Performance Evaluation of NICAM-DC-MINI using XcalableACC on Accelerated Cluster

Masahiro Nakao (RIKEN AICS)
Hitoshi Murai, Akihiro Tabuchi, Taisuke Boku, Mitsuhisa Sato

XcalableACC parallel language

Overview

XcalableACC (XACC) is a directive-based language extension of C and Fortran for accelerated cluster systems (C++ on the table).

- High productivity by directives and coarray features
- High performance by direct communication between accelerators

Components

- XcalableMP (XMP) for distributed-memory parallelism

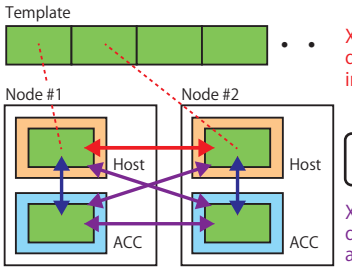
XcalableMP XMP is a directive-based language extension of C and Fortran for cluster systems

- OpenACC for offloading works for accelerators

OpenACC DIRECTIVES FOR ACCELERATORS OpenACC is also directive-based language extension for heterogeneous CPU/Accelerator systems

- XACC for communication of data on accelerators

Memory Model



XMP function defines ``Template'' as a dummy array that represents a global index space.

XACC function enables users to transfer data between accelerators and between accelerator and host memory directly.

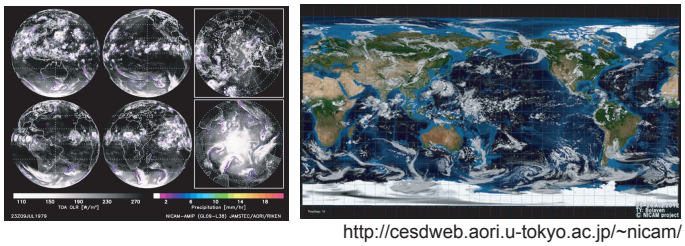
Omni XcalableACC Compiler

- *http://omni-compiler.org*
- Developed by RIKEN AICS and Center for Computational Sciences in University of Tsukuba

Implementation of NICAM-DC-MINI

What is NICAM-DC-MINI ?

- A subset of NICAM dynamical core package
  - NICAM stands for Nonhydrostatic ICosahedral Atmospheric Model, which is an application for Global Cloud Resolving Model
  - Developed by the Japan Agency for Marine-Earth Science and Technology (JAMSTEC), Atmosphere and Ocean Research Institute (AORI) at The University of Tokyo, and RIKEN Advanced Institute for Computational Science (AICS).



http://cesdweb.aori.u-tokyo.ac.jp/~nicam/

Implementation

- Based on the existing NICAM-DC-MINI using MPI and OpenACC
- To exchange sleeve regions among processes, we use coarray features instead of MPI
  - MPI_Send/Isend → coarray assignment
  - MPI_Recv/Irecv → (be deleted)
  - MPI collective communication → intrinsic subroutine (e.g. co_max)
  - MPI_Wait and MPI_Barrier → sync all statement

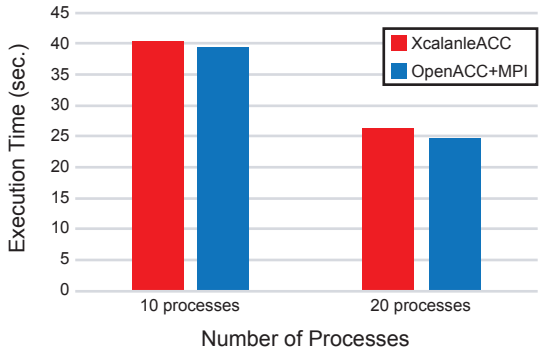
```
do ro=1,romax(halo)
   call mpi_irecv(recvbuf(1,ro), rsize(ro,halo)*cmax, mpi_double_precision,  &
                  sourcerank(ro,halo), recvtag(ro,halo),                     &
                  ADM_comm_run_world, areq(ro), ierr)
end do

do so=1,somax(halo)
   call mpi_isend(sendbuf(1,so),  ssize(so,halo)*cmax, mpi_double_precision, &
                  destrank(so,halo), sendtag(so,halo),                       &
                  ADM_comm_run_world, areq(so+romax(halo)), ierr)
end do

call mpi_waitall(acount,areq,stat,ierr)
```

Additional sync all statement is required to ensure that the array recvbuf on all images can be used.

```
sync all
do so=1,somax(halo)
   recvbuf(1:ssize(so,halo)*cmax, dstimg(so))[destrank(so,halo)+1] =          &
           sendbuf(1:ssize(so,halo)*cmax,so)
end do
sync all
```

Evaluation on HA-PACS/TCA

- On HA-PACS/TCA system located in University of Tsukuba
  - Each computer nodes has four GPUs (NVIDIA K20X)
- Data set is gl06rl01z80, which is executed with strong scaling
- The results of XACC are almost the same as those of OpenACC + MPI



Acknowledgement

This research was supported by Interdisciplinary Computational Science Program in the Center for Computational Sciences, University of Tsukuba and the JST CREST entitled ``Research and Development on Unified Environment of Accelerated Computing and Interconnection for Post-Petascale Era''.