XcalableMP

Directive-Based Language eXtension for Scalable Parallel Programming

T2K Open Supercomputer Alliance
University of Tsukuba University of Tokyo Kyoto University

What's XcalableMP?

MPI is widely used as a parallel programming model. However, the programming cost of MPI is high.

XcalableMP[1-3], ***XMP*** for short, is a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and to tune its performance by having minimal and simple notations.

XMP specification is being designed by XcalableMP Specification Working Group(XMP-WG). XMP-WG is a special interest group, which is organized to make a draft on "petascale" parallel language. XMP-WG consists of members from academia(U. Tsukuba, U. Tokyo, Kyoto U. and Kyusyu U.), research labs (RIKEN, NIFS, JAXA, JAMSTEC/ES) and industries (Fujitsu, NEC, Hitachi) in Japan.

XMP prototype compiler and tools are being developed in "Seamless and Highly-productive Parallel Programming Environment for High performance computing" project funded by Ministry of Education, Culture, Sports, Science and Technology, JAPAN.

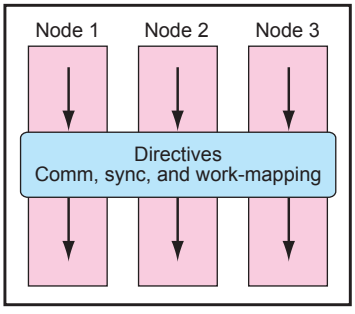
Features

To reduce code writing and educational costs

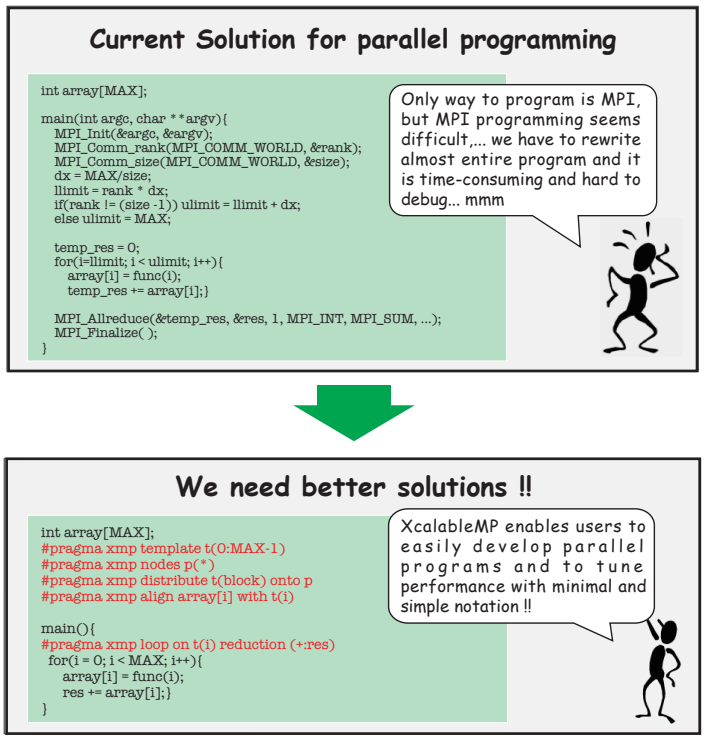
- Language extension of C99 and Fortran 95
- Supports typical parallelization based on data parallel paradigm and work mapping under "global-view" programming model
- Also includes Coarray Fortran like feature as "local-view" programming model
- Many concepts are inherited from High Performance Fortran

Performance awareness

- Execution model is a Single Program Multiple Data (SPMD)
- A thread starts execution in each node independently (as in MPI)
- Communication, synchronization and work-mapping occur when directives are encountered



- All actions are taken by directives for being "easy-to-understand" in performance tuning (different from High Performance Fortran)



Status

- XMP specification version 1.0 is available
- Omni XMP compiler 0.5.3 for C is available from University of Tsukuba
  - Download from http://www.xcalablemp.org
  - Supported platforms are Linux cluster, Cray platform, ..
  - Interface of Scalasca & tlog profiling tools
  - For accelerators(GPU, etc)
  - XMP Parallel I/O
  - Interface of MPI library

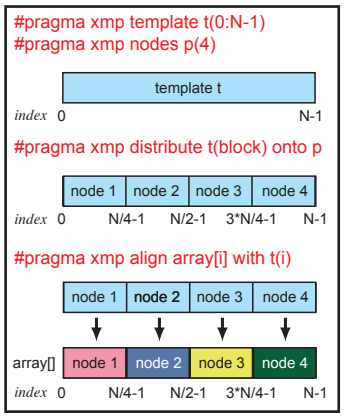
  } work in progress
- XMP will be used to program to K computer

Programming Model

Global-view Programming

The global-view programming model supports typical work-mapping and communication for parallel programs.

First, to parallelize a program in the global-view model, data distribution is described with a template. The template is a dummy array used to express an index space associated with an array. The concept of the template is shown in the next page' s figure. The template and a node set are defined by a ***template directive*** and a ***node directive***. Distribution of the template is described by a ***distribute directive***. Finally, data distribution of the array is specified by an ***align directive*** to align the array with the template.




```
1  int array[N];
2
3  #pragma xmp template t(0:N-1)
4  #pragma xmp nodes p(4)
5  #pragma xmp distribute t(block) onto p
6  #pragma xmp align array[i] with t(i)
7
8  main(void){
9     int i, res = 0;
10
11 #pragma xmp loop on t(i)
12    for(i = 0; i < N; i++){
13       array[i] = func(i);
14       res += array[i];
15       }
16
17 #pragma xmp reduction (+:res)
18 }
```

Above right figure shows a simple example of the global-view programming model in XMP C language. Line 3 defines the template with a start index (0) and a length of index (N). Line 4 defines the node set (4 nodes in this case). Line 5 specifies the data distribution of the template on the node set as a block distribution. XMP supports ***block***, ***cyclic***, ***block-cyclic***, and ***gen-block*** distributions. Line 6 defines the distribution to align the array with the template.

In line 11, a ***loop directive*** is used to parallelize a loop statement from Line 12 to Line 15. In this case, node 1 executes iterations from ***0*** to ***N/4 – 1***, and node 2 executes iterations from ***N/4*** to ***N/2 – 1***, independently. Line 17 executes a reduction operation on the variable ***res*** by a ***reduction directive***. The reduction directive supports typical operators (such as “+” , “MAX” , and so on).

XMP also supports rich communication and synchronize directives such as “shadow” , “gmove” , and “barrier” . Global-view communication directives are used to maintain the consistency of shadow area, move distributed data globally, and synchronize nodes.

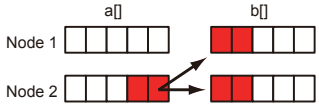
Local-view Programming

The local-view programming model attempts to increase performance by considering inter-node communication and local memory of each node explicitly. In this model, local data distributed on the node can be referred to using node number.

XMP adopts coarray notation as an extension of Fortran and C languages for local-view programming. In the case of Fortran as the base language, the notation of XMP coarray is compatible with that of Coarray Fortran. To use coarray notation in C, we propose a language extension of the C language. To access coarray, the node number is specified by following an array reference by “:” .

array_name[start : length[:step]]:[node_number]

```
double a[5], b[5];
#pragma xmp coarray a   // Declartion
   :
b[0:2] = a[3:2]:[2];          // Communication
```



The **array_name[start:length]:[node_number]** means elements from the **array_name[start]** to the **array_name[start+length-1]** located on compute node whose name is **node_number**.

XMP recommends users to choice the global-view programming model and the local-view programming model according to an algorithm.

Example

Laplace Solver by Global-view programming

```
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp nodes p(N_Y, N_X)
#pragma xmp distribute t(block,block) onto p
#pragma xmp align u[y][x] with t(x, y)
#pragma xmp align uu[y][x] with t(x, y)
#pragma xmp shadow uu[1:1][1:1]
        :
#pragma xmp loop (x, y) on t(x, y) threads
for(y = 1; y < YSIZE-1; y++)
  for(x = 1; x < XSIZE-1; x++)
    uu[y][x] = u[y][x];

#pragma xmp reflect uu

#pragma xmp loop (x, y) on t(x, y) threads
for(y = 1; y < YSIZE-1; y++)
  for(x = 1; x < XSIZE-1; x++)
    u[y][x] = (uu[y-1][x] + uu[y+1][x] +
               uu[y][x-1] + uu[y][x+1])/4.0;
```

Defines ***two dimensional process grid***

Defines ***shadow area*** and its width

Specifies ***additional thread parallelization***

Synchronizes data only on ***shadow area***

The laplace solver is the implementation of a laplace equation using Jacobi iteration with 4 points-stencil operations. This code is an example to use ***shadow*** and ***reflect directives*** which communicate and synchronize only overlapped region. Moreover, thread parallelization is used in loop statements. XMP can program simply by using shadow/reflect operations with keeping serial code image.

Integer Sort of NPB by Local-view programming

```
int key[SIZE];
#pragma xmp coarray key
        :
#pragma xmp barrier
for( i=0; i<comm_size; i++ )
  key[recv_displ[i]:count[i]]:[i]
              = buff[send_displ[i]:count[i]];
```

Defines ***co-array***

Exchanges data by using ***coarray***

The Integer Sort benchmark tests a sorting operation that is important in particle method codes. This code is an example to use ***coarray*** function which exchanges data. XMP coarray function will use efficient one-sided communication library GASNet, ARMCI, and so on.

Reference

[1] Jinpil Lee, Minh Tuan Tran, Tetsuya Odajima, Taisuke Boku and Mitsuhisa Sato: An Extension of XcalableMP PGAS Language for Multi-node GPU Clusters, Ninth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar 2011), 2011.

[2] Jinpil Lee and Mitsuhisa Sato: Implementation and Performance Evaluation of XcalableMP: A Parallel Programming Language for Distributed Memory Systems, The 39th international Conference on Parallel Processing Workshops (ICPPW10), pp.413-420, 2010.

[3] Masahiro Nakao, Jinpil Lee, Taisuke Boku, Mitsuhisa Sato. “XcalableMP Implementation and Performance of NAS Parallel Benchmarks” , Fourth Conference on Partitioned Global Address Space Programming Model (PGAS10), Oct., 2010.

For more information, please visit

- T2K Open Supercomputer Alliance (#5007@Level 6)
- Center for Computational Sciences, University of Tsukuba (#923@Level 4)
- http://www.xcalablemp.org