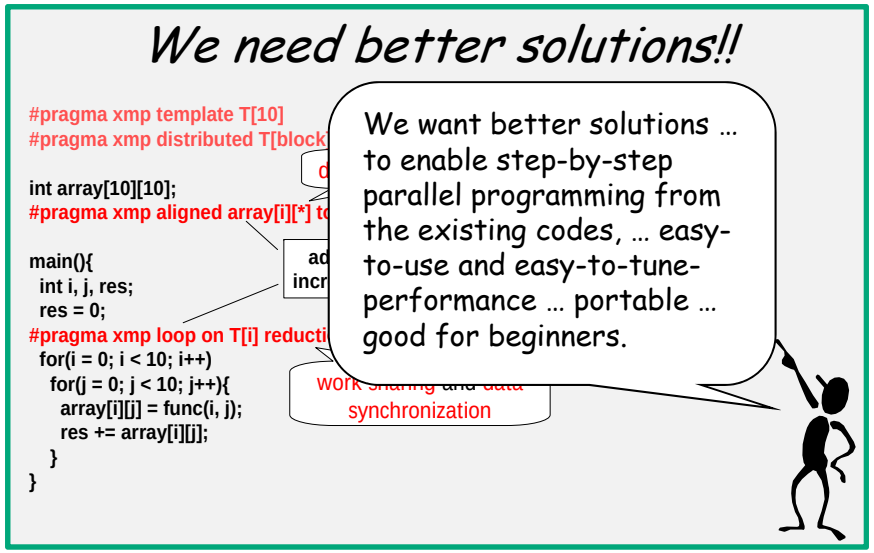
Why do we need parallel programming language researches?

- In 90's, many programming languages were proposed.
  - but, most of these disappeared.
- MPI is dominant programming in a distributed memory system
  - low productivity and high cost
- No standard parallel programming language for HPC
  - only MPI
  - PGAS, but…

XcalableMP

is our solution!





20

What's XcalableMP?

- XcalableMP (XMP for short) is:
  - A programming model and language for distributed memory , proposed by XMP WG
  - http://www.xcalablemp.org

- XcalableMP Specification Working Group (XMP WG)
  - XMP WG is a special interest group, which organized to make a draft on "petascale" parallel language.
  - Started from December 2007, the meeting is held about once in every month.
    - Mainly active in Japan, but open for everybody.

- XMP WG Members (the list of initial members)
  - Academia: M. Sato, T. Boku (compiler and system, U. Tsukuba), K. Nakajima (app. and programming, U. Tokyo), Nanri (system, Kyusyu U.), Okabe (HPF, Kyoto U.)
  - Research Lab.: Watanabe and Yokokawa (RIKEN), Sakagami (app. and HPF, NIFS), Matsuo (app., JAXA), Uehara (app., JAMSTEC/ES)
  - Industries: Iwashita and Hotta (HPF and XPFortran, Fujitsu), Murai and Seo (HPF, NEC), Anzaki and Negishi (Hitachi), (many HPF developers!)

- Funding for development
  - e-science project : "Seamless and Highly-productive Parallel Programming Environment for High-performance computing" project funded by MEXT,Japan
    - Project PI: Yutaka Ishiakwa, co-PI: Sato and Nakashima(Kyoto), PO: Prof. Oyanagi
    - Project Period: 2008/Oct to 2012/Mar (3.5 years)

HPF (high Performance Fortran) history in Japan

- Japanese supercomputer venders were interested in HPF and developed HPF compiler on their systems.
- NEC has been supporting HPF for Earth Simulator System.
- Activities and Many workshops: HPF Users Group Meeting (HUG from 1996-2000), HFP intl. workshop (in Japan, 2002 and 2005)
- Japan HPF promotion consortium was organized by NEC, Hitatchi, Fujitsu …
  - HPF/JA proposal
- Still survive in Japan, supported by Japan HPF promotion consortium
- XcalableMP is designed based on the experience of HPF, and Many concepts of XcalableMP are inherited from HPF

Lessons learned from HPF

- "Ideal" design policy of HPF
  - A user gives a small information such as data distribution and parallelism.
  - The compiler is expected to generate "good" communication and work-sharing automatically.
- No explicit mean for performance tuning .
  - Everything depends on compiler optimization.
  - Users can specify more detail directives, but no information how much performance improvement will be obtained by additional informations
    - INDEPENDENT for parallel loop
    - PROCESSOR + DISTRIBUTE
    - ON HOME
  - The performance is too much dependent on the compiler quality, resulting in "incompatibility" due to compilers.
  - **Lesson :*"Specification must be clear. Programmers want to know what happens by giving directives"***
  - The way for tuning performance should be provided.

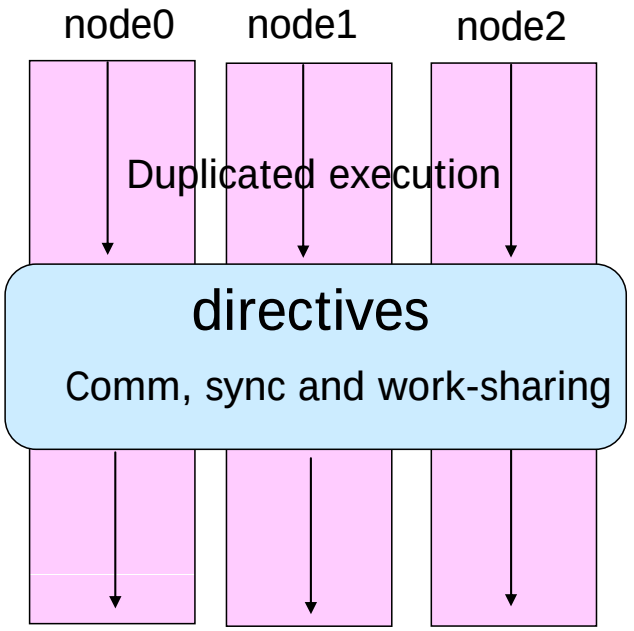
Performance-awareness: This is one of the most important lessons for the design of XcalableMP

XcalableMP

http://www.xcalablemp.org

XcalableMP : directive-based language eXtension
for Scalable and performance-aware Parallel Programming

- **Directive-based language extensions** for familiar languages F90/C (C++)
  - To reduce code-rewriting and educational costs.



- **"Scalable" for Distributed Memory Programming**
  - SPMD as a basic execution model
  - A thread starts execution in each node independently (as in MPI) .
  - Duplicated execution if no directive specified.
  - MIMD for Task parallelism
- **"performance-aware" for explicit communication and synchronization**.
  - Work-sharing and communication occurs when directives are encountered
  - All actions are taken by directives for being "easy-to-understand" in performance tuning (different from HPF)

Code Example

```
int array[YMAX][XMAX];

#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] with t(i)

main(){
  int i, j, res;
  res = 0;

#pragma xmp loop on t(i)  reduction(+:res)
  for(i = 0; i < 10; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      res += array[i][j];
    }
}
```

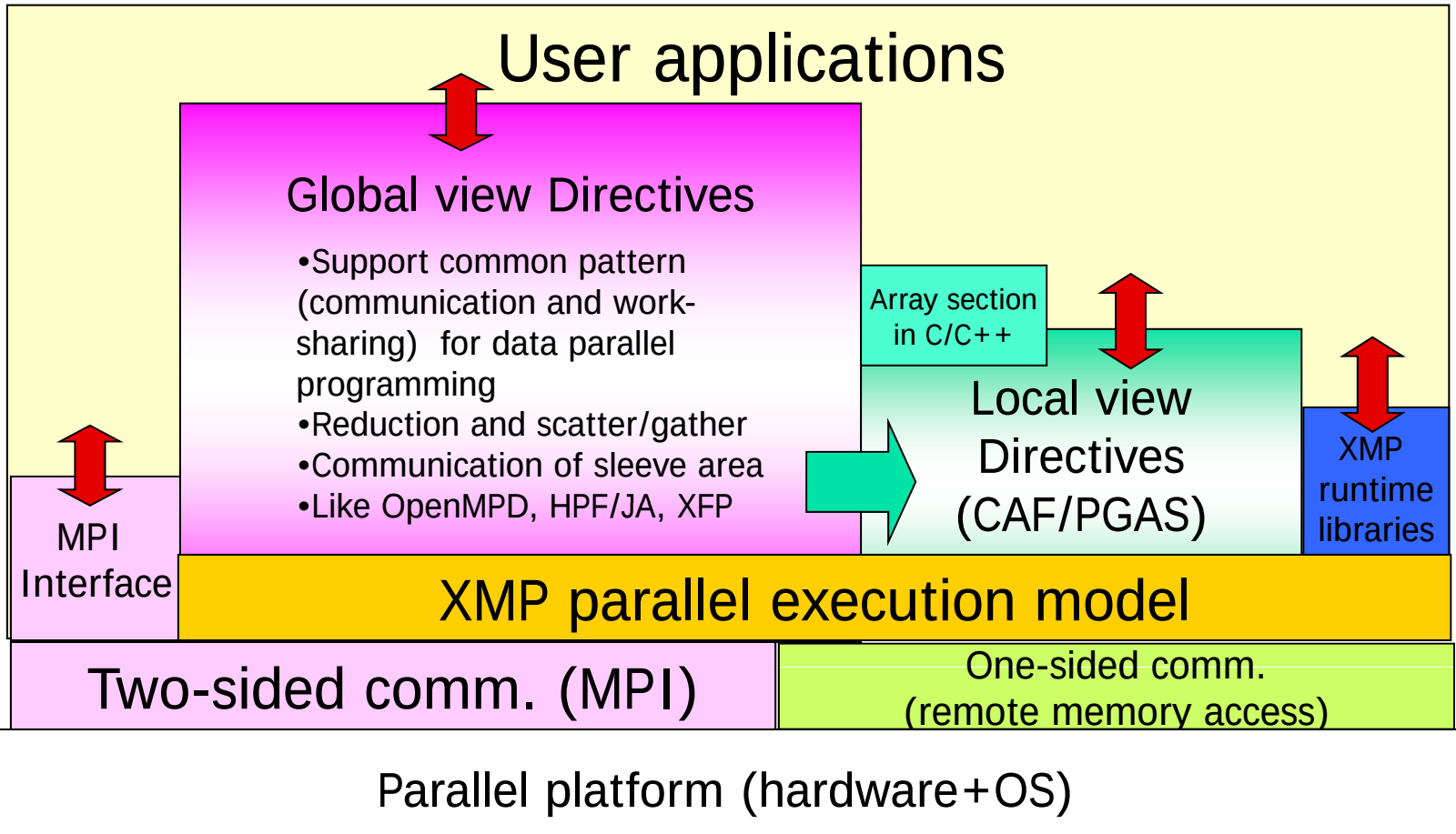
data distribution

add to the serial code : incremental parallelization

work sharing and data synchronization

Overview of XcalableMP

- XMP supports typical parallelization based on the **data parallel paradigm** and work sharing under "**global view**“
  - An original sequential code can be parallelized with **directives**, like OpenMP.
- XMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as "**local view**" programming.



Nodes, templates and data/loop distributions

- Idea inherited from HPF
- Node is an abstraction of processor and memory in distributed memory environment, declared by node directive.

```
#pragma xmp nodes p(32)
#pragma xmp nodes p(*)
```

- Template is used as a dummy array distributed on nodes

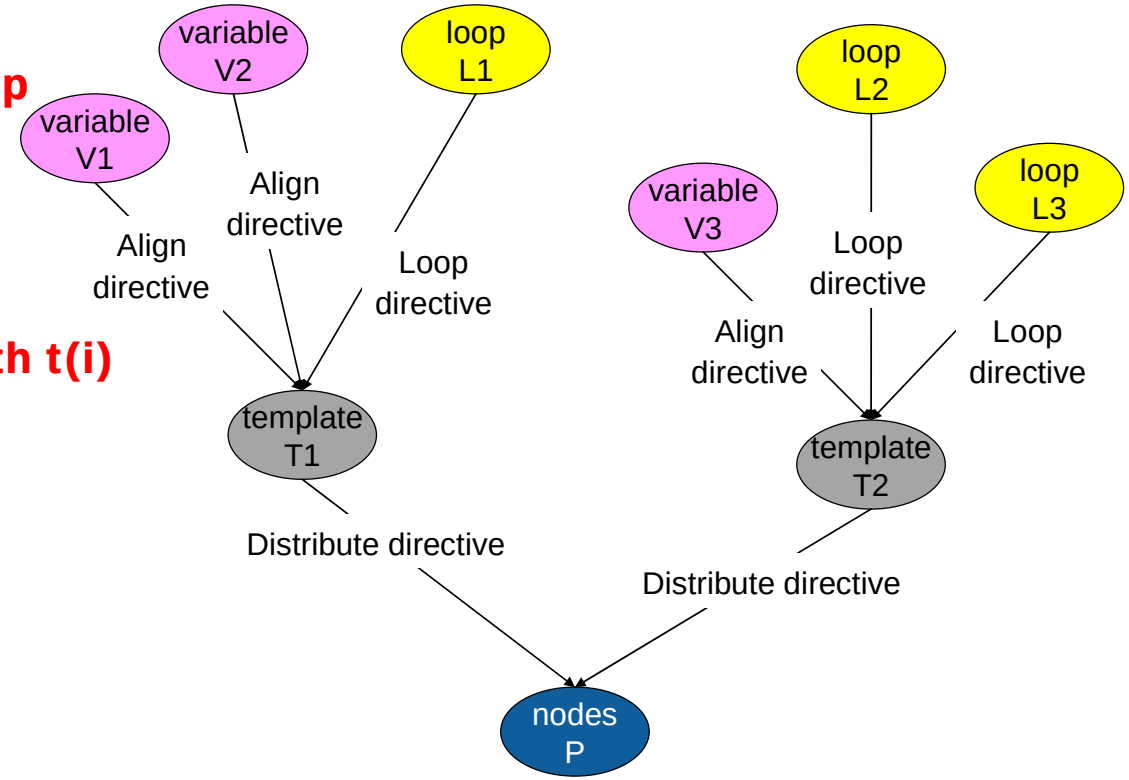
```
#pragma xmp template t(100)
#pragma xmp distribute t(block) onto p
```

- A global data is aligned to the template

```
#pragma xmp align array[i][*] with t(i)
```

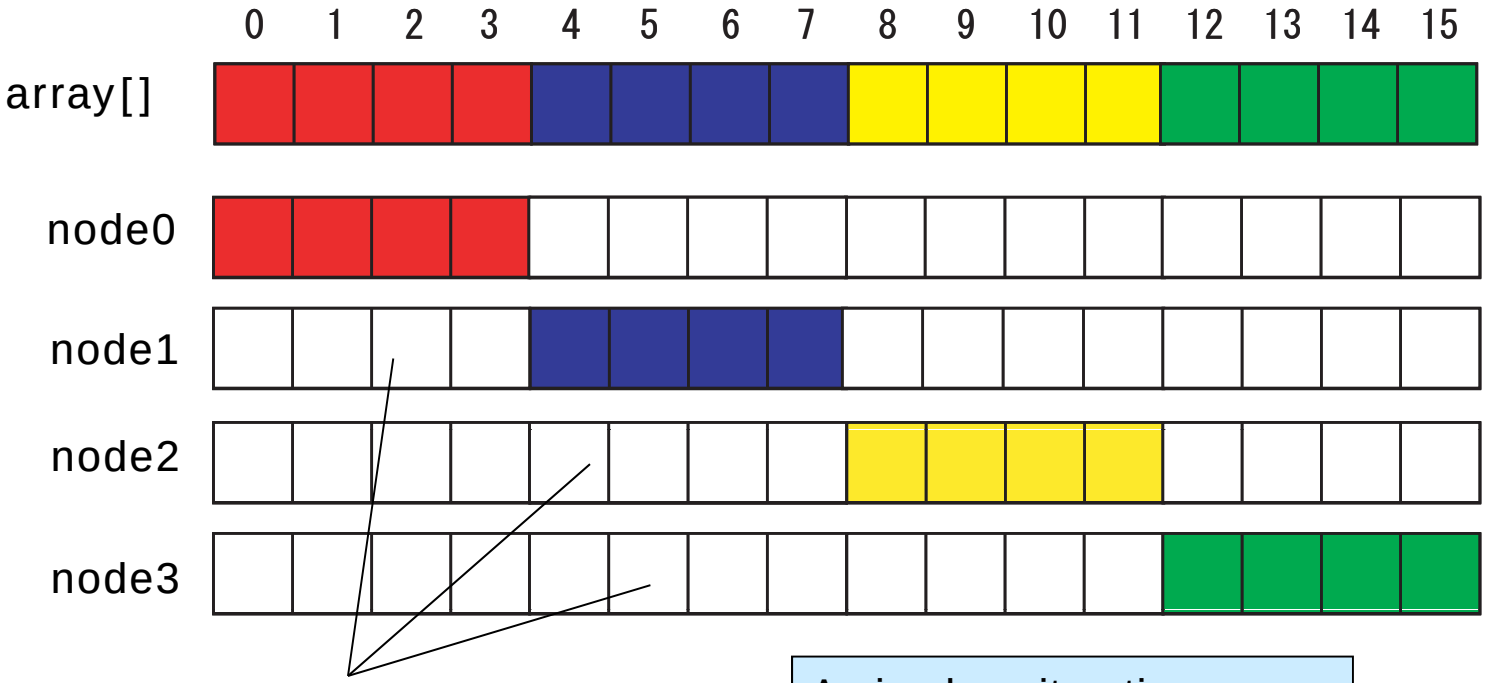
- Loop iteration must also be aligned to the template by on-clause.

```
#pragma xmp loop on t(i)
```



Array data distribution

- The following directives specify a data distribution among nodes
  - #pragma xmp nodes p(*)
  - #pragma xmp template T(0:15)
  - #pragma xmp distribute T(block) on p
  - #pragma xmp align array[i] with T(i)



Reference to assigned to other nodes may causes error!!

Assign loop iteration as to compute own data

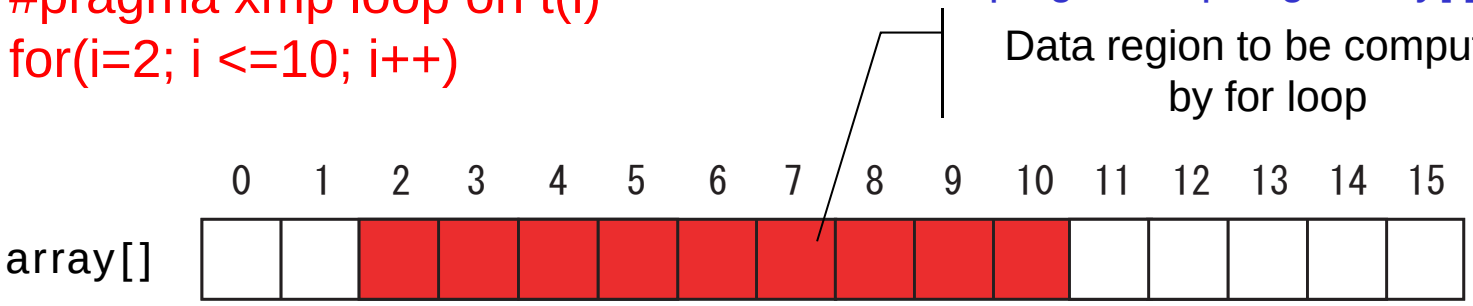
Communicate data between other nodes

Parallel Execution of “for” loop

- Execute for loop to compute on array

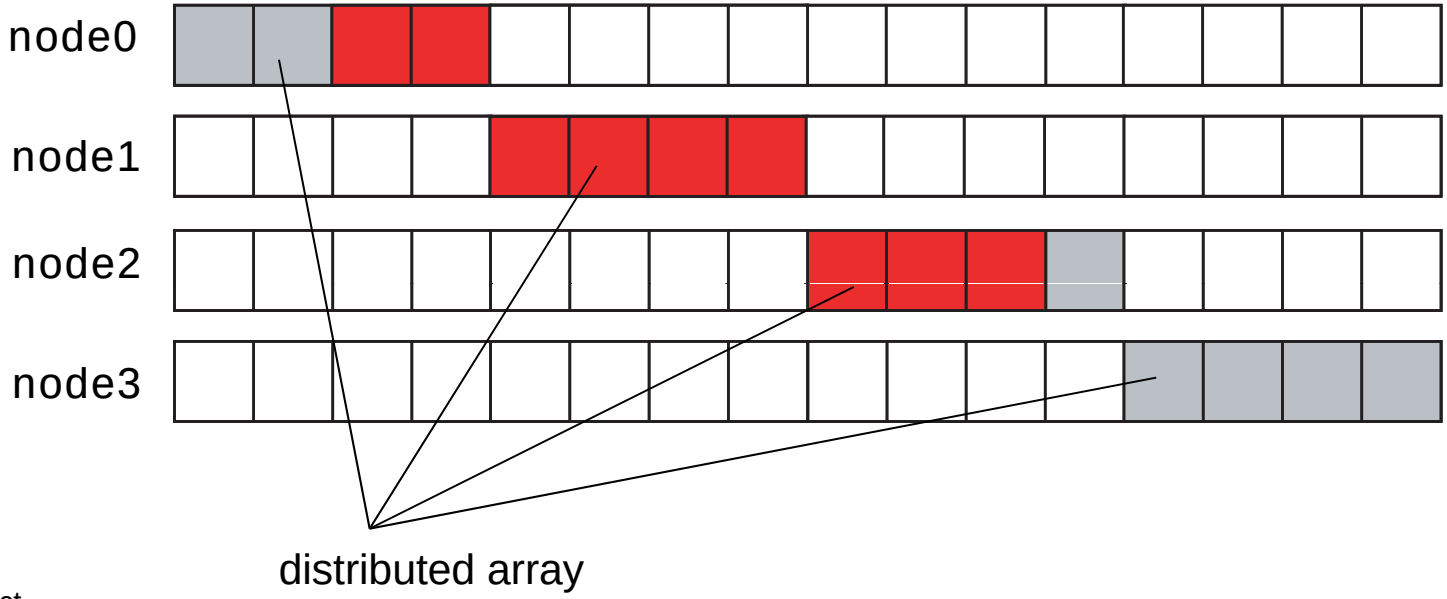
```
#pragma xmp nodes p(*)
#pragma xmp template T(0:15)
#pragma xmp distributed T(block) onto p
#pragma xmp align array[i] with T(i)
```

```
#pragma xmp loop on t(i)
for(i=2; i <=10; i++)
```



Execute “for” loop in parallel with affinity to array distribution by on-clause:

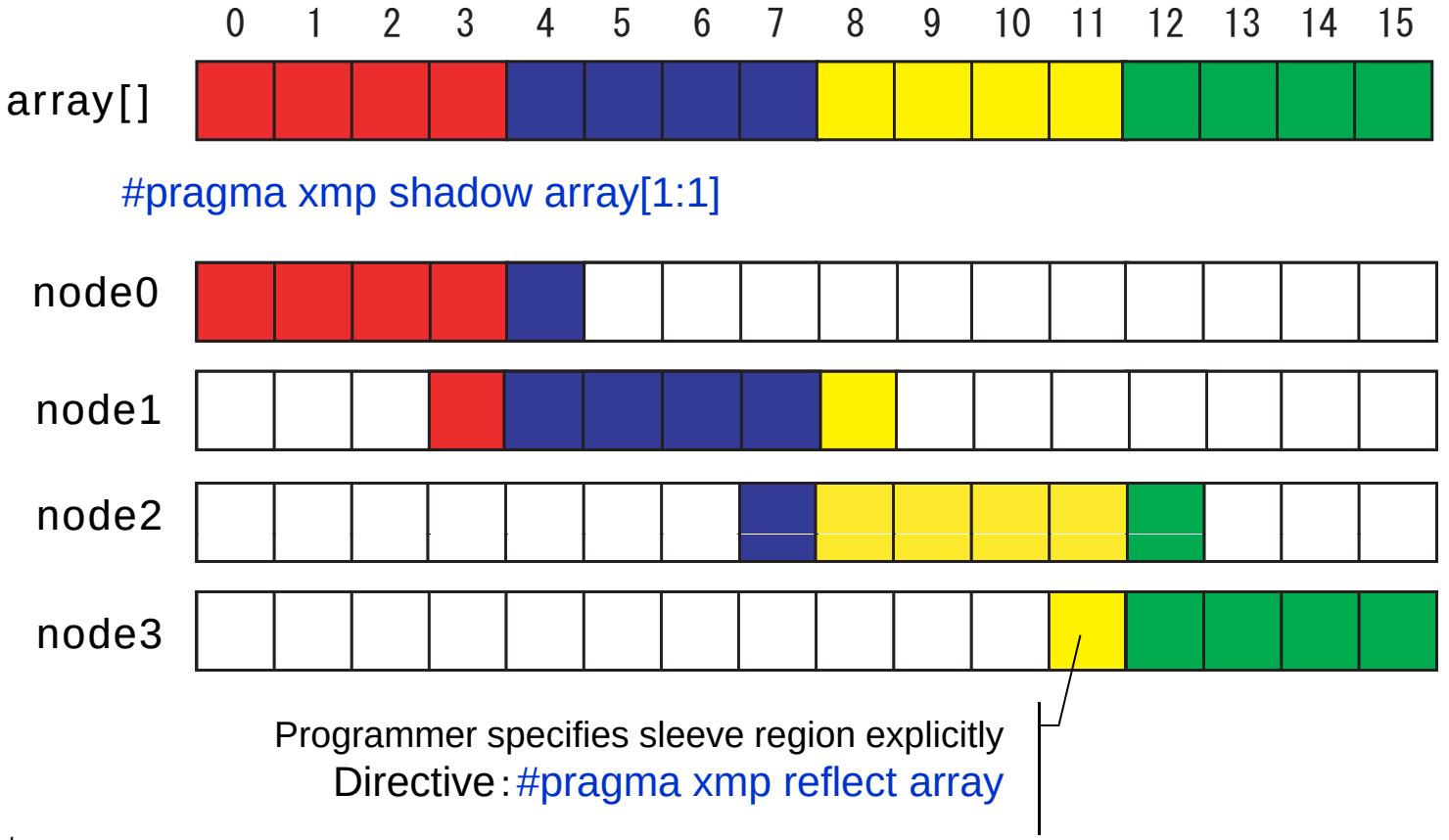
```
#pragma xmp loop on t(i)
```



Data synchronization of array (shadow)

- Exchange data only on “shadow” (sleeve) region
  - If neighbor data is required to communicate, then only sleeve area can be considered.
  - example：b[i] = array[i-1] + array[i+1]

#pragma xmp align array[i] with t(i)



XcalableMP コード例 (laplace, global view)

```
#pragma xmp nodes p[NPROCS]
#pragma xmp template t[1:N]
#pragma xmp distribute t[block] on p

double u[XSIZE+2][YSIZE+2],
       uu[XSIZE+2][YSIZE+2];
#pragma xmp aligned u[i][*] to t[i]
#pragma xmp aligned uu[i][*] to t[i]
#pragma xmp shadow uu[1:1]

lap_main()
{
 int x,y,k;
 double sum;
    …
```

ノードの形状の定義

Templateの定義とデータ分散を定義

データの分散は、templateにalign
データの同期のためのshadowを定義、この場合はshadowは袖領域

Work sharing
ループの分散

データの同期

```
for(k = 0; k < NITER; k++){
         /* old <- new */
#pragma xmp loop on t[x]
         for(x = 1; x <= XSIZE; x++)
           for(y = 1; y <= YSIZE; y++)
             uu[x][y] = u[x][y];
#pragma xmp reflect uu
#pragma xmp loop on t[x]
         for(x = 1; x <= XSIZE; x++)
           for(y = 1; y <= YSIZE; y++)
             u[x][y] = (uu[x-1][y] + uu[x+1][y
                  uu[x][y-1] + uu[x][y+1])/4.0
    }
    /* check sum */
    sum = 0.0;
#pragma xmp loop on t[x] reduction(+:sum)
    for(x = 1; x <= XSIZE; x++)
         for(y = 1; y <= YSIZE; y++)
           sum += (uu[x][y]-u[x][y]);
#pragma xmp block on master
    printf("sum = %g¥n",sum);
}
```

XcalableMP Global view directives

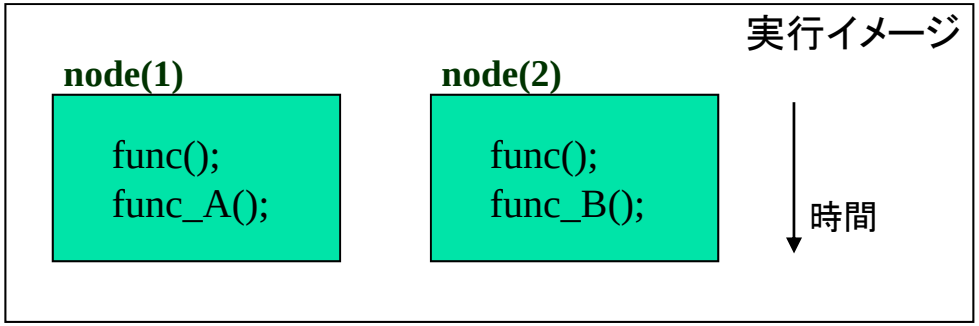
- Execution only master node
  - #pragma xmp block on master
- Broadcast from master node
  - #pragma xmp bcast (*var*)
- Barrier/Reduction
  - #pragma xmp reduction (*op: var*)
  - #pragma xmp barrier
- Global data move directives for collective comm./get/put
- Task parallelism
  - #pragma xmp task on *node-set*

タスクの並列実行

- **#pragma xmp task on *node***
  - 直後のブロック文を実行するノードを指定

例）

```
func();
#pragma xmp tasks
{
#pragma xmp task on node(1)
   func_A();
#pragma xmp task on node(2)
   func_B();
}
```

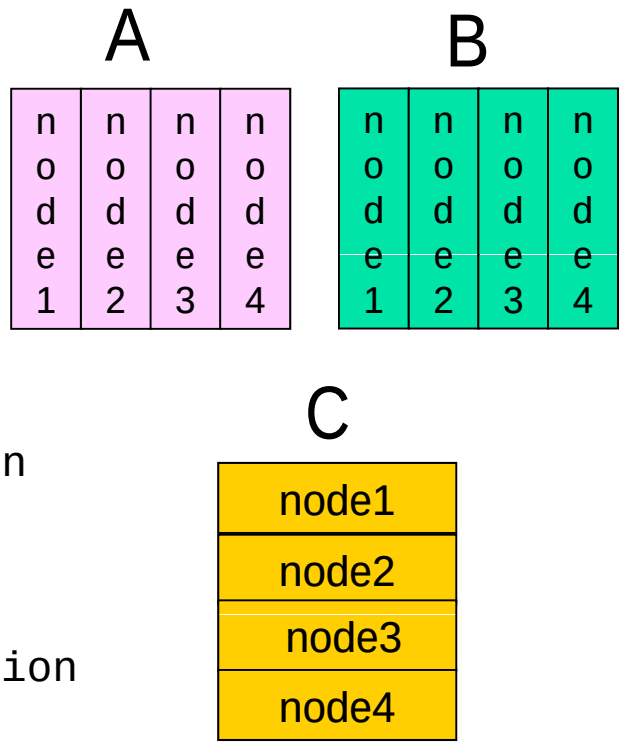


  - 異なるノードで実行することでタスク並列化を実現

gmove directive

- The "gmove" construct copies data of distributed arrays in global-view.
  - When no option is specified, the copy operation is performed *collectively* by all nodes in the executing node set.
  - If an "in" or "out" clause is specified, the copy operation should be done by one-side communication ("get" and "put") for remote memory access.

```
!$xmp nodes p(*)
!$xmp template t(N)
!$xmp distribute t(block) to p
real A(N,N),B(N,N),C(N,N)
!$xmp align A(i,*), B(i,*),C(*,i) with t(i)

    A(1) = B(20)       // it may cause error
!$xmp gmove
    A(1:N-2,:) = B(2:N-1,:) // shift operation
!$xmp gmove
    C(:,:) = A(:,:)      //  all-to-all
!$xmp gmove out
    X(1:10) = B(1:10,1) // done by put operation
```



XcalableMP Local view directives

- XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as "**local view**" programming.
  - The basic execution model of XcalableMP is SPMD
    - Each node executes the program independently on local data if no directive
  - We adopt Co-Array as our PGAS feature.
  - In C language, we propose array section construct.
  - Can be useful to optimize the communication

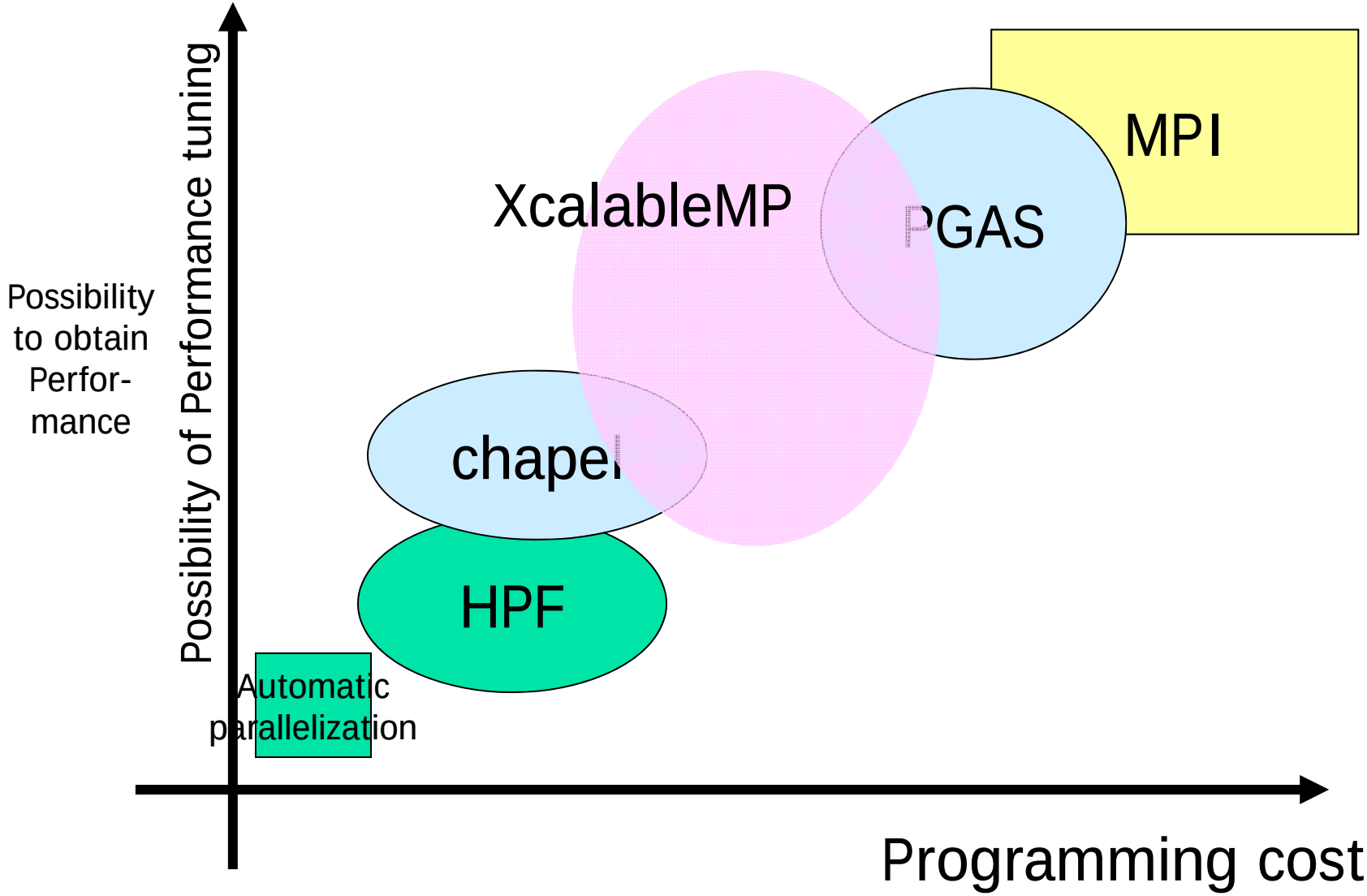
- Support alias Global view to Local view

Array section in C

```
int A[10]:
int B[5];

A[5:9] = B[0:4];
```

```
int A[10], B[10];
#pragma xmp coarray [*]: A, B
…
A[:] = B[:]:[10]; // broadcast
```

Target area of XcalableMP



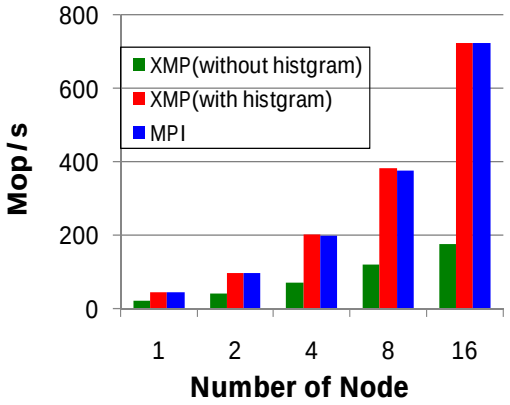
Status of XcalableMP

- Status of XcalableMP WG
  - Discussion in monthly Meetings and ML
  - XMP Spec Version 0.7 is available at XMP site.
  - XMP-IO and multicore extension are under discussion.
- Compiler & tools
  - XMP prototype compiler (xmpcc version 0.5) for C is available from U. of Tsukuba.
  - Open-source, C to C source compiler with the runtime using MPI
  - XMP for Fortran 90 is under development.
- Codes and Benchmarks
  - NPB/XMP, HPCC benchmarks, Jacobi ..
  - Honorable Mention in SC10/SC09 HPCC Class2
- Platforms supported
  - Linux Cluster, Cray XT5 ...
  - Any systems running MPI
    - The current runtime system designed on top of MPI

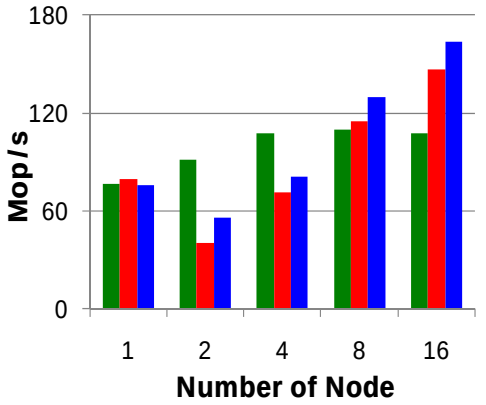
NPB IS performance

- Coarray is used
- Performance comparable to MPI

T2K Tsukuba System



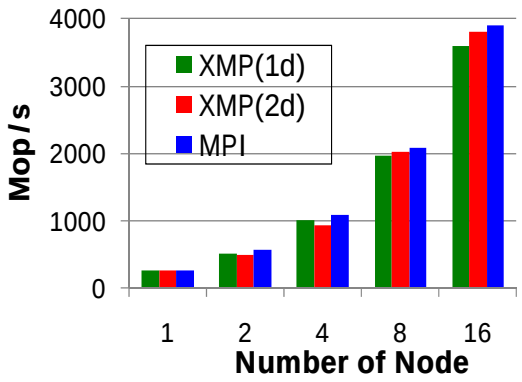
PC Cluster



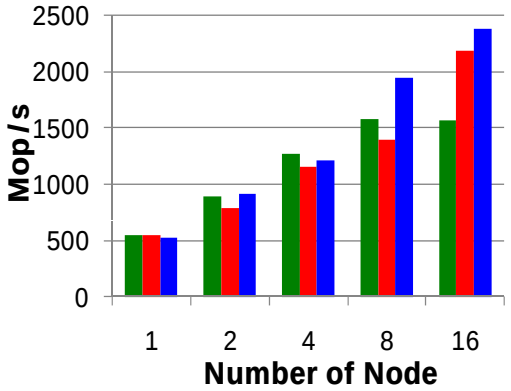
NPB CG performace

- Two-dimensional Parallelization
- Performance comparable to MPI

T2K Tsukuba System



PC Cluster



Agenda of XcalableMP

- Interface to existing (MPI) libraries
  - How to use high-performance lib written in MPI
- Extension for multicore
  - Mixed with OpenMP
  - Autoscoping
- XMP IO
  - Interface to MPI-IO
- Extension for GPU...

マルチコア対応

- 現状
  - ほとんどのクラスタがいまや、マルチコアノード（SMPノード）
  - 小規模では格コアにMPIを走らせるflat MPIでいいが、大規模ではMPI数を減らすためにOpenMPとのハイブリッドになっている。
    - ハイブリッドにすると（時には）性能向上も。メモリ節約も。
  - しかし、ハイブリッドはプログラミングのコストが高い。

- ２つの方法
  - OpenMPをexplicitに混ぜて書く方法
  - loop directiveから、implicitにマルチスレッド・コード(OpenMP)を出す方法　→　explicitに書くことになった

マルチコア対応

- loop directiveから、implicitにマルチスレッド・コード(OpenMP)を出す方法
  - loop directiveは基本的に、並列ループ（つまり、各iterationは並列に実行できる）
  - では、ノードの中でも並列に実行できるはず。
  - 問題となるケース

```
#pragma xmp loop (i) on ...
for( ... i ... ){
        x +=
        t = ...
        A(i) = t +1;
}
```

却下！
賛同を得られず！！

これをノード内で
実行すると、xとかtが
raceする。

マルチコア対応

- デフォールトでは、シングルスレッドで実行
- マルチスレッド実行する場合は、thread(=スレッド数)を指示
  - OpenMPでいろいろなものを指定するのは面倒なので、auto scopingも検討

```
#pragma xmp loop (i) on ...
for( ... i ...){
#pragma omp for
   for( ... j ...){
   ....
   }
}
```

```
#pragma xmp loop (i) on ... threads ... openmpの指示行
for( ... i ...){
    ....
}
```

XMP IO

- Design
  - Provide efficient IO for global distributed arrays directly from lang.
  - Mapping to MPI-IO for efficiency
  - Provide compatible IO mode for sequential program exec.

- IO modes
  - (native local IO)
  - Global collective IO (for global distributed arrays)
  - Global atomic IO
  - Single IO to compatible IO to seq. exec

XMP IO functions in C

- Open & close
  - xmp_file_t *xmp_all_fopen(const char *fname, int amode)
  - int xmp_all_fclose(xmp_file_t *fp)
- Independent global IO
  - size_t xmp_fread(void *buffer, size_t size, size_t count, xmp_file_t *fp)
  - size_t xmp_fwrite(void *buffer, size_t size, size_t count, xmp_file_t *fp);
- Shared global IO
  - size_t xmp_fread_shared(void *buffer, size_t size, size_t count, xmp_file_t *fp)
  - size_t xmp_fwrite_shared(void *buffer, size_t size, size_t count, xmp_file_t *fp);
- Global IO
  - size_t xmp_all_fread(void *buffer, size_t size, size_t count, xmp_file_t *fp)
  - size_t xmp_all_fwrite(void *buffer, size_t size, size_t count, xmp_file_t *fp);
  - int xmp_all_fread_array(xmp_file_t *fp, xmp_array_t *ap, xmp_range_t *rp, xmp_io_info *ip)
  - size_t xmp_all_fwrite_array(xmp_file_t *fp, xmp_array_t *ap, xmp_range_t *rp, xmp_io_info *ip)
  - Xmp_array_t is a type of global distributed array descriptor
- Need “set_view”?

Fortran IO statements for XMP-IO

- Signle IO
  - !$xmp io single
    open(10, file=…)
  - !$xmp io single
    read(10,999) a,b,c
    999 format(…)
  - !$xmp io single
    backspace 10

- C1. Collective IO
  - !$xmp io collective
    open(11, file=…)
  - !$xmp io collective
    read(11) a,b,c

- C1 Atomic IO
  - !$xmp io atomic
    open(12, file=…)
  - !$xmp io atomic
    read(12) a,b,c

注意： これは暫定版の仕様です。

並列ライブラリインタフェース

- すべてをXMPで書くことは現実的ではない。他のプログラミングモデルとのインタフェースが重要

- MPIをXMPから呼び出すインタフェース
- (MPIからXMPを呼び出すインタフェース)

- XMPから、MPIで記述された並列ライブラリを呼び出す方法
  - 現在、Scalapackを検討中
  - XMPの分散配列記述から、Scalapackのディスクリプタを作る
  - XMPで配列を設定、ライブラリを呼び出す
  - その場合、直によびだすか、wrapperをつくるか。

GPU/Manycore extension

- 別のメモリを持つ演算加速装置が対象
  - メモリをどのように扱うかが問題
  - 並列演算はOpenMP等でも行ける
- Device指示文
  - Offloadする部分を指定
  - ほぼ同じ指示文を指定できる（但し、どの程度のことができるかはそのdeviceによる）
  - GPU間の直接の通信を記述ができる。
- Gmove指示文で、GPU/host間のデータ通信を記述

```
#pragma xmp nodes p(10)
#pragma xmp template t(100)
#pragma xmp distribute t(block) on p

double A[100];
double G_A[100];
#pragma xmp align to t: A, G_A
#pragma device(gpu) allocate(G_A)
#pragma shadow G_A[1:1]

#pragma xmp gmove out
    G_A[:] = A[:]       // host->GPU

#pragma xmp deivce(gpu1)
{
#pragma xmp loop on t(i)
   for(...) G_A[i] = ...
#pragma xmp reflect G_A
}

#pragma xmp gmove in
   A[:] = G_A[:]        // GPU->host
```

他にも

- Performance tools interface
- Fault resilience / Fault tolerance
- ...

おわりに

- XMPを使うメリットは？
  - プログラムが（MPIと比べて）論理的に、簡単かける（はず）
  - 既存の言語C, Fortranから使える
  - Multi-node GPUに対応
  - マルチコア化が進むと、MPI-OpenMPは限界がある（とおもう）
- XMPは主流になれるのか？
  - 少なくとも、PGASはこの数年のトレンド。XMPは、CAFをサブセットとして含んでいる
  - HPFの経験がある（はず） HPFではある程度のプログラムはかけていた（はず）
  - GPUについては、わからない
  - すくなくとも、５年は開発・保守を続ける（つもり）
  - ポイントは、メーカーがついてくるか。現在のところ、富士通とCray…
- お願い
  - XMP/Fortranを鋭意、開発中。９月までには…
  - XMP/Cは一応使えているので、使ってみてください。
  - もちろん、京でも使えるようにします。