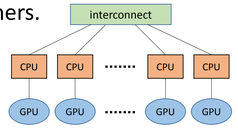
XcalableACC

a Directive-based Language Extension for Accelerated Parallel Computing

Hitoshi Murai, Masahiro Nakao, Takenori Shimosaka (RIKEN AICS)
Akihiro Tabuchi (U. Tsukuba)
Taisuke Boku, Mitsuhisa Sato (CCS, U. Tsukuba)

Background and Goal

- Accelerated parallel computers (e.g. GPU clusters) emerged.
- MPI+CUDA style programming is difficult for programmers.
- Direct communication between accelerators would be important to improve strong scaling.
  - NVIDIA GPUDirect
  - Tightly Coupled Accelerators (TCA)



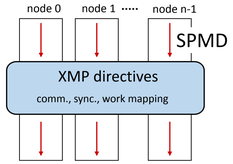
Our Approach: New language for accelerated parallel computers by combining XcalableMP and OpenACC

XcalableMP

www.xcalblemp.org

A directive-based language extension of C and Fortran for distributed-memory parallel systems

- Data and work mapping among *nodes*
- Global communication (e.g. stencil, reduction, etc.)
- One-sided communication like Fortran's coarray



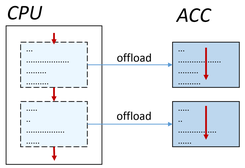

```
!$xmp nodes p(2,2)
!$xmp template t(n,n)
!$xmp distribute t(block,block) onto p
      real a(n,n)
!$xmp align a(i,j) with t(i,j)
!$xmp shadow a(1,1)

!$xmp reflect (a)

!$xmp loop (i,j) on t(i,j)
      do j = 2, n-1
      do i = 2, n-1
        w = a(i-1,j) + a(i+1,j) + ...
        ...
```

OpenACC

Directives for Accelerators

Another directive-based language extension for heterogeneous CPU/ACC systems

- Offloading programs from a host CPU to an attached accelerator device (ACC)
- Portability across operating systems and various types of CPUs and ACCs.

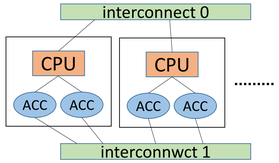



```
#pragma acc data copyin(A[N])
{
  #pragma acc kernels
  {
    #pragma acc loop independant
    for (int i = 0; i < N; ++i){
      A[i][0] = ...;
    }
    ...
  }
  #pragma acc update host(A)
  ...
}
```

XcalableACC = XcalableMP + OpenACC + Novel Extensions

Basic Concept

- XcalableMP for distributed-memory parallelism
- OpenACC for offloading works to ACC
- XACC extensions for:
  - Multiple ACCs
  - Direct communication between ACCs

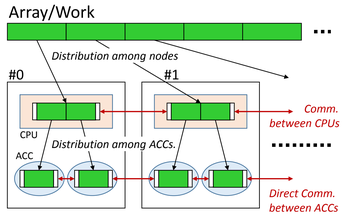


Target Architecture

- Homogeneous nodes composed of one CPU and one or more ACCs
- Two kinds of interconnect: between CPUs and between ACCs

XACC Extensions

- Data and work mapping onto multiple ACCs (two-level distribution)
  1. Distribution among nodes by XMP directives
  2. Distribution among ACCs within a node by the novel `layout` and `on_device` clauses.
- Direct communication between ACCs
  - Communication directives from XMP accept data on the ACC memory.



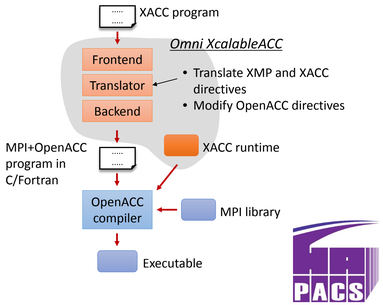

```
#pragma xmp nodes p(*)
#pragma acc device d(*)

#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] = ...;
   }
}
```

XACC extension directives/clauses (tentative)

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

Omni XcalableACC

- being developed as an additional function of the *Omni XcalableMP* compiler
- Translating an XACC program to an MPI+OpenACC program.
- primary target: HA-PACS/TCA at CCS, U. Tsukuba

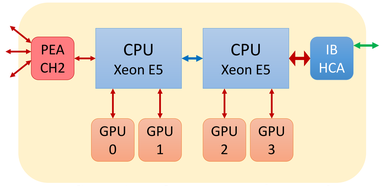


Preliminary Evaluation

Environment

- HA-PACS/TCA, based on the TCA architecture
- the "PEACH2 (PCI-Express Adaptive Communication Hub ver.2)" chip enables direct communication between accelerators
- Omni as the backend OpenACC compiler

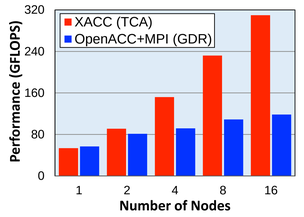
CPU	Intel Xeon-E5 2680v2 2.8 GHz x 2 Socket
peak	224 GFlops/CPU
memory	DDR3 1866 MHz x 4 channels, 128GB
GPU	NVIDIA Tesla K20X x 4 GPU
peak	1.31 TFlops/GPU
memory	GDDR5 6GB/GPU
Interconnect	Mellanox Connect-X3 Dual-port QDR
#nodes	64
compiler	gcc-4.7, CUDA6.0
comm. library	MVAPICH2-GDR



Configuration of an HA-PACS/TCA node

Evaluation

- The Himeno benchmark (a very typical stencil code) parallelized with XACC
- size = 128x128x256
- Up to 2.7 times faster than MPI+OpenACC
- About half SLOC of MPI+OpenACC.
- XACC could support both high performance and high productivity.




```
float p[MIMAX][MJMAX][MKMAX];
// data mapping directives of XMP
#pragma xmp shadow p[1:1][1:1][0]

#pragma acc data copy(p) ...
{
..
#pragma xmp reflect (p) acc
..
#pragma xmp loop (k,j,i) on t(k,j,i)
#pragma acc parallel loop collapse(3) ...
for (i=1; i < MIMAX; ++i)
  for (j=1; j < MJMAX; ++j){
    for (k=1; k < MKMAX; ++k){
      S0 = p[i+1][j][k] * ...;
```

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."