



XcalableMP: A performance-aware
scalable parallel programming language
and
"e-science" project
for Japanese Petascale supercomputer

Mitsuhisa Sato

Center for Computational Sciences, University of Tsukuba, Japan



hpcs lab
High Performance Computing System

Agenda



- "e-science" project
 - T2K alliance

- XcalableMP : directive-based language eXtension for Scalable and performance-aware Parallel Programming
 - Motivation and background
 - Concept and model
 - Some examples

T2K Open Supercomputer Alliance

- Primary aiming at design of common specification of new supercomputers.
- Now extending to collaborative work on research, education, grid operation, ..., for inter-disciplinary computational (& computer) science.
- *Open* hardware architecture with commodity devices & technologies.
- *Open* software stack with open-source middleware & tools.
- *Open* to user's needs not only in FP & HPC field but also IT world.

Kyoto Univ.

416 nodes (61.2TF) / 13TB
Linpack Result:
Rpeak = 61.2TF (416 nodes)
Rmax = 50.5TF



Univ. Tokyo

952 nodes (140.1TF) / 31TB
Linpack Result:
Rpeak = 113.1TF (512+256 nodes)
Rmax = 83.0TF



Univ. Tsukuba

648 nodes (95.4TF) / 20TB
Linpack Result:
Rpeak = 92.0TF (625 nodes)
Rmax = 76.5TF



What is (so-called) E-science project

- Precise Project Name
 - Research and Development of Software for System Integration and Collaboration to Realize the E-Science Environment
 - e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発
- September 2008 to March 2012 (Three and half years)
- Two Subprojects
 - Seamless and Highly-Productive Parallel Programming Environment Project
 - Univ. of Tokyo, Univ. of Tsukuba, and Kyoto Univ.
 - Research on resource sharing technologies to form research community (研究コミュニティ形成のための資源連携技術に関する研究)
 - NII, AIST, Osaka Univ., TITECH, Univ. of Tsukuba, Tamagawa Univ, KEK, and Fujitsu

Overview of our project

■ Objectives

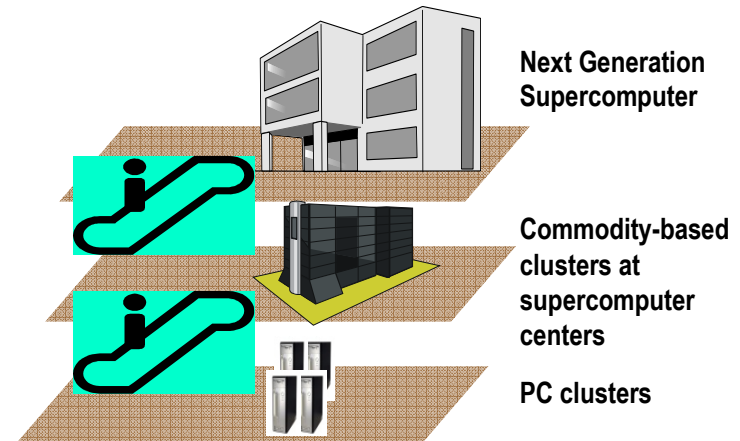
Providing a new seamless programming environment from small PC clusters to supercomputers, e.g., massively commodity-based clusters and the next generation supercomputer in Japan

- parallel programming language
- parallel script language
- Portable numerical libraries with automatic tuning
- Single runtime environment

■ Research Periods

Sept. 2008 – Mar. 2012

Funded by Ministry of Education, Culture, Sports, Science and Technology, Japan



■ Organization

- PI: Yutaka Ishikawa, U. Tokyo
- University of Tokyo
 - Portable numerical libraries with automatic tuning
 - Single runtime environment
- University of Tsukuba (Co-PI: Sato)
 - XcalableMP: parallel programming language
- Kyoto University (Co-PI: Nakashima)
 - Xcrpyt: parallel script language

- Needs for programming languages for HPC
 - 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...

Current solution for programming

```

int array[YMAX][XMAX];
main(int argc, char**argv){
  int i,j,res,temp_res, dx,llimit,ulimit,
  MPI_Init(argc, argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  dx = YMAX/size;
  llimit = rank * dx;
  if(rank != (size - 1)) ulimit = llimit + dx;
  else ulimit = YMAX;
  temp_res = 0;
  for(i = llimit; i < ulimit; i++)
  for(j = 0; j < 10; j++){
    array[i][j] = func(i, j);
    temp_res += array[i][j];
  }
  MPI_Allreduce(&temp_res, &res, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
  MPI_Finalize();
}
  
```

Only way to program is MPI, but MPI programming seems difficult, ... we have to rewrite almost entire program and it is time-consuming and hard to debug... mmm

We need better solutions!!

```

#pragma xmp template T[10]
#pragma xmp distributed T[block]
int array[10][10];
#pragma xmp aligned array[i][*] to
main(){
  int i, j, res;
  res = 0;
  #pragma xmp loop on T[i] reduction
  for(i = 0; i < 10; i++)
  for(j = 0; j < 10; j++){
    array[i][j] = func(i, j);
    res += array[i][j];
  }
}
  
```

We want better solutions ... to enable step-by-step parallel programming from the existing codes, ... easy-to-use and easy-to-tune-performance ... portable ... good for beginners.

■ Objectives

- Making a draft on “petascale” parallel language for “standard” parallel programming
- To propose the draft to “world-wide” community as “standard”

■ 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)

■ More than 10 WG meetings have been held (Dec. 13/2007 for kick-off)

■ Funding for development

- E-science project : “Seamless and Highly-productive Parallel Programming Environment for High-performance computing” project funded by Ministry of Education, Culture, Sports, Science and Technology, 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 vendors 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, Hitachi, 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

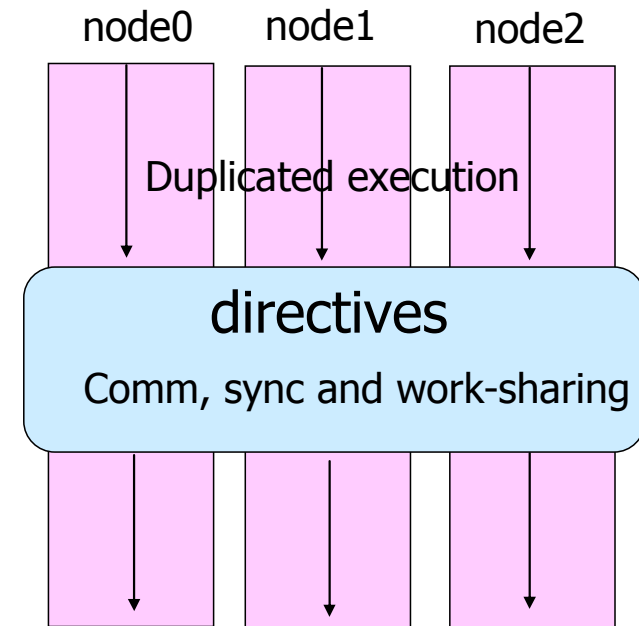
“Pitfalls” and Lessons learned from HPF

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

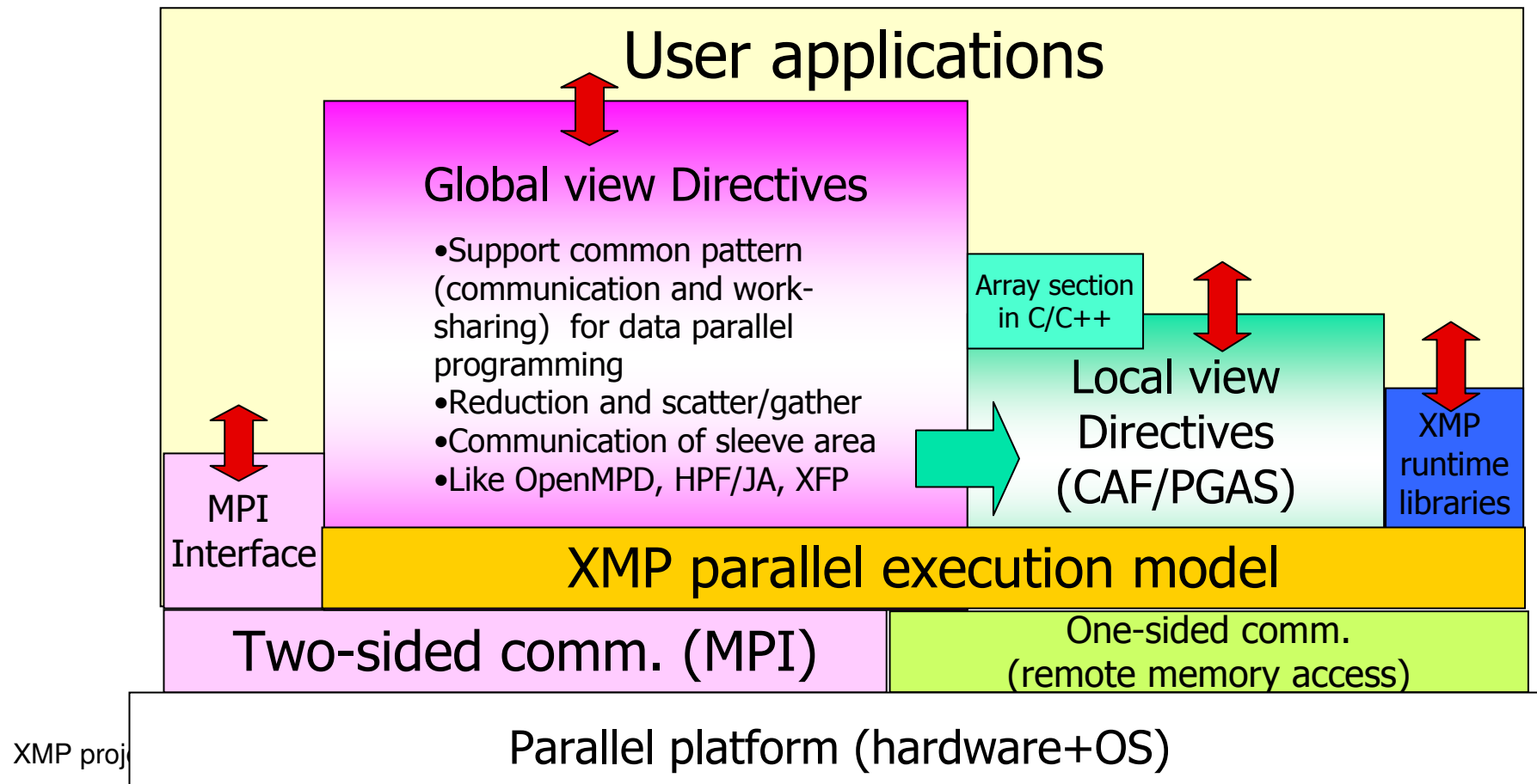
- **Directive-based language extensions** for familiar languages F90/C/C++
 - To reduce code-rewriting and educational costs.
- **“Scalable” for Distributed Memory Programming**
 - SPMD as a basic execution model
 - A thread starts execution in each node independently (as in MPI) .
 - Duplicated execution 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)



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.



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)
```

data distribution

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

add to the serial code : incremental parallelization

```
#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];  
    }  
}
```

work sharing and data synchronization

The same code written in MPI



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

main(int argc, char**argv){
  int i,j,res,temp_res, dx,llimit,ulimit,size,rank;

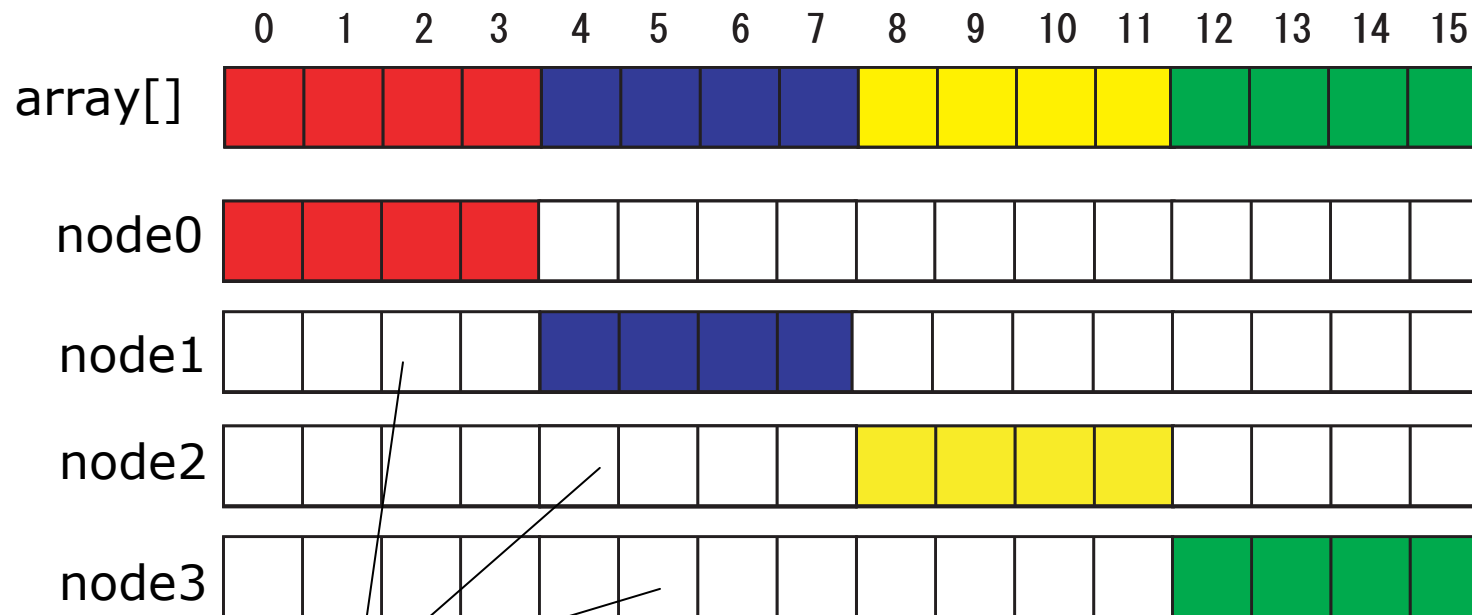
  MPI_Init(argc, argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  dx = YMAX/size;
  llimit = rank * dx;
  if(rank != (size - 1)) ulimit = llimit + dx;
  else ulimit = YMAX;

  temp_res = 0;
  for(i = llimit; i < ulimit; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      temp_res += array[i][j];
    }

  MPI_Allreduce(&temp_res, &res, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
  MPI_Finalize();
}
```

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

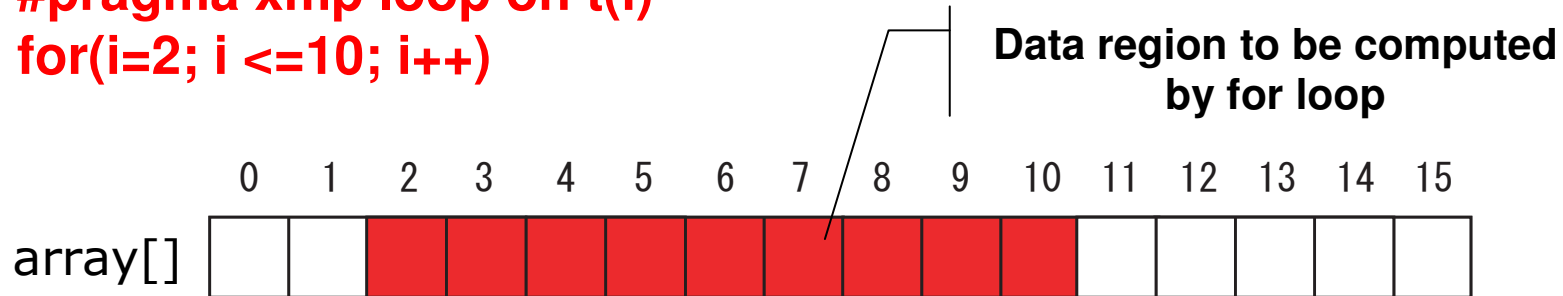
Parallel Execution of “for” loop



- Execute for loop to compute on array

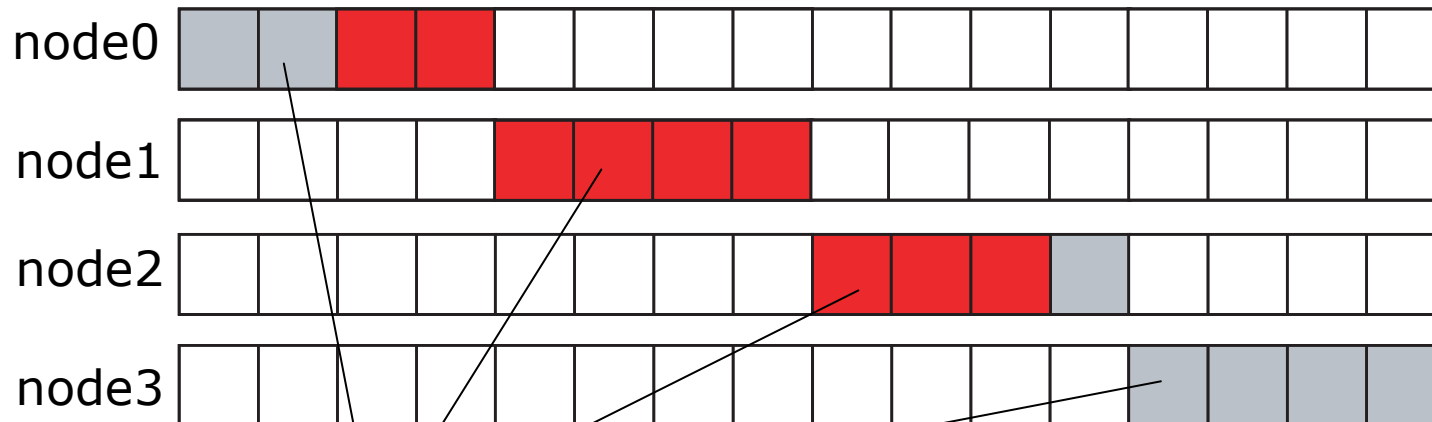
#pragma xmp loop on t(i)
for(i=2; i <=10; i++)

#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)



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

#pragma xmp loop on t(i)



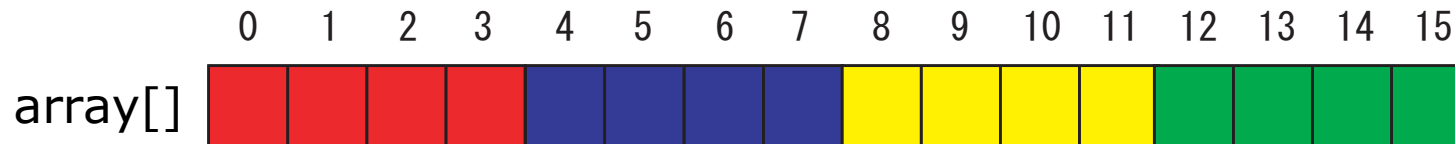
distributed array

Data synchronization of array (shadow)

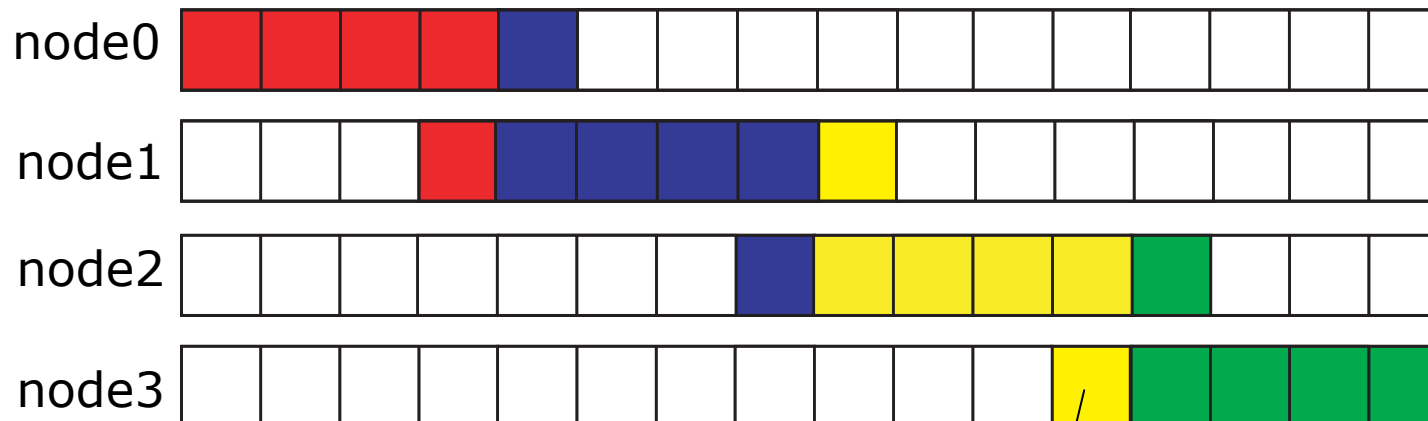


- Exchange data only on “shadow” (sleeve) region
 - If neighbor data is required to communicate, then only sleeve area can be considered.
 - example : $b[i] = \text{array}[i-1] + \text{array}[i+1]$

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



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



Programmer specifies sleeve region explicitly
Directive : `#pragma xmp reflect array`

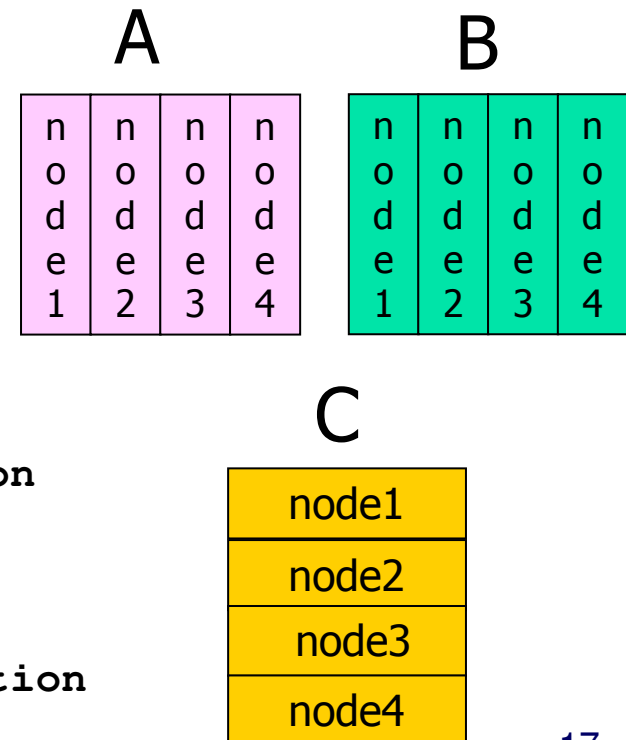
gmove directive



- 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 Local view directives



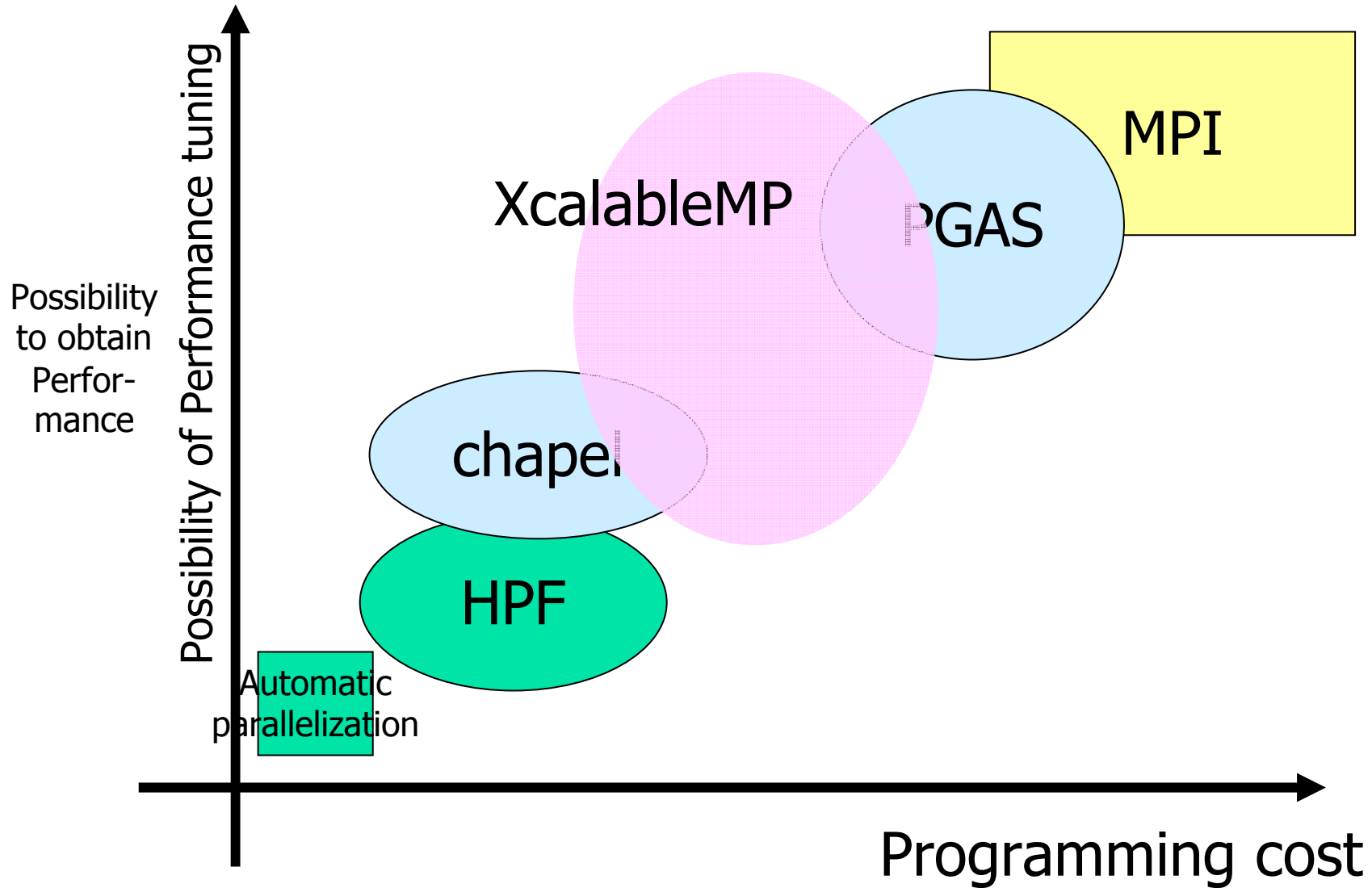
- XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as "**local view**" programming.
 - The basic execution model of XcalableMP is SPMD
 - Each node executes the program independently on local data if no directive
 - We adopt Co-Array as our PGAS feature.
 - In C language, 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 **X_{calable}MP**



Summary

- Our objective
 - High productivity for distributed memory parallel programming
 - Not just for research, but collecting ideas for “standard”
 - Distributed memory programming “easier than MPI” !!!

- XcalableMP project: status and schedule
 - [SC09 HPC Challenge benchmark Class 2 Finalist!](#)
 - Nov. 2009, draft of XcalableMP specification 0.7
 - <http://www.xcalablemp.org/xmp-spec-0.7.pdf>
 - 1Q/10 α release, C language version
 - 2Q/10 Fortran version

- Features for the next
 - IO
 - Fault tolerant
 - support for GPGPU programming
 - Others ...

About Collaborations



- Past projects on HPC between France and Japan
 - Mainly for Grid Computing technologies
 - Sakura (JSPS): Research on software and network technologies for an international large scale P2P distributed computing infrastructure (Sato and Cappello) : 2005-2006 (3years)
 - FJ Grid project of INRIA
 - JST-CNRS: NEGST (NEXt Grid Systems and Techniques): International Collaboration and Promotion on interoperability and advanced technologies of Grid computing (Miura@NII and Petiton): 2006-2008 (3 years)
 - These project ware very successful!!
 - Many Workshops, and exchanging students and researchers.
 - Now moving to the pataascale (and exascale) ...
 - France-Japan PAAP (Petascale Applications , Algorithms and Programming) workshop: 3 workshops were held.

About Collaborations



- High Performance Computing (HPC) can be a candidate for F-J Joint project
 - HPC is an important area to support computational science & engineering for future society and industries.
 - Japanese 10 petaflops computer project and Kobe HPC Center is planned.
 - Some group (T2K and Titech) is still active in this area
 - World-wide community of HPC is now moving to Exascale!

- How to co-operate
 - Setting a common ambitious goal and making plans and deliverables for HPC moving to exascale computing.
 - Cooperation may be complementary
 - e.g. France is strong in numerical analysis, and Japan is strong in system.
 - Form of cooperation
 - Meeting (f2f and on VTC), exchange of researchers and students