2. Programming Environment Research Team

2.1. Team members

Mitsuhisa Sato (Team Leader)
Hitoshi Murai (Research Scientist)
Tetsuya Abe (Postdoctoral Researcher)
Swann Perarnau (Postdoctoral Researcher)
Tomotake Nakamura (Research Associate)
Takenori Shimosaka (Research Associate)
Masahiro Yasugi (Visiting Researcher)
Tomio Kamada (Visiting Researcher)
Tomoko Nakashima (Assistant (Concurrent))

2.2. Research Activities

The K computer system running in AICS is a massively parallel system which has a huge number of processors connected by the high-speed network. In order to exploit full potential computing power to carry out advanced computational science, efficient parallel programming is required to coordinate these processors to perform scientific computing. We conducts researches and developments on parallel programming models and language to exploit full potentials of large-scale parallelism in the K computer and increase productivity of parallel programming.

In 2011FY, in order to archive these objectives above, we carried out the following researches:

1) We have been developing a new programming language, called XcalableMP(XMP), which is originally designed in Japanese HPC language research community. Although MPI is the de facto standard for parallel programming on distributed memory systems, writing MPI programs is often a time-consuming and complicated process. 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 has been designed by XcalableMP Specification Working Group which consists of members from academia and research labs to industries in Japan. In this year, we developed a prototype compiler of XMP Fortran, and deployed it to the K computer. We did a preliminary performance evaluation using a XMP version of the SCALEp code (a climate code for LES) on the K computer. We also design the interface to MPI programs in XMP, and IO supports of the XMP language. We have ported the GASnet communication library developed by LBNL/UCB for the K computer as an one-side communication layer of XMP.
2) We started the research for performance tuning tools for large-scale scientific applications running on the K computer. As a first step, we have ported the Scalasca performance turning

and analysis tool developed by JSC, Germany, to the K computer and examined the potential of this tool for the K computer.

3) We investigated methods and tools to support a correct parallel program. We proposed a light-weight XMP verification tool which helps users to verify XMP programs using descriptions of global-view programming directives in XMP.
4) The processor of the K computer has an interesting hardware mechanism called “sector cache”, which allows partition of L2 on-chip cache to optimize the locality for important data. We investigated the technique to optimize the usage of sector cache.
5) Our team supports Xcrypt, a scripting language developed by Kyoto University. We have developed an application using Xcrypt which perform an automatic tuning for Block Multi-Color Ordering Code.

We have started discussion with application developers in AICS for the collaborations on performance tuning and development of parallel programs.

2.3. Research Results and Achievements

2.3.1. Development of XcalableMP Fortran and Preliminary Performance on the K computer

In this year, we developed a prototype compiler of XMP Fortran, and deployed it to the K computer. We did a preliminary performance evaluation of XMP Fortran on the K computer. We parallelized SCALEp code by XMP Fortran. SCALEp is a kinetic core in SCALE developed by the SCALE project which develops (Parallel) Climate code for large eddy simulation under the collaboration with Climate science research team and computer science research teams. The parallelization steps of SCALEp codes are as follows:

1. Insert pragmas to specify 2D block distribution of 3D array.
2. Paralleling double nested loop by loop directives
3. Insert reflect directives for the communication periodic neighbor elements.
4. As a, we have examined the performance improvement by runtime optimization using Remote Direct Memory Access (RDMA) of K computer for neighbor communications.

Figure 1 shows the outline of XMP version of SCALEp code.

```
!$xmp nodes p(N1,N2)
!$xmp template t(IA,JA)
!$xmp distribute t(block,block) onto p

real(8) :: dens(0:KA,IA,JA)
!$xmp align (*,i,j) &
!$xmp    with t(i,j) :: dens, ...
!$xmp shadow (0,2,2) :: dens, ...

!$xmp reflect (dens(0,/periodic/2,&
!$xmp                /periodic/2), ...)

!$xmp loop (ix,jy) on t(ix,jy)
do jy = JS, JE
  do ix = IS, IE
    do kz = KS+2, KE-2
      ... dens(kz,ix+1,jy) ...
    end do
  end do
end do
```

Declarations for Node array and template

Data distribution

Neighbor comm

Loop paralization

Figure 1. The outline of SCALEp in XMP

The problem size is 512x512 in horizontal directions, 128 in vertical direction. In Figure 2, the execution time for 500 steps is shown. Note that XMP node is assigned to one node and the node program is parallelized by automatic paralleling compiler by Fujitsu.

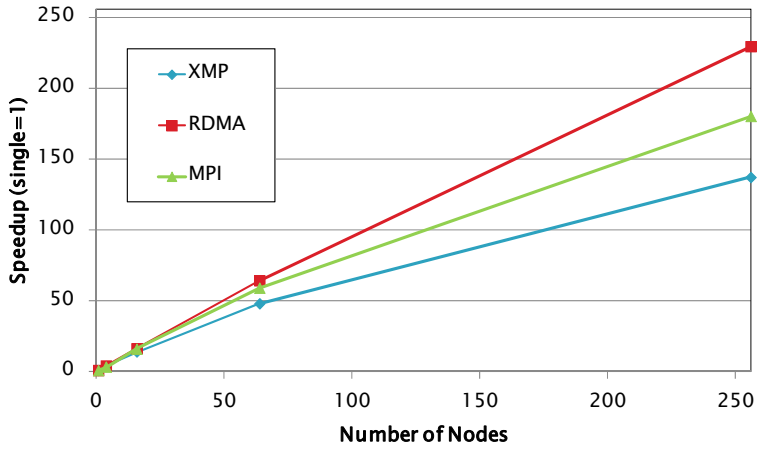


Figure 2. Performance of SCALEp code on the K computer.

We found some overhead of XMP version comparing to the hand-coded MPI version, and it can be improved significantly by the optimization using RDMA.

2.3.2. Design of the Interface of XcalableMP Programs to Parallel Numerical Libraries

Scientific parallel applications are often developed using high-performance parallel libraries. In development of scientific applications, re-writing every application programs using only XMP is unrealistic. In order to develop high performance parallel application programs easily, it is very

important to make use of existing parallel high performance libraries. We have designed an interface of XMP to parallel numerical libraries, which has the following features:

- Parallel numerical library routines can be invoked as an XMP procedure through the XMP interface procedure.
- When the parallel numerical library routine needs information on global distributed arrays, the interface extracts it from the descriptor using some query routines provided by XMP runtime library and passes it to the numerical library routine as arguments.
- The interface does not affect the behavior of numerical library routines expect for restrictions concerning the XMP specification.

We have implemented the interface between XMP and each of ScaLAPACK (Scalable Linear Algebra PACKage http://www.netlib.org/scalapack/) and MUMPS (A MUltifrontal Massively Parallel sparse direct Solver http://graal.ens-lyon.fr/MUMPS/). The example code of a XMP program using the interface to ScaLAPCK is shown in Figure 3.

```
#include “xmp.h”
int nrow=NROW,ncol=NCOL,nprow=NPROW,npcol=NPCOL;
double a[NCOL][NROW],b[NROW],ipiv[2*NROW][NPCOL];
#pragma xmp nodes p(npcol,nprow)
#pragma xmp nodes q(nprow)=p(1,1:nprow)
#pragma xmp template t(0:ncol-1,0:nrow-1)
#pragma xmp template t1(0:2*nrow-1,0:npcol-1)
#pragma xmp distribute t(block,block) onto p
#pragma xmp distribute t1(block,block) onto p
#pragma xmp align a[i][j] with t(i,j)
#pragma xmp align ipiv[i][j] with t1(i,j)
#pragma xmp align b[i] with t(*,i)

int main(int argc, char* argv[]){

  int i,j,contxt,myrow,mycol;
  int icontxt=-1,what=0;
  int nrhs=1,ia=1,ja=1,ib=1,jb=1,info;
  double a0[ncol][nrow],b0[nrow],btmp,err;
  char *order="R";
  blacs_get_(&icontxt,&what,&contxt);
  blacs_gridinit_(&contxt,order,&nprow,&npcol);
  blacs_gridinfo_(&contxt,&nprow,&npcol,&myrow,&mycol);

  for(i=0;i<ncol;i++){
    for(j=0;j<nrow;j++){
      a0[i][j] = rand()/(1.0e+10);
    }
  }

 for(j=0;j<nrow;j++){
    b0[j]=1.0;
  }

#pragma xmp loop (i,j) on t(i,j)
  for(i=0;i<ncol;i++){
    for(j=0;j<nrow;j++){
      a[i][j] = a0[i][j];
    }
  }

#pragma xmp loop on t(*,j)
  for(j=0;j<nrow;j++){
     b[j]=b0[j];
  }

ixmp_pdgesv(&nrow,&nrhs,a,&ia,&ja,xmp_desc_of(a),
ipiv,b,&ib,&jb,xmp_desc_of(b),&contxt,&info);
……
```

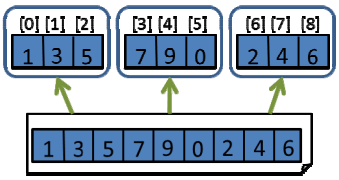
Figure 3. Example code of using the XMP interface for ScaLAPACK

2.3.3. Parallel I/O Facility of XcalableMP and Evaluation of I/O Performance

XcalableMP allows the programmer to define a distributed array over nodes to support typical data parallel programmer with global-view programming. In order to perform I/O operations of a distributed array efficiently, XMP-IO is defined a parallel I/O API for a distributed global array in the XcalableMP specification.

In a large-scale parallel system, some applications must often perform I/O for a large data efficiently. These I/O intensive operations may cause a serious problem in a large scale parallel system. To solve the I/O problem, parallel IO libraries such as MPI-IO and pNetCDF has been developed.

We have designed the XMP-IO using MPI-IO as an underlying parallel I/O library. The XMP interface to obtain the information of the distributed array described in the program is used to perform parallel I/O operations by MPI-IO. As other parallel I/O interface, we also designed a XMP library interface to a parallel library for PNetCDF which is a commonly used file format. Figure 4 shows an example code using XMP-IO and XMP pNetCDF interface and these MPI counterparts.



(a) File IO example (upper: node, lower: file)

XMP program

```
int a[9];
#pragma xmp nodes p(3)
#pragma xmp template t(0:8)
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i] with t(i)

xmp_file_t  fp;     // file handle
xmp_range_t  ra;   // range of array: ra
xmp_desc_t  desc;  // info of the specified array
desc = xmp_desc_of(a);
xmp_set_range(ra, 0, 0, 3, 1);

fp = xmp_fopen_all(fname, “r”);
xmp_file_set_view_all(fp, 0, desc, ra);
xmp_fread_darray_all(fp, desc, ra);
xmp_fclose_all(fp);
```

Translate

MPI program

```
int a[3];
gsize[0] = 9;
dist[0] = MPI_DISTRIBUTE_BLOCK;
darg[0] = MPI_DISTRIBUTE_DFLT_DARG;
psize[0] = 3;

MPI_File_open(comm, fname, mode, info, &fh);
MPI_Type_create_darray(3, rank, 1, gsize, dist,  psize,
MPI_ORDER_C, MPI_INTEGER, &type);
MPI_Type_commit(&type);
MPI_File_set_view(fh, 0, MPI_INTEGER, type, “native”,
 info);
MPI_File_read_all(fh, a, 3, MPI_INTEGER, &status);
MPI_File_close(&fh);
```

(b) XMP-IO and equivalent MPI program

XMP program using our library interface

```
int a[9];
xmp_desc_t  desc;
desc = xmp_desc_of(a);

xmpnc_open(comm, fname, NC_READ, info, &id);
xmpnc_inq_varid(fid, “data”, &vid);
xmpnc_get_varaw_int_all(id, vid, desc);
xmpnc_close(id);
```

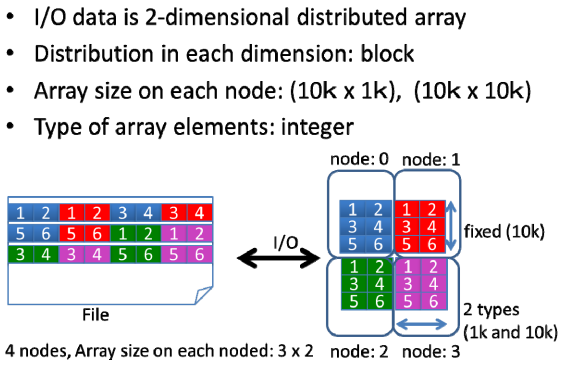
Translate

MPI program using PnetCDF

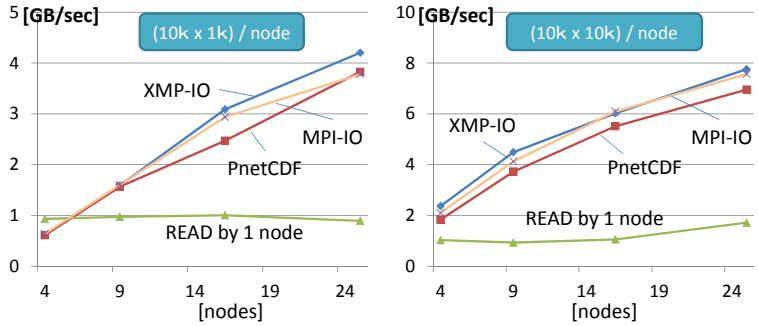
```
int a[3];

ncmpi_open(comm, fname, NC_READ, info, &id);
ncmpi_inq_varid(ncid, “data”, &vid);
start[0] = rank * 3;
count[0] = 3;
ncmpi_put_vara_int(id, vid, start, count, &a[0]);
ncmpi_close(id);
```

(c) XMP pNetCDF interface and equivalent MPI with PNetCDF call

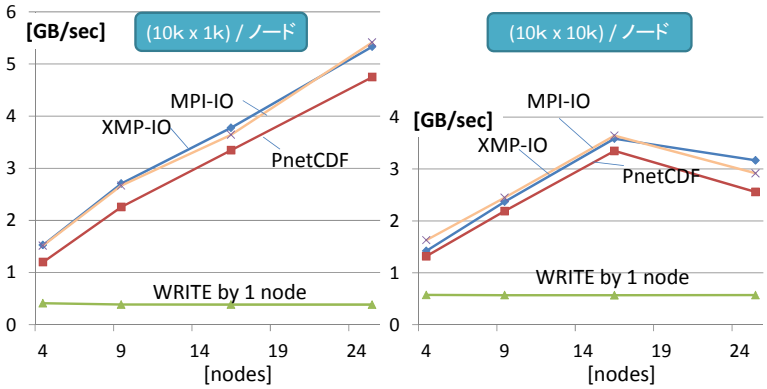
Figure 4. XMP-IO and XMP netCDF



(a) Experimental Setting



(b) Read performance



(c) Write performance

Figure 5. Performance results of XMP-IO and XMP pNetCDF interface

In Figure 5, the experimental results of these parallel I/O in XcalableMP are shown. We measured the IO performance for data with size 10kB x 1KB and 10kB and 10kB respectively. We found that these parallel I/O interfaces provide a good parallel I/O performance on the Lustre file system with reasonable programming cost.

2.4. Schedule and Future Plan

The major goal of researches in next financial year (2012) is to finish the full implementation of XcalableMP C and Fortran for the K computer, and release them to the users. With this release, we will provide XMP-IO and XMP interface to some high-performance parallel libraries written in MPI.

As for the performance tuning tool for the K computer, we will perform several case studies on real scientific applications using Scalasca as well as providing Scalasca to general users. Through these case studies, we will extend it for valuable performance analysis in the K computer.

We are also continuing the researches on the optimization for “sector cache” and parallel program verification tools for XMP program and MPI programs.

2.5. Publication, Presentation and Deliverables

(1) Journal Papers

- None

(2) Conference Papers

1. Yonezawa, Tadashi Watanabe, Mitsuo Yokokawa, Mitsuhisa Sato, and Kimihiko Hirao Advanced Institute for Computational Science (AICS), "Japanese National High-Performance Computing Research Institute and its 10-petaflops supercomputer "K"", Proceeding of SC11, SC '11 State of the Practice Reports , 2011.

[not refereed, in Japanese]

1. Tomotake Nakamura and Mitsuhisa Sato, "Parallel I/O Facility of PGAS Programming Language XcalableMP and Evaluation of I/O Performance on the Lustre File System", IPSJ SIG Technical Report (in Japanese), Vol. 2011-HPC-132 No. 36, pp. 1-9, 2011.
2. Hitoshi Murai and Mitsuhisa Sato, "Extensions of XcalableMP for User-Defined Distribution of Data and Loops", IPSJ SIG Technical Report (in Japanese), Vol. 2011-HPC-130, No. 59, pp. 1-9, 2011.
3. T. Shimosaka, H. Murai and M. Sato, "A Design of MPI Parallel Library Interface of Parallel Programming Language XcalableMP", IPSJ SIG Technical Reports (in Japanese), Vol.2011-HPC-130 (55), pp.1--8, July, 2011.
4. T. Abe, T. Hiraishi, Y. Miyake, T. Iwashita and H. Nakashima, "Job-level parallel executions for satisfying distributed constraints". Summer United Workshops on Parallel, Distributed and Cooperative Processing, IPSJ SIG Technical Reports (in Japanese), 2011-HPC-130(59), pp.1--8, July, 2011.
5. T. Abe and M. Sato, "Directive-based source code checking for verification-oblivious programming in high performance computing" (in Japanese). Summer Programming Symposium, pp.1--6, September, 2011.

(3) Invited Talks (From April 2011 to March 2012)

1. Mitsuhisa Sato, "The K Computer Project and Research on Parallel Programming Languages", PGAS 2011: Fifth Conference on Partitioned Global Address Space Programming Models, October 18, 2011, Galveston Island, Texas, USA.
2. Mitsuhisa Sato, "Challenges of programming environment and tools for peta-scale computers -- programming environment researches for the K computer ---," Invited Presentation, 4th Workshop on Productivity and Performance (PROPER 2011) Tools for HPC Application Development EuroPar 2011 Conference, Bordeaux/France, August 30th 2011.
3. Mitsuhisa Sato, "Perspective of HPC Development for Computational Science in Japan", HPC in Asia Workshop, ISC 2011.
4. Mitsuhisa Sato, "The K Computer and XcalableMP Parallel Language Project: Towards a Programming Environment for Peta-scale Computing", Joint ENCORE & PEPPHER Workshop on Programmability and Portability for Emerging Architectures (EPoPPEA), Jan 24, 2012, Paris, France. (in conjunction with the HiPEAC'12 conference)

(4) Posters and presentations

1. Takenori Shimosaka, Hitoshi Murai, Mitsuhisa Sato, “A Design of the Interface of XcalableMP Programs to Parallel Numerical Libraries”, the 2nd AICS International Symposium, Poster, March 2012.
2. Tomotake NAKAMURA and Mitsuhisa SATO, “Paralle I/O Facility of PGAS Programming Language XcalableMP and Evaluation of I/O Performance on Parallel File System Lustre”, the 2nd AICS International Symposium, Poster, March 2012.

(5) Patents and Deliverables

1. XcalableMP compiler ver. 0.5 for the K computer
2. Scalasca performance analysis tool for the K computer (test version)
3. GASnet one-sided communication Library for the K computer (test version)