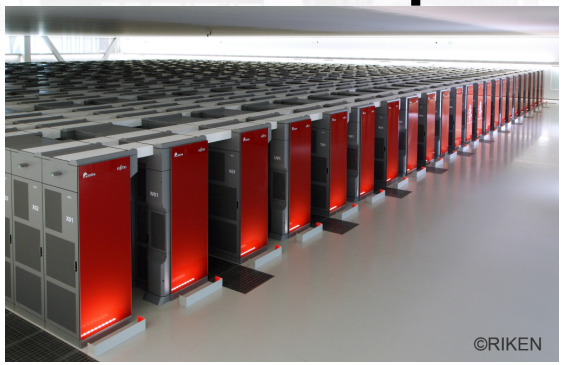
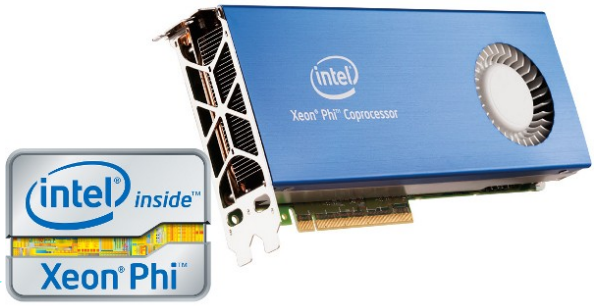
Parallel Programming for HPC in two decades

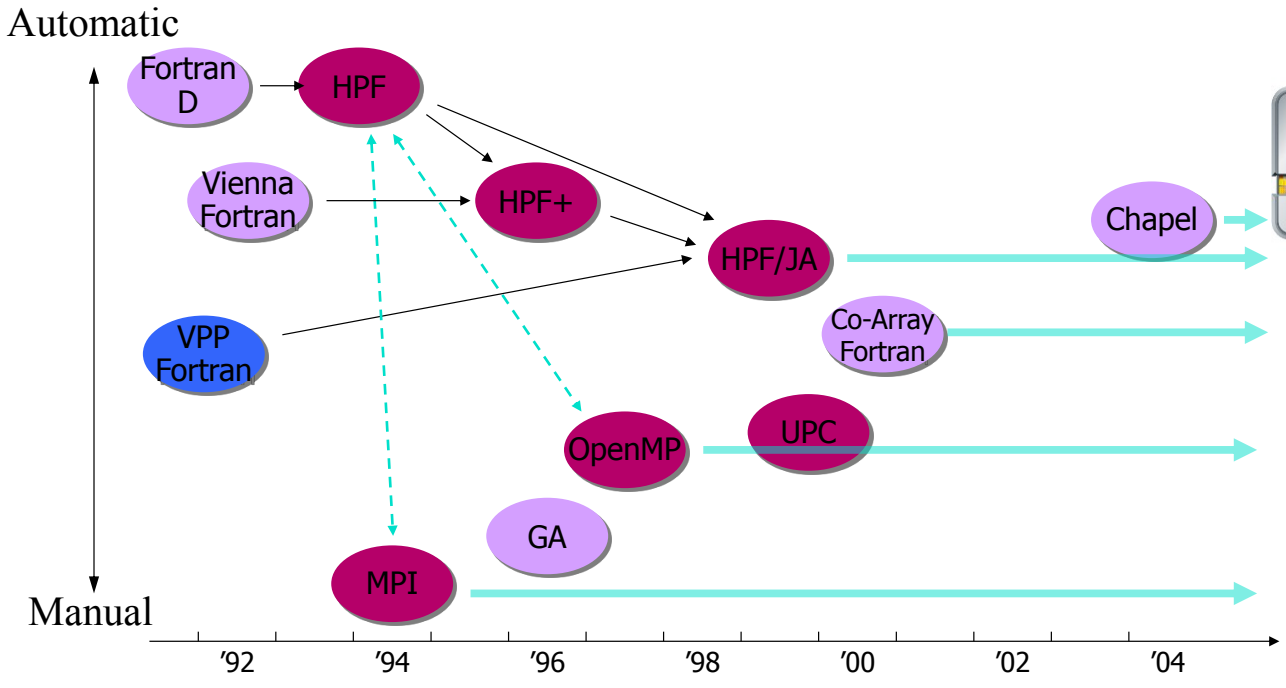
- (80's) message-passing libraries of MPP venders
- 1989: Initial release of PVM (Parallel Virtual Machine)
- 1993: HPF first report
- 1994: MPI 1.0 released
- (1995: Beowulf Cluster, and RWCP PC Cluster)
- 1997: OpenMP 1.0
- 2009: MPI 2.1

The K computer

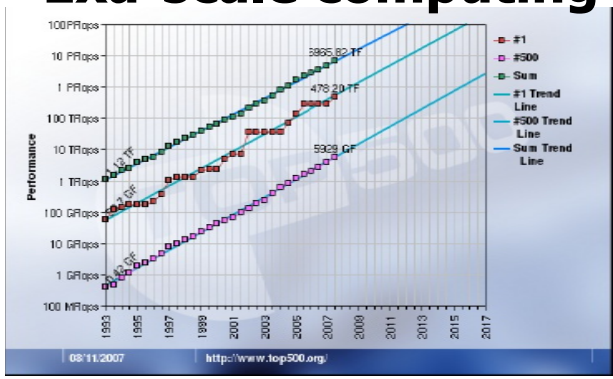


Manycore (Intel Xeon Phi)



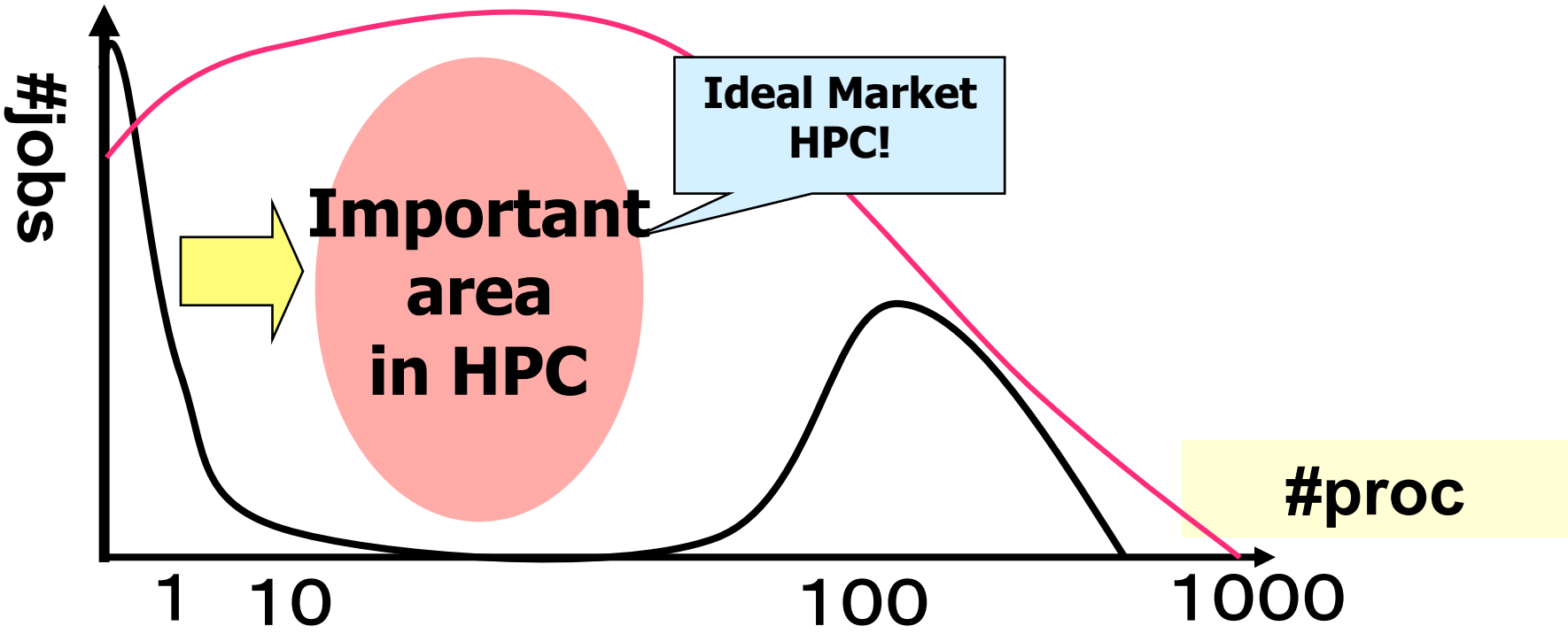


Exa-scale computing



“Blue Collar” Computing (in SC 2004)

- In SC2004, Stan Ahalt @ OSC
  - It is important to increase users in the middle class for more boarder HPC.
  - “The two major barriers for companies wanting to use High Performance Computing (HPC) to solve complex problems are the cost of implementation and the difficulty of installing, maintaining, programming and using HPC systems”
  - Increase Productivity! – Education, … and Programming

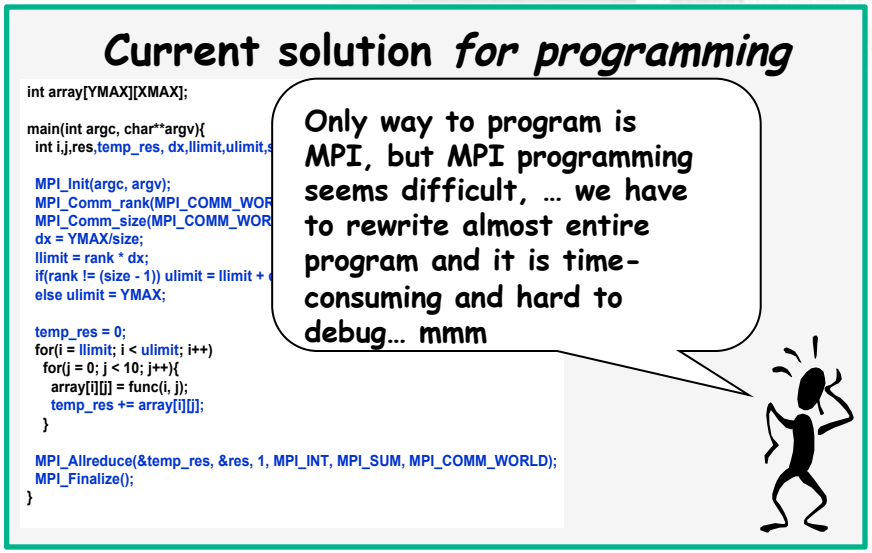


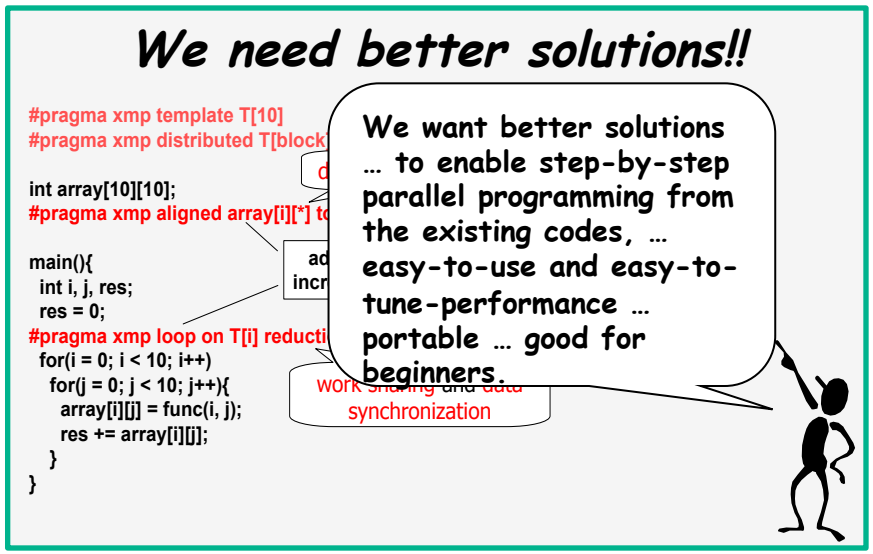
Why do we need parallel programming language researches?

- In 90's, many programming languages were proposed.
  - but, none of them has prevailed.
- MPI is dominant programming in a distributed memory system
  - low productivity and high cost
- No standard parallel programming language for HPC
  - only MPI
  - PGAS is now emerging, …

XcalableMP

is our solution!





History of HPC language projects in Japan

- VPPFortran for NWT (VPP500)
  - NWT(Numerical Wind Tunnel), a parallel Vector machine for CFD, 1st machine in Top500 (1993/Nov to 1995/Nov)
  - Fortran extensions for NWT, specifying global and local memory dedicated to VPP, proposed by Fujitsu
  - Renamed to XPFortran as a Fujistu product

Dr. Miyoshi

- HPF for Earth Simulator (SX-6)
  - ES, 1st machine in Top500 (2002-2004/June)
  - NEC has been supporting HPF for Earth Simulator System.
  - Japan HPF promotion consortium was organized by NEC, Hitatchi, Fujitsu …
  - Activities and many workshops: HPF Users Group Meeting (HUG from 1996-2000), HFP intl. workshop (in Japan, 2002 and 2005)

VPP500

HPF experience with IMPACT-3D

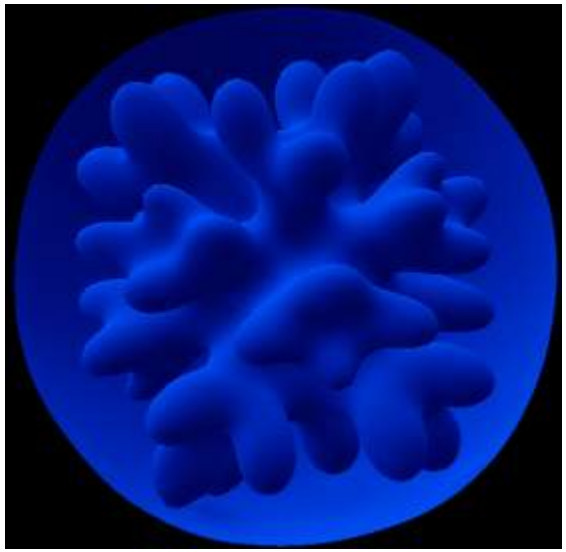
- IMPACT-3D: an implosion analysis code using TVD scheme
  - three-dimensional compressible and inviscid Eulerian fluid computation with the explicit 5-point stencil scheme for spatial differentiation
  - fractional time step for time integration.

- Gordon Bell winners of SC 2002
  - For achieving 14.9 TFLOPS on

  the Earth Simluator System with

  the IMPACT-3D code,

  written in High Performance Fortran (HPF)

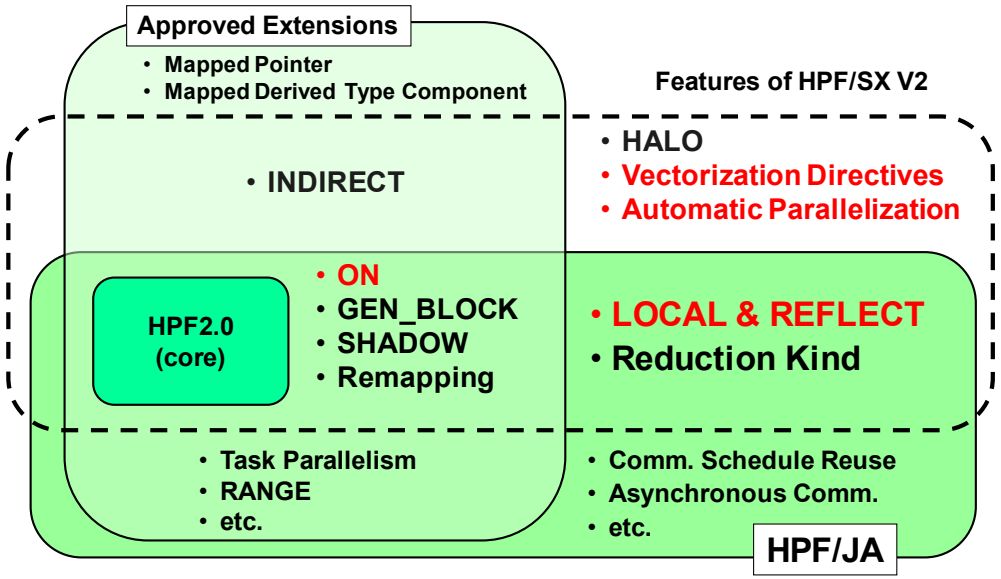


Parallelization of IMPACT-3D using HPF (ext)

- Parallelization only by DISTRIBUTE and SHADOW
  - Block distribution on the last (third) dimension of each arrays
  - Add shadow on the third dimension
- All loops are parallelized by the HPF/ES compiler

- 12.5TFLOPS
  (efficiency 38%)
  by 512 node(4096CPU)
  with mesh-size
  2048x2048x4096

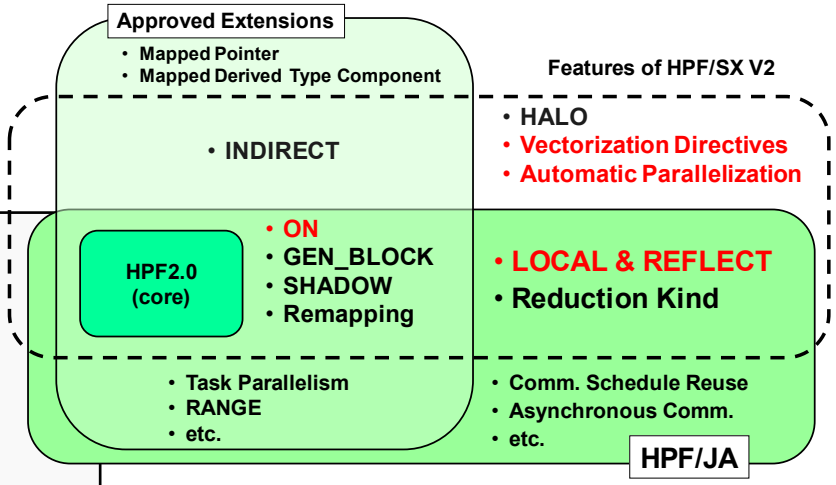
※ **MPI15.3TFLOPS**




```
!HPF$ distribute (*,*,block) ::
!HPF$&          sr,se,sm,sp,sn,sl,
!HPF$&          walfa1,walfa2,walfa3,walfa4,walfa5,
!HPF$&          wnue1,wnue2,wnue3,wnue4,wnue5,
 ...
!HPF$ shadow (0,0,0:1) ::
!HPF$&          sr,se,sm,sp,sn,sl,
!HPF$&          wg1,wg2,wg3,wg4,wg5,
!HPF$&          wtmp1,wtmp2,wtmp3
```

Optimization by HPF/JA extensions

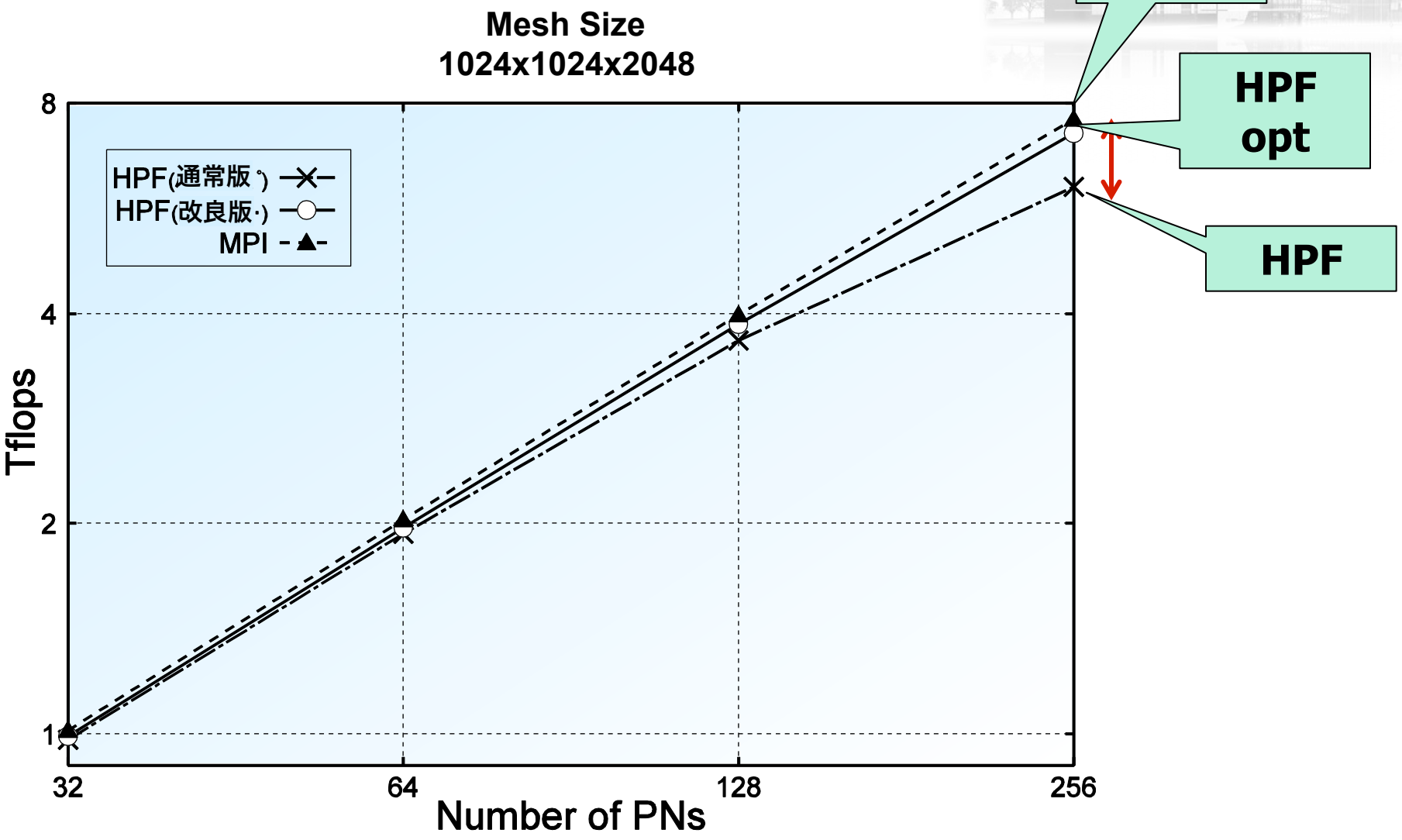
- Optimize communication by **`REFLECT`** and **`LOCAL`**
  - **`REFLECT`** explicitly updates SHADOW, with re-use of communication schedule
  - The LOCAL directive guarantees the accesses to arrays in a list do not require inter-processor communications.
  - The user can eliminate redundant communications for the shadow area by the combined use of the REFLECT and LOCAL directives.
- 14.9TFLOPS (efficiency 45%) by 512 node(4096CPU) with mesh-size **`2048x2048x4096`**




```
!HPFJ reflect sr, sm, sp, se, sn, sl

      do iz = 1, lz-1
!HPF$ on home( sm(:,:,iz) ), local begin
      do iy = 1, ly
      do ix = 1, lx
         wu0 = sm(ix,iy,iz  ) / sr(ix,iy,iz  )
         wu1 = sm(ix,iy,iz+1) / sr(ix,iy,iz+1)
         wv0 = sn(ix,iy,iz  ) / sr(ix,iy,iz  )
         ...
```

Scalability of IMPACT-3D in ES



Lessons learned from HPF

- Pros: Easy to programming
  - A user gives a small information such as data distribution and parallelism.
  - The compiler is expected to generate "good" communication and work-sharing automatically.

- Cons: No explicit means 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 information
    - INDEPENDENT for parallel loop
    - SHADOW & RELECT, ON HOME, LOCAL, …
  - The way for tuning performance should be provided.

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

"The Rise and Fall of High Performance Fortran ..." by Kennedy, Koelbel and Zima [HOPL 2007]

- A very highly suggestive literature for language projects

- We would focus on this point:

> **The difficulty was that there were only limited ways for a user to exercise fine-grained control over the code generated once the source of performance bottlenecks was identified, … The HPF/JA extensions ameliorated this a bit by providing more control over locality. However, it is clear that additional features are needed in the language design to override the compiler actions where that is necessary. Otherwise, the user is relegated to solving a complicated inverse problem in which he or she makes small changes to the distribution and loop structure in hopes of tricking the compiler into doing what is needed.**

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!)

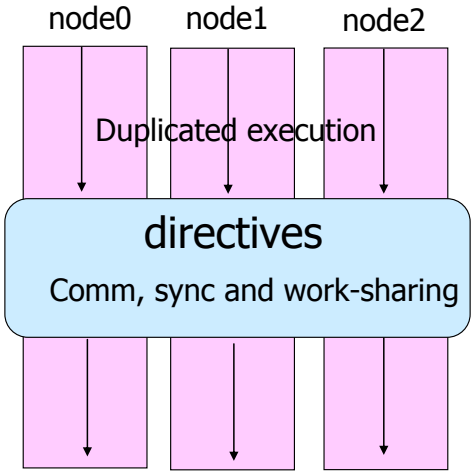
- A prototype XMP compiler is being developed by U. of Tsukuba and RIKEN AICS.
- XMP is proposed for a programming language for the K computer, supported by the programming environment research team.

XcalableMP

http://www.xcalablemp.org

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

- A PGAS language. Directive-based language extensions for Fortran95 and C99 for the XMP PGAS model
  - To reduce the cost of code-rewriting and education
- Global view programming with global-view distributed data structures for data parallelism
  - A set of threads are started as a logical task. Work mapping constructs are used to map works and iteration with affinity to data explicitly.
  - Rich communication and sync directives such as "gmove" and "shadow".
  - Many concepts are inherited from HPF
- Co-array feature of CAF is adopted as a part of the language spec for local view programming (also defined in C).




```
int array[N];
#pragma xmp nodes p(4)
#pragma xmp template t(N)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][ with t(i)

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

Code Example

```
program xmp_example
Integer array(XMAX,MAX)
!$XMP nodes p(4)
!$XMP template t(YMAX)
!$XMP distribute t(block) onto p
!$XMP align array(*,j) with t(j)

  integer :: i, j, res
  res = 0

!$XMP loop on t(j)  reduction(+:res)
  do j = 1, 10
    do I = 1, 10
       array(I,j) = func(i, j)
       res += array(I,j)
  enddo
 enddo
```

data distribution

add to the serial code : incremental parallelization

work mapping and data synchronization

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];
    }
}
```

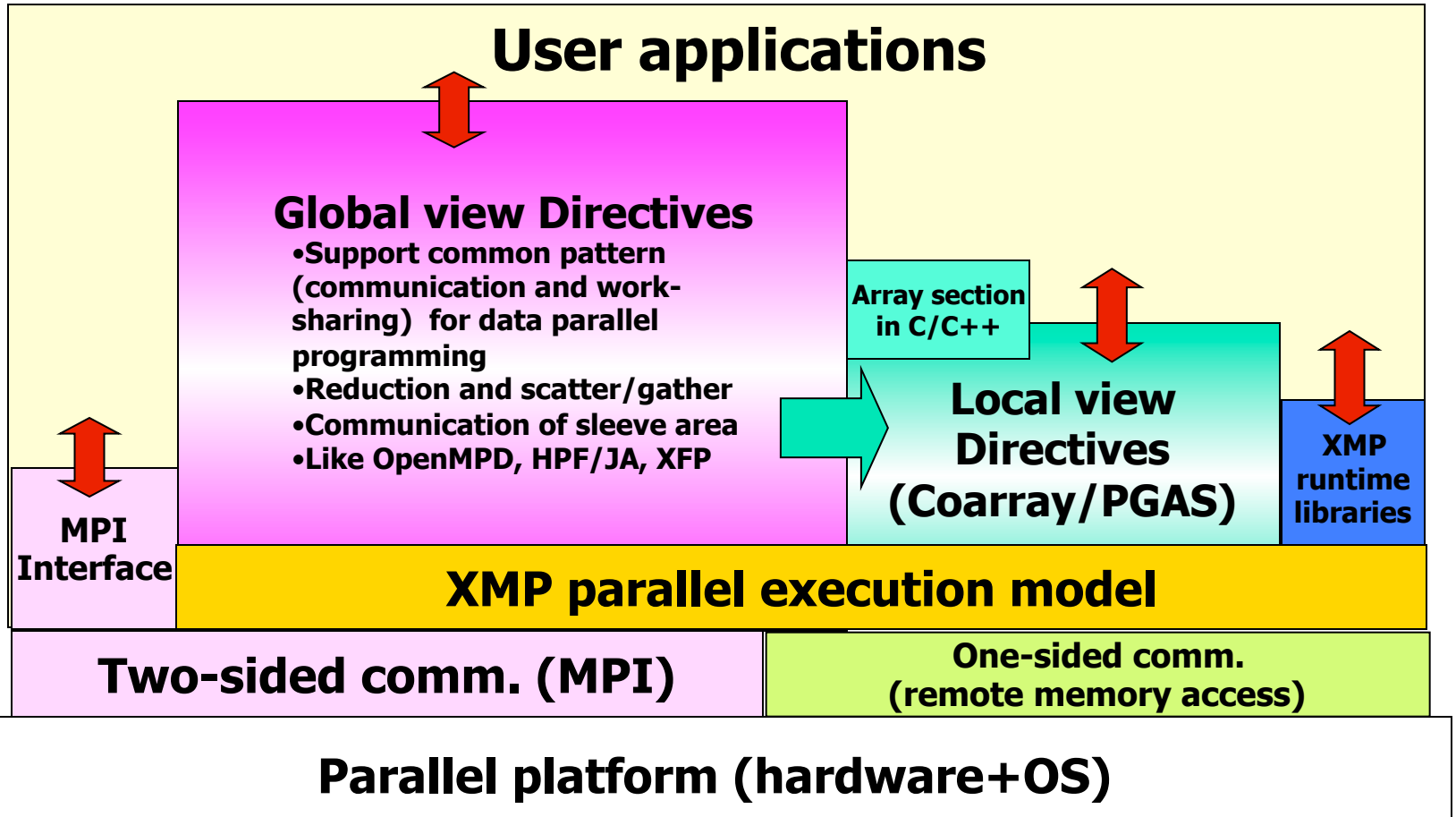
data distribution

add to the serial code : incremental parallelization

work mapping and data synchronization

Overview of XcalableMP

- XMP supports **typical data parallelization** with the description of data distribution and work mapping under "**global view**“
  - Some sequential code can be parallelized with **directives**, like OpenMP.
- XMP also includes Co-array notation of PGAS (Partitioned Global Address Space) feature as "**local view**" programming.



Nodes, templates and data/loop distributions

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

!$XMP nodes p(32)
!$XMP nodes p(*)

- Template is used as a dummy array distributed on nodes

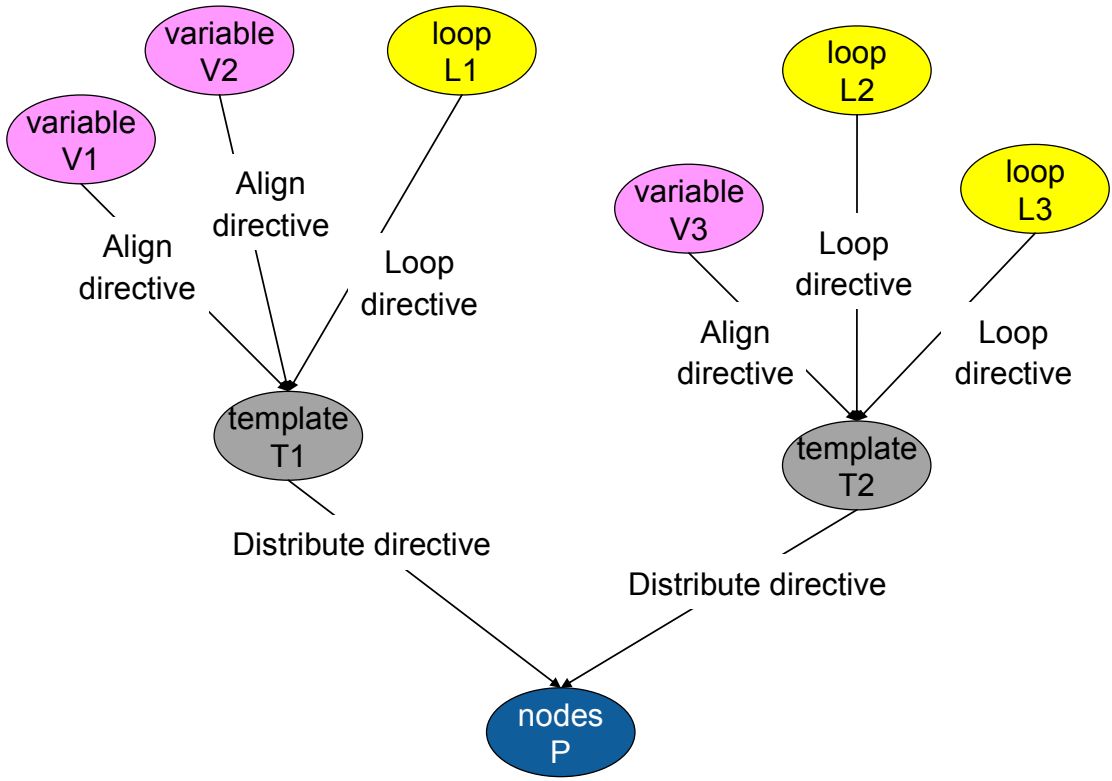
!$XMP template t(100)
!$XMP distribute t(block) onto p

- A global data is aligned to the template

!$XMP align array[i][*] with t(i)

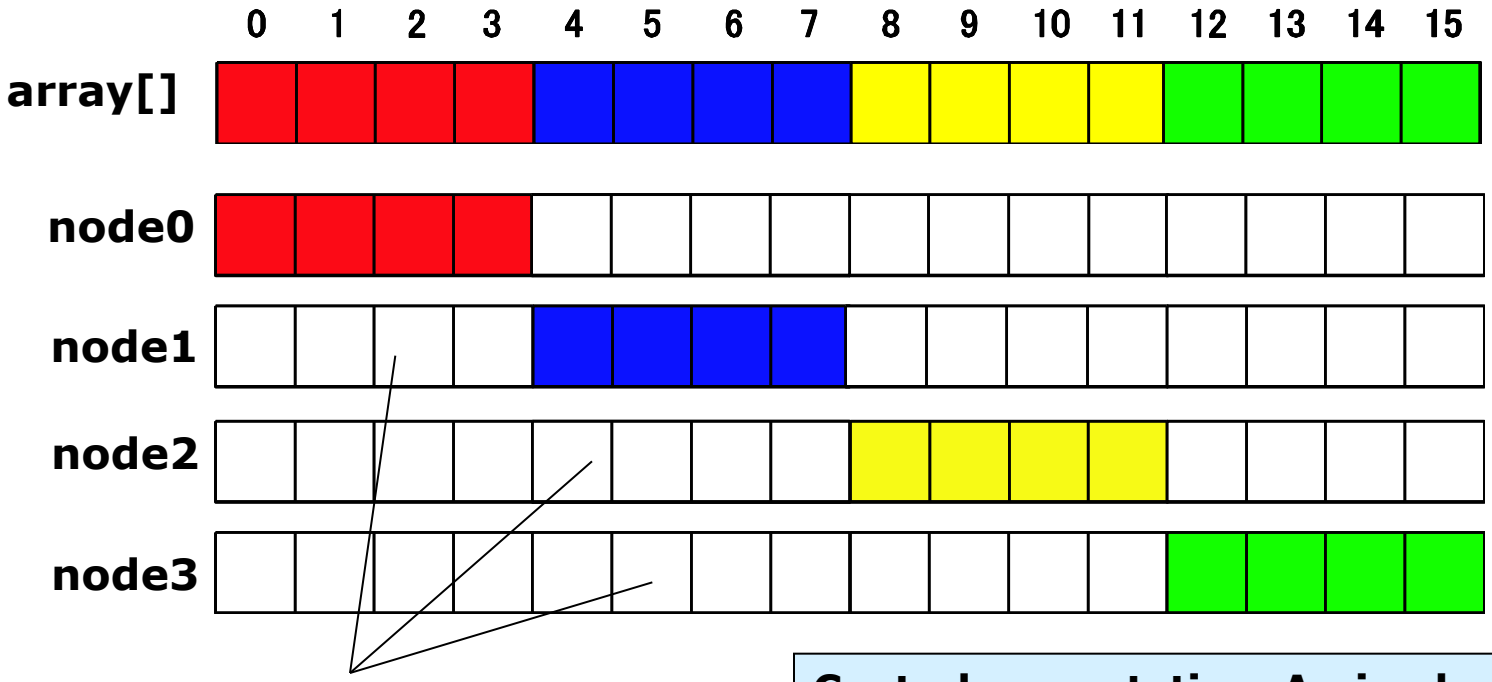
- Loop iteration must also be aligned to the template by on-clause.

!$XMP loop on t(i)



Array data distribution

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



Reference to assigned to other nodes may causes error!!

Control computation: Assign loop iteration to nodes which compute own data

This is different from HPF and UPC

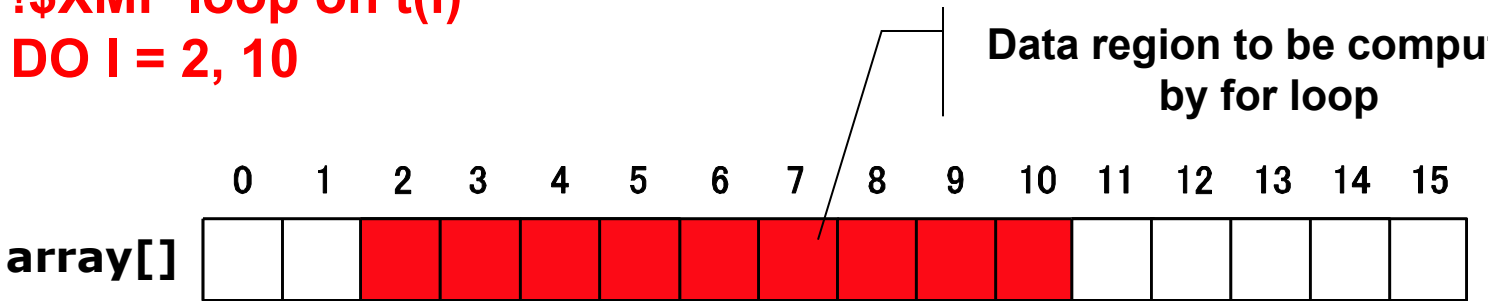
Explicit Communication between nodes

Parallel Execution of “for” loop

- Execute for loop to compute on array

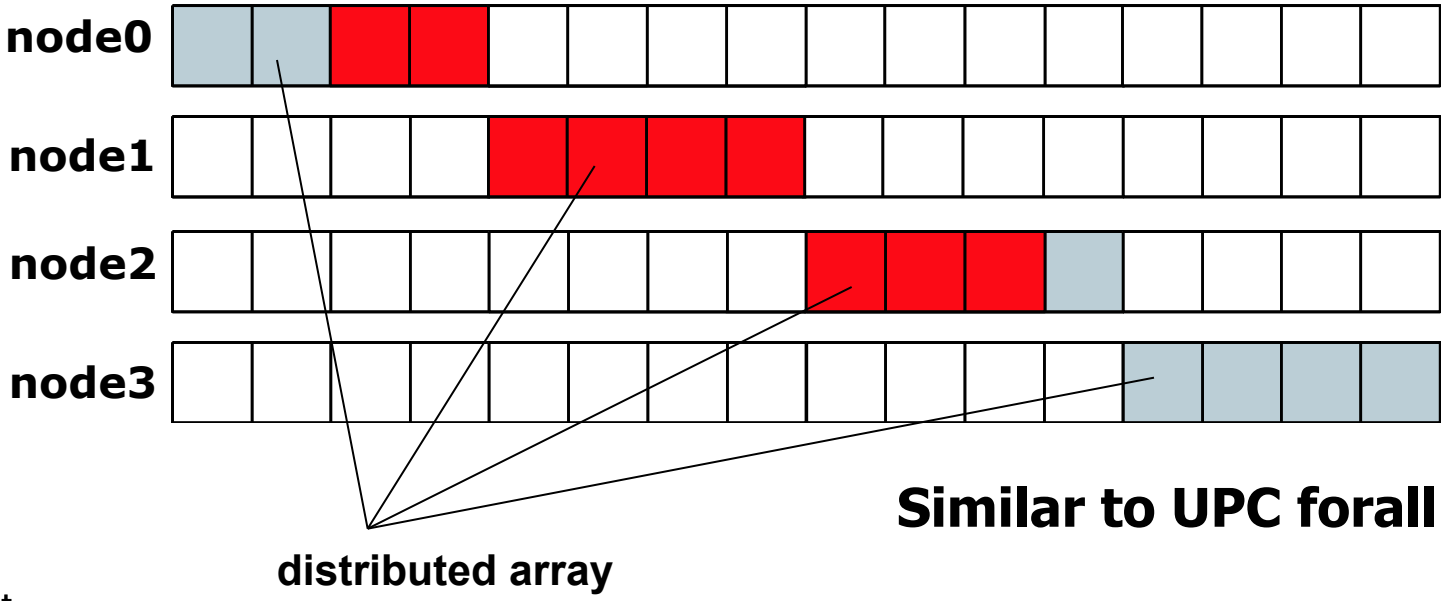
```
!$XMP nodes p(*)
!$XMP template T(0:15)
!$XMP distributed T(block) onto p
!$XMP align array(i) with T(i)
```

```
!$XMP loop on t(i)
DO I = 2, 10
```



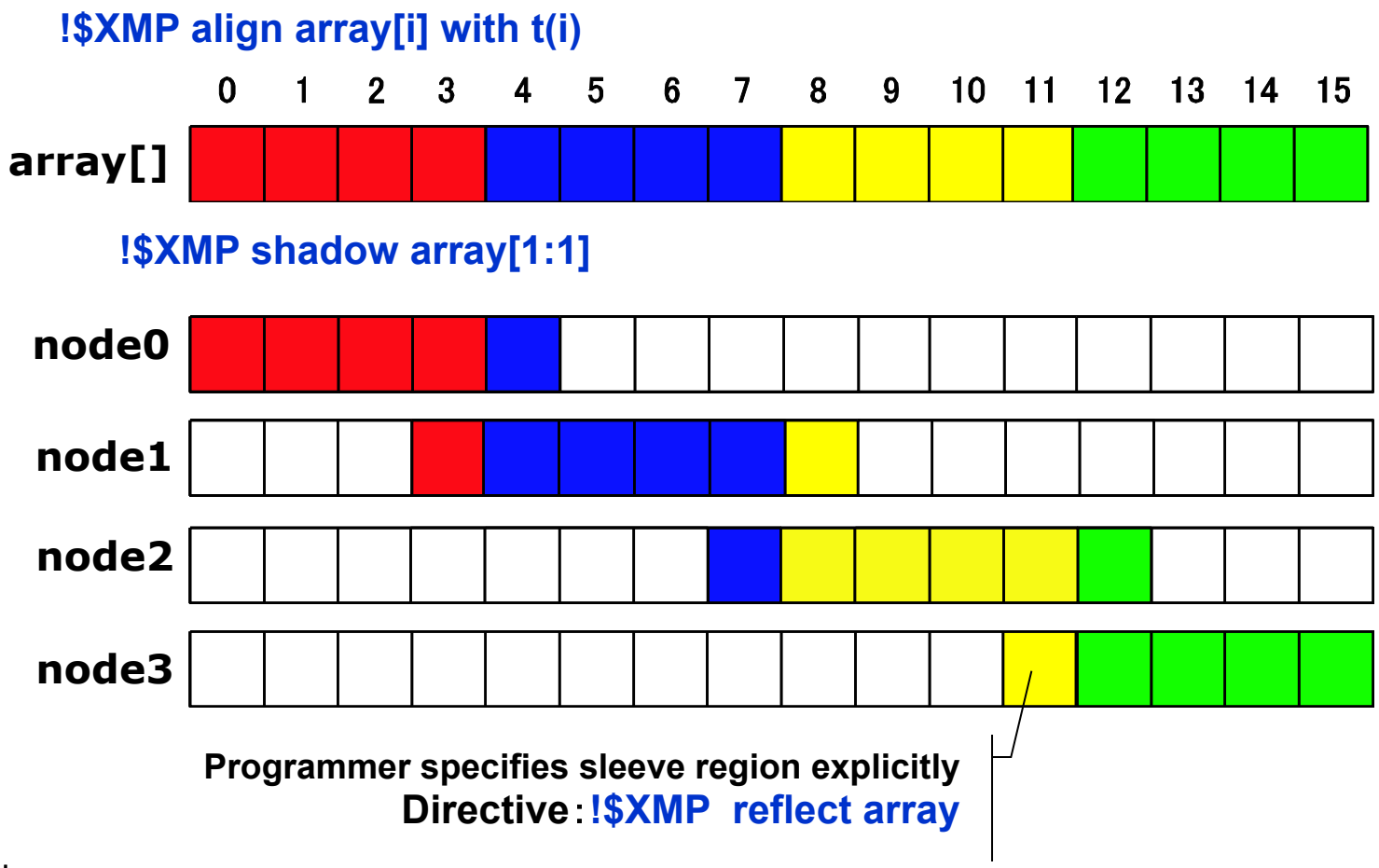
Execute “for” loop in parallel with affinity to array distribution by on-clause:

```
!XMP loop on t(i)
```



Shadow and reflect: Data synchronization of array

- 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)

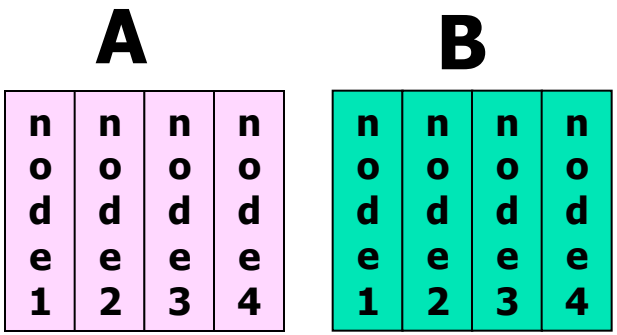


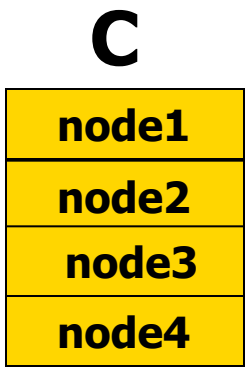
gmove directive

- 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 Global view directives

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

Co-array: XcalableMP Local view programming

- 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 (the same as Intel's)
  - Can be useful to optimize communications

- Support alias Global view to Local view

Array section in C

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

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

Co-array in C

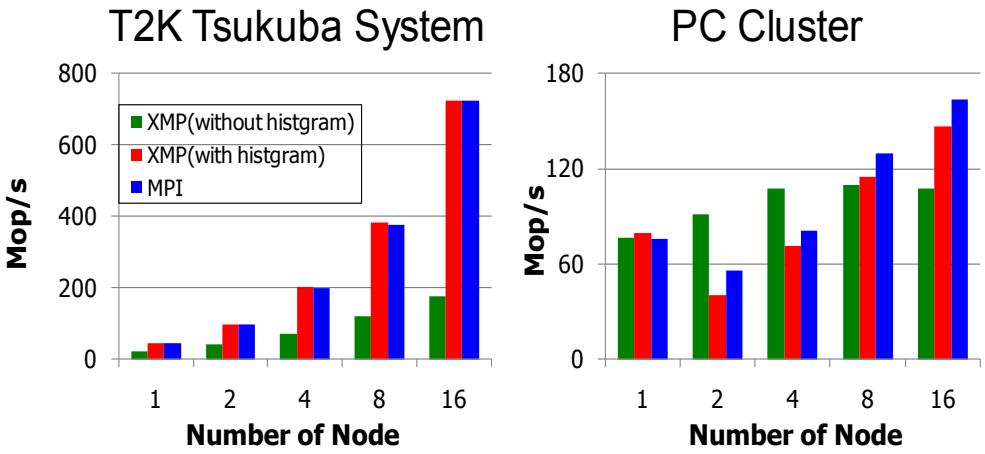
```
int A[10]:[*], B[10]:[*];
A[:] = B[:]:[10]; // broadcast
```

Status of XcalableMP

- Status of XcalableMP WG
  - Monthly Meetings and ML, supported by PC Cluster Consortium Japan.
  - XMP Spec Version 1.0 was published (at SC11). It includes XMP-IO and multicore extension as a proposal in ver 1.0.
  - Version 1.1: it will be revised at SC12
- Compiler & tools
  - XMP C prototype compiler (version 0.6, beta) for C is available.
  - XMP Fortran F95 is now in alpha release (limited).
  - Open-source, source-to-source compiler with the runtime using MPI
- Codes and Benchmarks
  - NPB/XMP, HPCC benchmarks, Jacobi …
  - Platforms supported
  - Linux Cluster, Cray XT5 … K computer
  - Any systems running MPI. The current runtime system designed on top of MPI

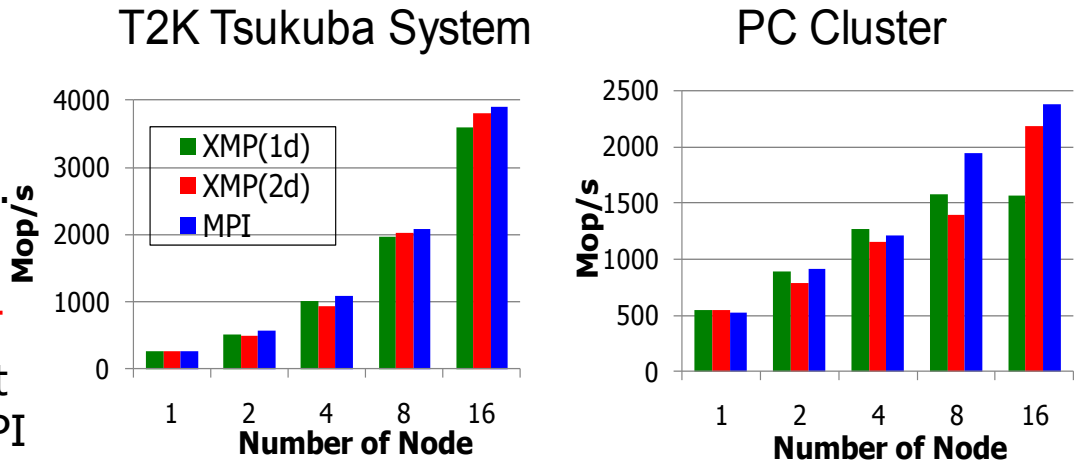
NPB IS performance

- **Coarray is used**
- **Performance comparable to MPI**



NPB CG performace

- **Two-dimensional Parallelization**
- **Performance comparable to MPI**



Experience on the K computer Parallelization of SCALEp by XMP

- What is SCALEp
  - SCALE project: (Parallel) Climate code for large eddy simulation
  - SCALEp is a kinetic core in SCALE
  - A typical stencil computation

- How to parallelize
  1. 2D block distribution of 3D array.
  2. Paralleling double nested loop by loop directives
  3. Insert reflect directives for the communication periodic neighbor elements.
     - Options: Runtime optimization using RDMA of K computer for neighbor communications

Parallelization of SCALEp by XMP

```
!$xmp nodes p(N1,N2)
!$xmp template t(IA,JA)
!$xmp distribute t(block,block) onto p

real(8) :: dens(0:KA,IA,JA)
!$xmp align (*,i,j) &
!$xmp    with t(i,j) :: dens, ...
!$xmp shadow (0,2,2) :: dens, ...

!$xmp reflect (dens(0,/periodic/2,&
!$xmp                  /periodic/2), ...)

!$xmp loop (ix,jy) on t(ix,jy)
do jy = JS, JE
  do ix = IS, IE
    do kz = KS+2, KE-2
      ... dens(kz,ix+1,jy) ...
    end do
  end do
end do
```

Declarations for Node array and template

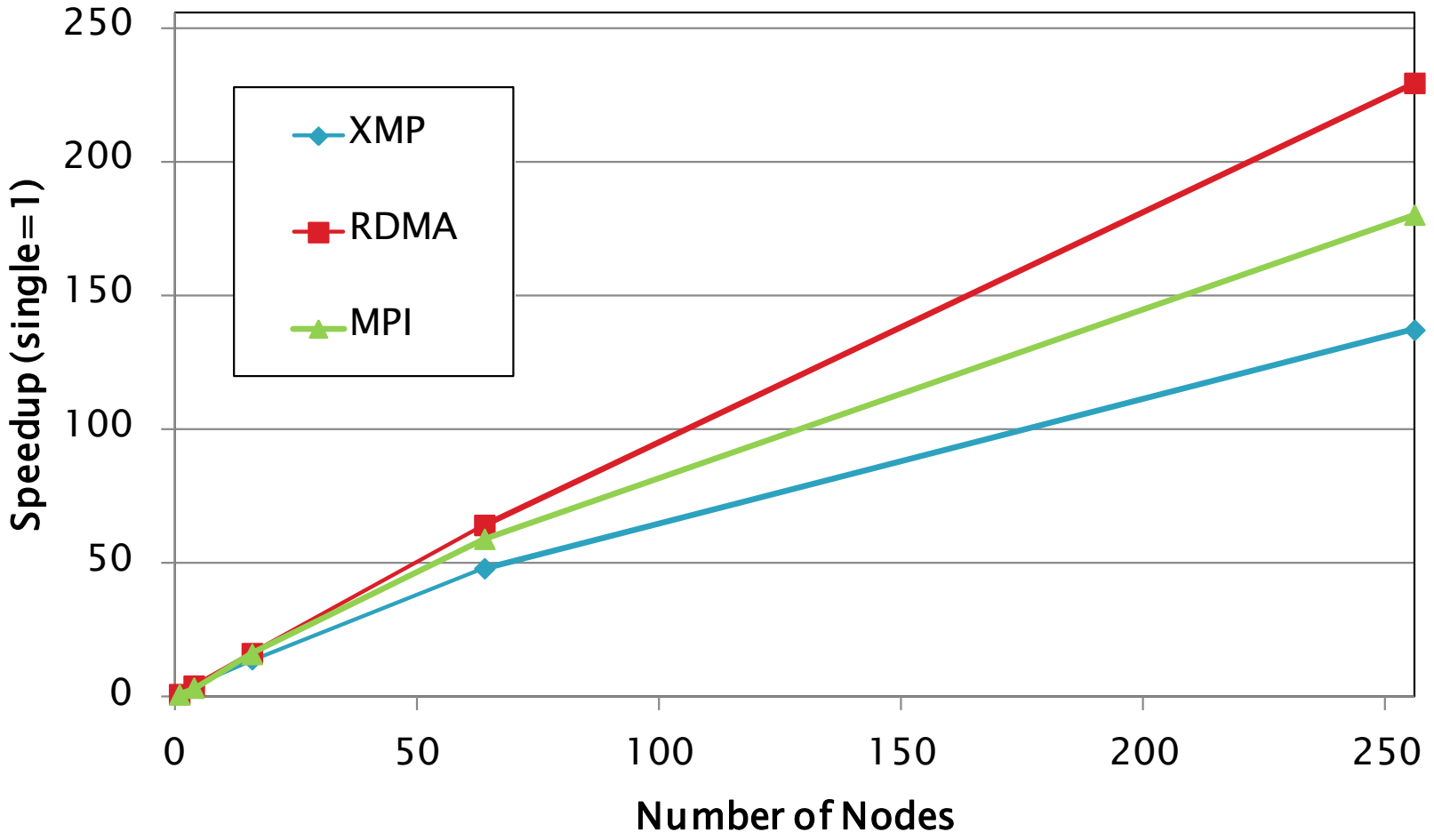
Data distribution

Neighbor comm

Loop paralization

Performance results of K computer

- Size horizontal 512x512, vertical128
- Execution time for 500 steps.
- Assign XMP node to one node. Local program is parallelized by automatic paralleling compiler by Fujitsu.



Research Agenda of XcalableMP for the K computer

- Exploiting network topology
  - It is found that the layout of nodes is very important to optimize communications in Tofu network (3D-torus)
  - Use node directive to describe the network topology.
- Optimization for one-sided communication
  - Design of one-sided communication layer using "Tofu" native library
- Exploiting Multi-node task group for multi-physics/multi-domain problems
  - XMP can define "nodes groups"
- Extensions for multicore
  - The K computer is a multi-core parallel system.
    - Flat-MPI can not be used any more …
    - Automatic parallelizing compiler is available, but …
  - Mixed with OpenMP
  - Autoscoping

What is different from at the time of HPF?

- Explicit message-passing using MPI still remains the dominant programming system for scalable applications (more than at the time of HPF?)
  - Many software stacks on top of MPI (Apps framework libraries, …)
- Fortran 90 is mature enough now. C (and C++) is used for HPC apps.
  - OpenMP supports both.
- Large-scale systems are more popular (BlueGene, the K-computer, …)
- Multicore and GPGPU/manycore make parallel programming more complicated.

- PGAS is emerging and getting attentions from the community
  - Model for scalable communication (than MPI?)

Research Agenda of XcalableMP

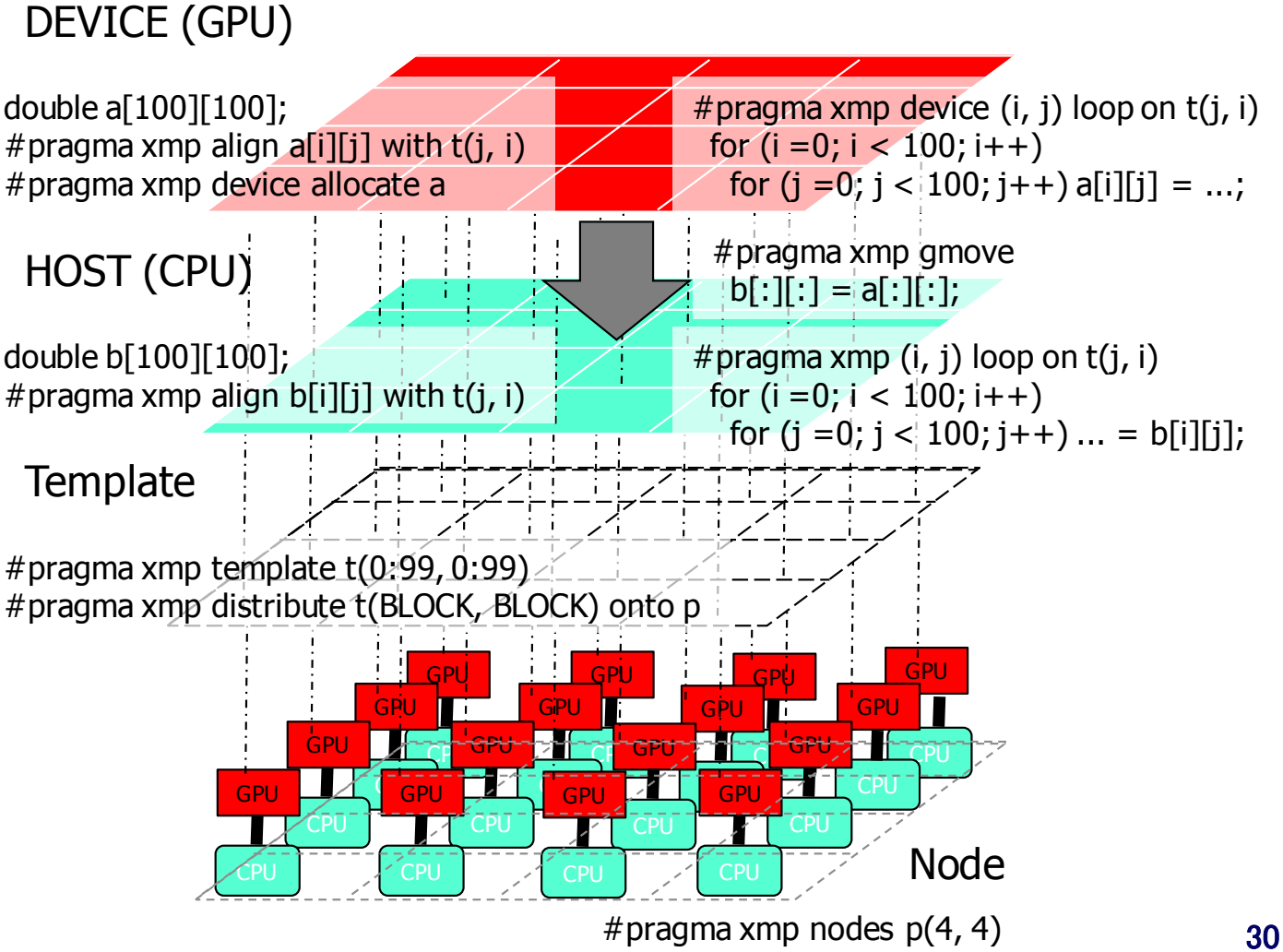
- Interface to existing (MPI) libraries
  - Rewriting every problem in XMP is not realistic.
  - Use of existing high performance parallel libraries written in MPI is crucial.
  - We are working on the design of interfaces for Scalapack, MUMPS, … etc.
- XMP IO
  - IO for distributed array
  - Interface to MPI-IO, netCDF, HDF5, …

For post-petascle …

- Extension for acceleration devices such as GPU, MIC, …
  - XMP-dev (XcalableMP acceleration device extension)
- User-defined distribution, helo, sparse matrix support, …
- Dynamic re-distribution directives, …
- Support of Fault-tolerance and Fault-resilience

XMP-dev: XcalableMP acceleration device extension [U of Tsukuba]

- Offloading a set of distributed array and operation to a cluster of GPU



Feasibility study projects for post-petascale computing in Japan

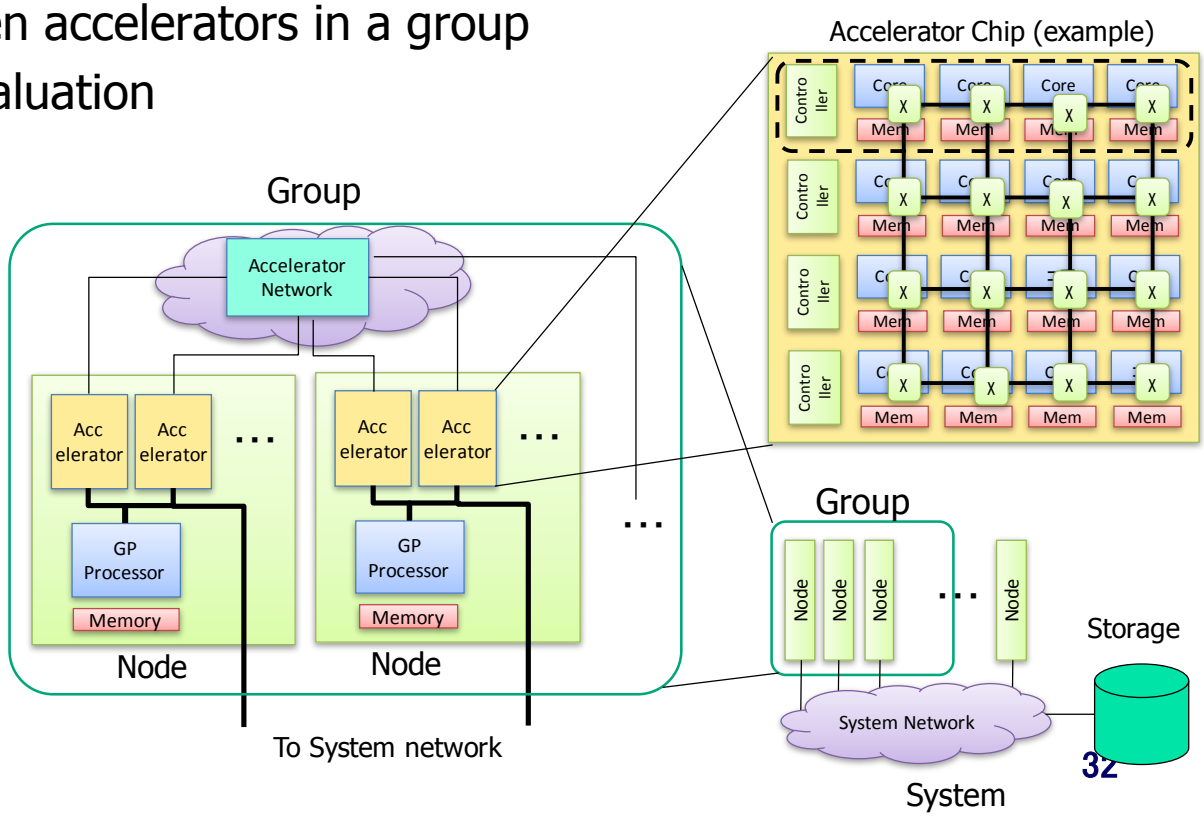
- 4 research teams are accepted (in the last week)
  - Application study team leaded by RIKEN AICS (Tomita)
  - System study team leaded by U Tokyo (Ishikawa)
    - Next-generation "General-Purpose" Supercomputer
  - System study team leaded by U Tsukuba (Sato)
    - Study on exascale heterogeneous systems with accelerators
  - System study team leaded by Tohoku U (Kobayashi)

- Projects were started from July 2012 (1.5 year) …

Study on exascale heterogeneous systems with accelerators (U Tsukuba proposal)

XcalableMP

- Two keys for exascale computing
  - Power and strong-scaling
- We study "exascale" heterogeneous systems with accelerators of many-cores
  - Architecture of accelerators, core and memory architecture
  - Special-purpose functions
  - Direct connection between accelerators in a group
  - Power estimation and evaluation
  - Programming model and computational science applications
  - Requirement for general-purpose system
  - etc …



Concluding Remarks

- XMP is a Parallel Programming Language for scientific programmers (not computer science experts!)
  - XcalableMP PGAS parallel programming language for better "productive" parallel programming than "MPI".
  - "downgraded HPF" as a reflection of HPF experience in Japan
- Programming environment researches for the K computer
  - We expect that the PGAS runtime will improve the performance of larger-scale parallel programs in the K computer.
  - Collaboration and feedback with application researchers are key to success
    - For improvement and dissemination of our software!

Thank you for your attention!