XcalableACC – a Directive-based Language Extension for Accelerated Parallel Computing

Hitoshi Murai, Masahiro Nakao
and Takehiro Shimosaka
Advanced Institute for Computational Science
RIKEN
Kobe, Japan
Email: {h-murai,masahiro.nakao,tshimoasaka}@riken.jp

Akihiro Tabuchi
Graduate School of Systems
and Information Engineering
University of Tsukuba
Tsukuba, Japan
Email: tabuchi@hpcs.cs.tsukuba.ac.jp

Taisuke Boku and Mitsuhisa Sato
Center for Computational Sciences
University of Tsukuba
Tsukuba, Japan
Email: {taisuke,msato}@cs.tsukuba.ac.jp

I. Introduction

A type of parallel computer, such as GPU clusters, which is composed of multiple nodes equipped with accelerator devices has become a popular HPC platform. In fact, four of the top ten supercomputers in the latest TOP500 list are of this type. We call it accelerated parallel computer (APC).

To program APCs, the combination of MPI for distributed-memory parallelism among nodes and a dedicated language or tool for offloading works to accelerator devices within a node (e.g. CUDA for NVIDIA's GPU) is usually adopted. However such programming model is complicated and difficult for most of application programmers, and a easier way to program APCs is strongly demanded. On the other hand, some technologies such as Tightly Coupled Accelerators (TCA) [1] and GPUDirect [2] that enable direct communication between accelerator devices are recently proposed.

Our research proposes a new language XcalableACC to meet those demands, which is a combination of two existing directive-based language extensions, XcalableMP and OpenACC.

XcalableMP (XMP) [3], developed by the XMP Specification Working Group of the PC Cluster Consortium, is a directive-based language extension for C and Fortran to program distributed-memory parallel computers. Using XMP, programmers can write parallel programs by inserting simple directives into their serial programs.

OpenACC [4], developed by Cray, CAPS, NVIDIA and PGI, is another directive-based language extension designed to program heterogeneous CPU/accelerator systems. It targets offloading programs from a host CPU to an attached accelerator device, and has an advantage of portability across operating systems and various types of host CPUs and accelerators.

XcalableACC (XACC) has features for handling distributed-memory parallelism, derived from XMP, and offloading tasks to accelerators, derived from OpenACC, and two additional functions: data/computation mapping among multiple accelerators and direct communication between accelerators.

II. Related Works

Extensions of PGAS languages [5]–[7] for supporting accelerators have been proposed but neither of them contain features for direct communication between accelerators.

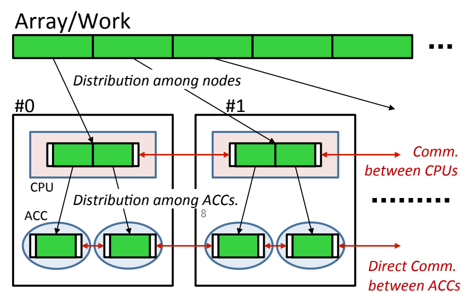

Fig. 1. Execution Model of XACC for data distribution, offloading, and communication

XcalableMP-dev [8] is the predecessor of XACC. XcalableMP-dev supports its own dedicated directives to program GPUs and direct communication between accelerators while XACC supports OpenACC, which is a standard of programming accelerators.

III. XcalableACC

A. Basic Concept

XACC can be said to be a combination of XMP, OpenACC and some novel extensions. Each of them has the function as follows.

- XMP for distributed-memory parallelism
- OpenACC for offloading works to accelerator
- XACC extensions for handling multiple accelerators and direct communication between accelerators

Fig. 1 shows the basic concept of XACC. An example code of XACC is given in Fig. 2.

B. Target Architecture

The target of XACC is accelerated parallel computers that are composed of homogeneous *nodes* each of which is:

- composed of a host CPU (maybe multi-core) and one or more attached accelerator devices; and

```
#pragma xmp nodes p(*)
#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 kernels loop layout(a[*][j]]) on_device(d)
   for (int j = 0; j < 100; j++){
      a[i][j] = ...
   }
}

...
```

Fig. 2. Example Code of XACC

- interconnected via two paths: one is between CPUs, and the other is between accelerators that allows accelerators communicate directly with each other.

C. XACC extensions

1) Data/Computation mapping onto multiple accelerators (two-level distribution):

- The novel `on_device` clause can be put on some OpenACC directives (e.g. `declare`, `data`, etc.) to explicitly specify their target device.
- Data and computation are distributed among nodes by an XMP directive, and further distributed among accelerators within one node by the `layout` clause of the `declare` and `loop` directives.

2) Direct Communication between accelerators:

- The XACC runtime system recognizes the arrangement of data in the host or device memory and autonomously selects the appropriate communication path for them.
- XMP's Communication directives, such as `reflect`, `bcast`, and `reduction`, would apply to data that reside in device memory if the `acc` caluse is specified.

D. Implementation

We are implementing *Omni XcalableACC* as an additional function of the Omni XMP compiler being developed by RIKEN AICS and University of Tsukuba. Its primary target is HA-PACS/TCA [1] in Center for Computational Science, University of Tsukuba.

IV. Preliminary Evaluation

We parallelized the Himeno benchmark [9] (size = 128x128x256), which is a typical stencil code, with Omni XACC to preliminarily evaluate the performance of an XACC

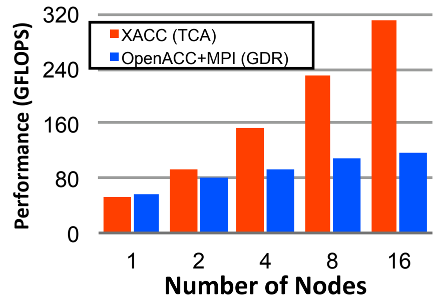

Fig. 3. Result of Preliminary Evaluation

program on HA-PACS/TCA. We used gcc-4.7 and CUDA6.0 as a backend compiler, and MVAPICH2-GDR 2.0b as a communication library. It can be seen from Fig. 3 that the XACC program using TCA is up to 2.7 times faster than the OpenACC+MPI(GDR) equivalent. In addition, the SLOC (source lines of codes) in XACC is about half of that in OpenACC+MPI(GDR).

V. Conclusion

We proposed a directive-based language extension for accelerated parallel computing, XcalableACC. It is basically a combination of XcalableMP and OpenACC, and has advanced features of data/computation mapping onto multiple accelerators and direct communication between accelerators. The preliminary evaluation showed that XcalableACC would be useful to program accelerated parallel computers.

Acknowledgment

The present study is supported in part by the 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."

References

[1] T. Hanawa *et al.*, *Interconnection Network for Tightly Coupled Accelerators Architecture*, High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on, pp. 79–82, 2013.

[2] NVIDIA Corporation, *NVIDIA GPUDirect*, https://developer.nvidia.com/gpudirect, 2014.

[3] XcalableMP Specification Working Group, *XcalableMP Specification Version 1.2*, http://www.xcalablemp.org/download/spec/xmp-spec-1.2.pdf, 2013.

[4] OpenACC-Standard.org, *The OpenACC Application Programming Interface Version 2.0*, http://www.openacc.org/sites/default/files/OpenACC.2.0a_1.pdf, 2013.

[5] D. Cunningham *et al.*, *GPU programming in a high level language: compiling X10 to CUDA*, Proc. 2011 ACM SIGPLAN X10 Workshop (X10 '11), New York, NY, USA, 2011

[6] A. Sidelnik *et al.*, *Performance Portability with the Chapel Language*, Proc. IEEE 26th International Parallel and Distributed Processing Symposium, pp. 582–594, 2012.

[7] L. Chen *et al.*, *Unified parallel C for GPU clusters: Language extensions and compiler implementation*, Languages and Compilers for Parallel Computing, pp. 151–165, 2011.

[8] J. Lee *et al.*, *An Extension of XcalableMP PGAS Language for Multi-node GPU Clusters*, Ninth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar 2011), Bordeaux, France, Aug. 2011.

[9] *The Riken Himeno CFD Benchmark*, http://accc.riken.jp/2444.htm.