XcalableMP parallel language project

Mitsuhisa Sato

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

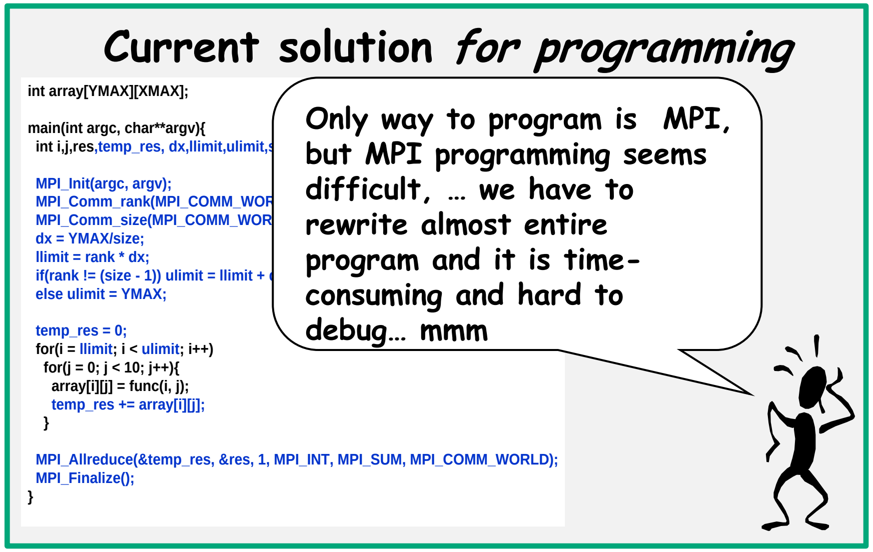
Outline

- Motivation and Background
  - Project organization
  - A short history of programming language research in Japan
  - Some thoughts about HPF
- XcalableMP PGAS parallel programming language
  - Basic model and concept
  - Overview of specification
  - performance on the K computer
  - Related researches and projects
- Summary

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, …

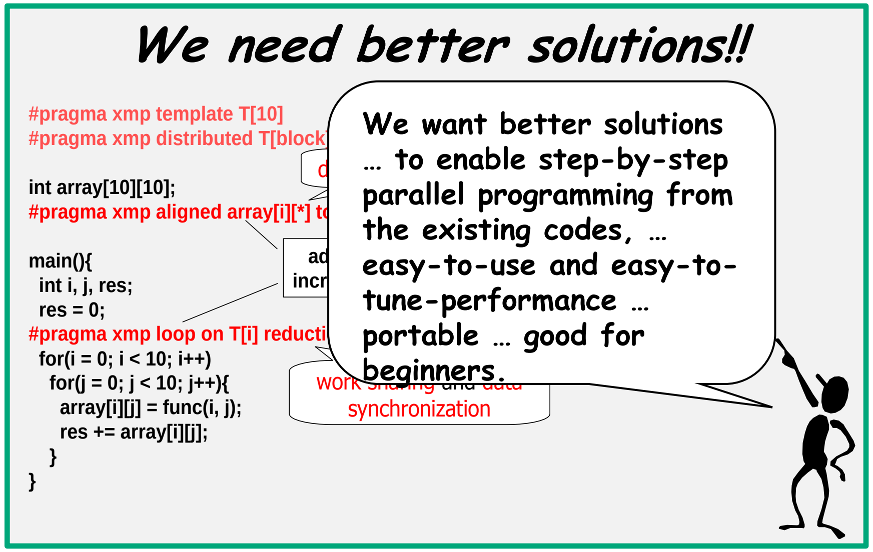
XcalableMP is our solution!



Current solution *for programming*

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

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

  MPI_Init(argc, argv);
  MPI_Comm_rank(MPI_COMM_WOR
  MPI_Comm_size(MPI_COMM_WOR
  dx = YMAX/size;
  llimit = rank * dx;
  if(rank != (size - 1)) ulimit = llimit + c
  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] reducti
  for(i = 0; i < 10; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      res += array[i][j];
    }
}
```

d
ad
incr
work sharing and data
synchronization

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.

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.

The 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)

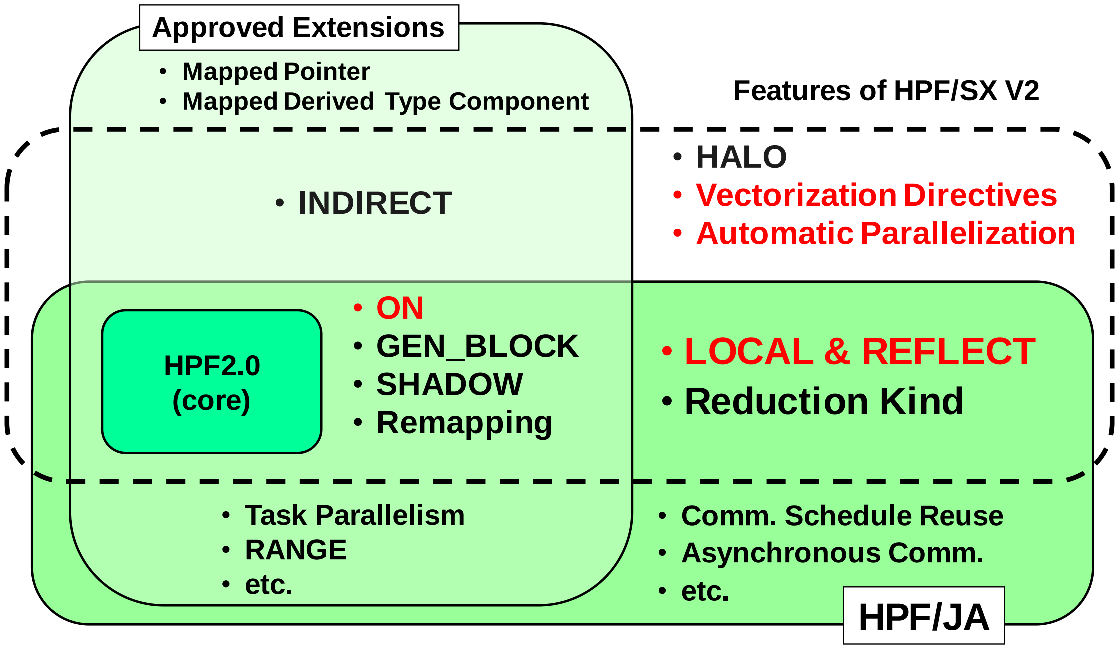
VPP500

HPF2.0 and HPF Activity in Japan

- Japanese supercomputer venders were interested in HPF and developed HPF compiler on their systems.
- HPF 2.0 (approved extension)
  - Independent & on clause
  - Shadow
  - GenBlock
- HPF/JA proposal by Japan

  HPF promotion consortium
  - Reduction kind
  - Reflect
  - Local
  - Full shadow
- HPF/ES extension by NEC

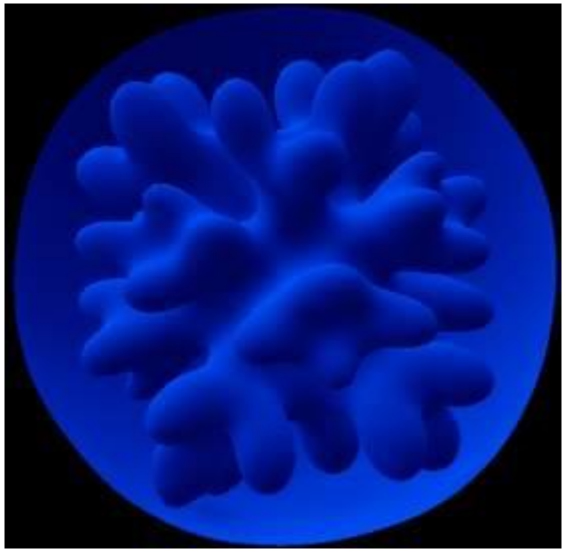
  for Earth Simulator System.
  - helo
  - Paralle I/O



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

- 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**

```
!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
```

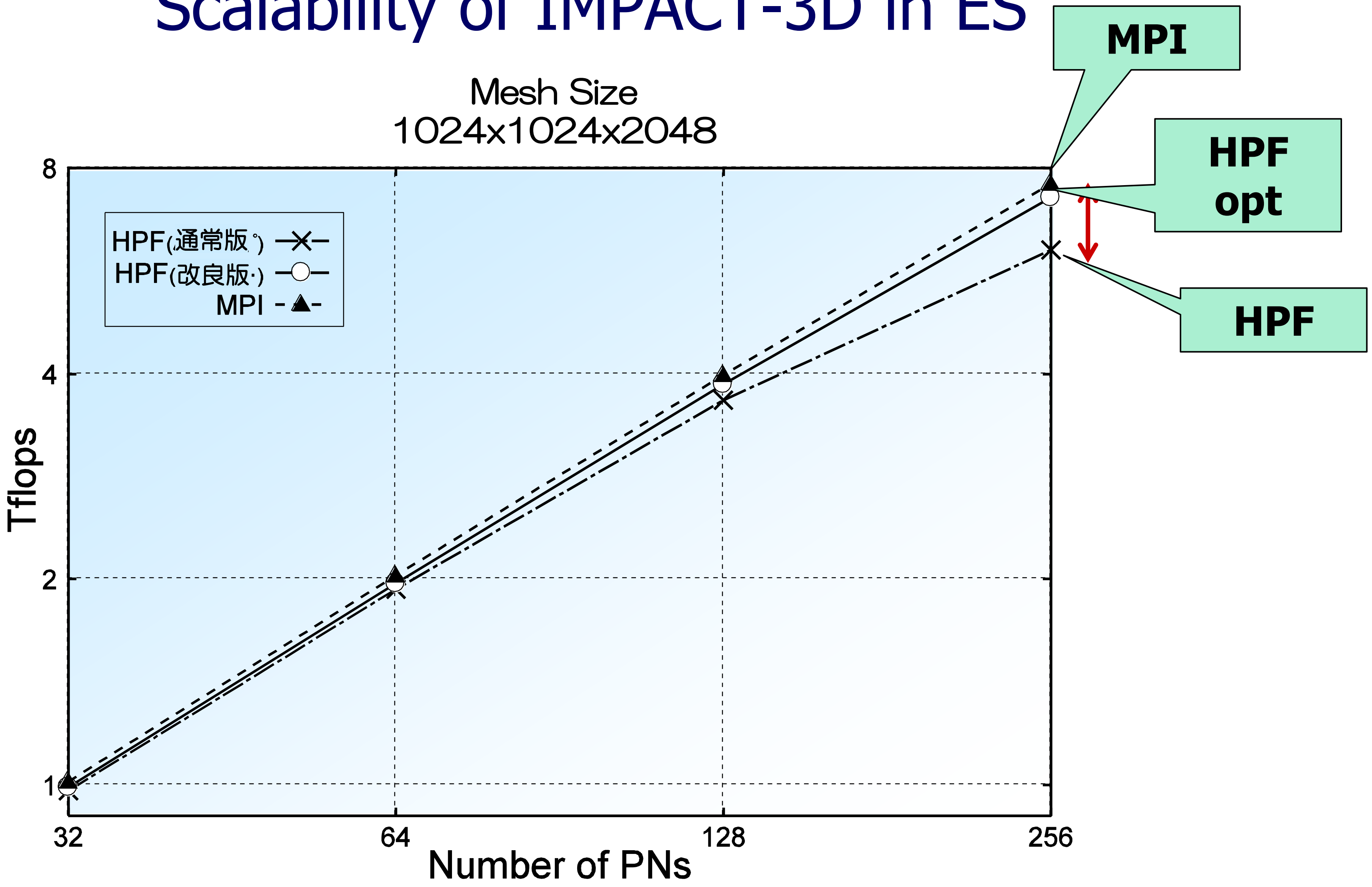
Optimization by HPF/JA extensions

- 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`**

※ MPI15.3TFLOPS

```
!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

- "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
    - SHADOW & RELECT, ON HOME, LOCAL, …
  - 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

"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?

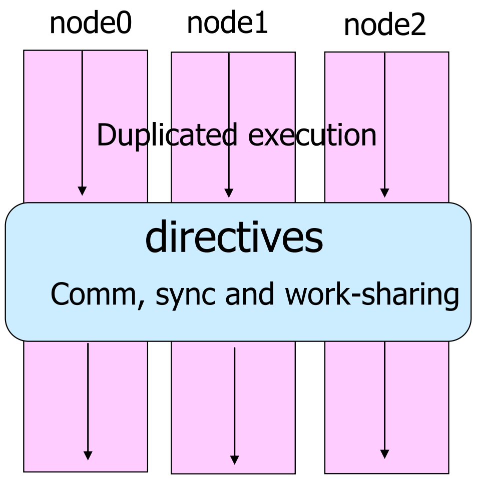
- Explicit message-passing using MPI still remains the dominant programming system for scalable applications (more than at the time of HPF?)
  - Many software stacks on top of MPI (Apps framework libraries, …)
- Fortran 90 is mature enough now. C (and C++) is used for HPC apps.
  - OpenMP supports both.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

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

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 += …;
   } }
```

XcalableMP Code Example (Fortran)

```
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

XcalableMP Code Example (C)

```
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];
    }
}
```

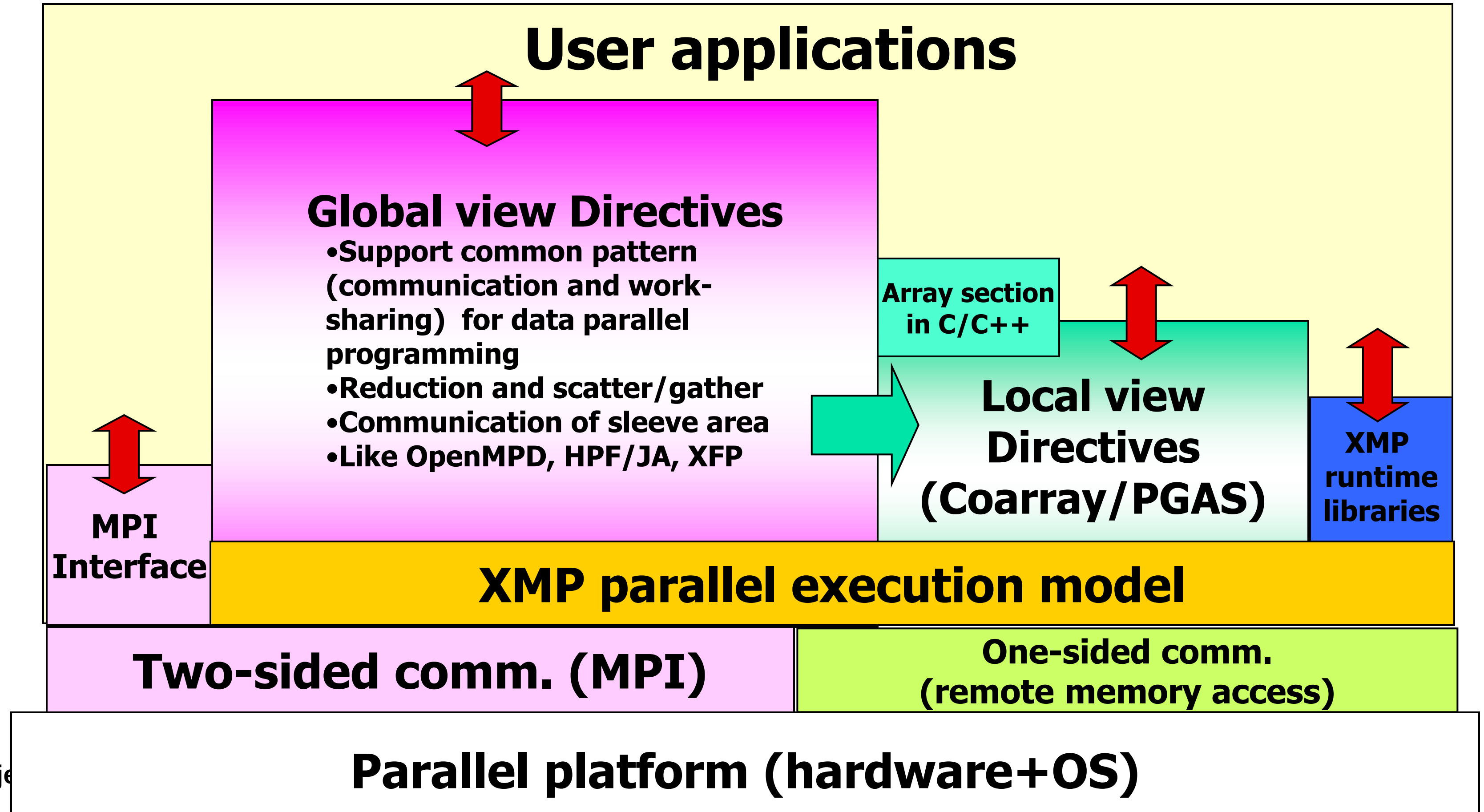
data distribution

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.


User applications
Global view Directives
•Support common pattern (communication and work-sharing) for data parallel programming
•Reduction and scatter/gather
•Communication of sleeve area
•Like OpenMPD, HPF/JA, XFP
Array section in C/C++
Local view Directives (Coarray/PGAS)
XMP runtime libraries
MPI Interface
XMP parallel execution model
Two-sided comm. (MPI)
One-sided comm. (remote memory access)
Parallel platform (hardware+OS)

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

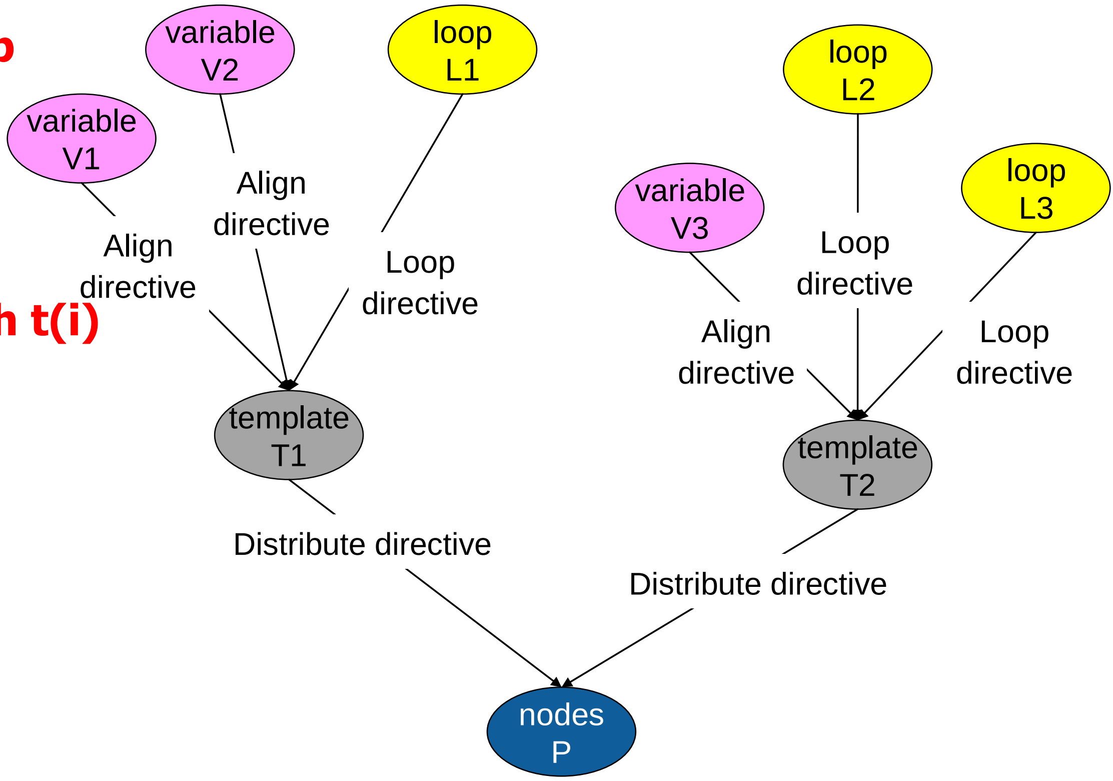
#pragma xmp template t(100)
#pragma distribute t(block) onto p

- A global data is aligned to the template

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

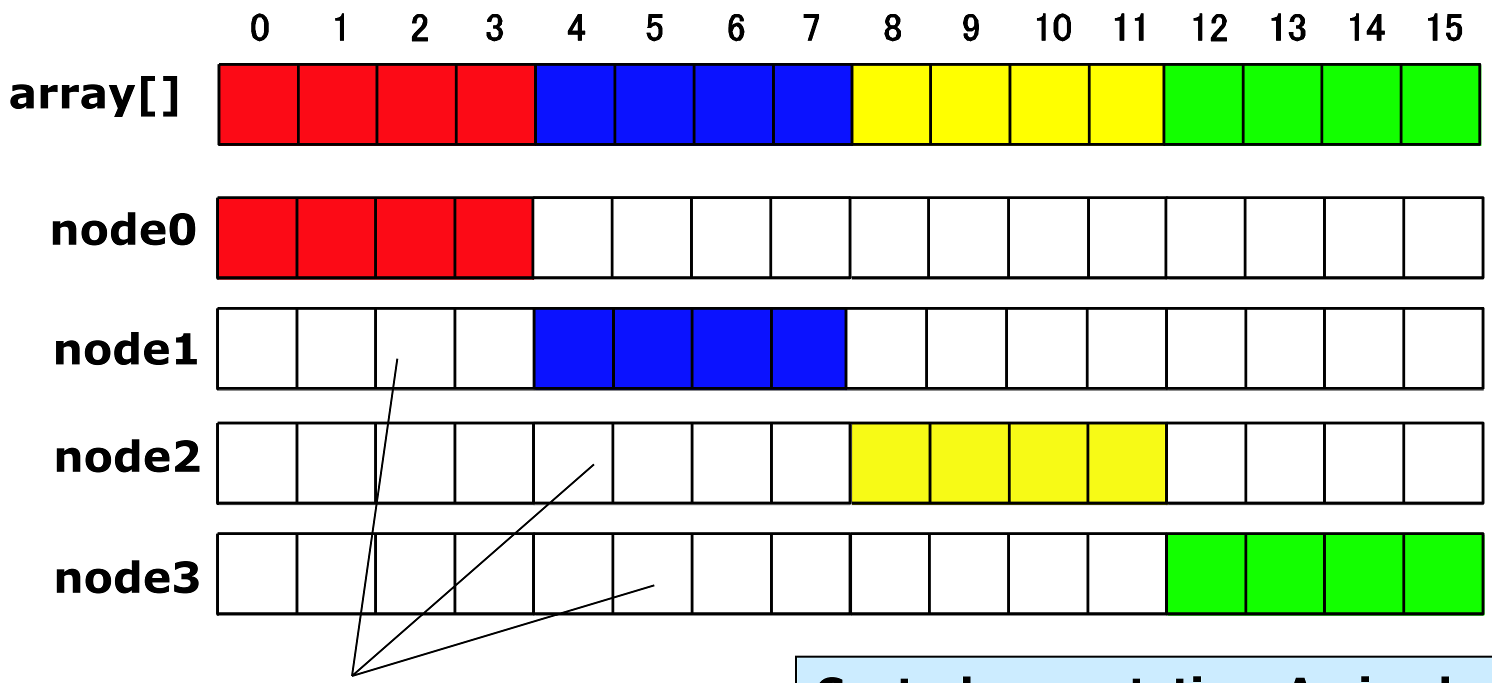
- Loop iteration must also be aligned to the template by on-clause.

#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

Explicit Communication between nodes

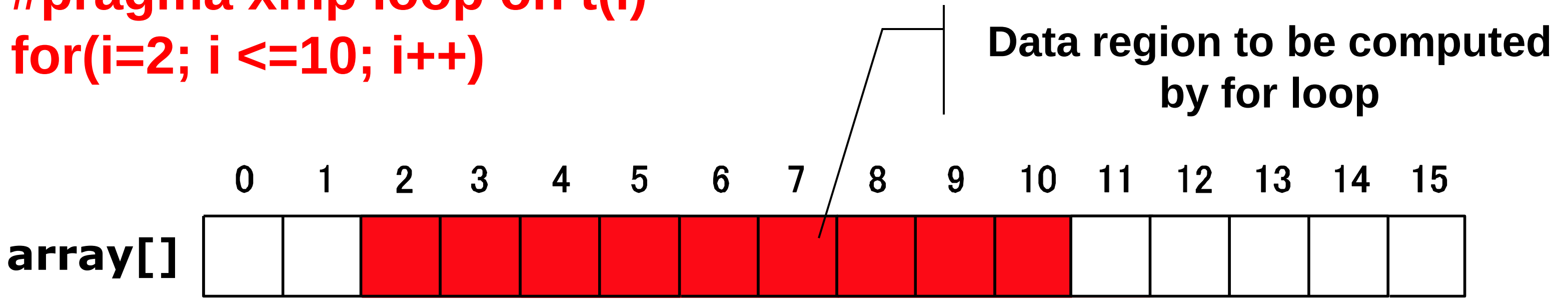
This is different from HPF and UPC

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)
```

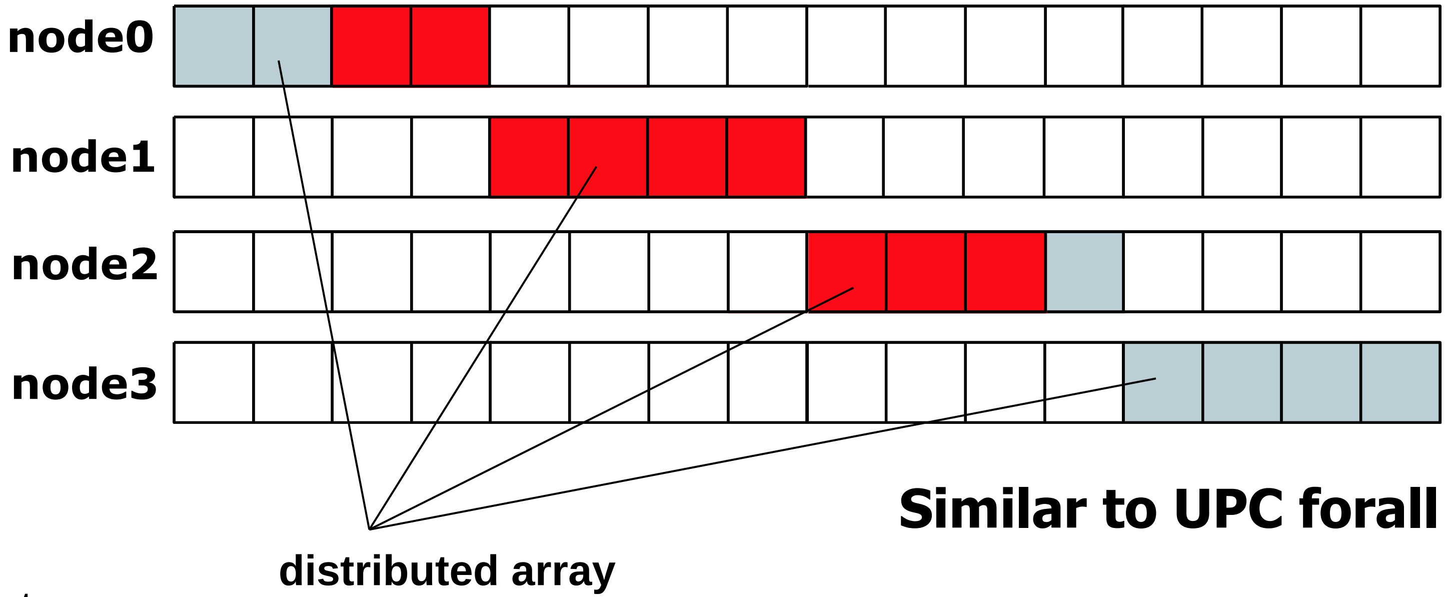
- Execute for loop to compute on array

```
#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:

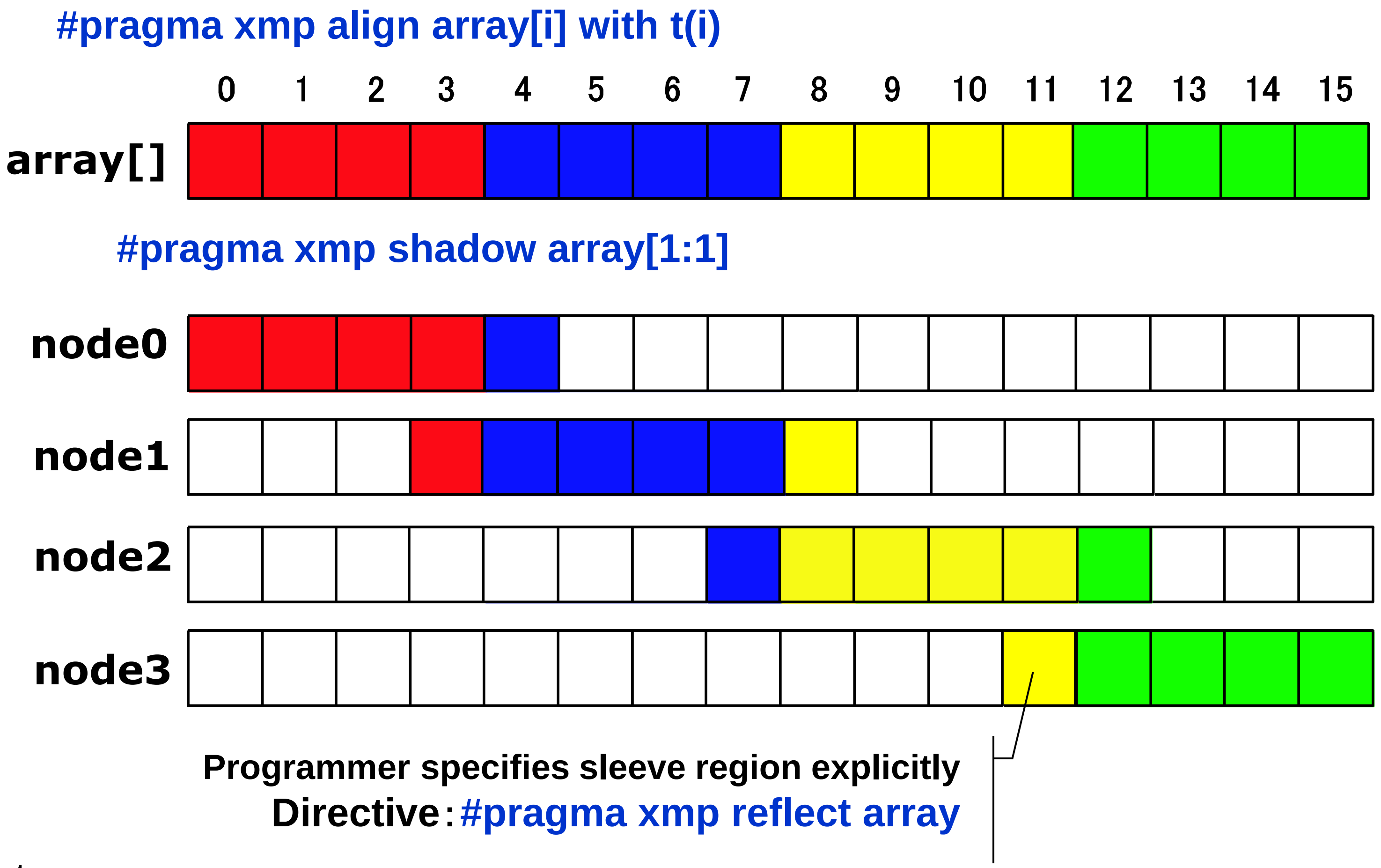
`#pragma xmp loop on t(i)`



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)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
array[]
#pragma xmp shadow array[1:1]
node0
node1
node2
node3
Programmer specifies sleeve region explicitly
Directive：#pragma xmp reflect array

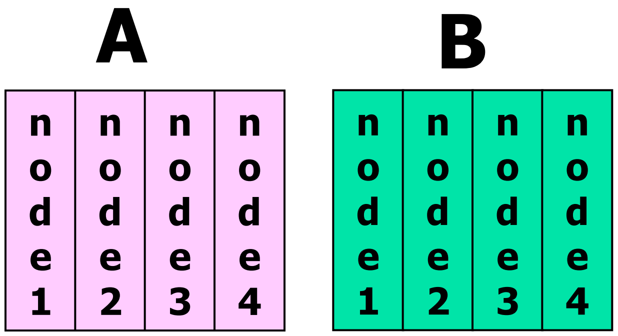
XcalableMP Global view directives

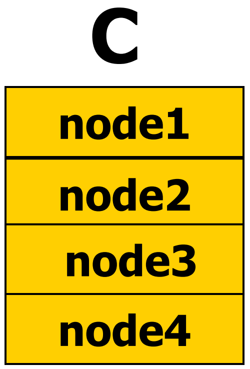
- Execution only master node
  - #pragma xmp block on master
- Broadcast from master node
  - #pragma xmp bcast (*var*)
- Barrier/Reduction
  - #pragma xmp reduction (*op: var*)
  - #pragma xmp barrier
- Global data move directives for collective comm./get/put
- Task parallelism
  - #pragma xmp task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

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
```





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
```

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

Research Agenda of XcalableMP

- Interface to existing (MPI) libraries, Mixed with MPI
  - 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, …

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

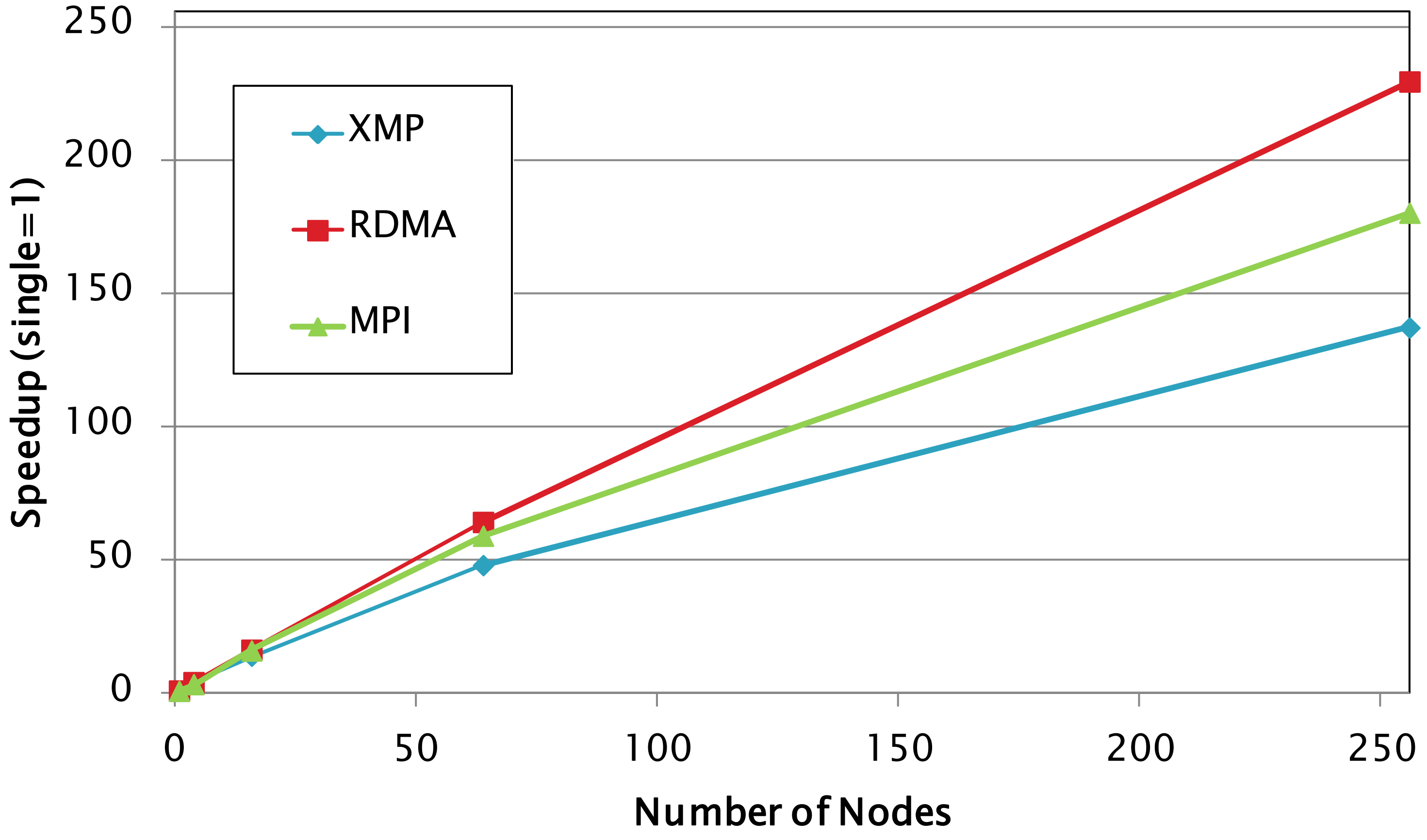
Data distribution

Neighbor comm

Loop paralization

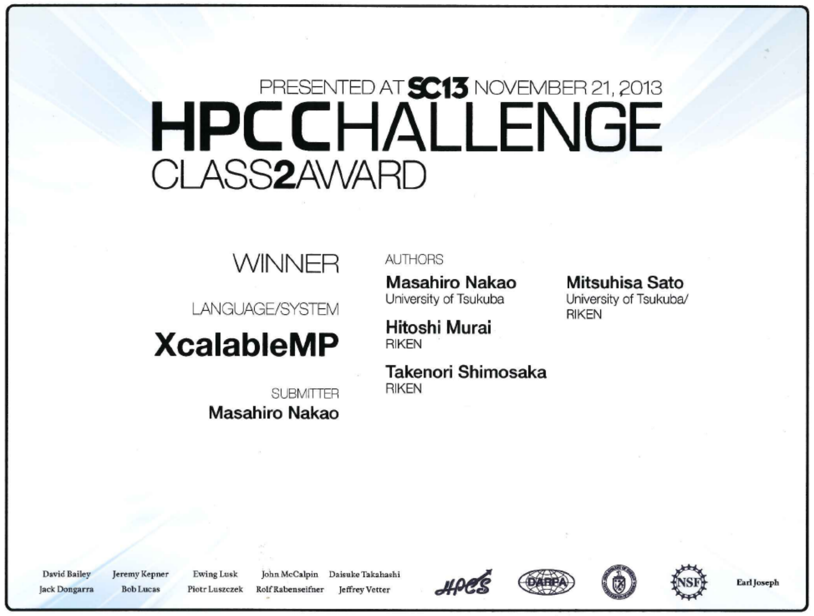
Performance results of K computer

- 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.
- <u>We can improve performance using RDMA of the K computer</u>



HPCC Class2 competition

- XMP (U. Tsukuba and RIKEN AICS team) won HPCC Class2 Awards in SC13.
  - Implementation and performance evaluation on the K computer
- HPC Challenge Benchmarks:
    - HPL, FFT, RandomAccess, Stream
  - Class 1 for Performance
  - Class 2 for Productivity of Prog. Language (Performance and Elegance)



PRESENTED AT SC13 NOVEMBER 21, 2013

HPC CHALLENGE

CLASS2AWARD

WINNER

LANGUAGE/SYSTEM

XcalableMP

SUBMITTER

Masahiro Nakao

AUTHORS

Masahiro Nakao
University of Tsukuba

Hitoshi Murai
RIKEN

Takenori Shimosaka
RIKEN

Mitsuhisa Sato
University of Tsukuba/
RIKEN

David Bailey, Jack Dongarra, Jeremy Kepner, Bob Lucas, Ewing Lusk, Piotr Luszczek, John McCalpin, Rolf Rabenseifner, Daisuke Takahashi, Jeffrey Vetter

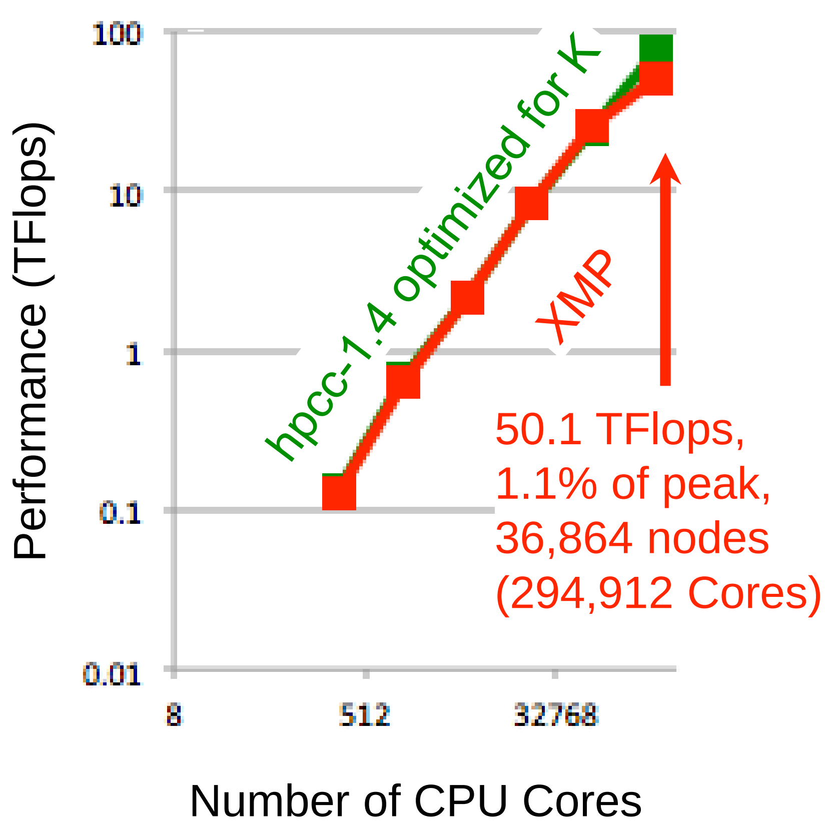
Earl Joseph

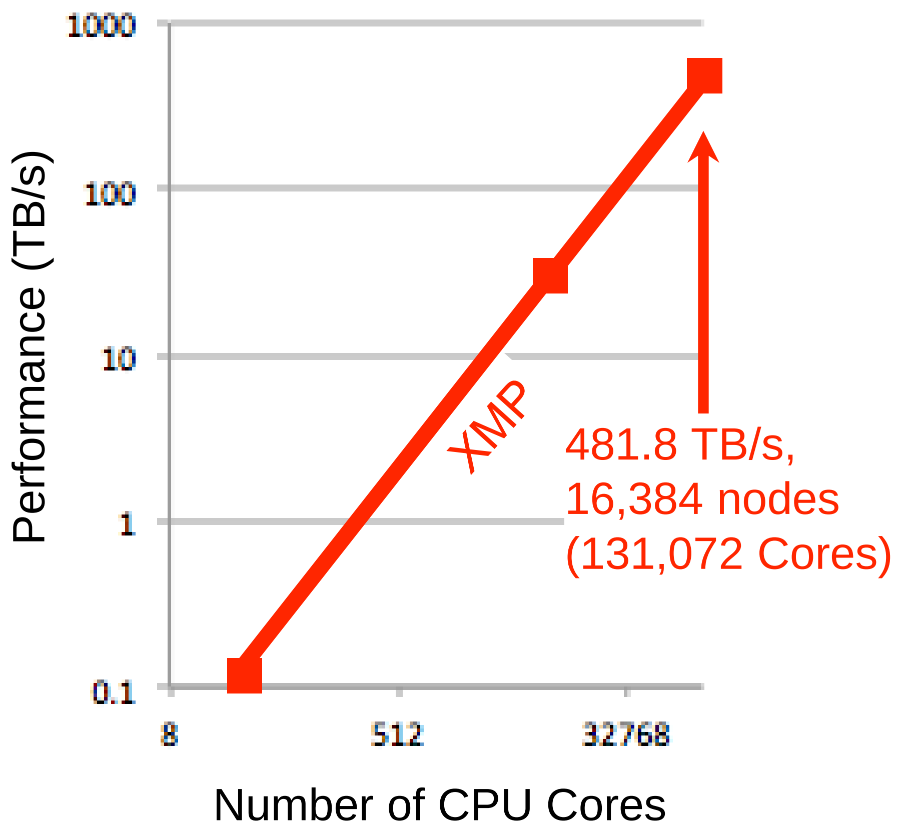
Benchmark	# Nodes	Performance	SLOC
HPL	16,384	933.8 TFlops (44.5% of peak)	306
RandomAccess	16,384	162.6 GUPs	250
FFT	36,864	50.1 TFlops (1.1% of peak)	239 + 283 + 1892
STREAM	16,384	481.8 TB/s	66
HIMENO	82,944	1.3 PFlops (12.7% of peak)	137

Full compute nodes

Results (2/3)

- FFT (1 process/node with 8 threads)
- STREAM (1 process/node with 8 threads)

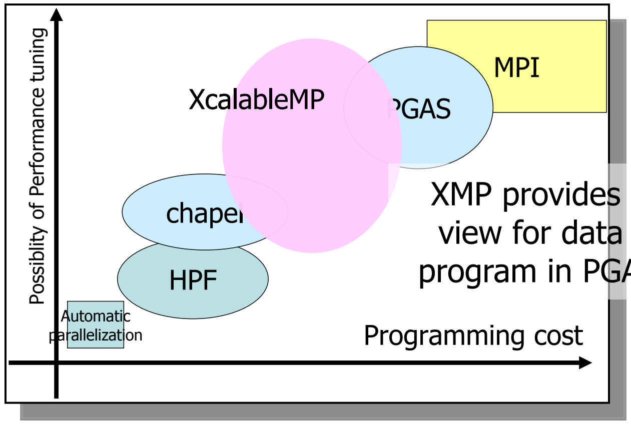




slide at SC12 PGAS BoF

XcalableMP(XMP) http://www.xcalablemp.org

- What's XcalableMP (XMP for short)?
  - A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
  - XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.
- Project status (as of Nov. 2013)
  - XMP Spec **Version 1.2** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
  - Reference implementation by U. Tsukuba and Riken AICS: **Version 0.7 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to- Source compiler to code with the runtime on top of MPI and GasNet.



XMP provides a global view for data parallel program in PGAS model

- Language Features
  - Directive-based language extensions for Fortran and C for PGAS model
  - Global view programming with global-view distributed data structures for data parallelism
    - SPMD execution model as MPI
    - pragmas for data distribution of global array.
    - Work mapping constructs 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).

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][*] to 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];
    }
}
```

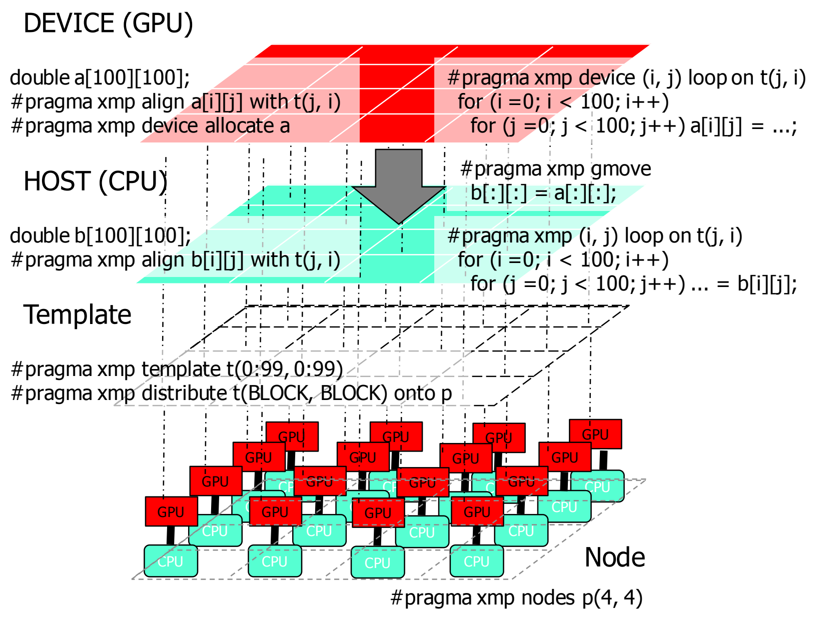
data distribution

add to the serial code : incremental parallelization

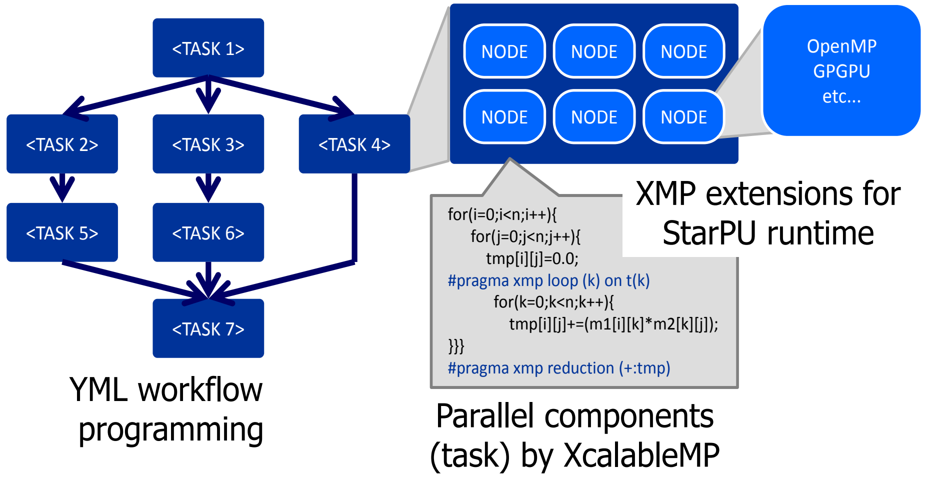
work sharing and data synchronization

Researches and projects related to XMP

- XMP extension for GPU: XMP-dev and XMP/OpenACC integration (JST CREST)
- FP2C (Framework for Post-Petascale Computing) : the multilevel programming as a solution for post-petascale system (FP3C Japan-French project)
  - Integration with YML flow lang.
  - XMP/StarPU integration scheduling CPU/GPU
- Porting to other platform
  - NEC SX, IBM BG/Q
- Optimization for Intel Xeon Phi
- User-defined Distribution ...
- Dynamic Tasking with XMP



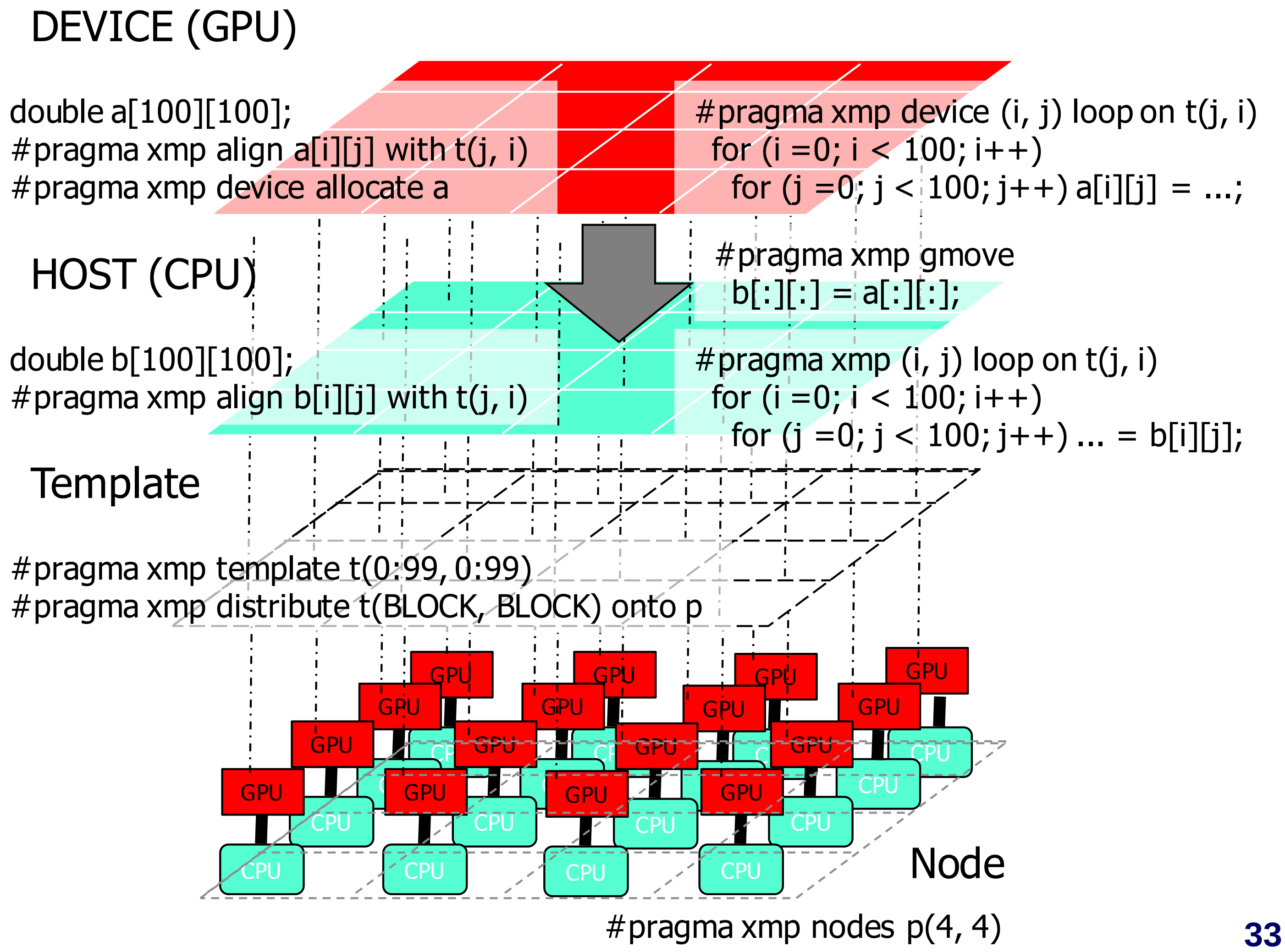
```
double a[100][100];
#pragma xmp align a[i][j] with t(j, i)
#pragma xmp device allocate a

#pragma xmp device (i, j) loop on t(j, i)
  for (i =0; i < 100; i++)
    for (j =0; j < 100; j++) a[i][j] = ...;

#pragma xmp gmove
  b[:][:] = a[:][:];

double b[100][100];
#pragma xmp align b[i][j] with t(j, i)

#pragma xmp (i, j) loop on t(j, i)
  for (i =0; i < 100; i++)
    for (j =0; j < 100; j++) ... = b[i][j];

#pragma xmp template t(0:99, 0:99)
#pragma xmp distribute t(BLOCK, BLOCK) onto p

#pragma xmp nodes p(4, 4)
```



```
for(i=0;i<n;i++){
  for(j=0;j<n;j++){
    tmp[i][j]=0.0;
#pragma xmp loop (k) on t(k)
      for(k=0;k<n;k++){
        tmp[i][j]+=(m1[i][k]*m2[k][j]);
}}}
#pragma xmp reduction (+:tmp)
```

XMP-dev: XcalableMP acceleration device extension

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



```
double a[100][100];
#pragma xmp align a[i][j] with t(j, i)
#pragma xmp device allocate a
```

```
#pragma xmp device (i, j) loop on t(j, i)
  for (i =0; i < 100; i++)
    for (j =0; j < 100; j++) a[i][j] = ...;
```

```
#pragma xmp gmove
  b[:][:] = a[:][:];
```

```
double b[100][100];
#pragma xmp align b[i][j] with t(j, i)
```

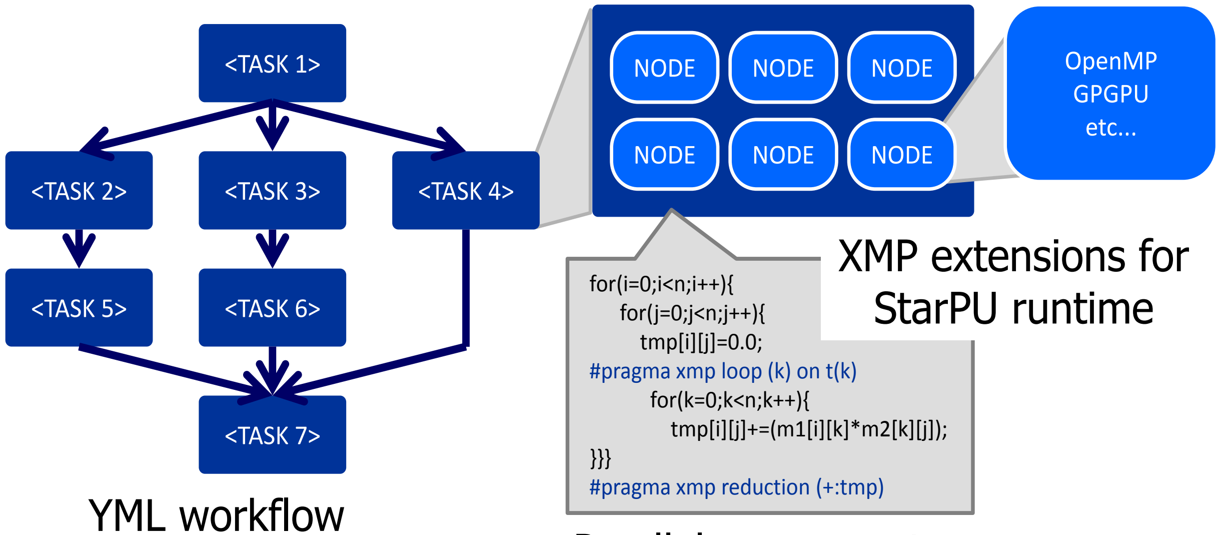
```
#pragma xmp (i, j) loop on t(j, i)
  for (i =0; i < 100; i++)
    for (j =0; j < 100; j++) ... = b[i][j];
```

```
#pragma xmp template t(0:99, 0:99)
#pragma xmp distribute t(BLOCK, BLOCK) onto p
```

```
#pragma xmp nodes p(4, 4)
```

A Selected Project Main Result in FP3C project

- FP2C (Framework for Post-Petascale Computing) : the multilevel programming as a solution for post-petascale system and XMP/StarPU integration
  - Enables to make use of a huge number of processors and attached accelerators in an efficient and hierarchical way.
  - Parallel algorithms in YML workflow language with parallel components written by XcalableMP (XMP) and its accelerators extensions supported by StarPU runtime technology in XMP language.
  - New algorithms implemented using this paradigm and evaluated on “K” RIKEN supercomputer in Kobe and the Hooper supercomputer in USA.



```
for(i=0;i<n;i++){
   for(j=0;j<n;j++){
      tmp[i][j]=0.0;
#pragma xmp loop (k) on t(k)
      for(k=0;k<n;k++){
         tmp[i][j]+=(m1[i][k]*m2[k][j]);
}}}
#pragma xmp reduction (+:tmp)
```

YML workflow programming

Parallel components (task) by XcalableMP

Concluding Remarks

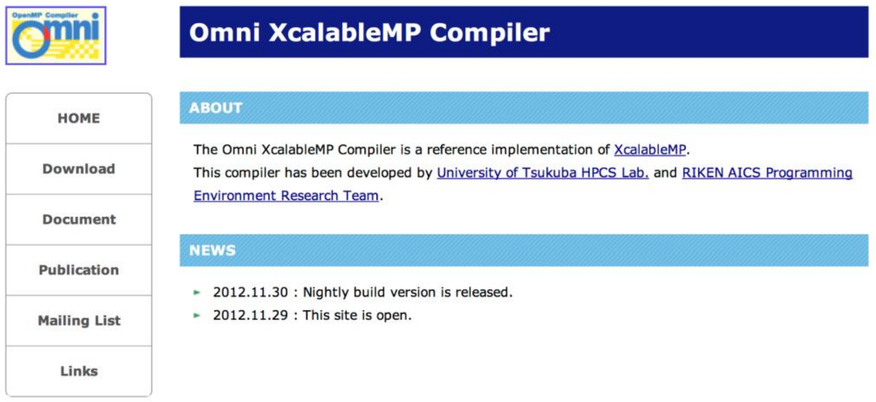
- Programming environment researches for peta-scale systems
  - XcalableMP PGAS parallel programming language for better "productive" parallel programming than "MPI".
  - "downgraded HPF" as a reflection of HPF experience in Japan
  - We expect that the PGAS runtime will improve the performance of larger-scale parallel programs in the K computer.
- XMP is also proposed for HPCI-FS accelerator architecture.

Thank you for your attention!

リファレンス実装について

- Omni XMP Compiler v0.6.1をリリース（2013年3月）
  - Coarray機能のストライド通信の対応
  - 京コンピュータにおけるCoarray機能のサポート
  - http://www.hpcs.cs.tsukuba.ac.jp/omni-compiler/xcalablemp/

- 2013年11月にOmni XMP Compiler v0.7をリリース
  - OpenMP指示文やOpenACC指示文との混在利用
  - 京コンピュータにおけるハードウェアサポートを利用した実装
  - XMP/Fortranの片側通信機能のサポートは、未



Omni XcalableMP Compiler

ABOUT

The Omni XcalableMP Compiler is a reference implementation of XcalableMP.
This compiler has been developed by University of Tsukuba HPCS Lab. and RIKEN AICS Programming Environment Research Team.

NEWS

- 2012.11.30 : Nightly build version is released.
- 2012.11.29 : This site is open.