XcalableMP Ver. 1.0

Highly Productive Parallel Programming Language

Masahiro NAKAO
Center for Computational Sciences
University of Tsukuba

Background

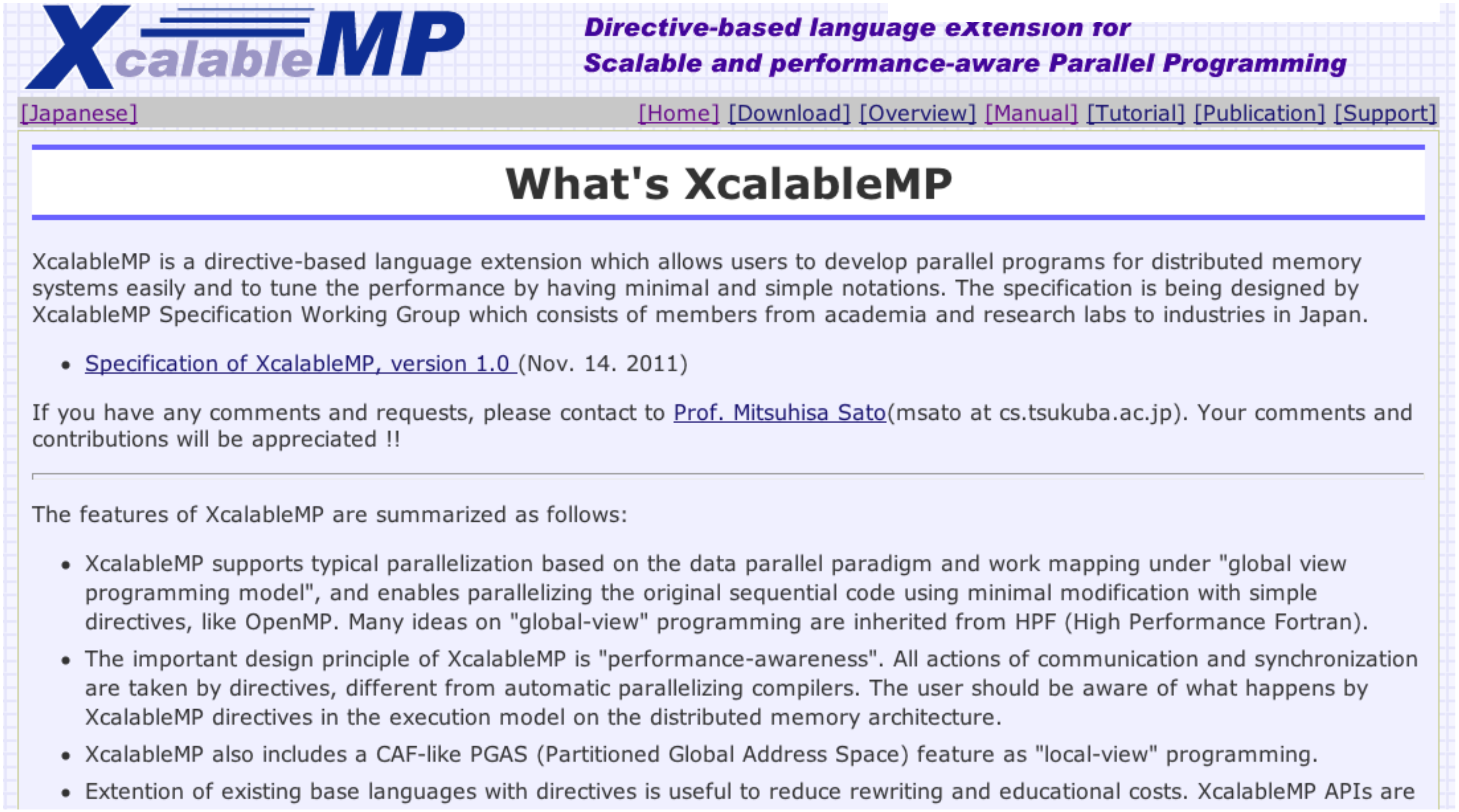
- MPI is widely used as a parallel programming model on distributed memory systems
  - Time-consuming
  - Complicated process
- Another programming model is needed !!
  - High performance
  - Easy to program (High productivity)

→ Development of XcalableMP (XMP)

e-Science Project

XcalableMP

- XMP Working Group designs XMP specification
    - XMP Working Group consists of members from
        - academia : U. Tsukuba, U. Tokyo, Kyoto U. and Kyusyu U.
        - research labs : RIKEN AICS, NIFS, JAXA, JAMSTEC/ES
        - industries : Fujitsu, NEC, Hitachi
    - Specification Version 1.0 is released !! (Nov. 2011)
- University of Tsukuba develops an Omni XMP compiler as a reference implementation
    - the K Computer, CRAY platforms(HECToR), Linux cluster
- Evaluation of Performance and Productivity


XcalableMP
Directive-based language extension for
Scalable and performance-aware Parallel Programming
[Japanese]
[Home] [Download] [Overview] [Manual] [Tutorial] [Publication] [Support]

What's XcalableMP

XcalableMP is a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and to tune the performance by having minimal and simple notations. The specification is being designed by XcalableMP Specification Working Group which consists of members from academia and research labs to industries in Japan.

- Specification of XcalableMP, version 1.0 (Nov. 14. 2011)

If you have any comments and requests, please contact to Prof. Mitsuhisa Sato(msato at cs.tsukuba.ac.jp). Your comments and contributions will be appreciated !!

The features of XcalableMP are summarized as follows:

- XcalableMP supports typical parallelization based on the data parallel paradigm and work mapping under "global view programming model", and enables parallelizing the original sequential code using minimal modification with simple directives, like OpenMP. Many ideas on "global-view" programming are inherited from HPF (High Performance Fortran).
- The important design principle of XcalableMP is "performance-awareness". All actions of communication and synchronization are taken by directives, different from automatic parallelizing compilers. The user should be aware of what happens by XcalableMP directives in the execution model on the distributed memory architecture.
- XcalableMP also includes a CAF-like PGAS (Partitioned Global Address Space) feature as "local-view" programming.
- Extention of existing base languages with directives is useful to reduce rewriting and educational costs. XcalableMP APIs are

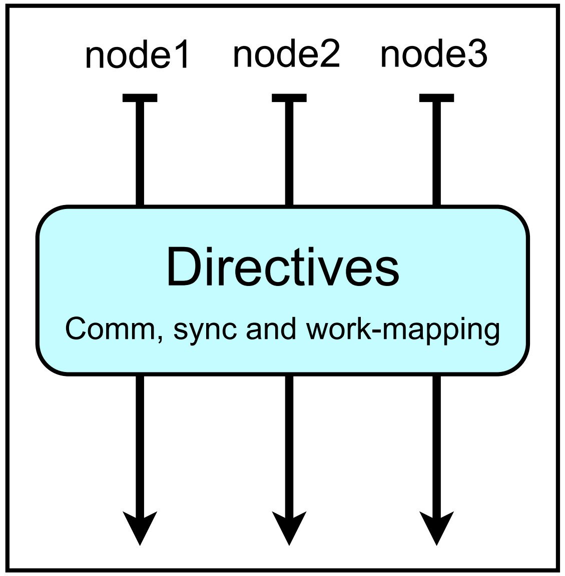
http://www.xcalablemp.org/

Agenda

- Overview of XMP
- XMP Programming Model
- Evaluation of Performance and Productivity of XMP

Overview of XMP

- XMP is a directive-based language extension like OpenMP and HPF based on C and Fortran95
  - To reduce code-writing and educational costs
- The basic execution model of XMP is SPMD
  - A thread starts execution in each node independently (as in MPI)
- "Performance-awareness" for explicit communication, sync. and work-mapping
  - All actions occur when thread encounters directives or XMP's extended syntax
  - XMP compiler generates communication only where a user inserts them to facilitate performance tuning



XMP Code Example

XMP C version

```
int array[100];
#pragma xmp nodes p(*)
#pragma xmp template t(0:99)
#pragma xmp distribute t(block) onto p
#pragma xmp align array[i] with t(i)

main(){
#pragma xmp loop on t(i) reduction(+:res)
   for(i = 0; i < 100; i++){
      array[i] = func(i);
      res += array[i];
   }
}
```

data distribution

work mapping & reduction

XMP Code Example

XcalableMP

XMP Fortran version

```
real a(100)
!$xmp nodes p(*)
!$xmp template t(100)
!$xmp distribute t(block) onto p
!$xmp align a(i) with t(i)
 :

!$xmp loop on t(i) reduction(+:res)
   do i=1, 100
      a(i) = func(i)
       res = res + a(i)
   enddo
```

data distribution

work mapping & reduction

The same code written in MPI

```
int array[100];

main(int argc, char **argv){
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    dx = 100/size;
    llimit = rank * dx;
    if(rank != (size -1)) ulimit = llimit + dx;
    else ulimit = 100;

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

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

Agenda

- Overview of XMP
- XMP Programming Model
- Evaluation of Performance and Productivity of XMP

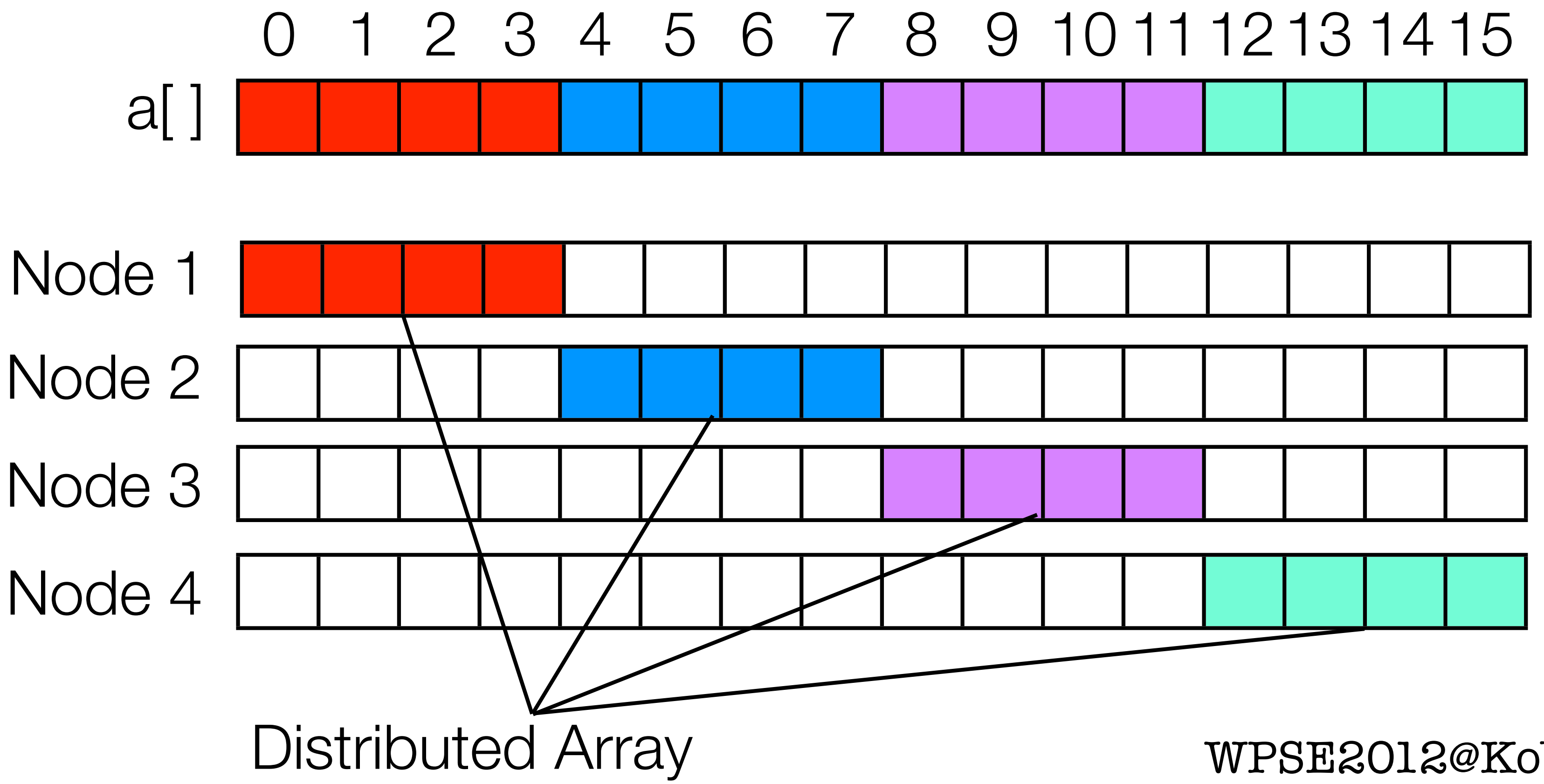
Programming Model

- Global View Model (like as HPF)
  - programmer describes data distribution, work mapping, communication and sync. by using directives
    - supports typical techniques for data-mapping and work-mapping
    - rich communication and sync. directives, such as “shadow”, “reflect” and “gmove”
- Local View Model (like as Coarray Fortran)
  - enables programmer to transfer data by using one-sided comm. easily

Data Distribution

- The directives define a data distribution among nodes

```
#pragma xmp nodes p(4)
#pragma xmp template t(0:15)
#pragma xmp distribute t(block) on p
#pragma xmp align a[i] with t(i)
```

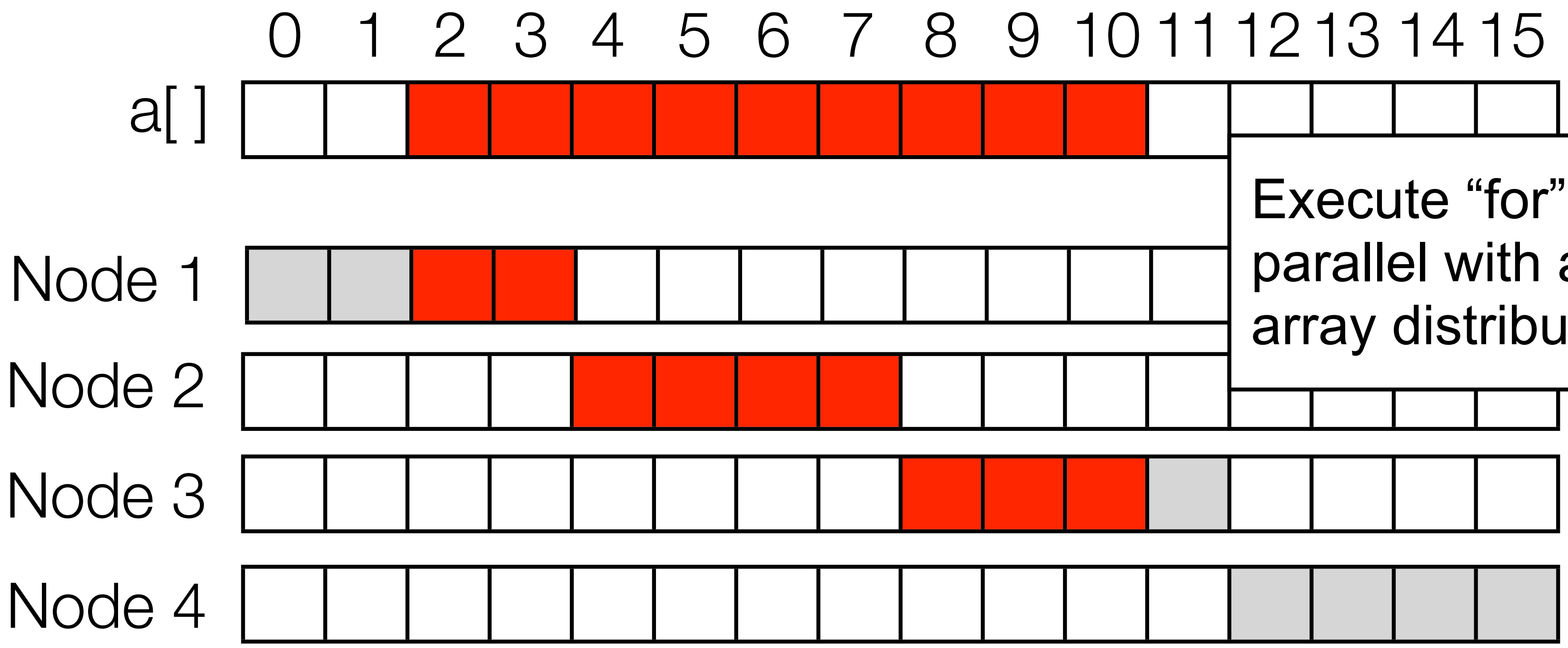


Parallel Execution of loop XcalableMP

- Loop directive is inserted before loop statement

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

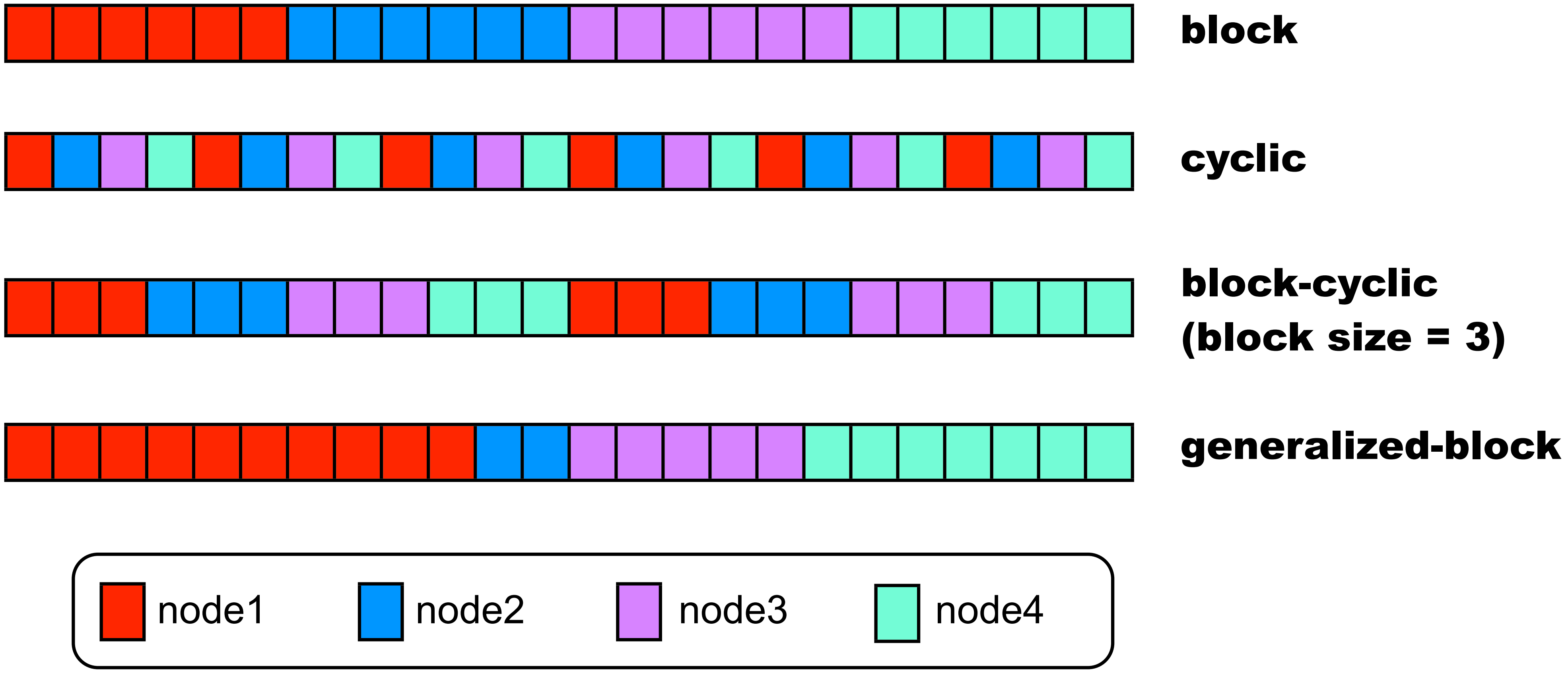
```
#pragma xmp nodes p(4)
#pragma xmp template t(0:15)
#pragma xmp distribute t(block) on p
#pragma xmp align a[i] with t(i)
```



Execute “for” loop in parallel with affinity to array distribution

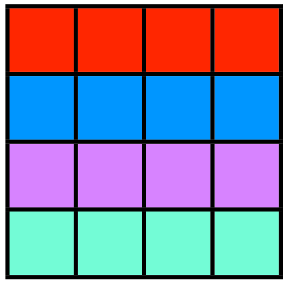
Each node computes Red elements in parallel

Example of Data Mapping

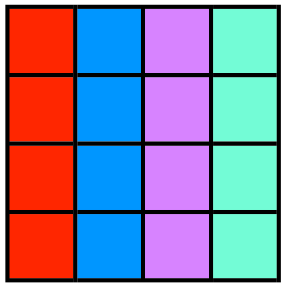


Multi Dimensional Array

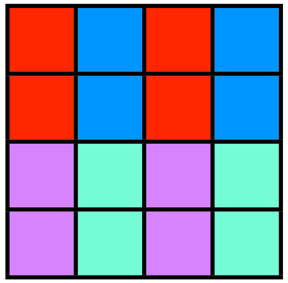
```
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i][*] with t(i)
```



```
#pragma xmp distribute t(block) onto p
#pragma xmp align a[*][i] with t(i)
```



```
#pragma xmp distribute t(block,cyclic) onto p
#pragma xmp align a[i][j] with t(i,j)
```

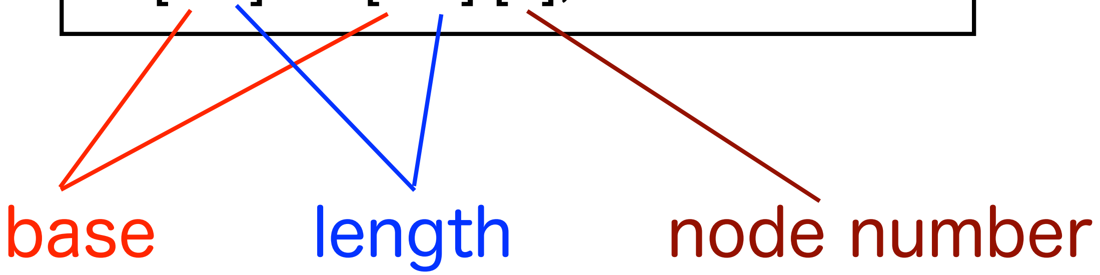


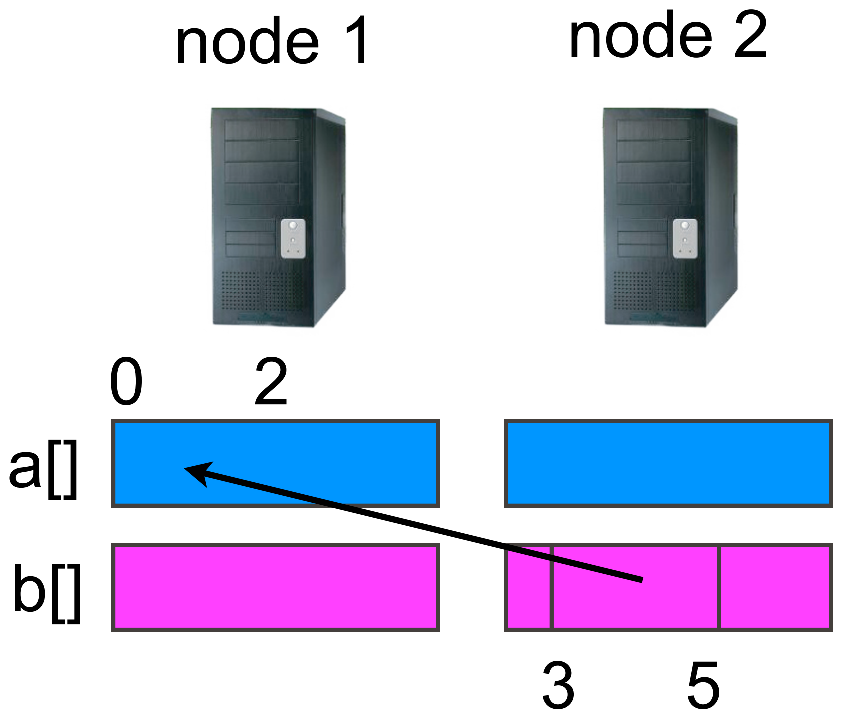
🟥 node1 🟦 node2 🟪 node3 🟩 node4

Local View Model

- One-sided communication for local data(Put/Get)
- In XMP Fortran, this function is compatible with that of CAF
- In XMP C, C is extended to support the ***array section*** notation
- Uses GASNet/ARMCI, which are high-performance communication layer

```
#pragma xmp coarray b:[*]
      :
if(me == 1)
  a[0:3] = b[3:3]:[2];        // Get
```





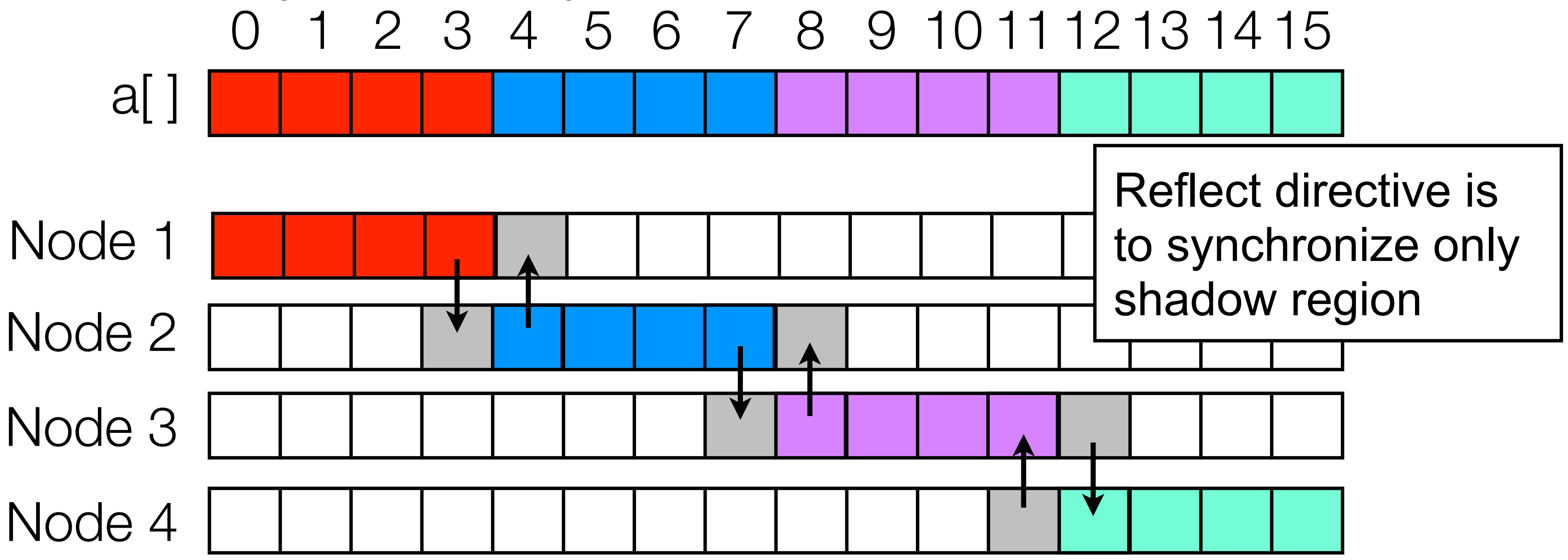
Other directives

Directive	Function
reduction	Aggregation
bcast	Broadcast
barrier	Synchronization
shadow/reflect	Create shadow region/sync.
gmove	Transfer for distributed data

shadow/reflect directives

- If neighbor data is required, then only shadow area can be synchronized
  - Shadow directive defines width of shadow area
  - b[i] = array[i-1] + array[i+1];

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



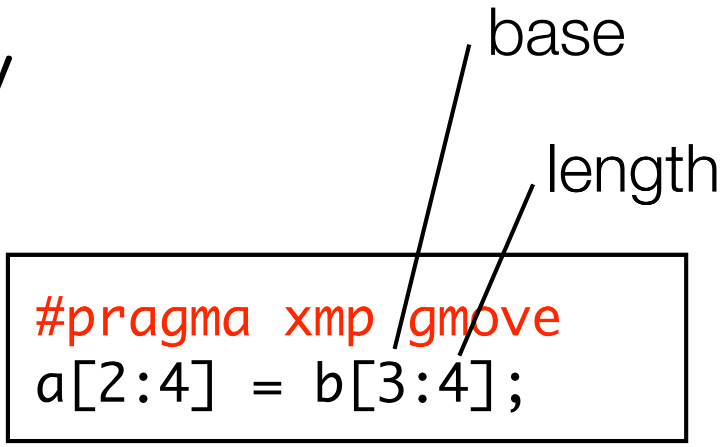
Reflect directive is to synchronize only shadow region

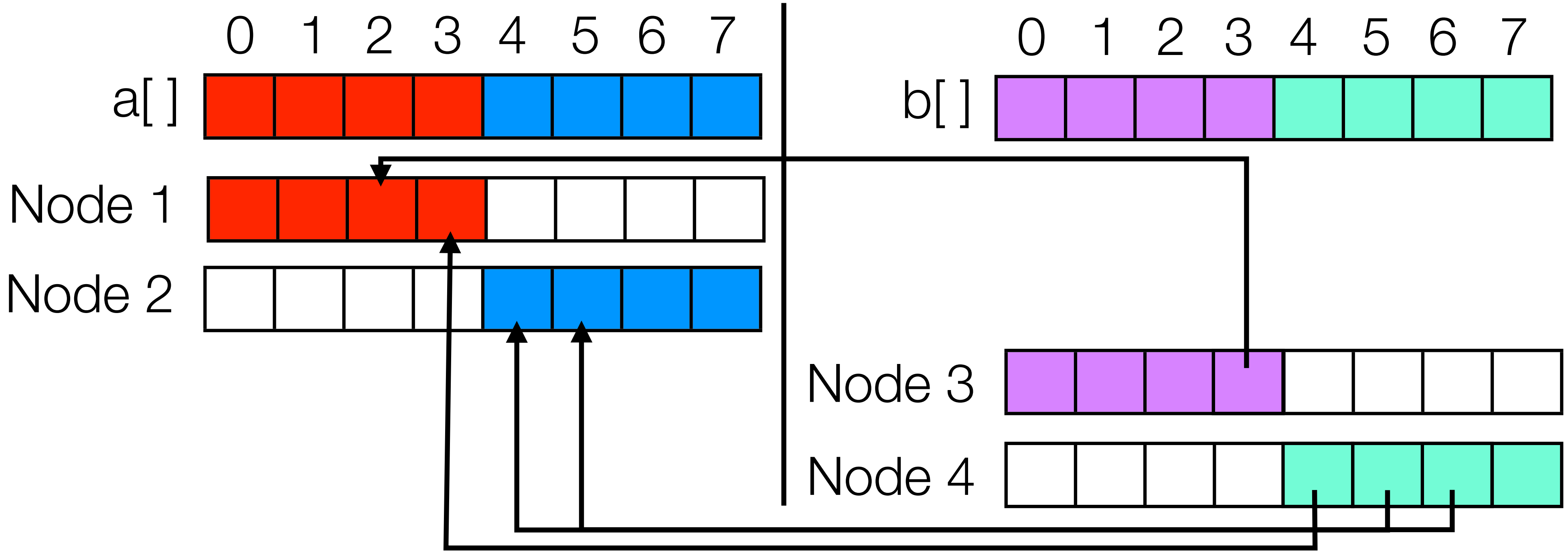
```
#pragma xmp reflect (array)
```

Gmove directive

- Communication for distributed array
  - uses array section notation in XMP C
  - Programmer doesn't need to know where each data is distributed

```
#pragma xmp gmove
a[2:4] = b[3:4];
```





Agenda

XcalableMP

- Overview of XMP
- XMP Programming Model
- Evaluation of Performance and Productivity of XMP

Evaluation

- Examines the performance and productivity of XMP
- Implementations of benchmarks
  - NAS Parallel Benchmarks
    - CG, EP, IS, BT, LU, FT, MG
  - HPC Challenge Benchmarks
    - HPL, FFT, RandomAccess, STREAM
    - Finalist of HPCC Class2 in SC10 and SC09
  - Laplace Solver
  - Himeno Benchmark, and so on

HPC CHALLENGE

FINALIST XcalableMP

CLASS 2 AWARD

Jinpil Lee
University of Tsukuba

Mitsuhisa Sato
University of Tsukuba

Environment

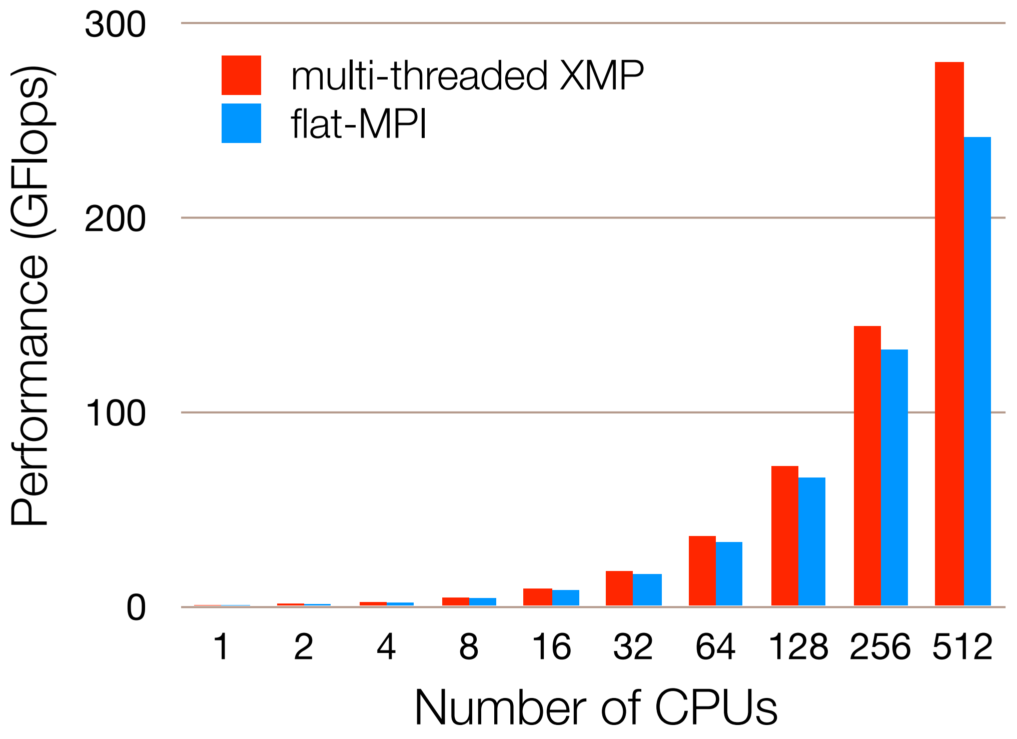
- T2K Tsukuba System
  - CPU : AMD Opteron Quad-Core 8356 2.3GHz (4 sockets)
  - Memory : DDR2 667MHz 32GB
  - Network : Infiniband DDR(4rails) 8GB/s

Laplace Solver

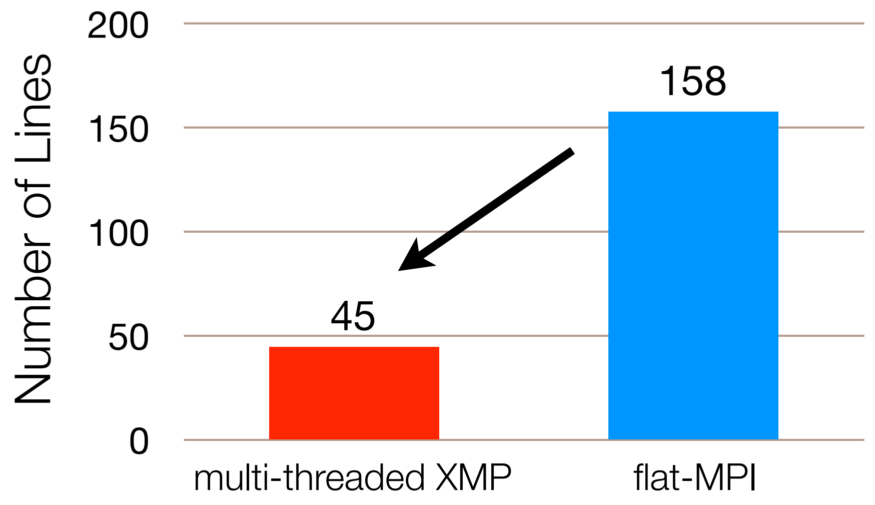
- “thread” clause for multicore cluster
- shadow/reflect directive

```
#pragma xmp loop (x, y) on t(x, y) threads
for(y = 1; y < N-1; y++)
    for(x = 1; x < N-1; x++)
        tmp_a[y][x] = a[y][x];
```

Performance



Productivity



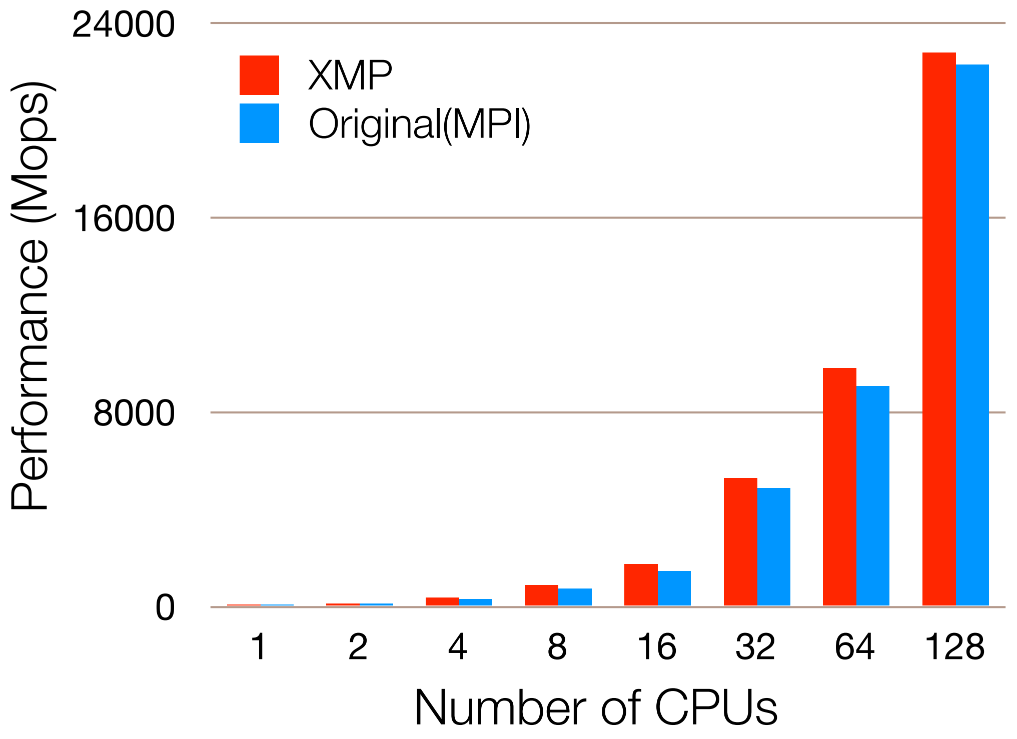
Conjugate Gradient

XcalableMP

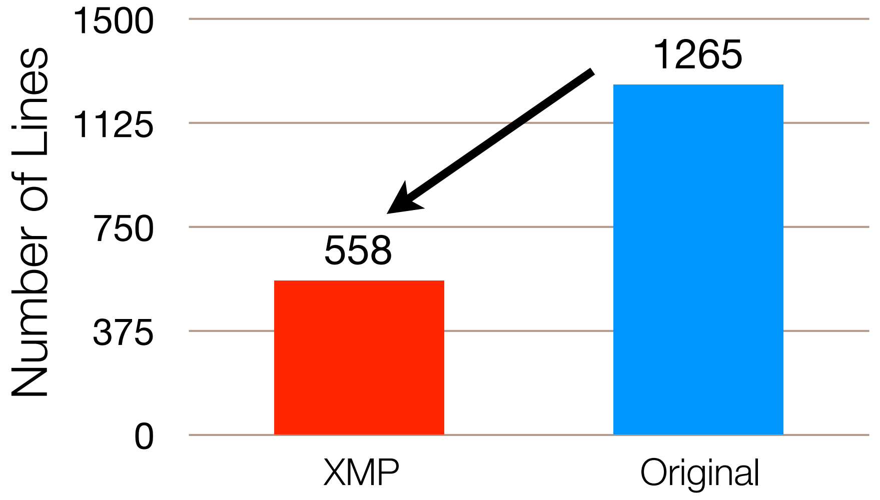
- Local view programming
- reduction directives for local variable

```
#pragma xmp coarray w, w1:[*]
    :
for( i=ncols; i>=0; i-- ){
  w[l:count[k][i]] += w1[m:count[k][i]]:[k];
```

Performance



Productivity

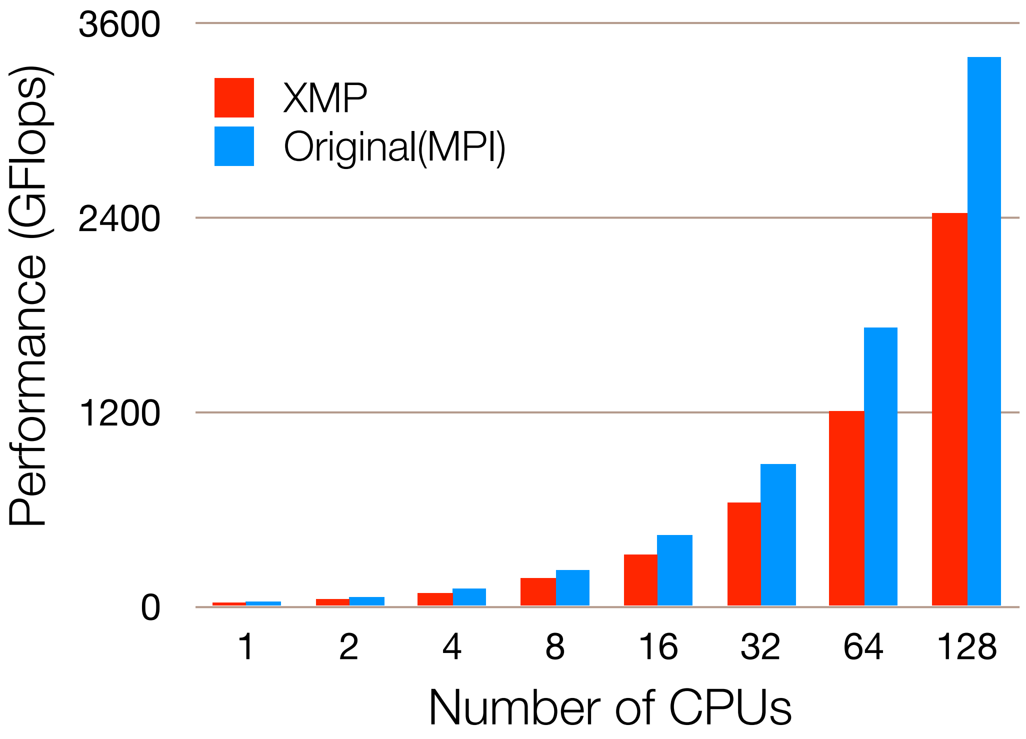


High Performance Linpack XcalableMP

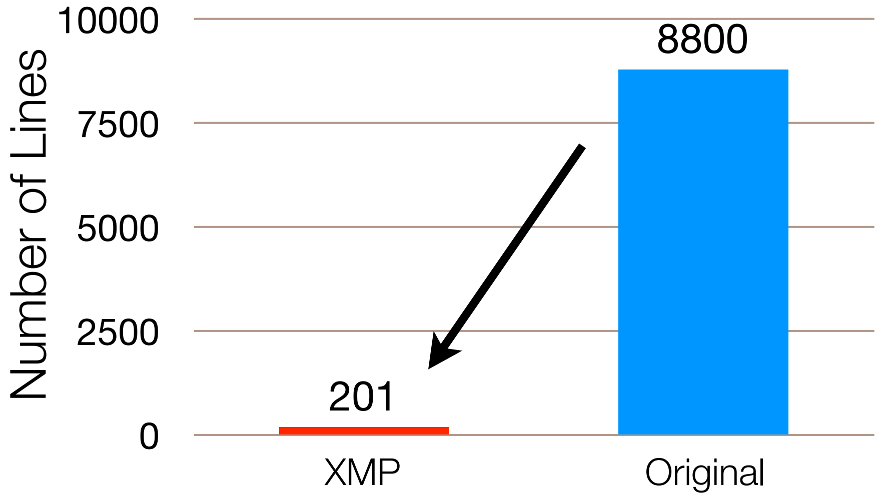
- block-cyclic distribution
- BLAS Lib. is used directly from distributed array

```
#pragma xmp distribute \
        t(cyclic(NB), cyclic(NB)) onto p
#pragma xmp align A[i][j] with t(j, i)
     :
cblas_dgemm(..., &A[y][x], ...);
```

Performance



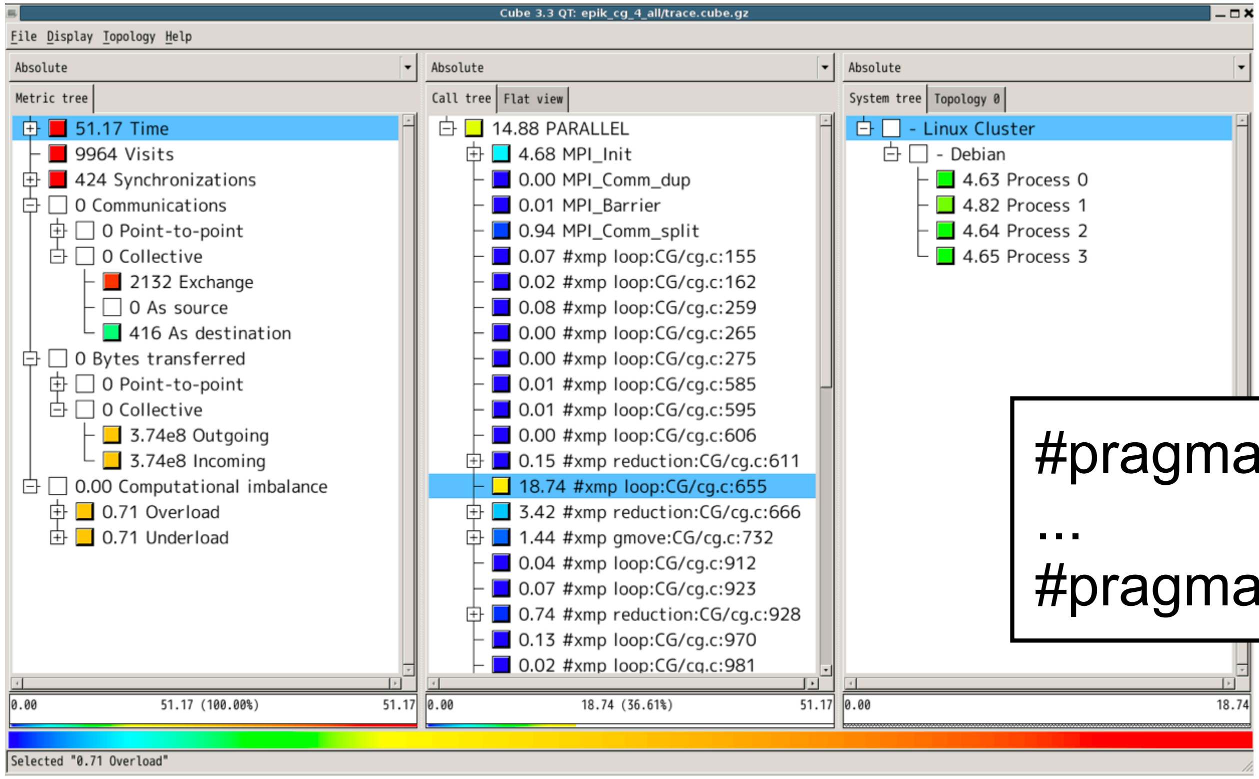
Productivity



International Collaboration

XcalableMP

- Interface of XMP program profile to Scalasca
- Scalasca is a software tool that supports the performance optimization of parallel programs
- Scalasca is developed by JÜLICH FORSCHUNGSZENTRUM and German Research School for Simulation Sciences



```
Cube 3.3 QT: epik_cg_4_all/trace.cube.gz
File  Display  Topology  Help

Absolute
Metric tree
51.17 Time
9964 Visits
424 Synchronizations
0 Communications
  0 Point-to-point
  0 Collective
    2132 Exchange
    0 As source
    416 As destination
0 Bytes transferred
  0 Point-to-point
  0 Collective
    3.74e8 Outgoing
    3.74e8 Incoming
0.00 Computational imbalance
  0.71 Overload
  0.71 Underload
0.00          51.17 (100.00%)          51.17

Absolute
Call tree | Flat view
14.88 PARALLEL
  4.68 MPI_Init
  0.00 MPI_Comm_dup
  0.01 MPI_Barrier
  0.94 MPI_Comm_split
  0.07 #xmp loop:CG/cg.c:155
  0.02 #xmp loop:CG/cg.c:162
  0.08 #xmp loop:CG/cg.c:259
  0.00 #xmp loop:CG/cg.c:265
  0.00 #xmp loop:CG/cg.c:275
  0.01 #xmp loop:CG/cg.c:585
  0.01 #xmp loop:CG/cg.c:595
  0.00 #xmp loop:CG/cg.c:606
  0.15 #xmp reduction:CG/cg.c:611
  18.74 #xmp loop:CG/cg.c:655
  3.42 #xmp reduction:CG/cg.c:666
  1.44 #xmp gmove:CG/cg.c:732
  0.04 #xmp loop:CG/cg.c:912
  0.07 #xmp loop:CG/cg.c:923
  0.74 #xmp reduction:CG/cg.c:928
  0.13 #xmp loop:CG/cg.c:970
  0.02 #xmp loop:CG/cg.c:981
0.00          18.74 (36.61%)          51.17

Absolute
System tree | Topology 0
- Linux Cluster
  - Debian
    4.63 Process 0
    4.82 Process 1
    4.64 Process 2
    4.65 Process 3
0.00                                  18.74

Selected "0.71 Overload"
```

```
#pragma xmp gmove profile
...
#pragma xmp loop on t(i) profile
```

Summary & Future work

- XcalableMP was proposed as a new programming model to facilitate program parallel applications for distributed memory systems
- Evaluation of Performance and Productivity
  - Performance of XMP is compatible with that of MPI
  - Productivity of XMP is higher than that of MPI
- Future work
  - Performance evaluation on larger environment
  - For accelerators(GPU, etc), Parallel I/O , and Interface of MPI library