# Toward Highly Productive Parallel Programming on Large Scale Accelerated Computing

Taisuke Boku[+],[*]
in cooperation with Hitoshi Murai[$], Masahiro Nakao[$], Jinpil Lee[$], Mitsuhisa Sato[$],
Akihiro Tabuchi[*] and Keisuke Tsugane[*]
+: Center for Computational Sciences, University of Tsukuba
*: Graduate School of Systems and Information Engineering, University of Tsukuba
$: Advanced Institute for Computational Science, RIKEN
Contact: *taisuke@cs.tsukuba.ac.jp*

## Extended Abstract

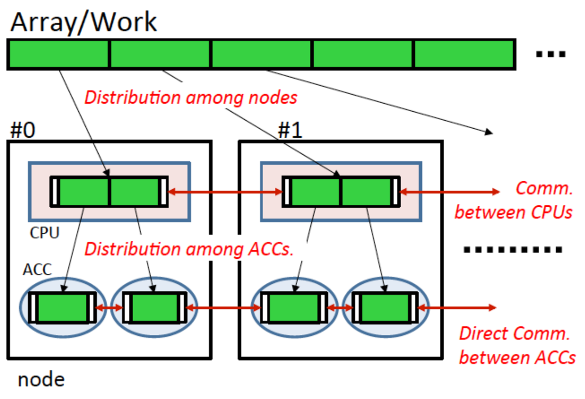
PGAS (Partitioned Global Address Space) languages have been widely proposed and designed for general parallel processing systems with distributed memory architecture. In a cooperation among Center for Computational Sciences (CCS), University of Tsukuba and RIKEN Advanced Institute for Computational Science (AICS), we have been designing and deploying an original PGAS language named XcalableMP (XMP) for large scale systems such as ordinary high-end PC clusters and MPPs. Actually, XMP is implemented on the K Computer, the former world fastest supercomputer built in a national flagship project, IBM BlueGene/Q system, Cray MPP and many of PC clusters. The unique features of XMP over ordinary PGAS language include two memory views with global-view and local-view, coarray remote memory access, easy performance tuning with semi-visible communication among nodes, etc. We have been implementing several types of benchmark codes and real application codes over these features to exploit high performance comparable with traditional MPI+X (X=OpenMP, etc.) style of programming keeping high productivity of coding to release scientists from troublesome program and debugging by careless coding.

On the other hand, the accelerated computing based on various accelerating devices such as Graphic Processing Unit (GPU), Field Programmable Gate Array (FPGA) or early version of many-core processor like Intel Knights Corner, plays important role on the high-end parallel computing especially for efficient performance/power ratio to solve a serious problem of power consumption toward Exascale systems. For the programming of these devices, relatively easy language framework such as CUDA, OpenACC or OpenCL is provided, however the programmability for general computational scientists is still very low compared with traditional parallel coding such as MPI or OpenMP. Because of high programming cost, only the limited applications with highly promised performance by these accelerating devices can be the targets. One of the typical style of parallel accelerated programming on GPU-ready clusters is a combination of MPI+CUDA+OpenMP where the remote node communication, device acceleration and multithreading on host node are programmed, respectively. The users have to be patient not only for acceleration coding but also for internode communication with memory coherence between host and device memories.

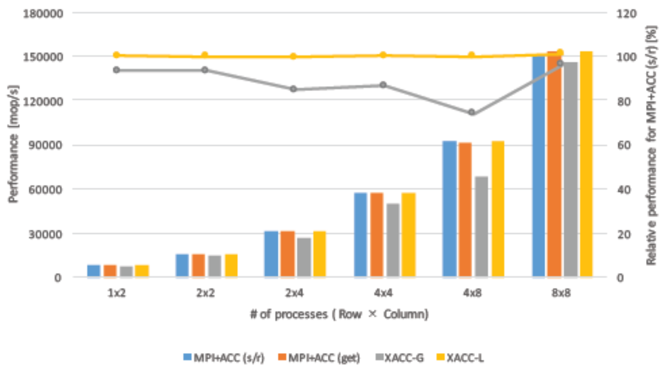
To solve such a complicated programming to cause low productivity on high performance computing, we have been proposing a PGAS base accelerated programming language framework named XcalableACC (XACC) based on XMP language. In a generic view, XACC is an orthogonal combination of XMP and OpenACC to support highly parallel HPC coding on large scale accelerated clusters. Since XMP is a directive base language, it is not easy to apply OpenACC directives directly to the partitioned array space by XMP description. Therefore, we need a dedicated compiler for XACC. In the compilation of XACC code, a global array declared in PGAS space and its related parallel loop iteration are distributed to computation nodes as sub-array and sub-indexed loop where the coherence of partial data is taken by MPI communication in a hidden manner from the programmer. Then, each OpenACC directive is applied for partial array computation keeping the same semantic with ordinary OpenACC. In this concept, XMP directives and OpenACC directives are basically orthogonal where the code neglecting OpenACC directives is compatible with XMP only code and the code neglecting XMP directives is compatible with OpenACC only code. To keep the computation efficiency, we introduce a small fraction of

incompatibility among them.

The two memory views in XMP, global view for distributed array and local view for remote data copy among nodes are also kept in XACC. In the local view model, the users can code a remove memory data copy with local data by array sub-indexing in coarray access manner. In OpenACC, it is user's due to keep the consistency between data in the host memory and the accelerator device memory. XMP directives provide the communication and synchronization among these data as like as original XMP programming. Keeping such compatibility between XMP and OpenACC, the users can enjoy the incremental code modification for performance improvement even on large scale accelerated computing. Currently, XACC in C language version for MPI and Nvidia CUDA translation is available for generic GPU-ready clusters.



The above figure shows the concept of global array distribution and sub-array management over the host and device memories in XACC. Internode communication is basically performed over the distributed array on host memory, but it is also possible among device memories over nodes to introduce CUDA-aware MPI such as MVAPICH2 with GDR feature.



This is a performance result of NAS Parallel Benchmark CG (Class D) in four types of programming (at CCGrid2017); MPI-send/receive and OpenACC, MPI-put/get and OpenACC, global view coding with XACC and local view coding of XACC. All the evaluation is done on HA-PACS/TCA cluster at CCS, University of Tsukuba, where two Nvidia K20X GPUs and dual-rail InfiniBand QDR are equipped on each node. Here, the performance of the local view coding in XACC is comparably high with MPI+OpenACC while the global view coding of XACC degrades the performance.

The code line counts of MPI+OpenACC and XACC are not drastically different, however the contents of code are much easier to understand and describe on XACC. We believe that the high productivity and maintenancebility of large scale accelerated codes are so important in near future of high-end programming. It is one of the most desired features to prepare the next generation of HPC programming for low power and high performance computing.

The development of XACC language system is partially supported by JST/CREST program entitled "Research and Development on Unified Environment of Accelerated Computing and Interconnection for Post-Petascale Era" in the research area of "Development of System Software Technologies for post-Peta Scale High Performance Computing".