Programming Environment Research Team

1. Team members

Mitsuhisa Sato (Team Leader)
Hitoshi Murai (Research Scientist)
Miwako Tsuji (Research Scientist)
Masahiro Nakao (Research Scientist)
Hidetoshi Iwashita (Research Associate)
Makoto Ishihara (Agency Staff)
Masahiro Yasugi (Senior Visiting Researcher)
Hitoshi Sakagami (Senior Visiting Researcher)
Brian Wylie (Visiting Researcher)
Christian Feld (Visiting Researcher)
Kengo Nakajima (Senior Visiting Researcher)
Tomoko Nakashima (Assistant)

2. Research Activities

The K computer system 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 conduct 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 2014FY, in order to archive these objectives above, we carried out the following researches:

1) We are working on the development and improvement of XcalableMP (XMP) programming languages. XcalableMP is a directive-based language extension, designed by XcalableMP Specification Working Group (XMP Spec WG) including some members from our team as a community effort in Japan. It allows users to develop parallel programs for distributed memory systems easily and to tune the performance by having minimal and simple notations. In this year, we have implemented Coarray functions in Fortran and C using one-sided communication supported efficiently by the K computer.
2) As an extension of XcalableMP to exascale computing, we are proposing a new programming model, XcalableACC, for emerging accelerator clusters, by integrating XcalableMP and OpenACC. We are working on the language design and the compiler development of XcalableACC. We have submitted the results of XcalableACC and XcalableMP to SC14 HPCC class 2 competition, and awarded "HPCC Class2 Performance Award". This research is funded

by JST CREST project on “post-petascale computing”.

3) As a follow-up of Japan-France project FP3C, "Framework and Programming for Post Petascale Computing" (2010-2013), and have been investigated fault tolerance mechanism for a multiple SPMD programming environment, FP2C (Framework for Post-Petascale Computing), developed in FP3C project. This work focuses on the technique of fault tolerance on workflow by replaying faulted tasks.
4) As a study on performance evaluation of the K computer, we are working on HPGMG, a new HPC benchmark program developed by Lawrence Berkeley National Laboratory (LBNL). For large-scale parallel applications, we continue the design of parallel communication library to support the communication between a set of multiple processes in Multiple processes Multiple Data (MPMD), named MPMPI library.
5) We conducted several collaborations on the performance evaluation with JSC, University of Tsukuba and other groups.

In addition to the research activities, we conduct promotion activities to disseminate our software. To promote XcalableMP as a means for parallelization of programs, we made the XcalableMP workshop, seminars or lectures as follows.

- XcalableMP workshop (Oct. 24)
- FOCUS seminar on XMP (Sep. 18, Dec. 18)

The seminar or lecture consists of both classroom and hands-on learning

3. Research Results and Achievements

3.1. Development of an XcalableMP compiler

We are developing Omni XcalableMP that is an open-source XcalableMP compiler, in cooperation with the university of Tsukuba. The latest version 0.9.1 has been released in April, 2015

3.1.1. Coarray Fortran and C

Coarray Fortran (CAF) is a parallel language that is a language extension of Fortran. Since it was accepted as a part of the latest Fortran standard Fortran2008, it is expected to be popular gradually. CAF is a lower-level language than XMP we have been developing and seems to be suitable for fine performance tuning. Therefore, we expect CAF not to be a competitor but to be a complementary to XMP.

This year we started to develop Coarray features into our XMP translator. In general, it is not easy to develop such low-level language as a translator, a source-to-source compiler. The gorl of

development is both functionality to be worth using real-world applications and high performance comparable with MPI library. And we will extend the experience of CAF also into C language.

Now we have developed major features of CAF1.0 specification such as declaration of coarray variables, reference and definition of remote coarray data objects, and dynamic allocation and explicit and automatic deallocation. During the development, we encountered and solved some issues as follows.

- Memory allocation of a coarray variable brings inter-node synchronization as a side effect. It had caused overhead cost at the start point of every procedure containing coarray. To move out and aggregate the costs, we developed a new mechanism which works just before the linkage-editor. It scans all object and archive files to be linked, finds coarrays to be allocated and initialized at the runtime, and generates instructions to allocate and initialize them previously before execution of the user program.
- For effective inter-code communication of coarray data, it is required at runtime to detect how long data is contiguous and how the contiguous data appear frequently. We developed a new interface, which has information made from parameters of multi-dimensional subarray data in Fortran language and is easy to detect how contiguous the data at runtime.

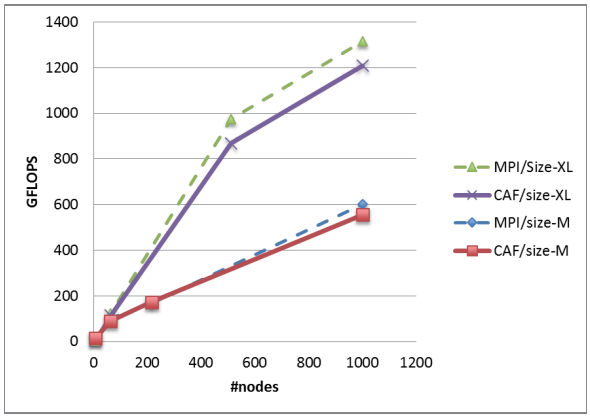

Fig. 1 Comparison of CAF and XMP

We evaluated the current compiler we have developed and concludes that the compiler works correctly with a program written by CAF and the performance is very close to the one of MPI. Fig. 1 shows comparison of CAF and MPI with Himeno benchmark. The MPI program is basically the same as the original and the CAF program was ported from it. In spite of the data size M and XL, the difference of the performance was 10 percent or less. And we still have room for improvement both the compiler and the CAF program.

According to the development and evaluation, we found CAF can be implemented even as a translator with high performance. And the program written in CAF seems much easier than the one of MPI. The latest XMP compiler with the CAF features was released in April 2015.

3.1.2. Evaluation of productivity and performance of XcalableMP

In order to evaluate productivity and performance of XcalableMP, we have implemented HPC Challenge (HPCC) benchmarks. The HPCC benchmarks are a set of benchmarks to evaluate multiple attributes of an HPC system. The HPCC benchmarks consist of High Performance Linpack (HPL), Fast Fourier Transform (FFT), STREAM, and RandomAccess.

Table 1 shows the source lines of code (SLOC) of our implementations. The SLOCs of XcalableMP are smaller than those of the reference implementations by using MPI.

Table 1: Source lines of code of HPCC benchmarks

	HPL	FFT	STREAM	RandomAccess
XcalableMP	313	204	69	253
Reference	8,800	787	329	938

All benchmarks were compiled by using the Omni compiler 0.9.0-alpha. In order to evaluate the performances of these benchmarks, we used all compute nodes at a maximum on the K computer. For comparison, we also evaluated the some reference implementations. For HPL, we compared our performance with the theoretical performance. Fig. 2 shows the performance results. The performances of XcalableMP implementations are almost the same as those of the reference implementations.

Through these implementations and performance evaluations, we have revealed that XcalableMP has good productivity and performance. We have submitted the results to the SC14 HPC Challenge Benchmark Class2 Competition, and we have awarded the HPC Challenge Class 2 Best Performance Award.

Last year, we had implemented the HPCC benchmarks. This year, we have tuned the Omni Compiler for the K computer and algorithms of HPCC benchmarks. Fig. 3 shows the comparison between implementations of last year and this year.

While the SLOCs of this year are almost the same as those of last year, the performance of this year are better than those of last year.

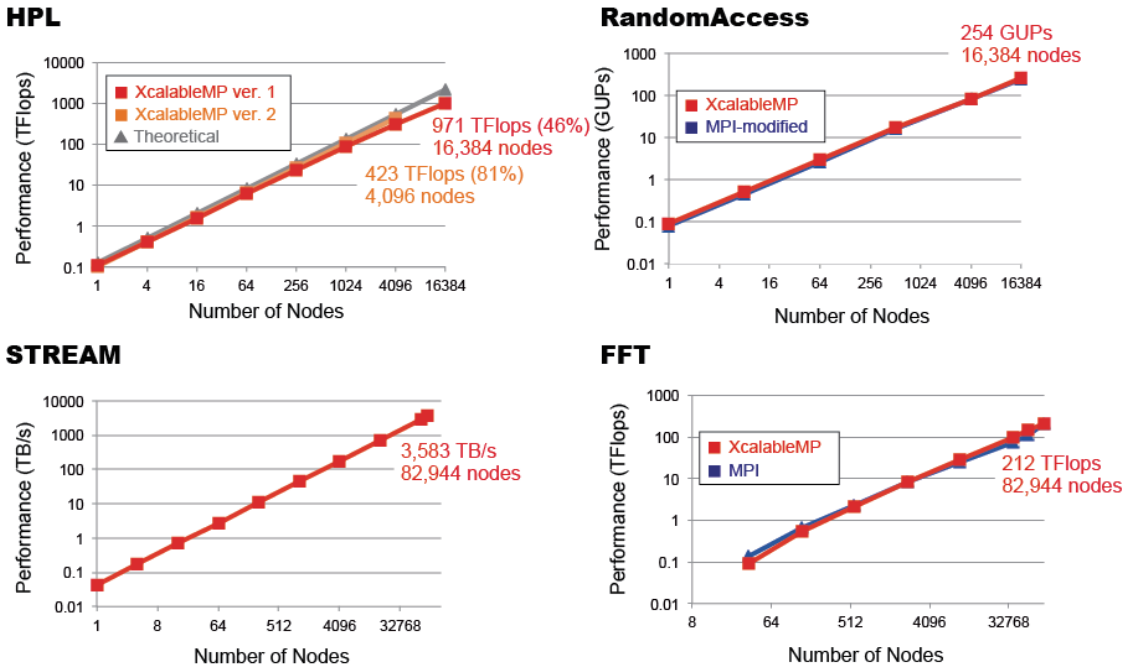

Fig. 2 Performance results of HPCC benchmarks

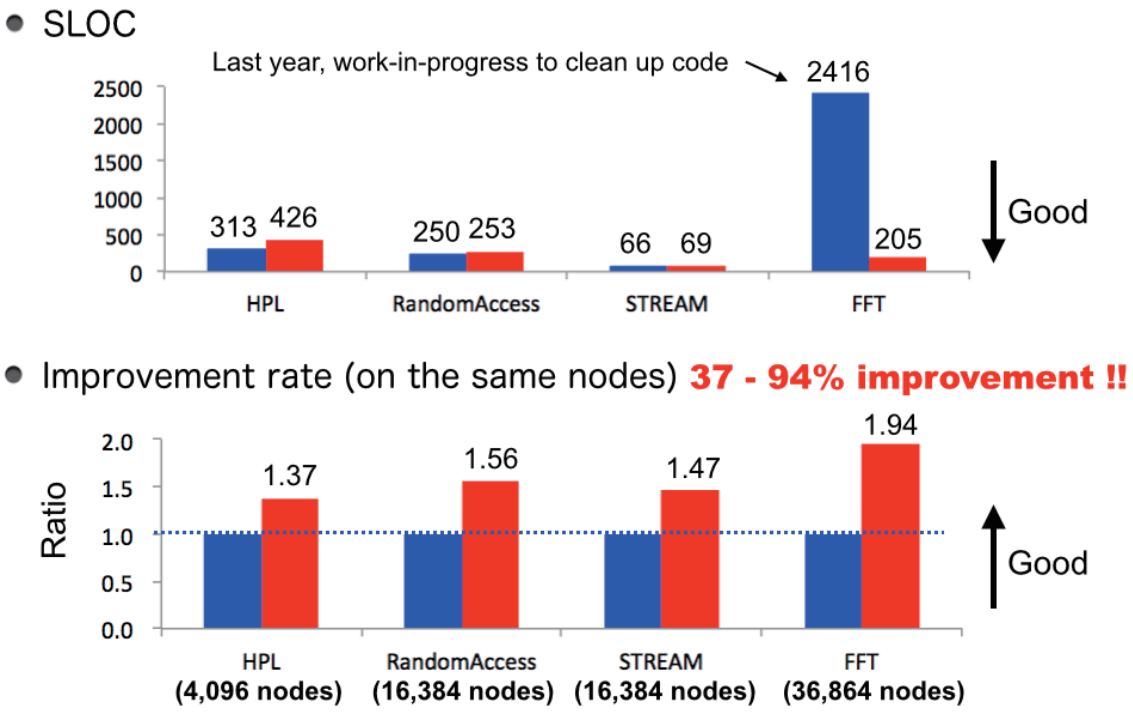

Fig. 3 Comparison between last year and this year

3.2. Multiple SPMD programming environment for fault tolerance

To develop a multiple SPMD programming environment supporting fault tolerance feature, we have extended middleware and developed the prototype environment by incorporating the middleware to the multiple SPMD programming environment.

During recent years, we have developed a multiple SMPD programming environment called FP2C (Framework for Post-Petascale Computing) with our domestic and international collaborators such as University of Tsukuba, University of Versailles, Maison de la Simulation etc… The FP2C combines several programing methodologies such as workflow, distributed parallel, shared memory and allows programmers to make hierarchical programs across multiple architectural levels such as intra-node, inter-node and inter-cluster. In FP2C, a workflow application is executed by YML --- a development and execution environment for a workflow --- and each task in the workflow can be distributed parallel program described XMP, MPI and so on.

YML workflow scheduler uses different middlewares for different programming environments. Especially, in clusters and supercomputers, it adopts OmniRPC-MPI, which is an extension of OmniRPC (Remote Procedure call library) for parallel remote programs. To realize fault tolerance, we have extended OmniRPC-MPI to OmniRPC-MPI-FT by implementing heartbeat messages. Moreover, FJMPI, which is MPI library provided by Fujitsu to realize "safe" communication between two communicators.

Although the OmniRPC-MPI had been developed for FP2C, it can work as an independent RPC library. Therefore, we have implemented the OpenFMO, which is one of fragment molecular orbital methods, with the OmniRPC-MPI-FT. According to the experimental results, while the overhead of the heartbeat messages seems to be ignorable, FJMPI is not efficient.

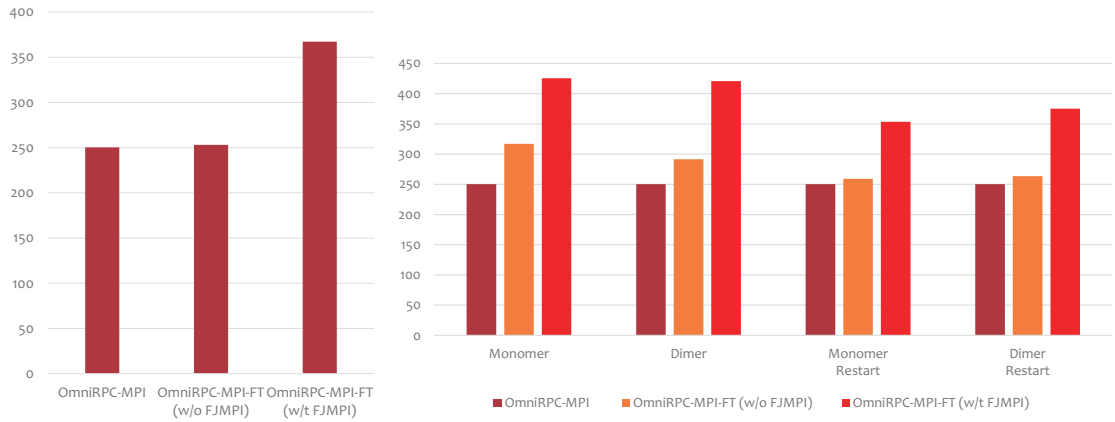

Fig. 4 elapse time.

We have also incorporated the OmniRPC-MPI-FT to the FP2C and improve the algorithm of our workflow scheduler in order to realize automatic fault detection and recovery. FJMPI "safe" communication had not been adopted. We have considered a "task-wise" strategy, in which if the scheduler detects an error in a task, it retries the task until successful completion. We have performed computational experiments with a block Gauss Jordan application and confirmed that

- If FP2C have several tasks simultaneously, it can recover error and complete the whole application.
- The overheads of heartbeat messages and fault recovery are acceptable.

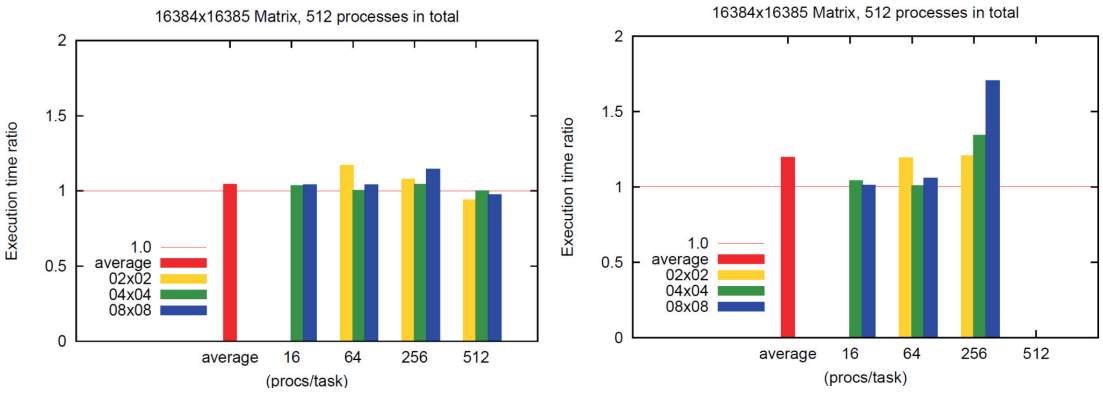

Fig. 5 The ratio of execution time without error(left) and with error(right)

3.3. XcalableMP Extension for Accelerator Clusters

We are designing a new programming model and developing its compiler for emerging accelerator clusters. Specifically, we target accelerator clusters that are capable of direct communication between two accelerator devices on different nodes, called Tightly Coupled Accelerators (TCA). TCA is a next-generation technology for communication between accelerators. This research is supported by CREST, JST.

3.3.1. XcalableACC language

In order to improve productivity of applications using TCA and accelerators, we have developed a new programming model XcalableACC (XACC), which is a combination of XcalableMP and OpenACC, enhanced by features for controlling multi-devices and direct communication between devices (Table 2)

Table 2 XcalableACC directives

XMP directives	distributed-memory parallelism among nodes
OpenACC directives	parallelism inside a device
XACC extensions	parallelism among devices and direct communication

Fig. 6 illustrates the basic concept of XACC.

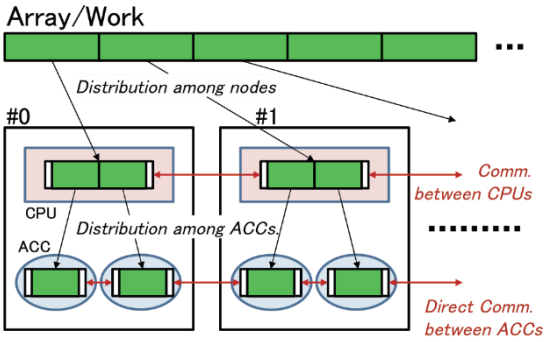

Fig. 6 Execution Model of XACC for data distribution, offloading, and communication

The XACC extensions for parallelism among devices and direct communication include the following clauses and directives.

- acc clause, which specifies which instance of the data (on CPU or ACCs) is to be communicated;
- device directive, which declares an *XACC device* that may be a set of ACCs;
- on_device clause, which specifies the target ACC of OpenACC directives;
- layout clause, which specifies data/work mapping onto an XACC device;
- shadow clause, which specifies the stencil area of a distribute array; and
- barrier_device directive, which specifies a barrier among devices.

Programmers could write their programs for ACC cluseters using XACC with less dificullty than using the de facto approach, the combination of MPI and CUDA. An example code of XACC is given in Fig. 7

```
void foo(){

#pragma xmp nodes p(4)
#pragma acc device d(*)

#pragma xmp template t(0:99)
#pragma xmp distribute t(block) onto p

  float a[100][100];
#pragma xmp align a[i][*] with t(i)
#pragma xmp shadow a[1:1][0]
#pragma acc declare copy(a) layout([*][block]) shadow([0][1:1]) on_device(d)

#pragma xmp reflect (a) acc

#pragma xmp loop (i) on t(i)
  for (int i = 0; i < 100; i++){
#pragma acc parallel loop layout(a[*][j+1]) on_device(d)
    for (int j = 0; j < 99; j++){
      a[i][j+1] = 1;
    }
  }

}
```

Fig. 7 Example code of XACC

3.3.2. Omni XcalableACC Compiler

We are also developing a compiler for XACC, named Omni XcalableACC, in collaboration with the university of Tsukuba. It works as an extended function of the Omni XcalableMP compiler (Fig. 8).

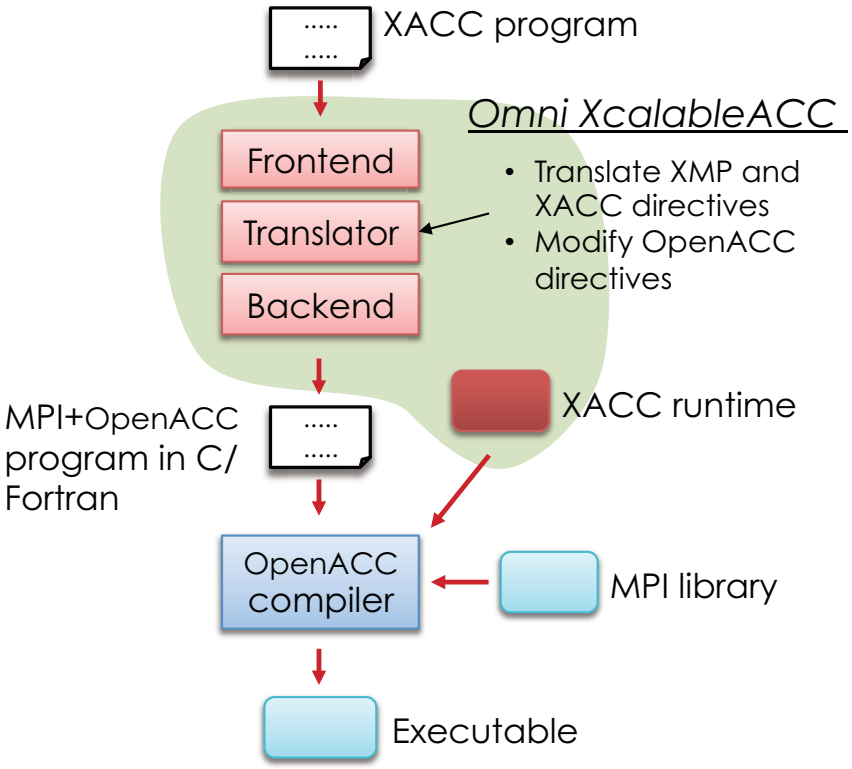

Fig. 8 Configuration of Omni XcalableACC

In this year, we have implemented basic functions of XACC, which include the feature of stencil communication (the reflect directive) based on TCA.

We parallelized the HIMENO benchmark, which is a typical stencil code, with Omni XACC and evaluated its performance on HA-PACS/TCA, which is an accelerator cluster based on the TCA architecture located at Center for Computational Sciences, University of Tsukuba. Fig. 9 and Table 3 show that, using XACC, programmers could enjoy both higher performance and productivity than using the normal means of OpenACC+MPI.

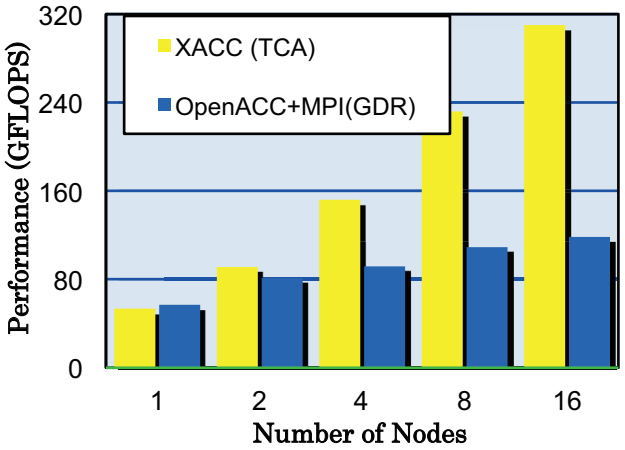

Fig. 9 Performance on HA-PACS/TCA

Table 3 Comparison of Source Lines of Code

	total	XMP	OpenACC	others
XACC	213	28	9	176
OpenACC + MPI	488	-	15	473

Other achievements in this year include prototyping the features of multi-devices and communication on a hybrid network of Infiniband and TCA.

3.3. Evaluation of HPGMG and HPCG

HPGMG is a new HPC benchmark program developed by Lawrence Berkeley National Laboratory (LBNL). Current most famous HPC benchmark program is High Performance Linpack (HPL), which is adopted by TOP500 (http://www.top500.org). HPL has a weakness of very long execution time for large-scale parallel computers because the execution time is proportional to the problem size. HPGMG is developed as an alternative to HPL.

HPCG (https://software.sandia.gov/hpcg/) has also the similar purpose. Both HPGMG and HPCG use iterative solvers based on the multigrid method: a finite-element solver (HPGMG-FE) and a finite-volume solver (HPGMG-FV) in HPGMG, and a symmetric Gauss-Seidel preconditioned conjugate gradient solver in HPCG.

Our team started cooperation in evaluating HPGMG-FV on Apr. 2014. We adjusted tuning parameters of HPGMG-FV for the K computer and measured its weak scaling performance.. As the result, we achieved the first rank of performance in Mar. 2015, as shown at the website of LBNL

(http://crd.lbl.gov/departments/computer-science/performance-and-algorithms-research/research/hpgmg/). Moreover we investigated performance characteristics of HPGMG-FV and HPCG using the profiler of the K computer.

Table 4 shows the weak scaling performance, the ratio of GFLOPS to the peak, and the SIMD occupation rate, as well as the performance without SIMD for comparison. Table 5 shows performance characteristics of HPCG optimized by RIKEN AICS software development team.

Table 4 Weak scaling performance of HPGMG on the K computer

Number of nodes	1	8	64	512	4096	32768	82944
DOF/s (SIMD)	3.99E+07	3.06E+08	2.40E+09	1.89E+10	1.50E+11	1.14E+12	2.83E+12
GFLOPS/Peak(%)	10.3107	9.8652	9.6761	9.5043	9.3916	8.8612	8.6821
SIMD rate(%)	61.5931	61.3401	61.9923	61.1174	61.0208	60.5174	60.425
DOF/s(NOSIMD)	3.88E+07	2.96E+08	2.33E+09	1.84E+10	1.46E+11	1.10E+12	

DOF/s: Degree of Freedom per Second

Table 5 Weak scaling performance of HPCG on the K computer

Number of nodes	1	8	64	512	4096
GFLOPS/Peak(%)	4.37229	4.2616	4.1228	4.10028	4.10913
SIMD rate(%)	17.1305	17.0965	17.0283	17.0269	17.0357

The ratio of GFLOPS to the peak and the SIMD occupation rate in Table 4 are different from those in Table 5 because smoothers of HPGMG and HPCG use different methods: chebychev polynomial and Gauss-Seidel, respectively. Currently, for both benchmarks, SIMD instruction have no effect.

4. Schedule and Future Plan

One of important action for XcalableMP project in recent years is to disseminate our XcalableMP to applications users. As in last year, we organized several schools and hands-on, workshop with potential users also in this year. We will continue these promotion activities for the next year while we will study more optimization technique of XcalableMP compiler to improve the performance. As a research agenda especially for the K computer, we will contribute the scalability of large-scale applications for the K computer.

The programming models for post-petascale will be investigated, including programming models and runtime techniques to support manycore and accelerators such as GPU in large-scale parallel system. XcalableACC is our solution for accelerator-based system, which is to be explored in the

JST CREST project. As the post-K computer will be a large-scale multicore-based system, we will investigate programming models for manycore-based parallel systems including dynamic tasking and load balancing as well as advanced PGAS models for distributed memory systems.

5. Publication, Presentation and Deliverables

(1) Journal Papers

(2) Conference Papers

1. Swann Perarnau, Mitsuhisa Sato: Victim Selection and Distributed Work Stealing Performance: A Case Study. IPDPS 2014: 659-668
2. Masahiro Nakao, Hitoshi Murai, Takenori Shimosaka, Akihiro Tabuchi, Toshihiro Hanawa, Yuetsu Kodama, Taisuke Boku, Mitsuhisa Sato. ``XcalableACC: Extension of XcalableMP PGAS Language using OpenACC for Accelerator Clusters," Workshop on accelerator programming using directives (WACCPD), New Orleans, LA, USA, Nov., 2014.
3. Takenori Shimosaka, Hitoshi Murai, Mitsuhisa Sato, "A Design of a Communication Library between Multiple Sets of MPI Processes for MPMD," 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), pp.1886,1893, 19-21 Dec. 2014

(3) Invited Talks

1. Hitoshi Murai, “XcalableACC: a PGAS Language for Accelerated Parallel Computers,” JST/CREST International Symposium on Post Petascale System Software, Kobe, Japan, 2014-12-04
2. Mitsuhisa Sato, HPC research and development in Japan - Post T2K and post K project -, EuroMPI/Aisa 2014.

(4) Posters and presentations

1. Miwako Tsuji, FP2C a multi-level programming paradigm, J-F Conference Extreme Performance Computational Science, Apr. 2014
2. Miwako Tsuji,The extension of OmniRPC-MPI toward fault tolerant computation, 1st workshop on middleware for fault tolerant, July. 2014 (in Japanese)
3. Takenori Shimosaka, Hitoshi Murai, Mitsuhisa Sato, "Efficient FFT implementation in XcalableMP". IPSJ SIG Technical Report, 2014-HPC-145, Jul. 2014 (in Japanese)
4. Masahiro Nakao, Hitoshi Murai, Takenori Shimosaka, Akihiro Tabuchi, Toshihiro Hanawa, Yuetsu Kodama, Taisuke Boku, Mitsuhisa Sato. “XcalableACC : An extension of XcalableMP for accelerator cluster system”, 2014-HPC-146(7),1-11, Oct. 2014 (in Japanese).

5. Masahiro Nakao, “Performance result and implementation of HPC Challenge Benchmarks by using XcalableMP”, 2nd XcalableMP workshop, Oct. 2014 (in Japanese).
6. Miwako Tsuji and Mitsuhisa Sato, The extension of OmniRPC-MPI toward fault tolerant computation in a multi SMPD environment, 2014-HPC-146, Oct. 2014 (in Japanese).
7. Miwako Tsuji, XMP/YML for a multi SPMD programming environment, 2nd XcalableMP workshop, Oct. 2014 (in Japanese).
8. Masahiro Nakao, Hitoshi Murai, Hidetoshi Iwashita, Takenori Shimosaka, Akihiro Tabuchi, Taisuke Boku and Mitsuhisa Sato. SC14 The 2014 HPC Challenge Awards BoF, New Orleans, LA, USA, Nov., 2014.
9. Masahiro Nakao, “Evaluation of XcalableMP by using HPC Challenge Benchmarks”, PC cluster workshop in Osaka, Feb. 2015 (in Japanese).
10. Miwako Tsuji and Mitsuhisa Sato, An investigation of workflow scheduling to realize fault tolerant in a multi SPMD programming environment, 2015-HPC-148, Mar. 2015 (in Japanese)
11. Miwako Tsuji, Fault Tolerance features of YML-XMP, Workshop on Language and Programming Paradigm for Exascale Applications, Mar. 2015
12. Miwako Tsuji, Fault resilient in a multi SPMD programming environment FP2C, 2nd workshop on middleware for fault tolerant, Mar. 2015 (in Japanese)
13. Masahiro Nakao, Hitoshi Murai, Hidetoshi Iwashita, Takenori Shimosaka, Mitsuhisa Sato, “Evaluation and Implementation of HPC Challenge Benchmarks by using PGAS language XcalableMP, ” 2015-HPC-148(21), Mar. 2015, (in Japanese).
14. Masahiro Nakao, Hitoshi Murai, Miwako Tsuji, Takenori Shimosaka, Ryuhei Harada, Tetsuya Odajima, Akihiro Tabuchi, Keisuke Tsugane, Laurence Beaude, Mitsuru Ikei, Taisuke Boku, Mitsuhisa Sato. “Development and evaluation of parallel language for cluster system equipped with accelerators”. 6th cross-disciplinary symposium of computational science – development, assimilation, and construction of new knowledge, Oct. 2014 (in Japanese).
15. Miwako Tsuji and Mitsuhisa Sato, Programming model for post peta sacale computing, 6th cross-disciplinary symposium of computational science - development, assimilation, and construction of new knowledge, Oct. 2014 (in Japanese).
16. Hidetoshi Iwashita. Coarray Features Contained in Parallel Language XcalableMP. Short Lecture at the booth of SC2014, November, 2014.
17. Hitoshi Murai, Masahiro Nakao, Takehiro Shimosaka, Akihiro Tabuchi, Taisuke Boku, and Mitsuhisa Sato, “XcalableACC - a Directive-based Language Extension for Accelerated Parallel Computing,” SC14, New Orleans, LA, USA, Nov. 2014.

(5) Patents and Deliverables

- Omni XcalableMP compiler ver. 0.9.1 (registered as an AICS-supported software)