Programming with CAF is very similar to MPI, in that it provides primitive elements for parallel programming. Since all communications are clear to the users, performance tuning can easily be done. Since CAF does not provide global array indices for the entire data space, the user can only use local array indices in the communication statement.

XcalableMP is being designed based on the experiences of HPF, Fujitsu XPF (VPP Fortran) and OpenMPD.

Programming Model

All variables except those specified by XcalableMP directives as a distributed object are private and cannot be directly referenced from other processes.

Distributed array is defined by aligned directive to align the object to the template distributed onto nodes. Shadow directives are used to declare the shadow region of each array, which can be synchronized by reflect directive.

The work sharing in loop iteration is described by the loop directive as in OpenMP. For work sharing in a loop, each iteration must be assigned by on clause so that each node refers to a partial region of the array to be assigned to that process. The programmer can specify the reduction operation on the variable by reduction clause. In XcalableMP, the user is responsible for describing the necessary directives for loop parallelization and data distribution.

A fragment of code taken from Laplace solver using simple Jacobi iteration is shown in the box. The Laplace solver is an example to use shadow and reflect directives which communicates and synchronize the overlapped regions.

```
#pragma xmp nodes p(YProc, XProc)
#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]
    • • •
for(k = 0; k < niter; k++){
#pragma xmp loop (y, x) on t(x, y)  // old <- new
  for(y = 1; y < YSIZE-1; y++)
    for(x = 1; x < XSIZE-1; x++)
      uu[y][x] = u[y][x];

#pragma xmp reflect uu          // synchronization

#pragma xmp loop (y, x) on t(x, y) // update
  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;
} // end of iteration
```

Acknowledgement

This research is carried out as a part of “Seamless and Highly-productive Parallel Programming Environment for High-performance computing” project funded by Ministry of Education, Culture, Sports, Science and Technology, JAPAN.

Reference

[1] Jinpil Lee and Mitsuhisa Sato, “Implementation and Perfor-mance Evaluation of XcalableMP“, A Parallel Programming Language for Distributed Memory Systems, 39th Annual International Conference on Parallel Processing (2010)

[2] 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), NewYork, USA, Oct., 2010.

XcalableMP is now under design by XcalableMP specification WG. If you are interested in XcalableMP activity, JOIN US!!! Please contact to M. Sato (msato@cs.tsukuba.ac.jp)