# Updates on XcalableMP PGAS Language


Mitsuhisa Sato

Director of Center for Computational Science (CCS),
University of Tsukuba,
Team leader of programming environment research team,
Advanced Institute for Computational Science (AICS), RIKEN

# 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.

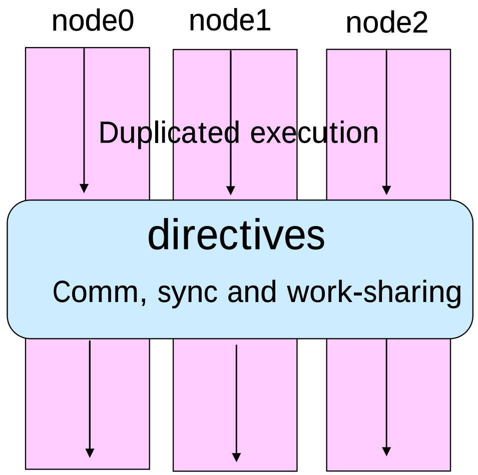
- XMP WG Members (the list of initial members)
  - Academia: M. Sato, T. Boku (compiler and system, U. Tsukuba), K. Nakajima (app. and programming, U. Tokyo), Nanri (system, Kyusyu U.), Okabe (HPF, Kyoto U.)
  - Research Lab.: Watanabe and Yokokawa (RIKEN), Sakagami (app. and HPF, NIFS), Matsuo (app., 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.
- 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 Fortran and C 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

```
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];
    }
}
```

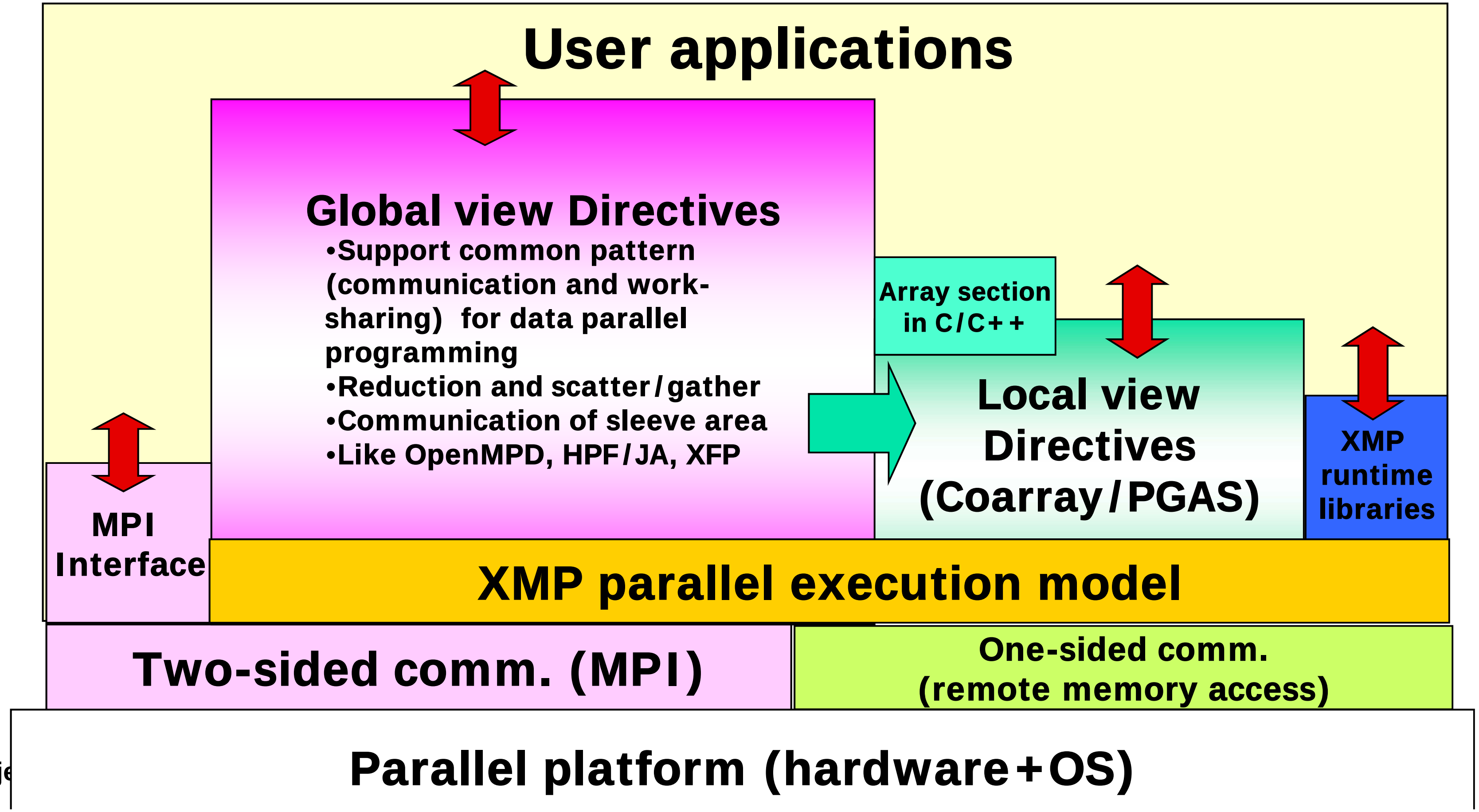
data distribution

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.

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

- Template is used as a dummy array distributed on nodes

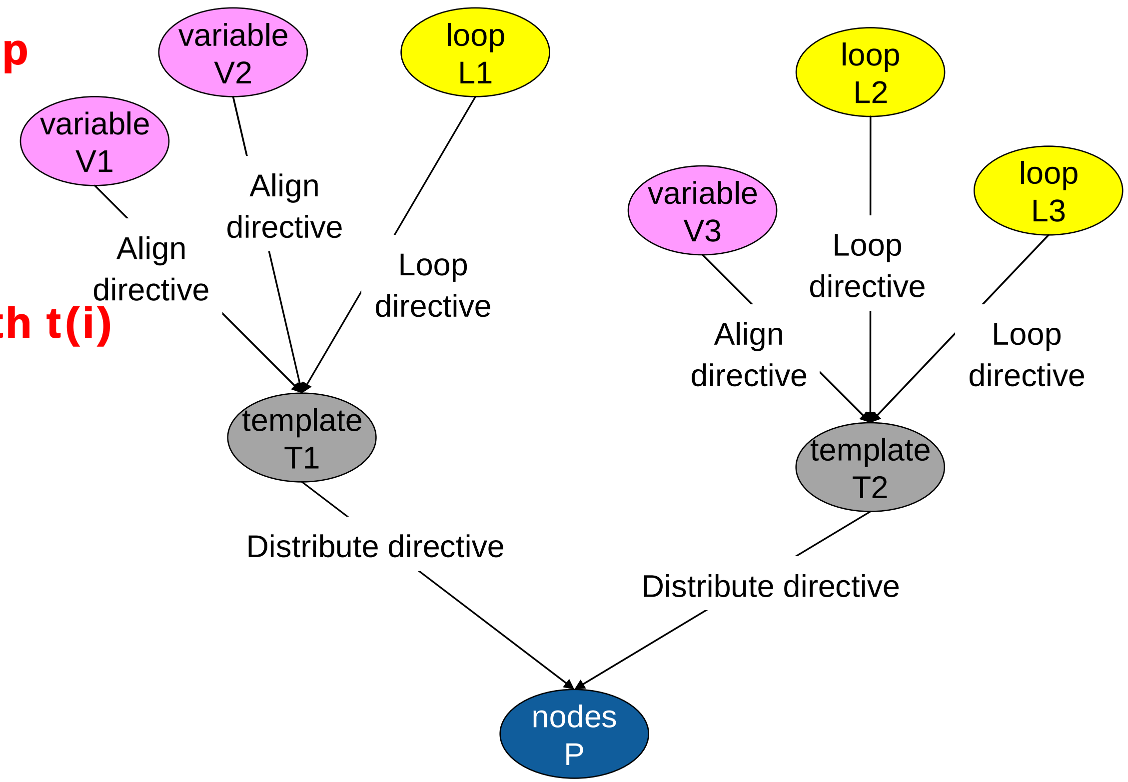
```
#pragma xmp template t(100)
#pragma distribute t(block) onto p
```

- A global data is aligned to the template

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

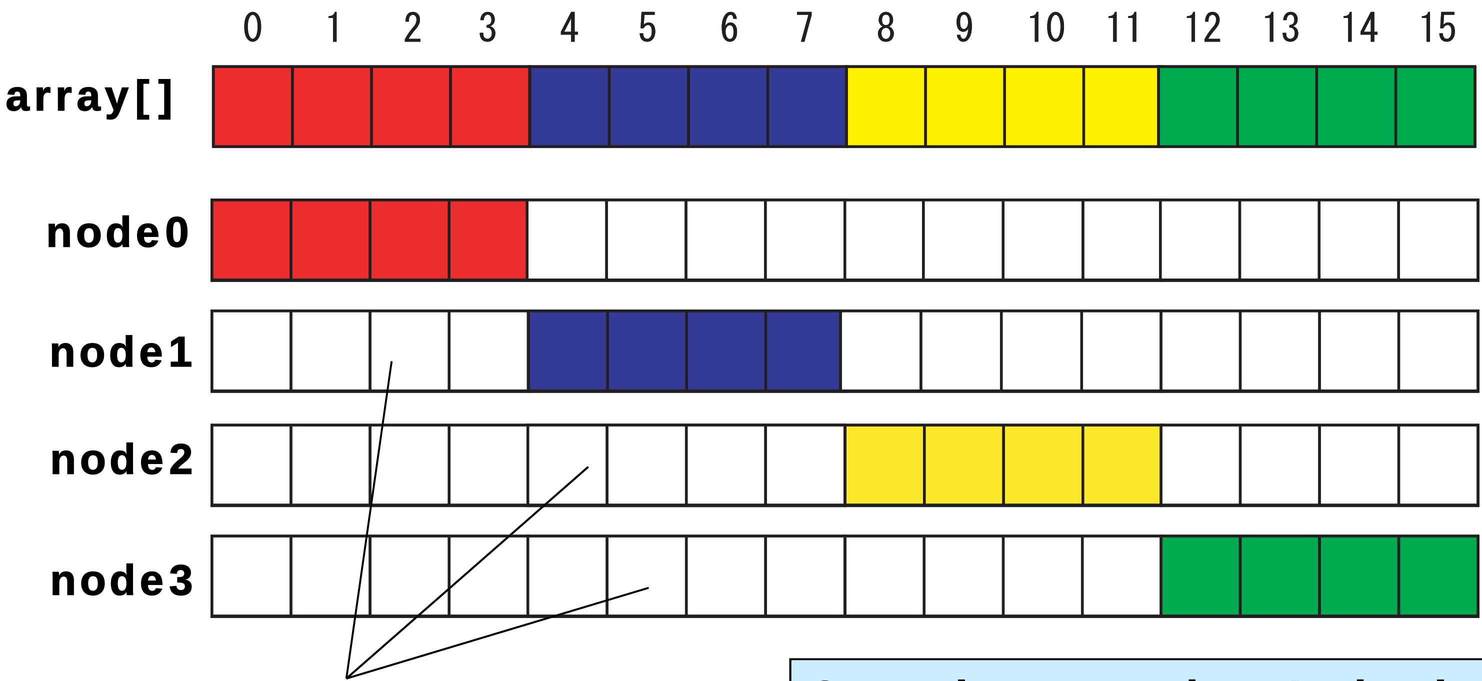
- Loop iteration must also be aligned to the template by on-clause.

```
#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!!**

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

**This is different from HPF and UPC**

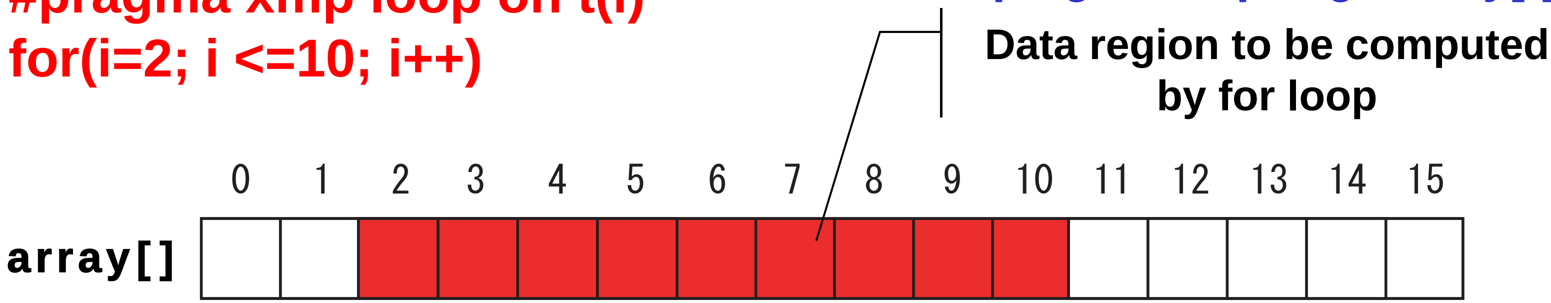
**Explicit Communication between nodes**

# Parallel Execution of “for” loop

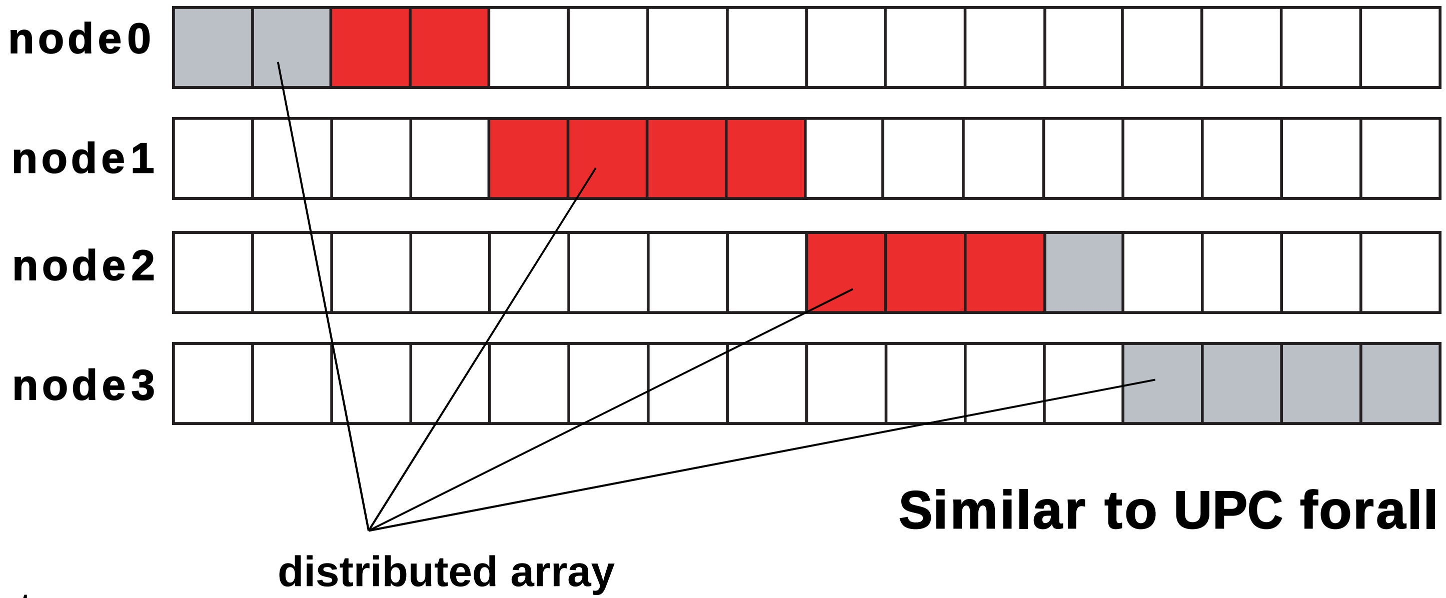
```
#pragma xmp nodes p(*)
#pragma xmp template T(0:15)
#pragma xmp distributed T(block) on
#pragma xmp align array[i] with T(i)
```

- Execute for loop to compute on array

```
#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)**

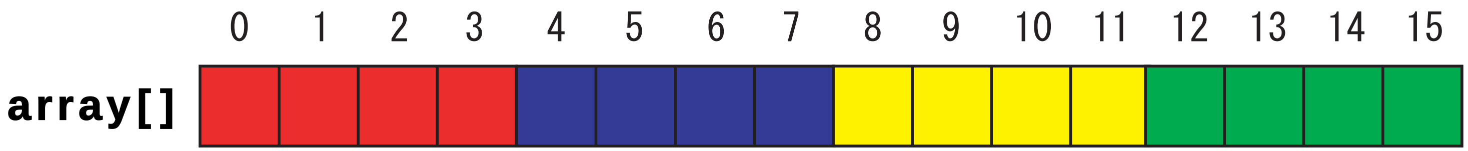



**Similar to UPC forall**

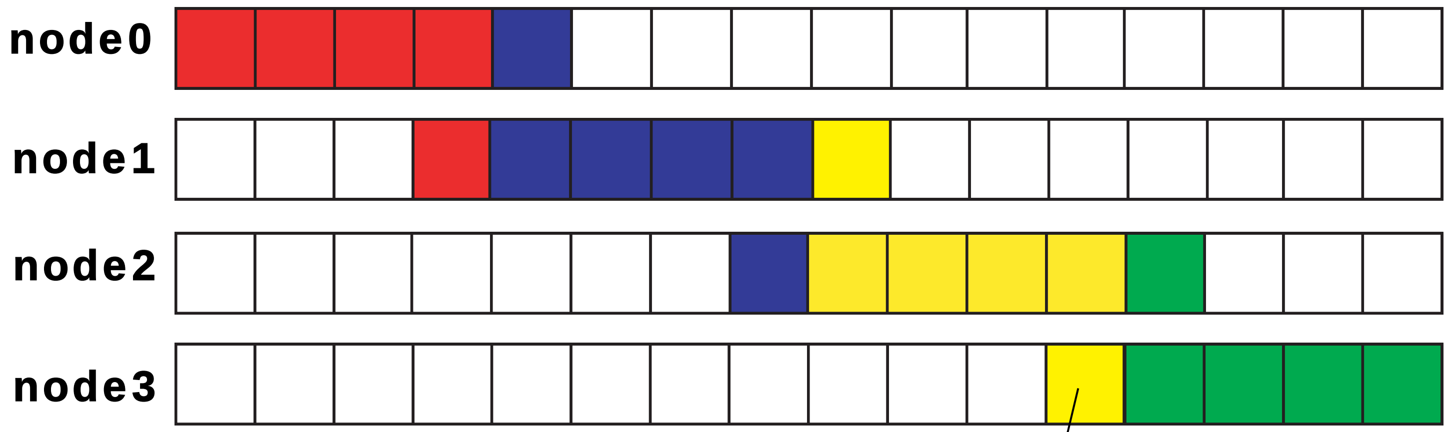
# Shadow and reflect: Data synchronization of array

- 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]

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




**#pragma xmp shadow array[1:1]**




**Programmer specifies sleeve region explicitly**

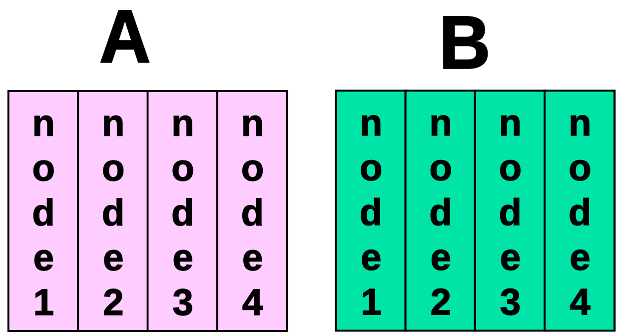
**Directive：#pragma xmp reflect array**

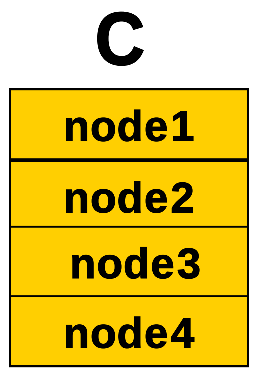
# gmove directive

- 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
  - #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*

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

# "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 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.

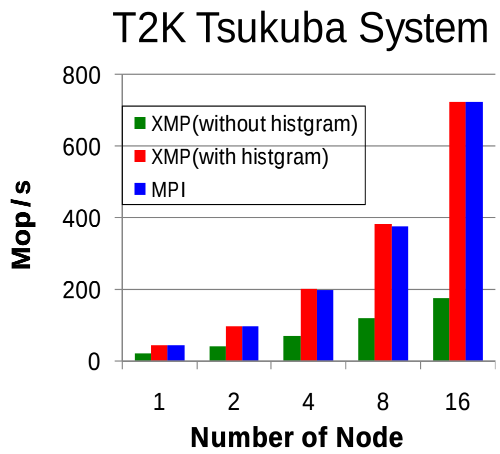
- PGAS is emerging and getting attentions from the community
  - Model for scalable communication (than MPI?)

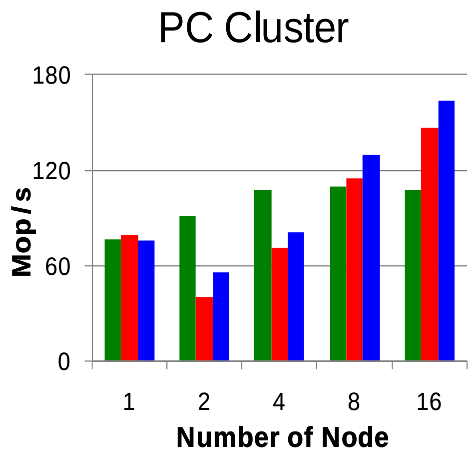
# Status of XcalableMP

- 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**

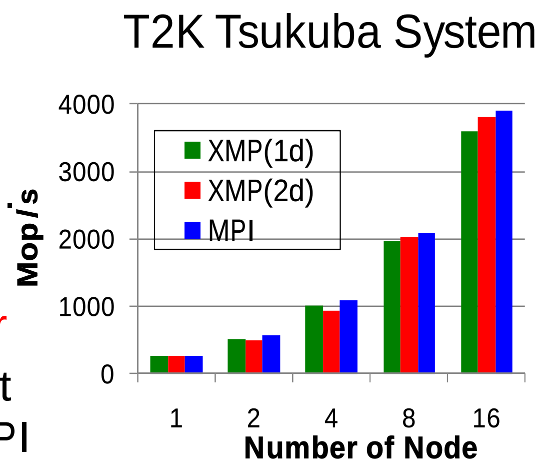
- **Coarray is used**
- **Performance comparable to MPI**

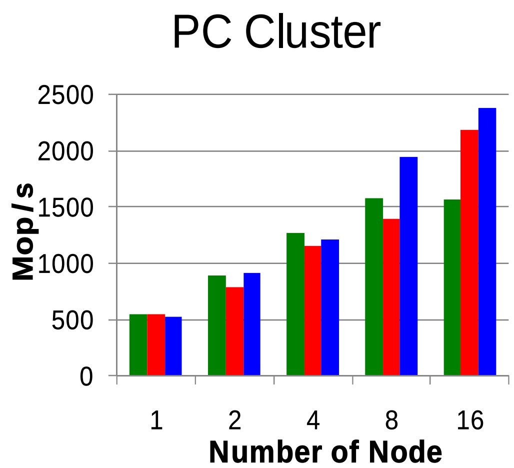



**NPB CG performace**

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



# 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.