The K computer and XcalableMP parallel language project

--- Towards programming environment for peta-scale computing ---

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

Outline

- The K computer : “Japanese next generation supercomputer “ project
  - Objective and organization
  - Hardware and software, applications
  - RIKEN Advanced Institute for Computational Science (AICS)

- Research on parallel programming languages
  - A short history of programming language research in Japan
  - Some thoughts about HPF
  - XcalableMP PGAS parallel programming language, with the K computer project

Objectives of the NGS (the K computer) project

- Design, build, and set up the **general-purpose** next-generation supercomputer to be one of **most powerful** supercomputers in the world. It will have a performance of 10 petaflops in the LINPACK benchmark with a system manufactured by **Fujitsu**.
- Develop and distribute **large-scale software applications** ("Grand Challenge" software) that exploit large fraction of the supercomputer
- Set up a research institute to run the supercomputer, to be an COE **institute** in computer science and computational science (AICS)

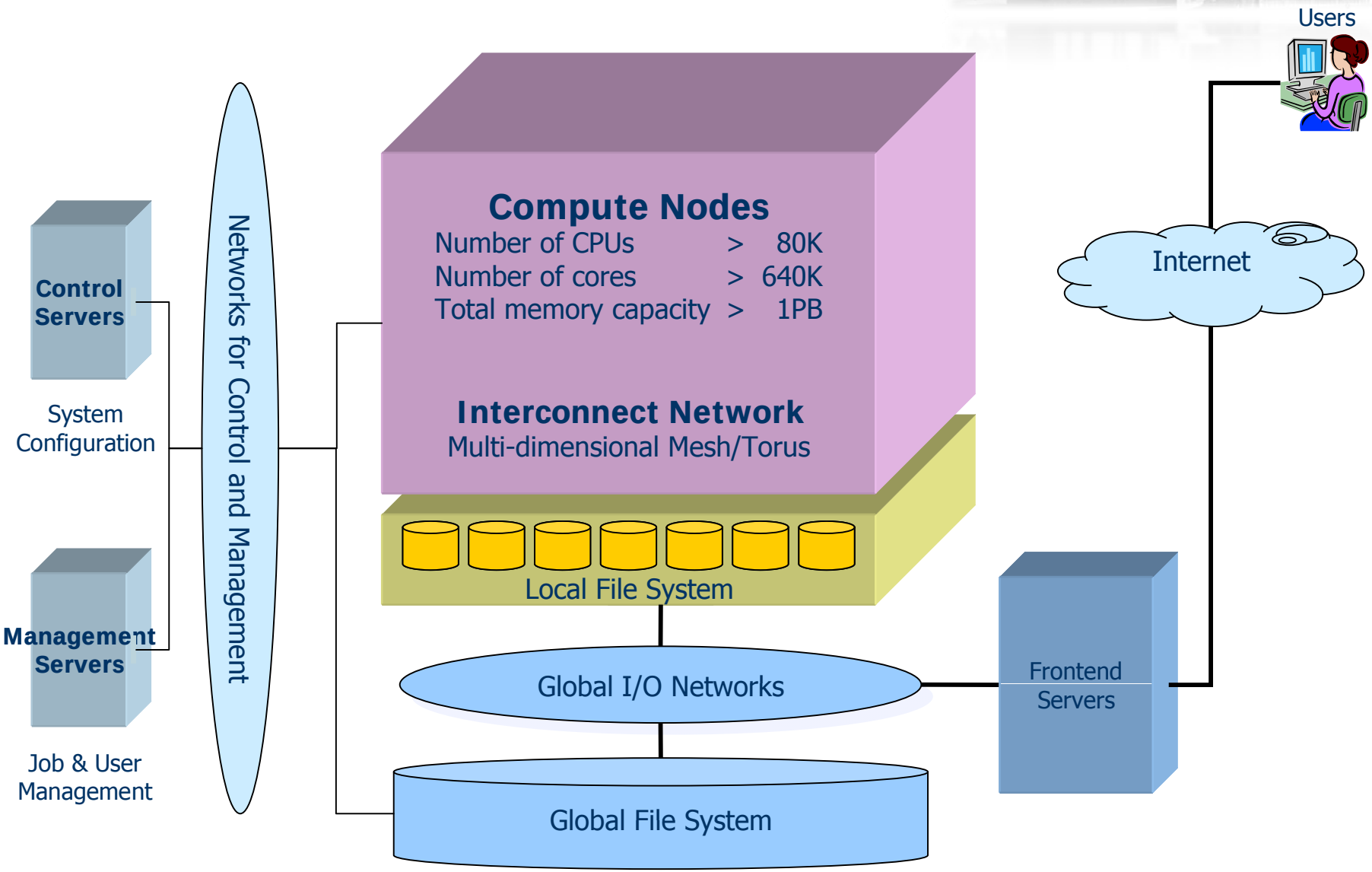
Nicknamed the "K computer"

Kei (京) represents the numerical unit of 10 Peta (10^{16}) in the Japanese language, representing the system's performance goal of 10 Petaflops. The Chinese character 京 can also be used to mean " a large gateway" so it could also be associated with the concept of a new gateway to computational science.

一、 十、 百、 千、 万、 億、 兆、 京、 垓、 杼、 穰、 溝、 澗、 正、 載、 極、
10^0 10^1 10^2 10^3 10^4 10^8 10^{12} 10^{16} 10^{20} 10^{24} 10^{28} 10^{32} 10^{36} 10^{40} 10^{44} 10^{48}

恒河沙、阿僧祇、那由他、不可思議、 無量大数
10^{52} 10^{56} 10^{60} 10^{64} 10^{68}

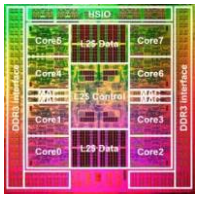
System Configuration



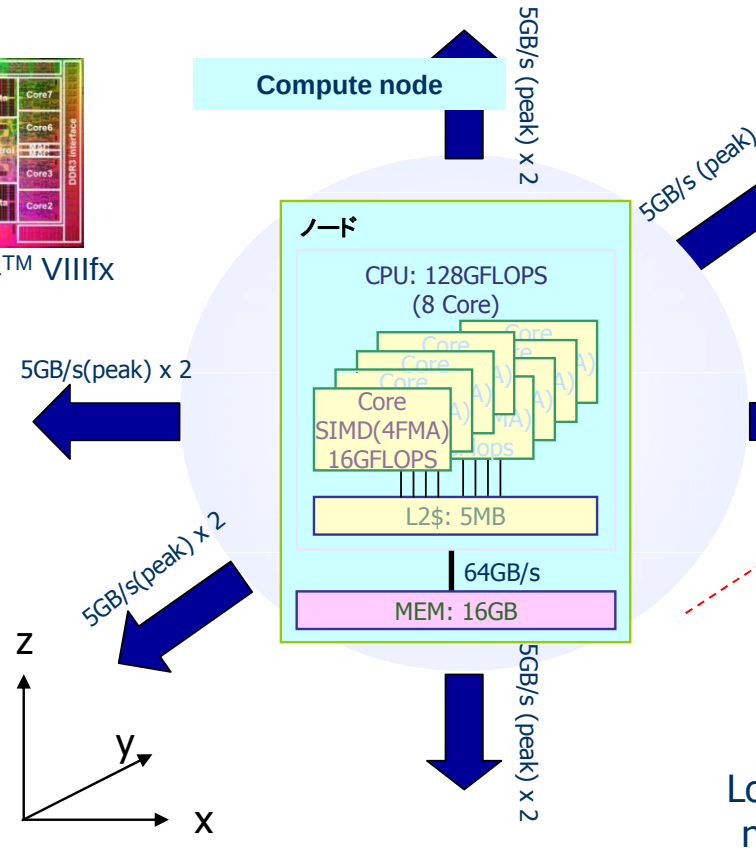
K computer: compute nodes and network

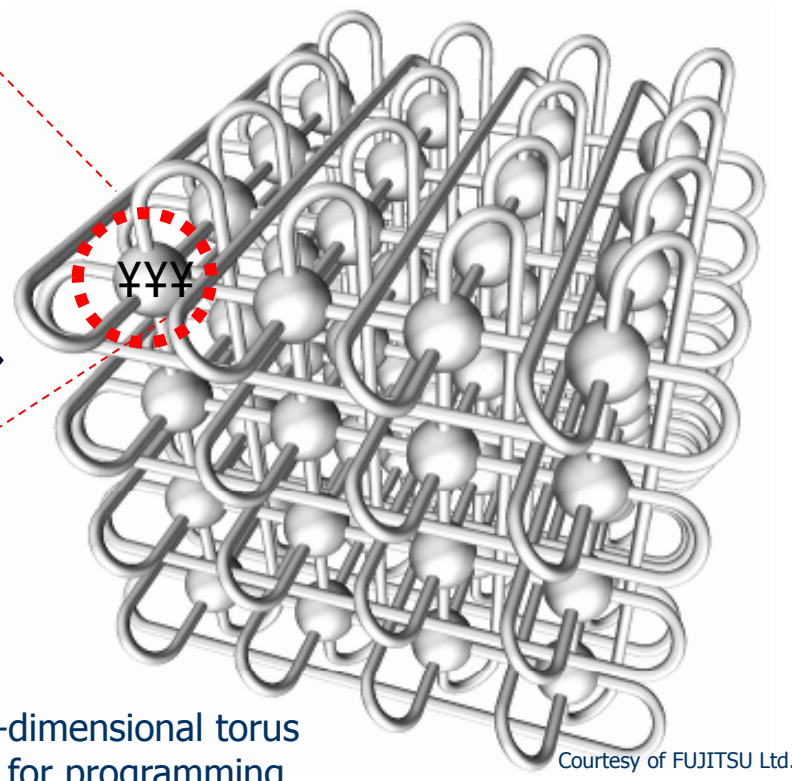
- Compute nodes (CPUs): > 80,000
  - Number of cores: > 640,000
- Peak performance: > 10PFLOPS
- Memory: > 1PB (16GB/node)

- Logical 3-dimensional torus network
- Peak bandwidth: 5GB/s x 2 for each direction of logical 3-dimensional torus network
- bi-section bandwidth: > 30TB/s
- Re-route mechanism for fault-tolerance



SPARC64™ VIIIfx





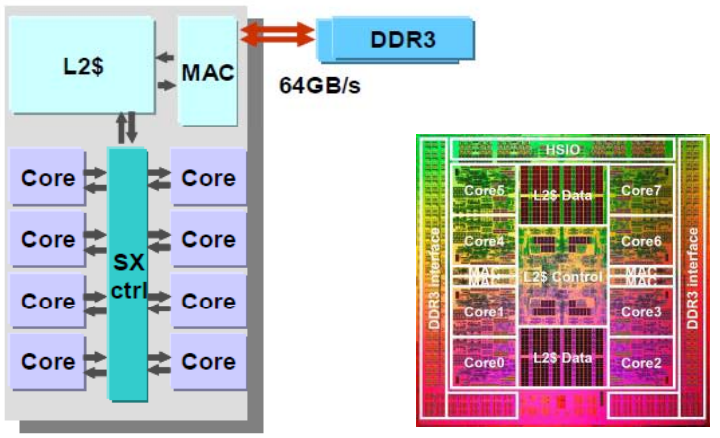
Logical 3-dimensional torus network for programming

Courtesy of FUJITSU Ltd.

CPU Features (Fujitsu SPARC64™ VIIIfx)

- 8 cores
- 2 SIMD operation circuit
  - 2 Multiply & add floating-point operations (SP or DP) are executed in one SIMD instruction
- 256 FP registers (double precision) to extract more parallelism
- Shared 5MB L2 Cache (10way)
  - Hardware barrier
    - Parallelization of inner loop by vectorization
  - Prefetch instruction
  - Software controllable cache
    - Sectored cache
      - Sector 0 : Normal cache access (default)
      - Sector 1 : Operand access explicitly specified by instructions
- Performance
  - 16GFLOPS/core, 128GFLOPS/CPU

16GF/core(2*4*2G)

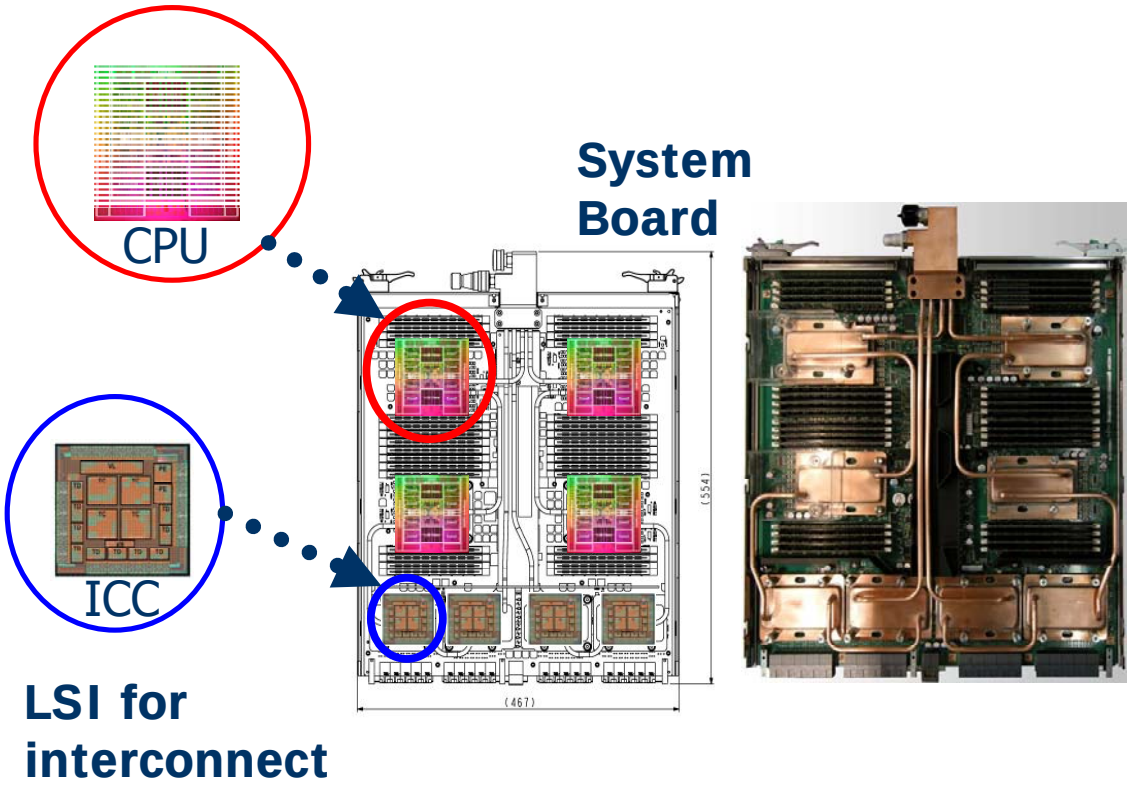


45nm CMOS process, 2GHz
22.7mm x 22.6mm
760 M transisters
58W（at 30℃ by water cooling)

Reference: SPARC64™ VIIIfx Extensions
http://img.jp.fujitsu.com/downloads/jp/jhpc/sparc64viiifx-extensions.pdf

Implementation: Board and Rack

- Water-cooled CPU &ICC and Memory air-cooled
- Several system boards are compiled and set into a cabinets.



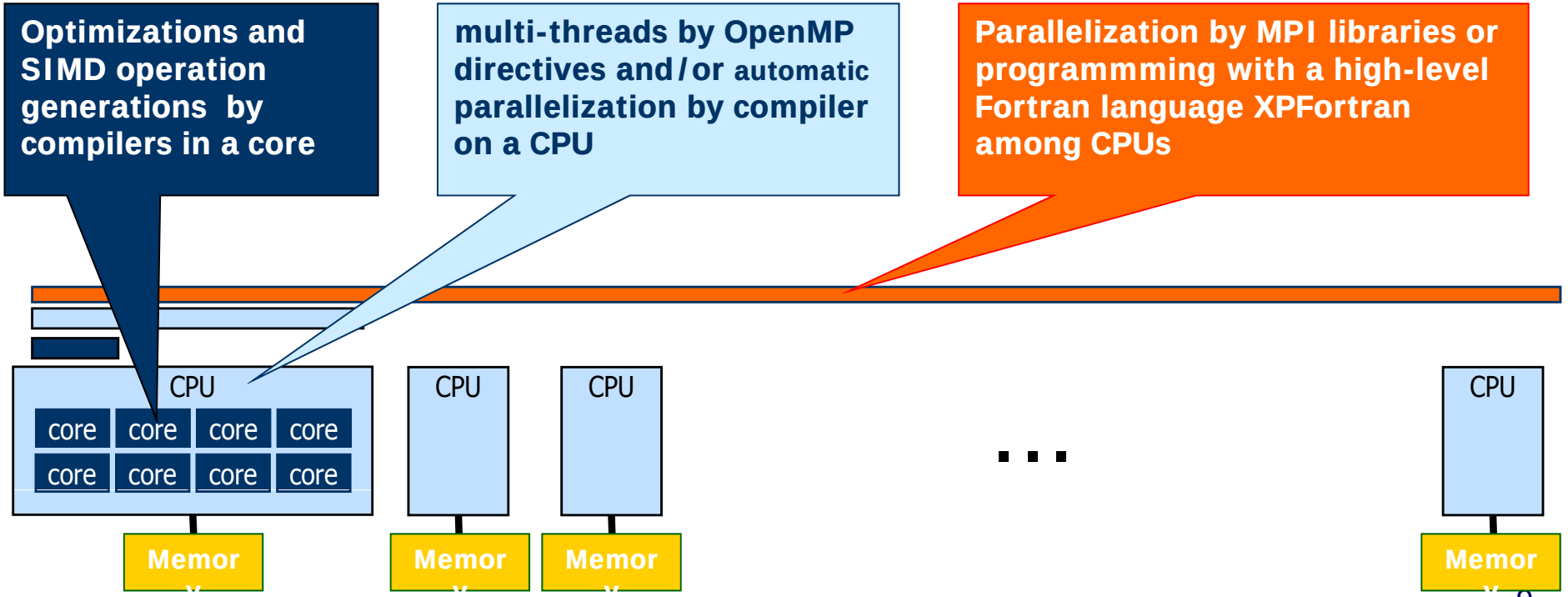
Courtesy of FUJITSU Ltd.

Highlights: the Massive Parallel System to Meet Various Application Environments

- Sustained Peta-Flops System in Real Applications
  - High-Performance/Low Power CPU with 8 cores : 128GFlops@2GHz, 58W
  - High Throughput/Low Latency Torus Network (Tofu), 1μsec latency in MPI
  - Optimized Compilers and Libraries : Fortran, C/C++,MPI,BLAS andLAPACK
- Highly Reliable System
  - Low Operating Temperature in CPU/ICC : 30℃ by Water Cooling
  - Auto-Recovery Functions with Strict Error Detections
  - Reliable Torus Network with Auto-Rerouting
  - Back-up Servers and Dual Data Paths in I/O
- Highly Efficient and Usable System for Diverse Work Loads
  - Distributed Parallel File System
  - Hierarchical I/O System with Staging Functions
  - Efficient Job Scheduler to Support 3-D Torus Network
  - Unified Portal System to Support Application Development, File Handling,
  - Job & Resource Monitoring, etc

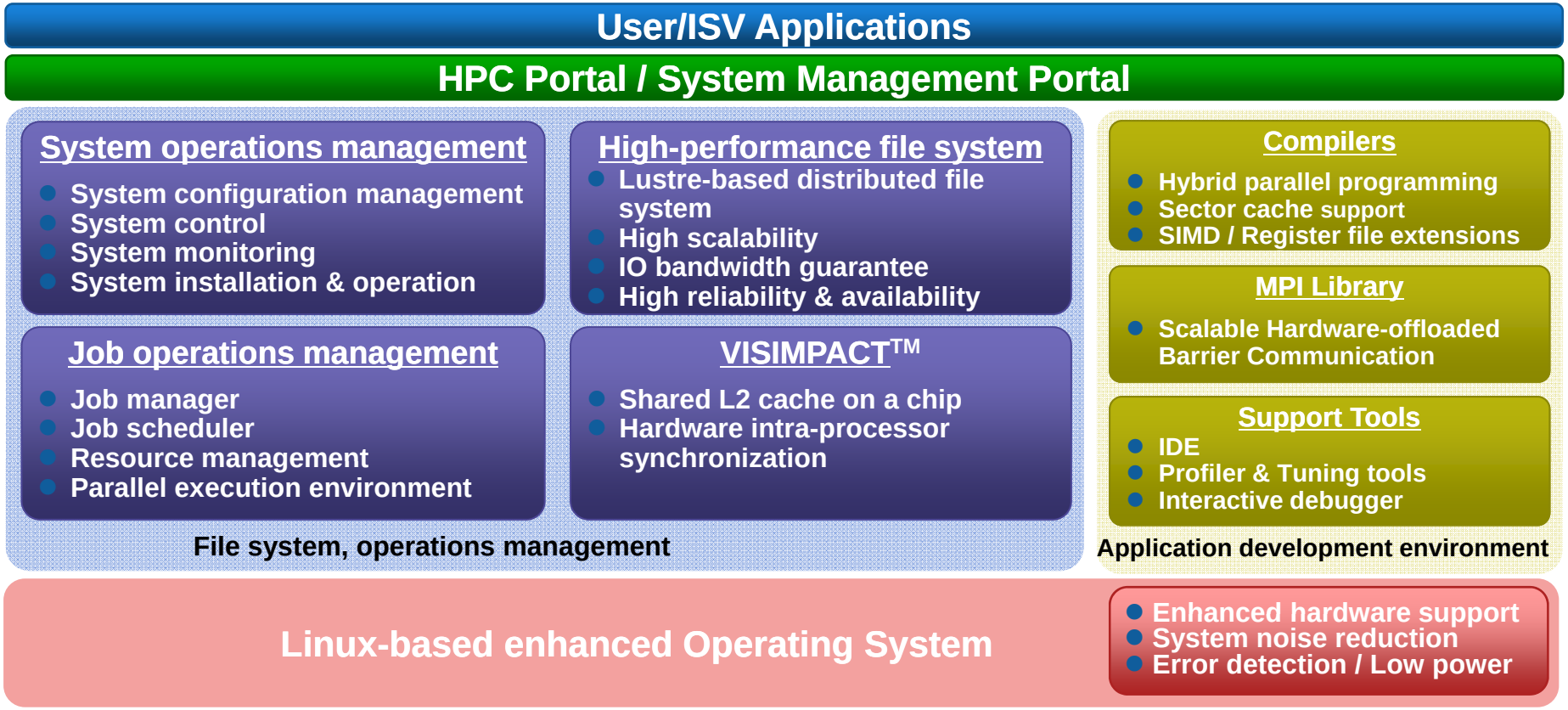
Programming models for the K computer

- A hybrid programming model with multi-threads and MPI is strongly recommended.
  - Too many MPI processes may cause the overhead (and trouble).
- A flat programming model by MPI only is also supported (not recommend)
- Automatic parallelizing compiler for inside-node is supported.
- XFP (Fujitsu dialects) (and XcalableMP, proposed)



System Software Stack

FUJITSU



K computer Delivery Began in Late September 2010

- The first eight racks of the K computer were delivered to Kobe from Fujitsu on September 28, 2010. More than 800 racks are required for a 10 Peta Flops Performance.
- A computer rack weighs about 1,300 kg in average. The rack contains 96 water-cooled Fujitsu SPARC64 VIIIfx CPU chips, each of which performs 128 GFlops, interconnected with the 3D Torus network that Fujitsu named Tofu.

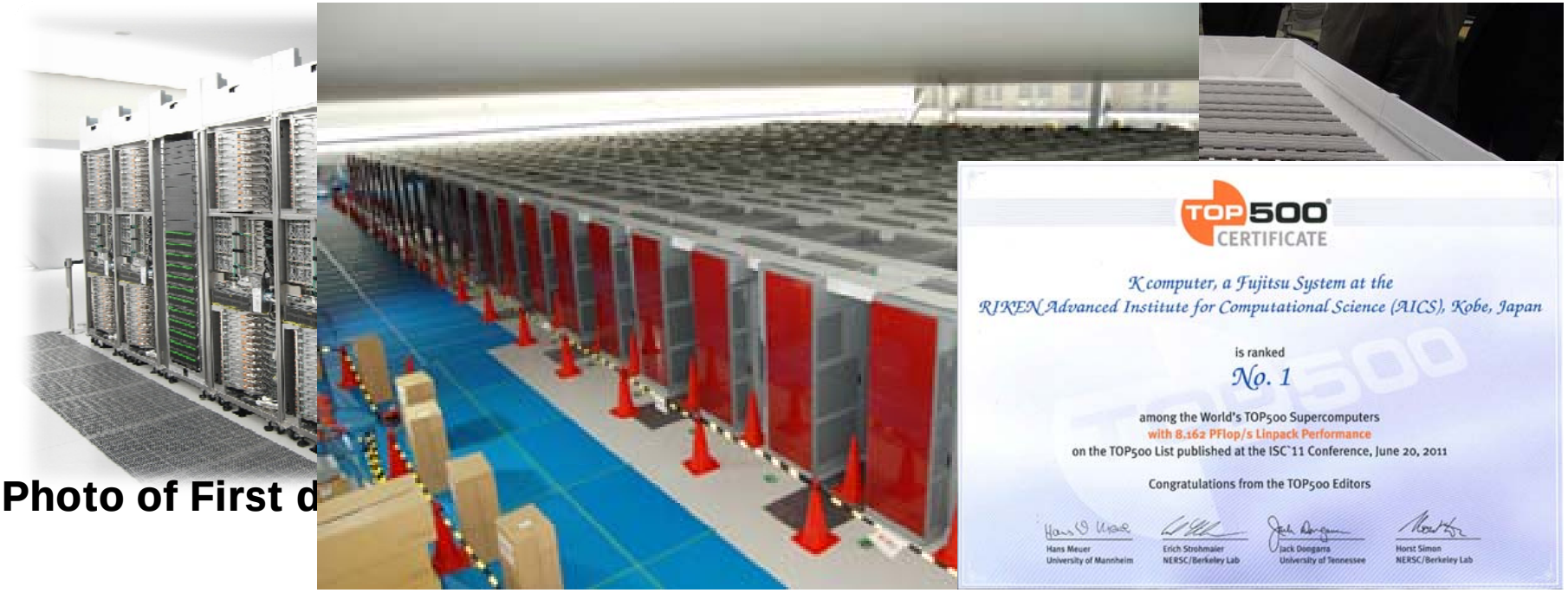
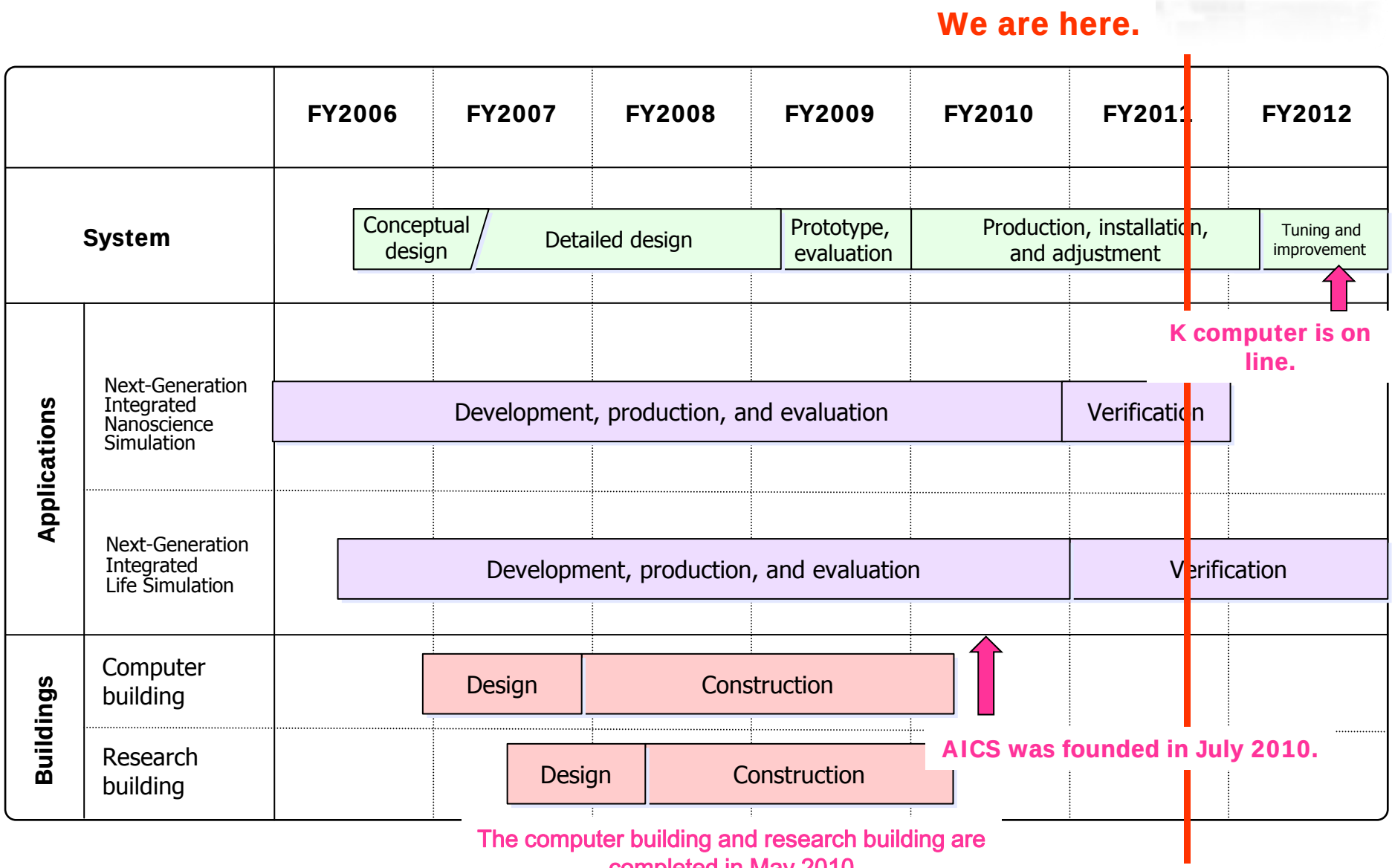


Photo of First d

Schedule of development

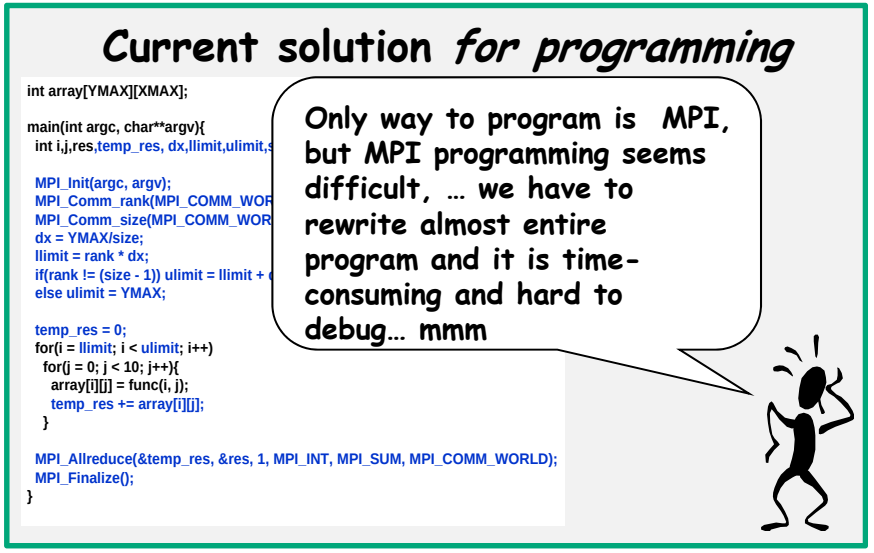


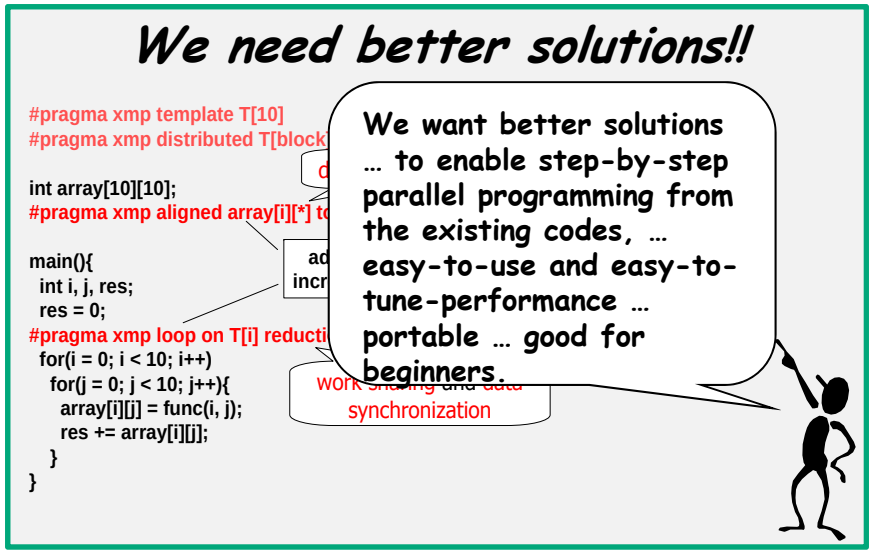
Programming Language projects for HPC in Japan

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, …

is our solution!





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.

The history of HPC language projects in Japan

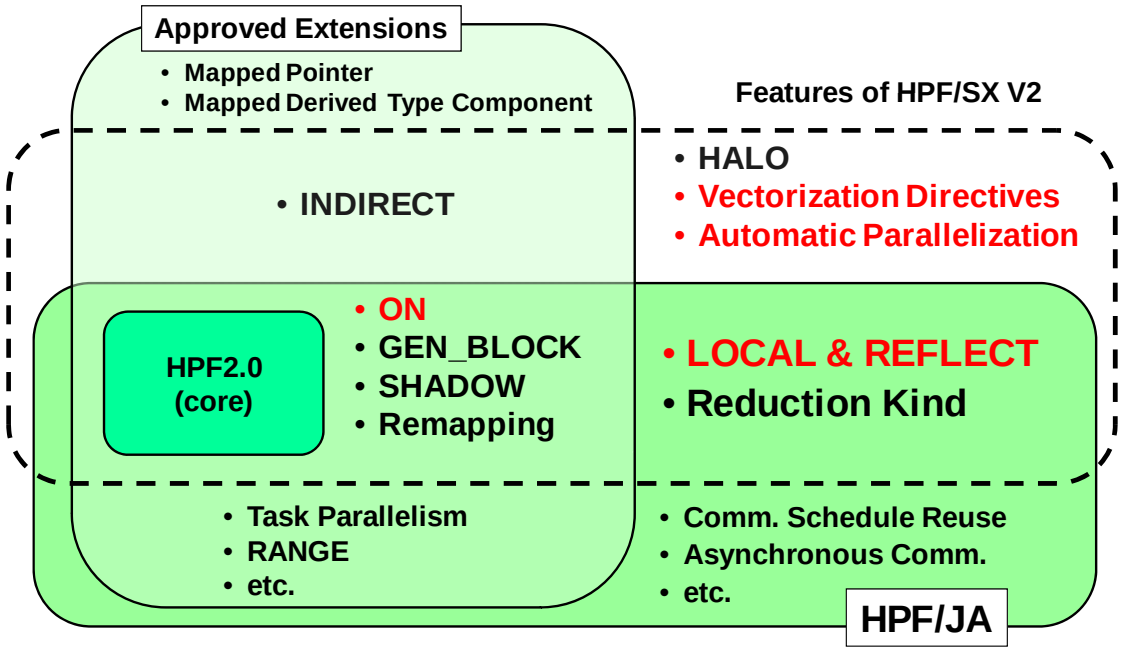
- VPPFortran for NWT (VPP500)
  - NWT(Numerical Wind Tunnel), a parallel Vector machine for CFD, 1st machine in Top500 (1993/Nov to 1995/Nov)
  - Fortran extensions for NWT, specifying global and local memory dedicated to VPP, proposed by Fujitsu
  - Renamed to XPFortran as a Fujistu product

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)

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

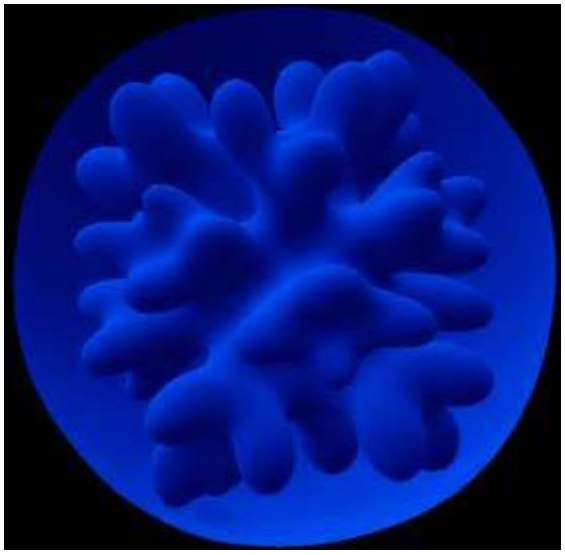
- IMPACT-3D: an implosion analysis code using TVD scheme
  - three-dimensional compressible and inviscid Eulerian fluid computation with the explicit 5-point stencil scheme for spatial differentiation
  - fractional time step for time integration.

- 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

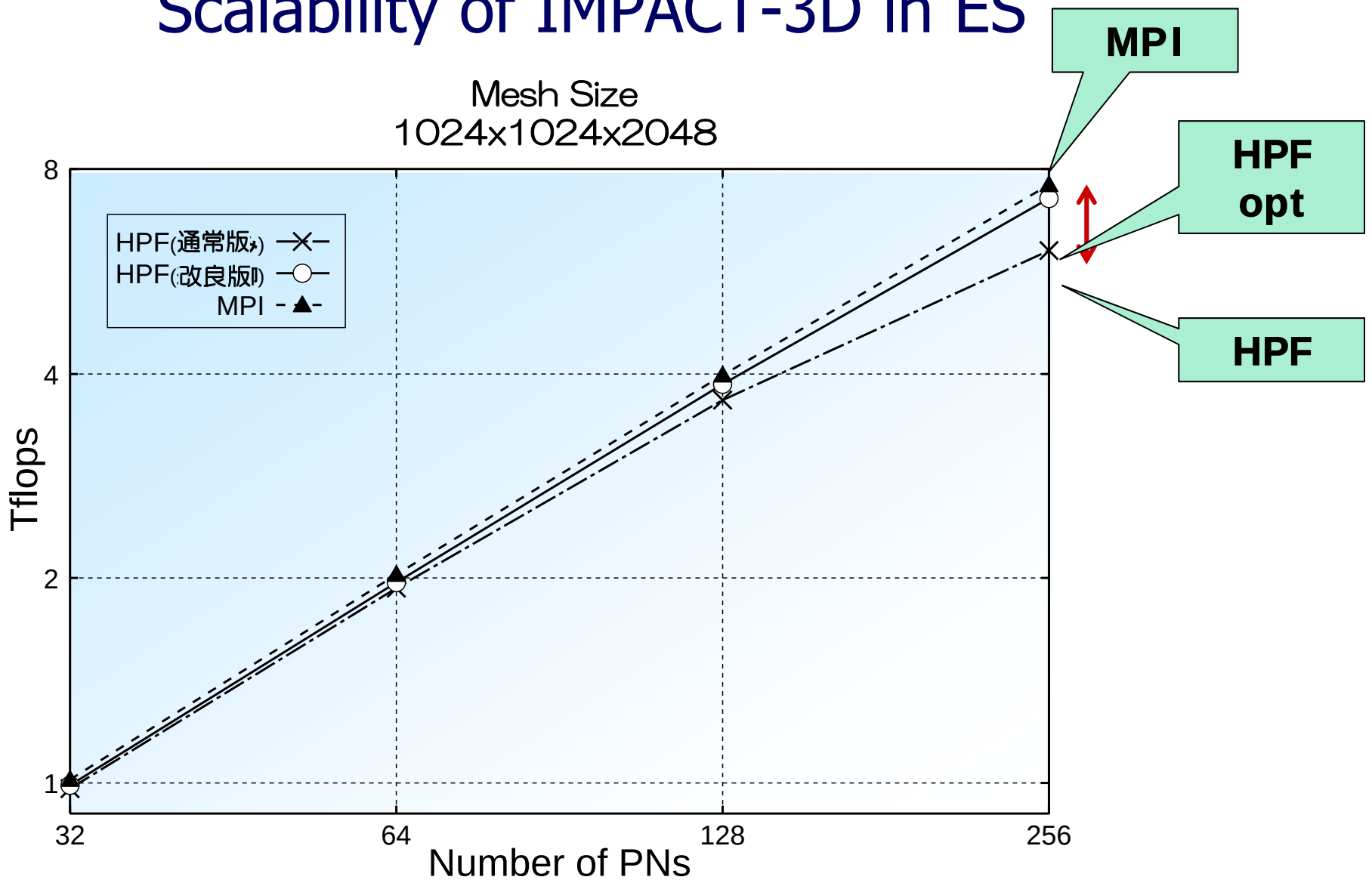
- Optimize communication by **REFLECT** and **LOCAL**
  - **REFLECT** explicitly updates SHADOW, with re-use of communication schedule
  - The LOCAL directive guarantees the accesses to arrays in a list do not require inter-processor communications.
  - The user can eliminate redundant communications 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.

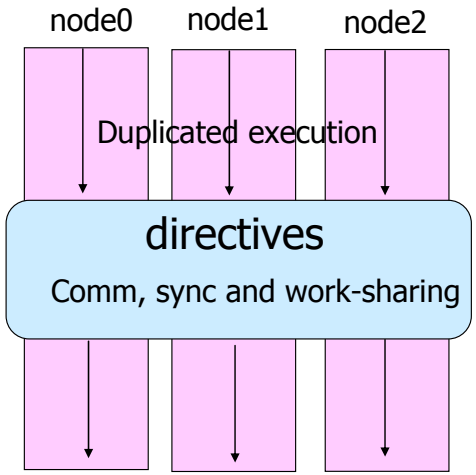
- PGAS is emerging and geting attentions from the community
  - Model for scalable communication (than MPI?)

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 += …;
   } }
```

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];
    }
}
```

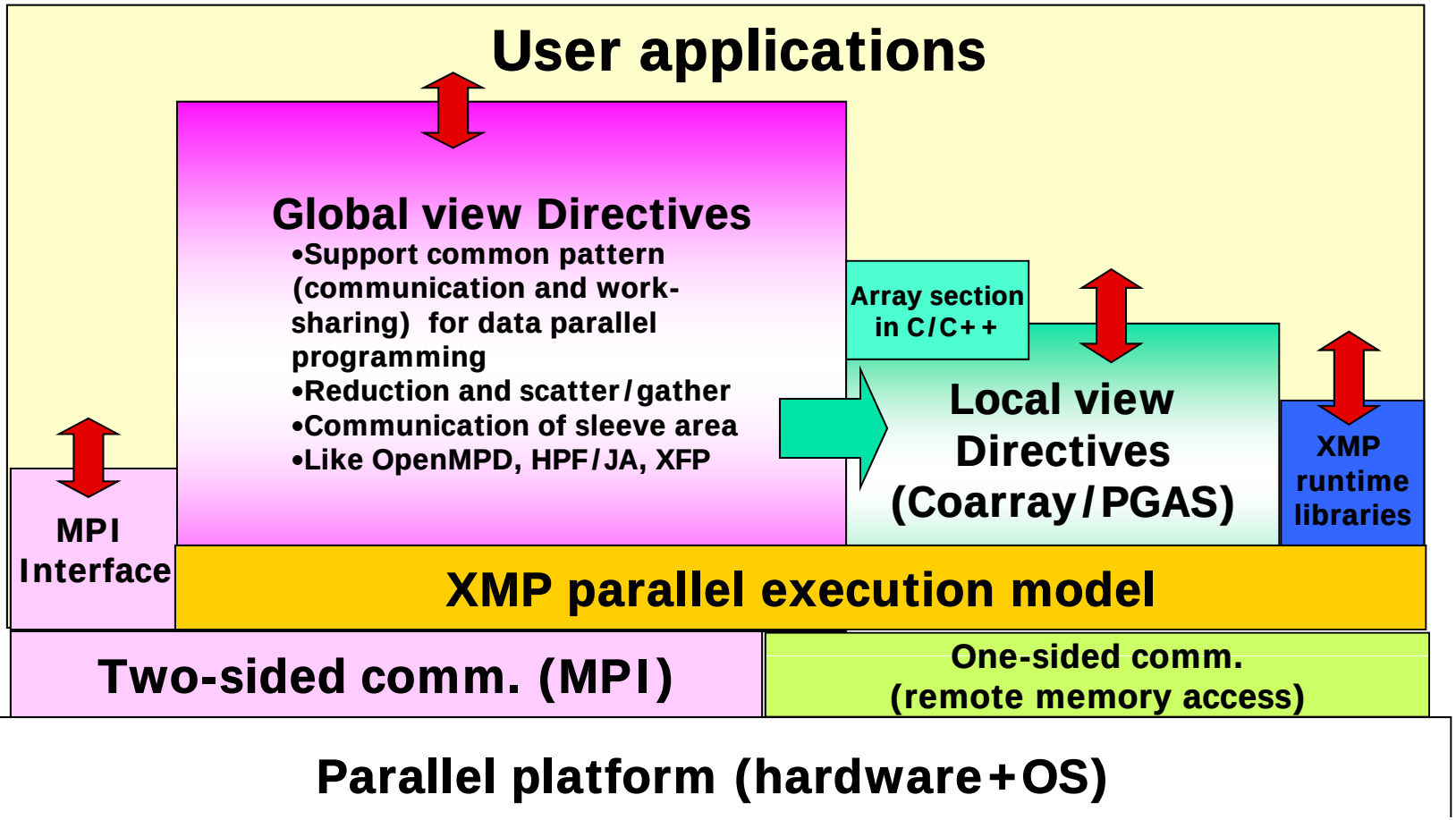
data distribution

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

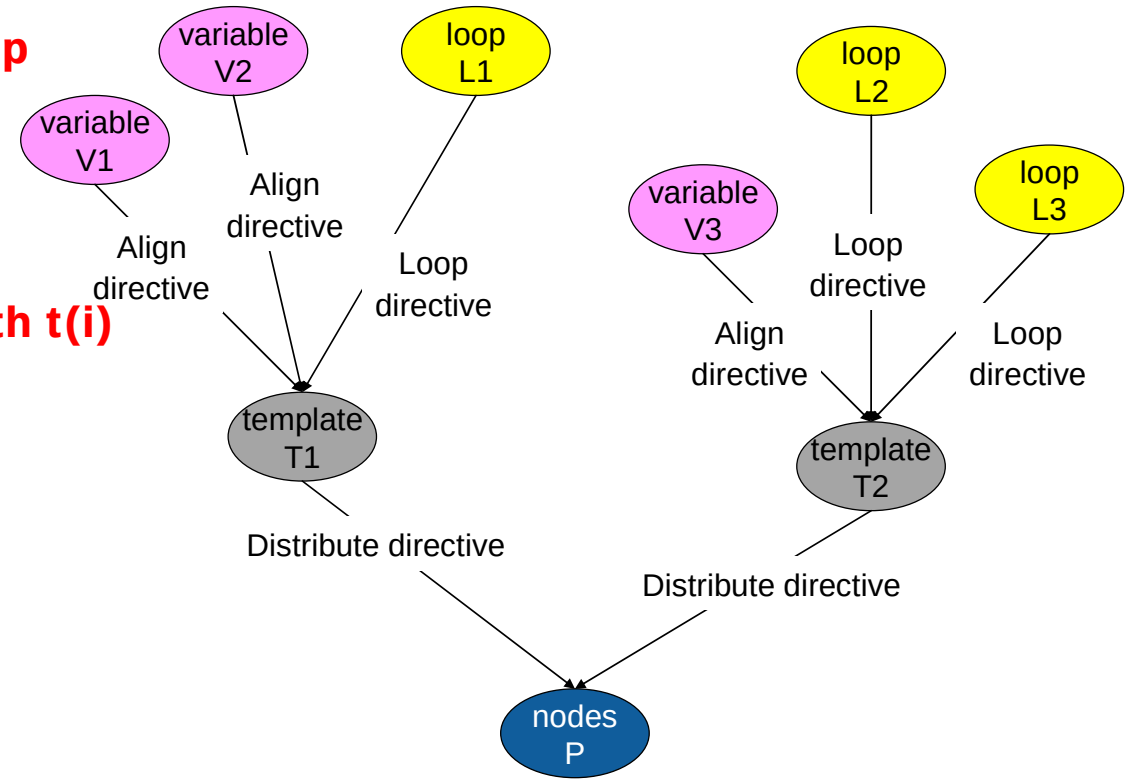
```
#pragma xmp template t(100)
#pragma distribute t(block) onto p
```

- A global data is aligned to the template

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

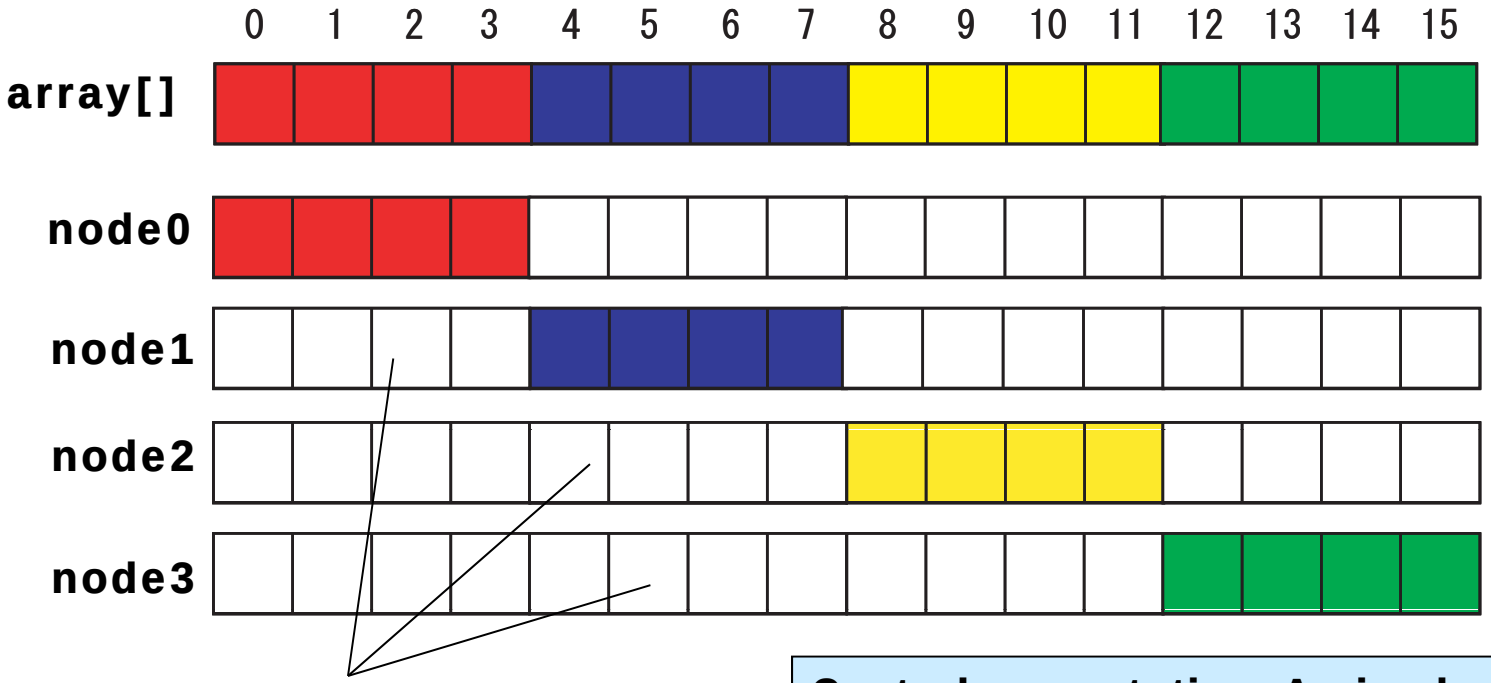
- Loop iteration must also be aligned to the template by on-clause.

```
#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

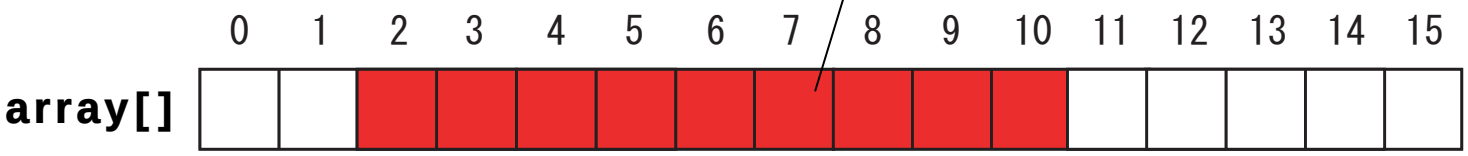
Parallel Execution of “for” loop

- Execute for loop to compute on array

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

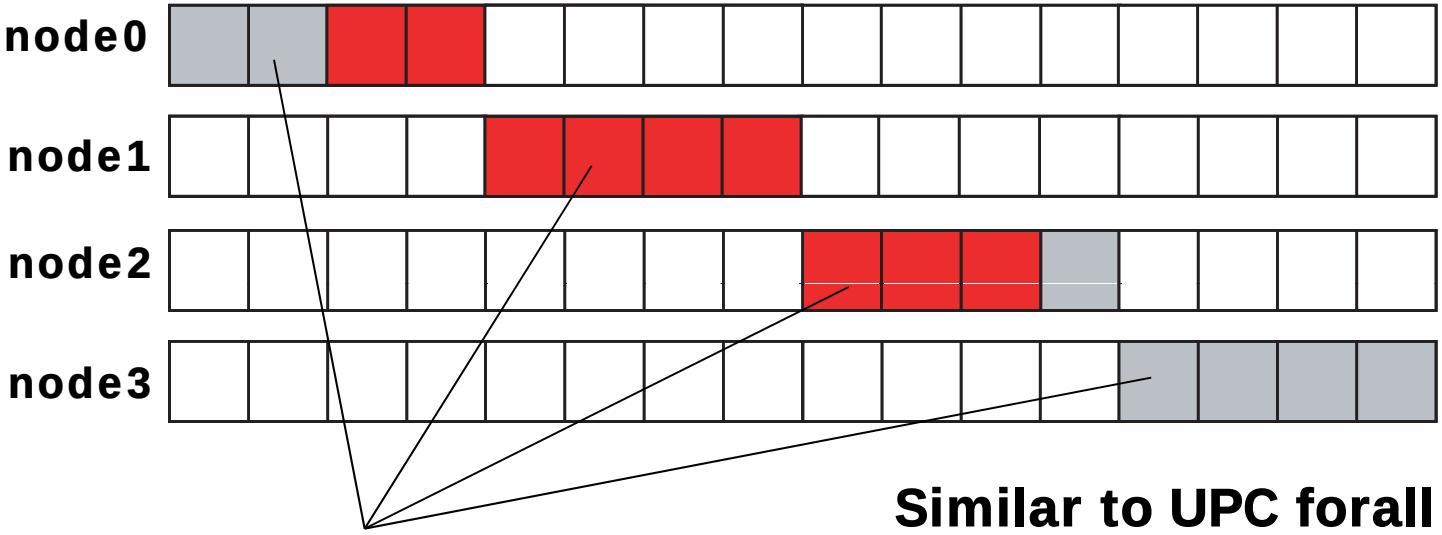
```
#pragma xmp loop on t(i)
for(i=2; i <=10; i++)
```

Data region to be computed by for loop



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

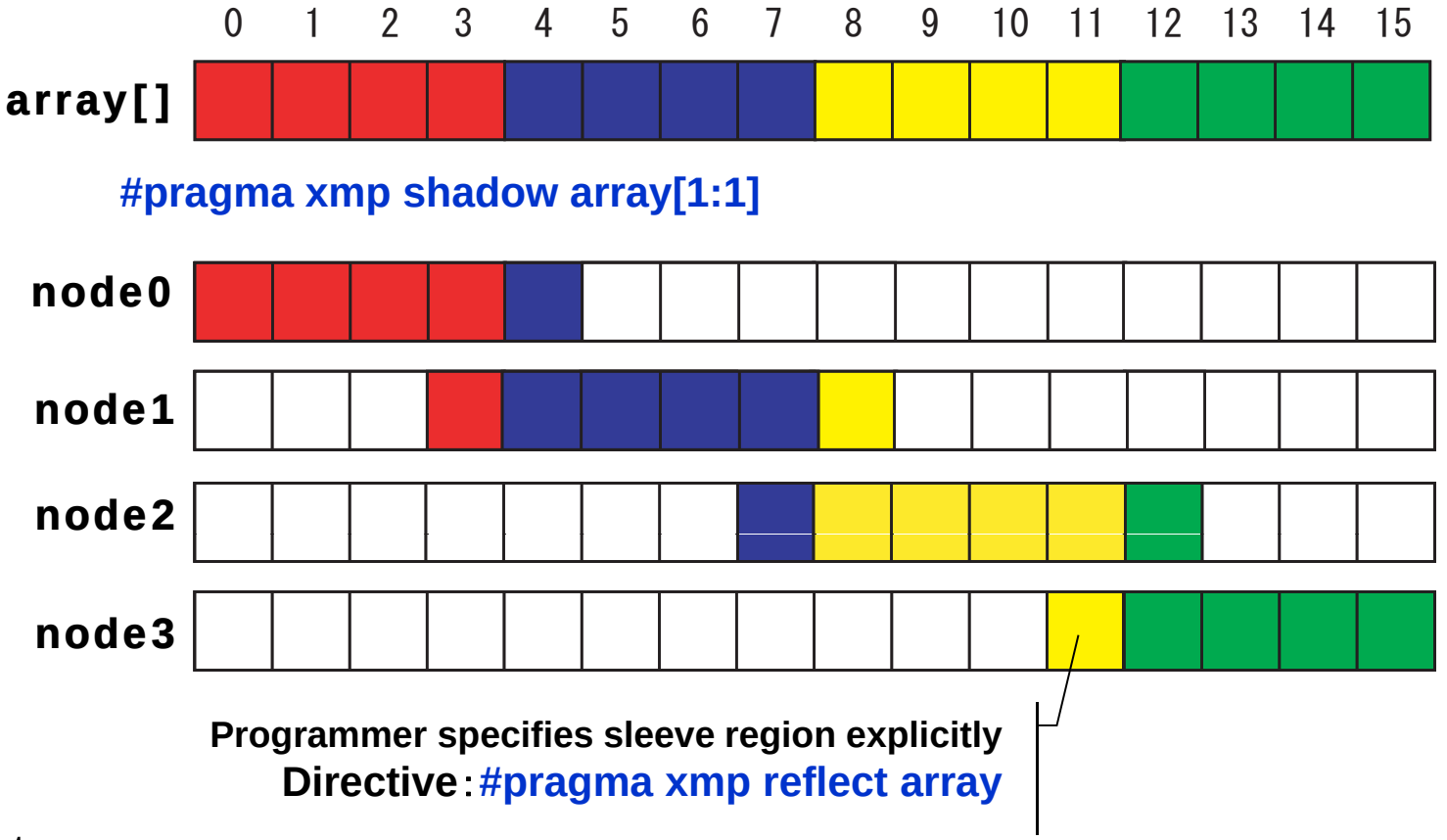
```
#pragma xmp loop on t(i)
```



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]

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



XcalableMP Global view directives

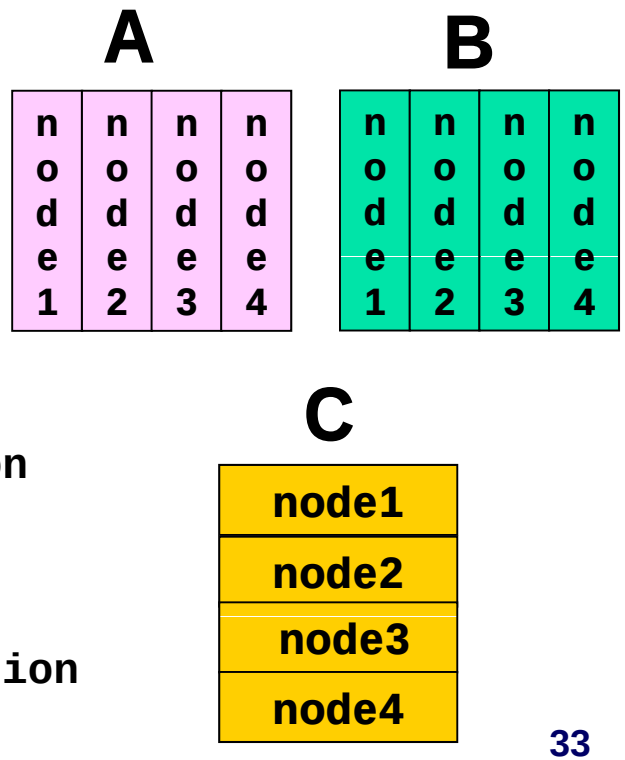
- Execution only master node
  - #pragma xmp block on master
- Broadcast from master node
  - #pragma xmp bcast (*var*)
- Barrier/Reduction
  - #pragma xmp reduction (*op: var*)
  - #pragma xmp barrier
- Global data move directives for collective comm./get/put
- Task parallelism
  - #pragma xmp task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

Status of XcalableMP

- Status of XcalableMP WG
- Discussion in monthly Meetings and ML
- XMP Spec Version 1.0 was published (at SC11).
- It includes XMP-IO and multicore extension as a proposal in ver 1.0.

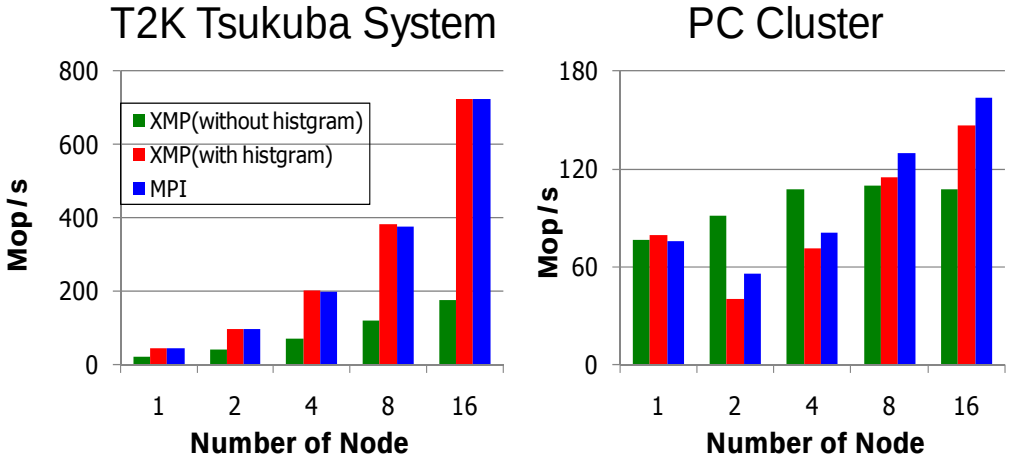
- Compiler & tools
- XMP prototype compiler (xmpcc version 0.5) for C is available from U. of Tsukuba.
- Fortran will be coming soon!
- Open-source, C to C source compiler with the runtime using MPI

- Codes and Benchmarks
- NPB/XMP, HPCC benchmarks, Jacobi ..

- Platforms supported
- Linux Cluster, Cray XT5 …
- Any systems running MPI. The current runtime system designed on top of MPI

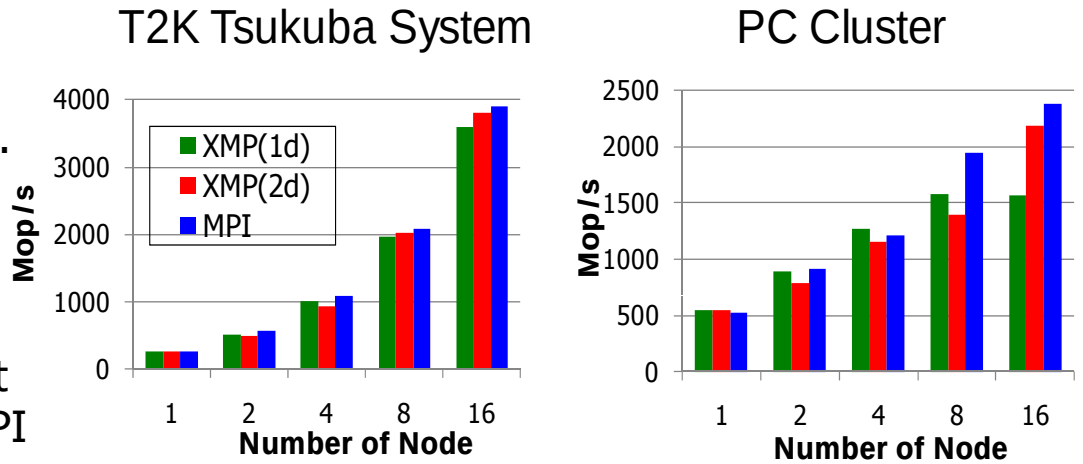
NPB IS performance

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



NPB CG performace

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



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

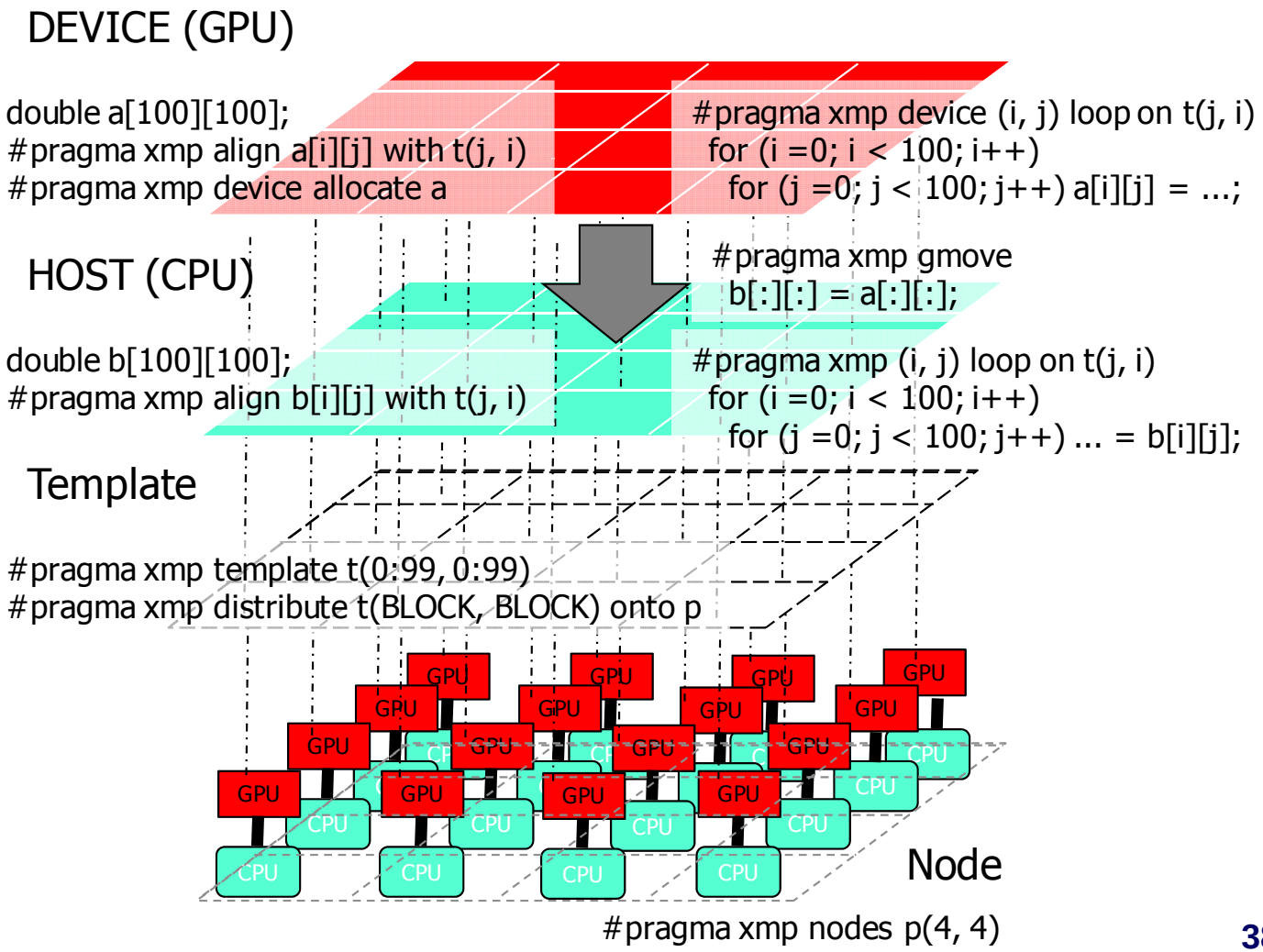
- Interface to existing (MPI) libraries
  - Rewriting every problem in XMP is not realistic.
  - Use of existing high performance parallel libraries written in MPI is crucial.
  - We are working on the design of interfaces for Scalapack, MUMPS, … etc.
- XMP IO
  - IO for distributed array
  - Interface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



Concluding Remarks

- K-computer project
  - The installation of the K computer are still going on.
  - … and some projects are getting started for post-petascale and exascale.
- Programming environment researches for the K computer
  - 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.
  - Collaboration and feedback with application researchers are key to success
    - For improvement and dissemination of our software!