XcalableMP

Directive-based language eXtension for Scalable Parallel Programming

Overview

What's XcalableMP ?

- XcalableMP is a directive-based PGAS language for distributed memory system
- Designed by XcalableMP Specification Working Group

  Members from academia (U. Tsukuba, U. Tokyo, Kyoto U., and Kyusyu U.), research labs(RIKEN, NIFS, JAXA, and JAMSTEC/ES), and industries(Fujitsu, NEC, Hitachi) in Japan
- Omni XcalableMP compiler is developed in "Seamless and Highly-productive Parallel Programming Environment for High performance computing" project funded by MEXT in Japan

Implementation Status

- XcalableMP specification ver. 1.0 is available
- Omni XcalableMP compiler ver. 0.5.3 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)
  - XcalableMP Parallel I/O
  - Interface of MPI library

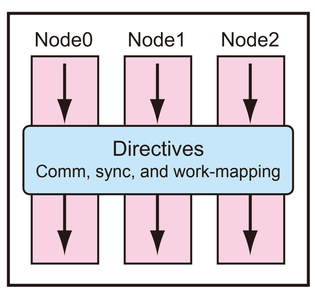
  } work in progress
- For K computer

  XcalableMP will be used to program to K computer

Programming Model

Language Features

- Language extension of C99 and Fortran 95
- SPMD as a basic execution model
- 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 HPF)



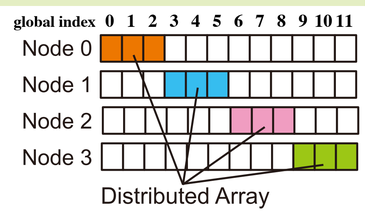
Global-view Programming

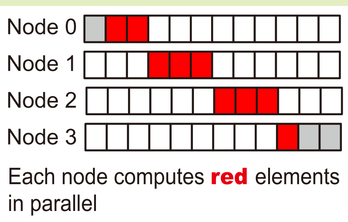
- Supports typical parallelization based on the data parallel paradigm and work mapping

Example: a[12] is distributed onto 4 nodes

```
int a[12];
#pragma xmp nodes p(4)
#pragma xmp template t(0:11)
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i] with t(i)
```

```
#pragma xmp loop on t(i) reduction(+s)
for(i = 1; i < 10; i++) {
    a[i] = func(i);
    s += a[i];
}
```





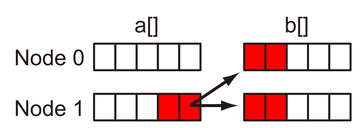
- Many concepts of XcalableMP are inherited from HPF

Local-view Programming

- Also includes Co-Array Fortran like feature

Example: Declaration & communication of coarray

```
double a[5], b[5];
#pragma xmp coarray a    // Declartion
   :
b[0:2] = a[3:2]:[1];     // Communication
```



- Extends C for an array section

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

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**

Benchmarks

Laplace Solver by Global-view

```
double u[YSIZE][XSIZE], uu[YSIZE][XSIZE];
#pragma xmp nodes p(N_Y, N_X)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#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;
```

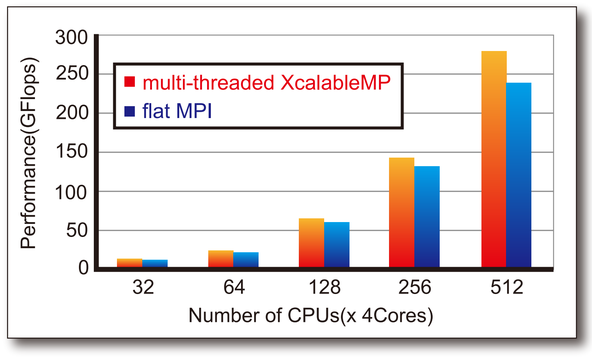
Define two-dimensional process grid

Define shadow area and its width

Specify additional thread parallelization

XcalableMP also supports hybrid parallelization for multicore cluster

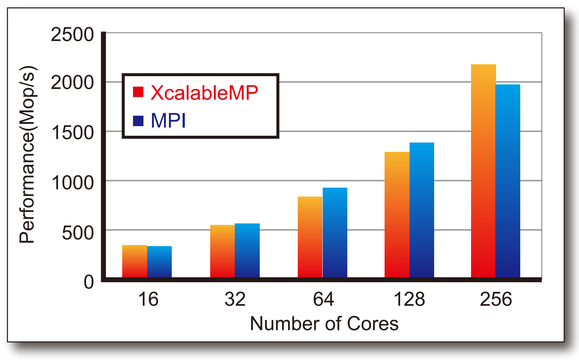
Synchronize data only on shadow area



Integer Sort of NPB by Local-view

```
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]];
```

Exchange data by using Co-array



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