# XcalableACC: Overview and Prospects

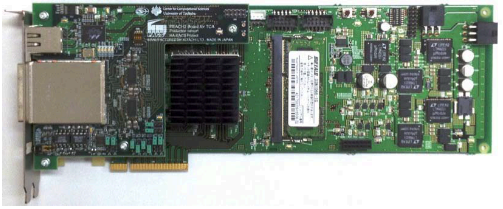
Hitoshi Murai

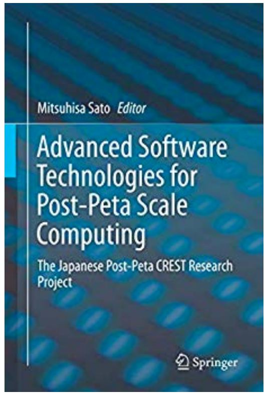
RIKEN R-CCS

# Background

- “Accelerator and Communication Unification for Scientific Computing” Project (2012-2018) `https://post-peta-crest.github.io/`
  - part of JST CREST post-petascale software program
  - PI: Prof. Boku @ U. Tsukuba
  - U. Tsukuba, Keio U. & RIKEN

- Tightly-coupled accelerators (TCA)
  - communication support by FPGA (PEACH2)
    - enables direct comm. between GPUs
  - GPU-accelerated language





Mitsuhisa Sato *Editor*

Advanced Software Technologies for Post-Peta Scale Computing

The Japanese Post-Peta CREST Research Project

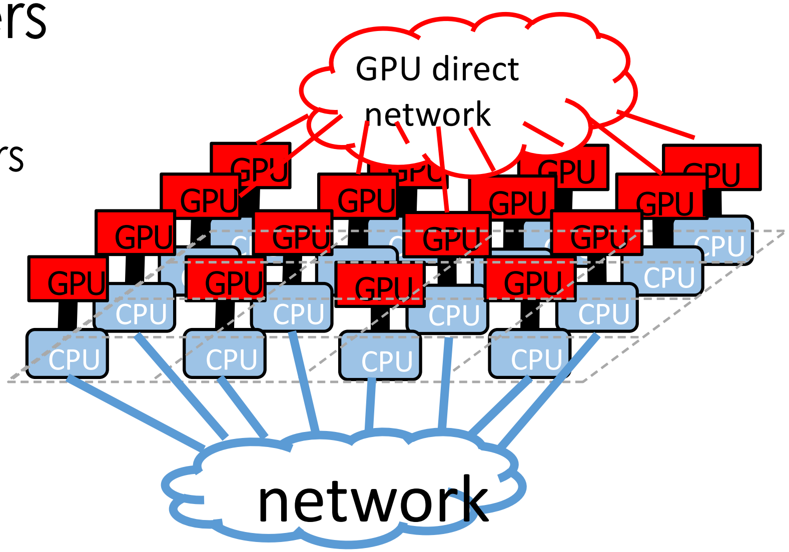
Springer

# Introduction (1)

- Accelerated clusters (e.g. GPU clusters) have become very popular HPC platforms.
- MPI+CUDA style programming lowers productivity.
- Two directive-based languages exist:
  - XcalableMP (XMP) as an alternative to MPI
  - OpenACC Directives for Accelerators as an alternative to CUDA

# Introduction (2)

- GPU accelerated cluster HPC system
  - A collection of nodes has GPUs as an accelerator
  - GPUs may have their own direct interconnect (i.e. *tightly-coupled accelerators*)
- Challenges in such accelerated clusters
  - Efficient (unified) programming model (not MPI+X)
  - Support of direct connection between accelerators
    - TCA by PEACH2
    - GPUDirect
    - etc.



# Goals

- Proposing a new programming language for accelerated clusters by combining XcalableMP and OpenACC
  - unified programming
  - direct communication among accelerators
- Developing its compiler

➡ Realizing high performance and productivity on accelerated clusters

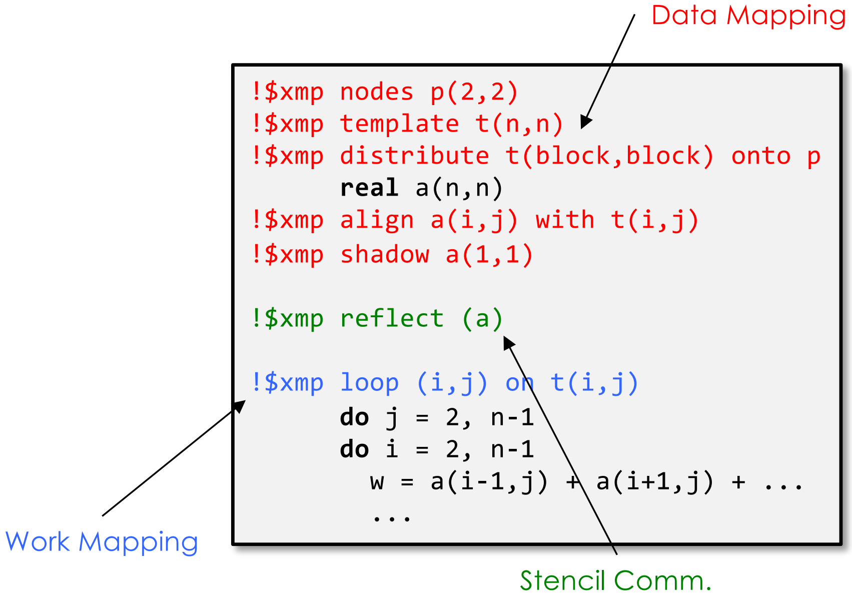
# Outline of This Talk

- What's XcalableMP (and OpenACC)?
- Design of the XcalableACC language
- Implementation of the Omni XcalableACC compiler
- Case study (QCD code)
- Prospects

# What's XcalableMP?

xcalablemp.org

- Directive-based PGAS extension for Fortran and C
  - Proposed by XMP Spec. WG of PC Cluster Consortium.
  - C++ support planned.
- Supports two parallelization paradigms:
  - Global-view (with HPF-like data/work mapping directives)
  - Local-view (with coarray)
- Allows mixture with MPI and/or OpenMP.



Data Mapping

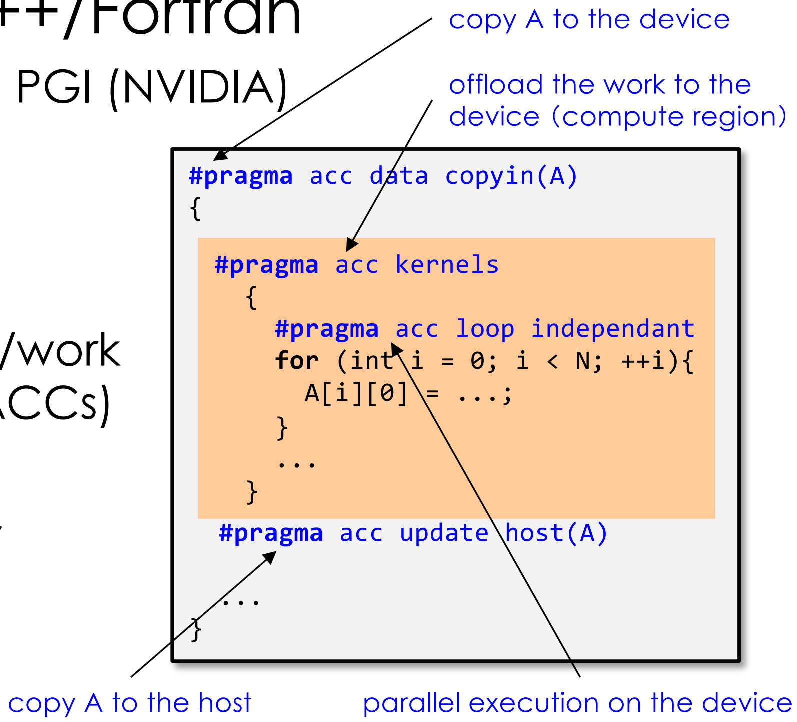
```
!$xmp nodes p(2,2)
!$xmp template t(n,n)
!$xmp distribute t(block,block) onto p
      real a(n,n)
!$xmp align a(i,j) with t(i,j)
!$xmp shadow a(1,1)

!$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) + ...
        ...
```

Work Mapping

Stencil Comm.

# What's OpenACC?

- Directive-based extension to program accelerators for C/C++/Fortran
  - Developed by Cray, CAPS, PGI (NVIDIA)
- Based on the <u>offload model</u>
  - A host (CPU) offloads data/work to devices (accelerators, ACCs)
- Portability across OSs, CPUs and ACCs.



copy A to the device

offload the work to the device (compute region)

```
#pragma acc data copyin(A)
{

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

    ...
}
```

copy A to the host

parallel execution on the device

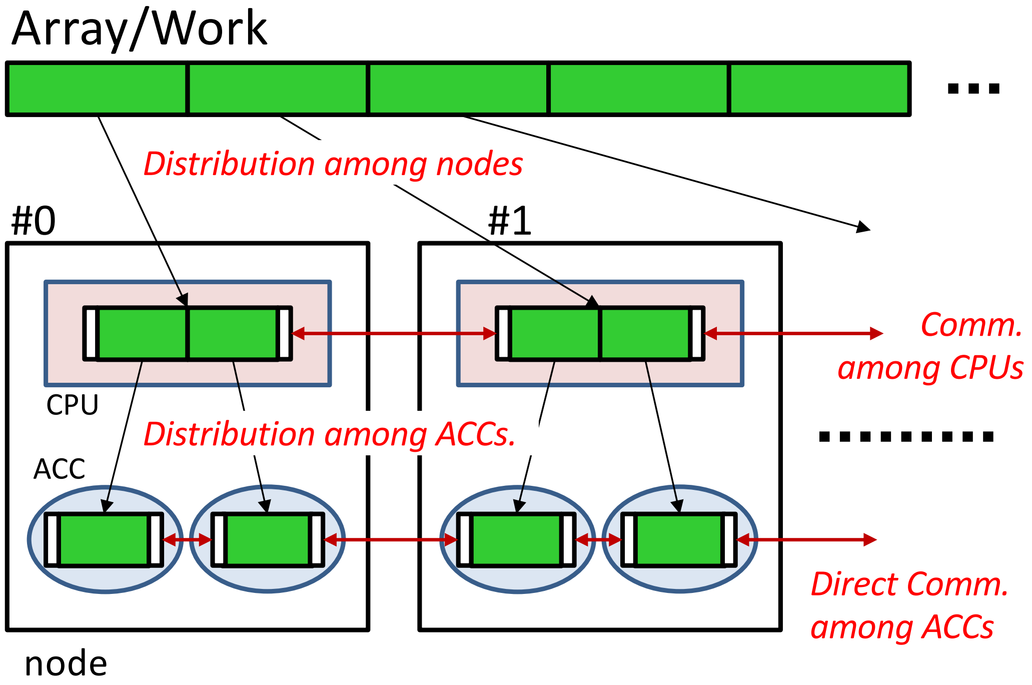
# Basic Concepts of XACC

- XACC = XMP + OpenACC + XACC Extensions

| | |
|---|---|
| XMP directives | distributed-memory parallelism among nodes |
| OpenACC directives | accelerator(s) within a node |
| XACC Extensions | • (hierarchical parallelism)<br>• direct comm. between ACCs |

- With XACC, XMP features (including coarray) can be applied to ACCs for productivity.

# Execution Model of XACC



# Syntax of XACC

- Diagonal combination of XMP and OpenACC
  - XMP outer and OpenACC inner (first distribute among nodes, and then onto accelerators)
- XACC extension
  - XMP's comm. directives with the `acc` clause target data on the device

```
#pragma xmp reflect (a) acc
```

# Example ( Himeno BMT )

Serial (original) code

```
float p[MIMAX][MJMAX][MKMAX];





...

...


for(i=1 ; i<MIMAX ; ++i)
  for(j=1 ; j<MJMAX ; ++j){
    for(k=1 ; k<MKMAX ; ++k){
      S0 = p[i+1][j][k] * ..;
```

# Example ( Himeno BMT )

XMP code

```
float p[MIMAX][MJMAX][MKMAX];
#pragma xmp align p[i][j][k] with t[i][j][k]
#pragma xmp shadow p[1:1][1:1][0]




...
#pragma xmp reflect (p)
...
#pragma xmp loop (k,j,i) on t(k,j,i)

for(i=1 ; i<MIMAX ; ++i)
  for(j=1 ; j<MJMAX ; ++j){
    for(k=1 ; k<MKMAX ; ++k){
      S0 = p[i+1][j][k] * ..;
```

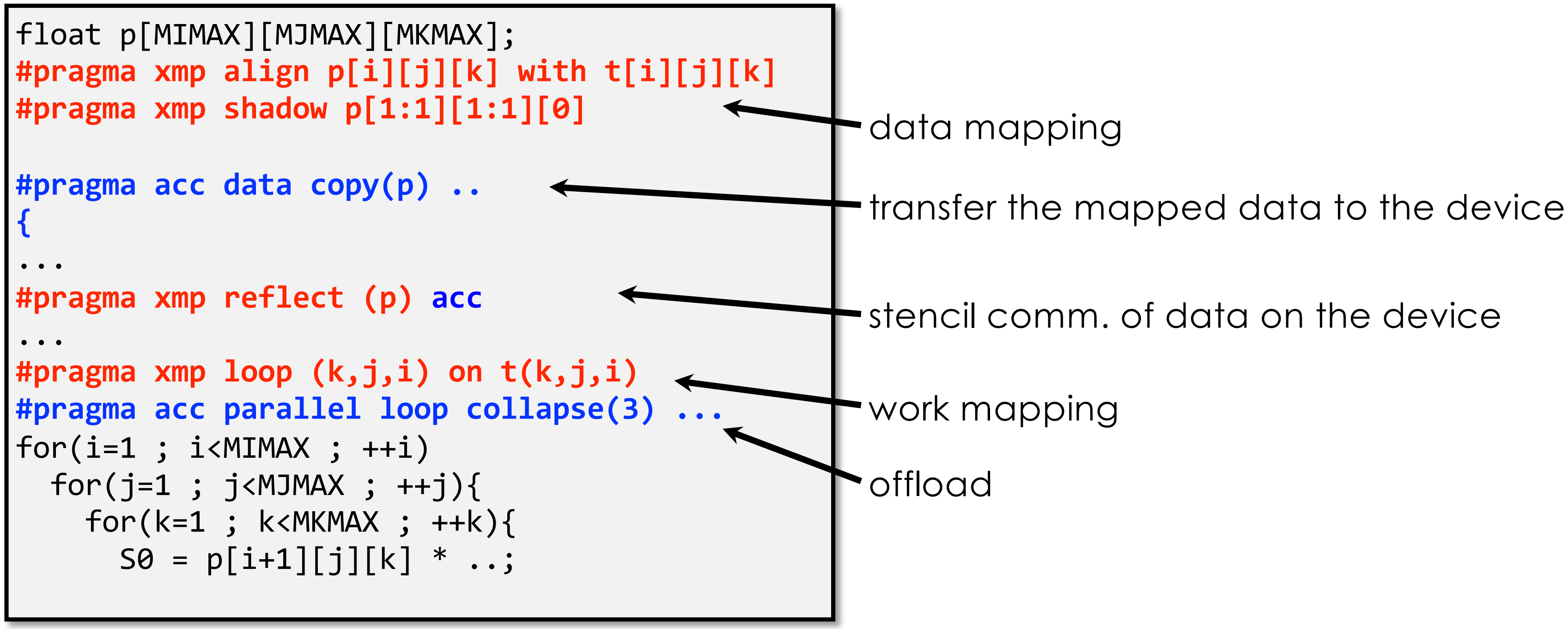
data mapping

stencil comm.

work mapping

# Example ( Himeno BMT )

XACC code



```
float p[MIMAX][MJMAX][MKMAX];
#pragma xmp align p[i][j][k] with t[i][j][k]
#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] * ..;
```

data mapping

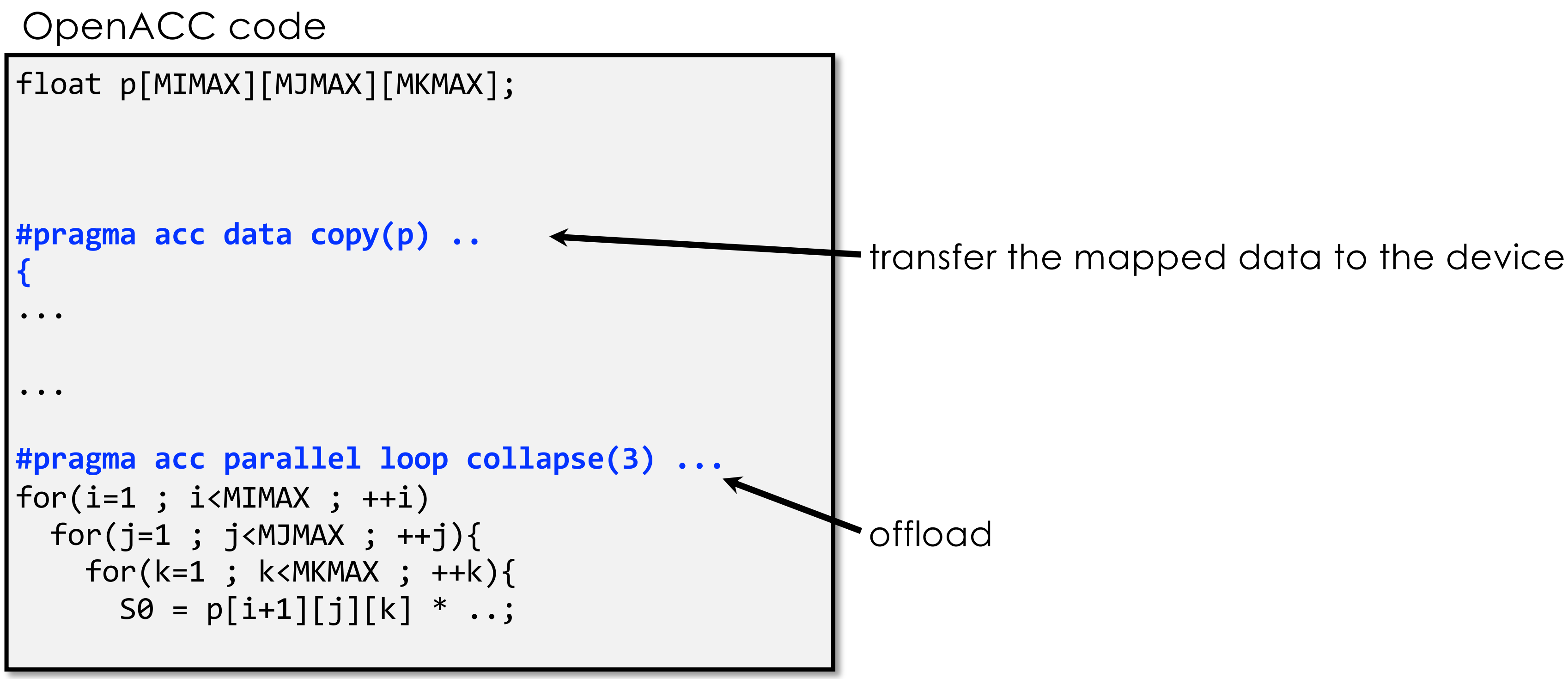
transfer the mapped data to the device

stencil comm. of data on the device

work mapping

offload

# Example ( Himeno BMT )



OpenACC code

```
float p[MIMAX][MJMAX][MKMAX];



#pragma acc data copy(p) ..
{
...

...

#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] * ..;
```

transfer the mapped data to the device

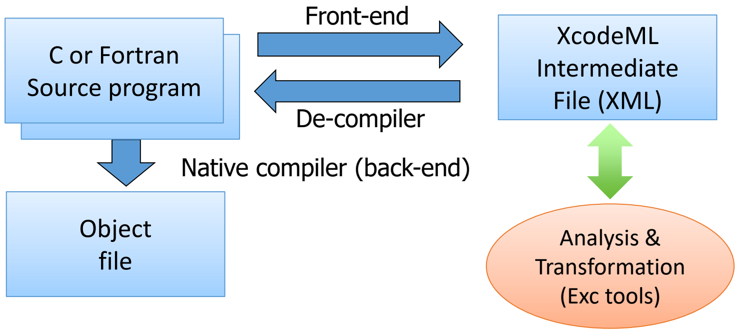
offload

omni-compiler.org

# Omni Compiler Infrastructure

- A collection of programs and libraries that allow users to build code transformation compilers.
- Source-to-source translation
- Supported base languages:
  - C99
  - Fortran 2008
  - C++ (planned)

- Supported directives:
  - OpenMP (C, F)
  - OpenACC (C)
  - XcalableMP (C, F)
  - XcalableACC (C, F)

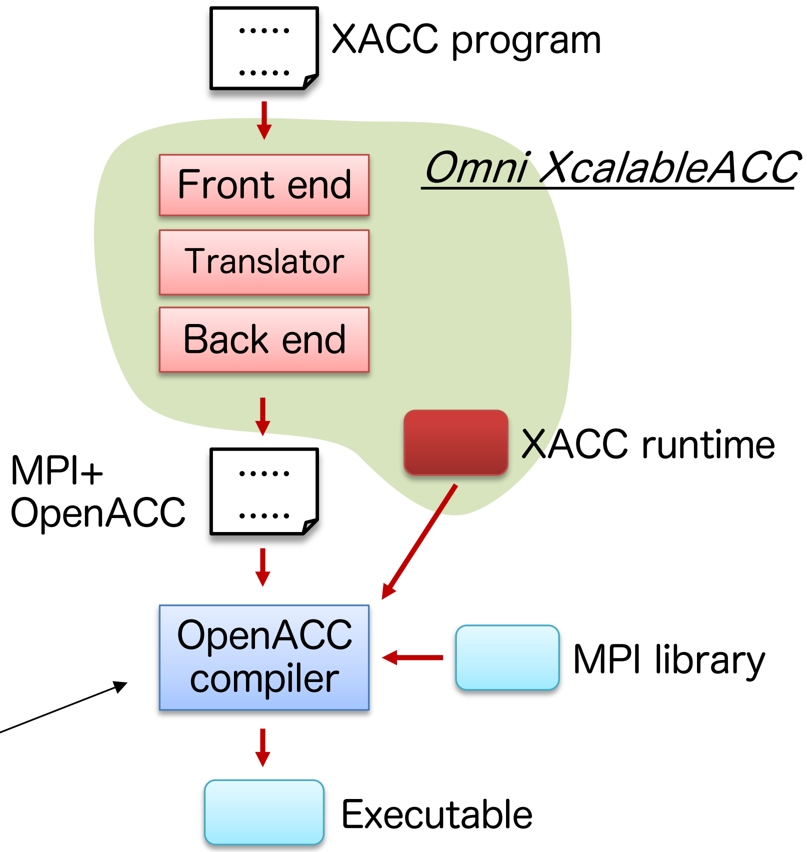

C or Fortran Source program
Front-end
De-compiler
XcodeML Intermediate File (XML)
Native compiler (back-end)
Object file
Analysis & Transformation (Exc tools)

# Omni XcalableACC

- Based on the Omni compiler infrastructure
- Direct comm. between devices is based on:
  - TCA by PEACH2 for HA-PACS/TCA, or
  - GPUDirect for general machines

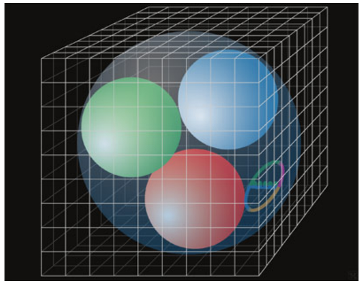
Omni OpenACC

- Accepts C with OpenACC 1.0 (+ part of 2.0).
- Translates OpenACC into CUDA or OpenCL.
- Can work as the back-end compiler of Omni XACC.



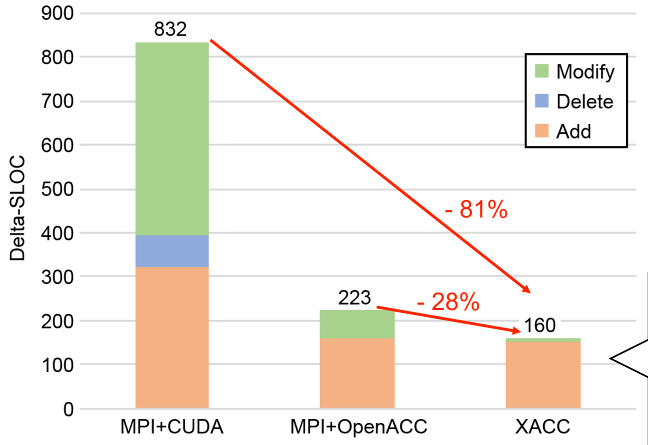
# XACC Case Study: QCD Mini-apps

- The XACC code based on an existing Lattice QCD mini-application (http://research.kek.jp/people/matufuru/Research/Programs/index.html)
  - By High Energy Accelerator Research Organization, Japan
  - Written in C, SLOC (Source Lines of Code) is 842
  - Implemented by extracting the main kernel of the Bridge++
- Parallelized in the directive-based global-view model by XcalableACC



# How Do We Evaluate Productivity ?

- Delta Source Lines of Codes (Delta-SLOC) metric
  - Indicates how many lines are changed from a serial code to a parallel code
  - Sum of three components: how many lines are added, deleted and modified
  - When the Delta-SLOC is small, productivity is good.



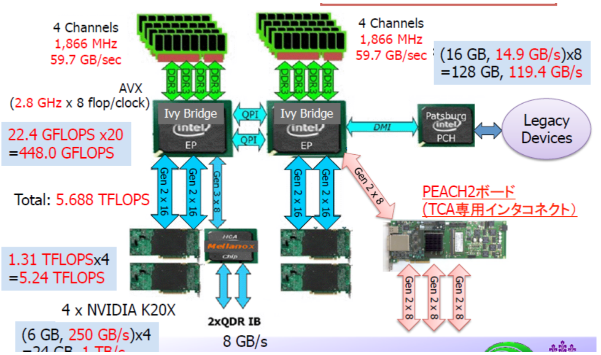
By using XACC,
a programmer develops
a parallel code by adding and
modifying 160 lines to the
serial code

SLOC of the serial code is 842

# HA-PACS/TCA Cluster : TCA test-bed

| | |
|---|---|
| *CPU/Memory* | *Intel Xeon-E5 2680v2 2.8 GHz / DDR3 SDRAM 128GB 59.7GB/s x 2* |
| *GPU/Memory* | *NVIDIA Tesla K20X / GDDR5 6GB 250GB/s x 4* |
| *Network* | *InfiniBand Mellanox Connect-X3 4xQDR x 2rails 8GB/s* |

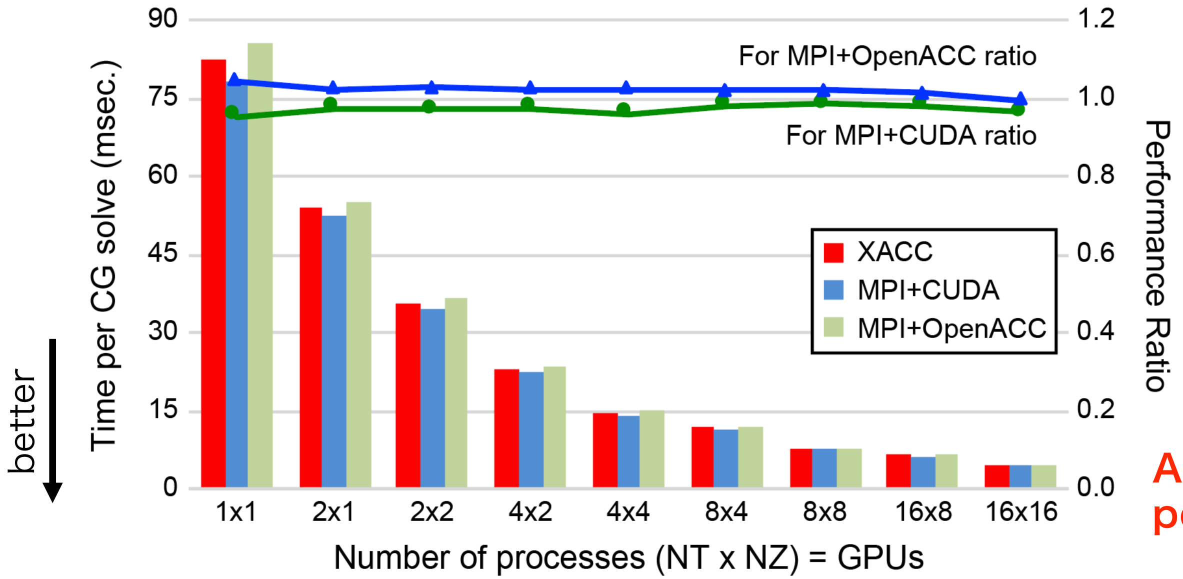
- 64 nodes, 364TF total
- Each node has:
  - Intel IvyBridge x 2
  - NVIDIA Tesla K20X x 4
  - PEACH2 board
  - IB QDR x 2



- Shutdown on Oct. 2018

# Performance of QCD in XACC

Data size is 32x32x32x32 (T x Z x Y x X axes) with strong scaling. Each process deals with a single GPU, 4 processes run on a single compute node

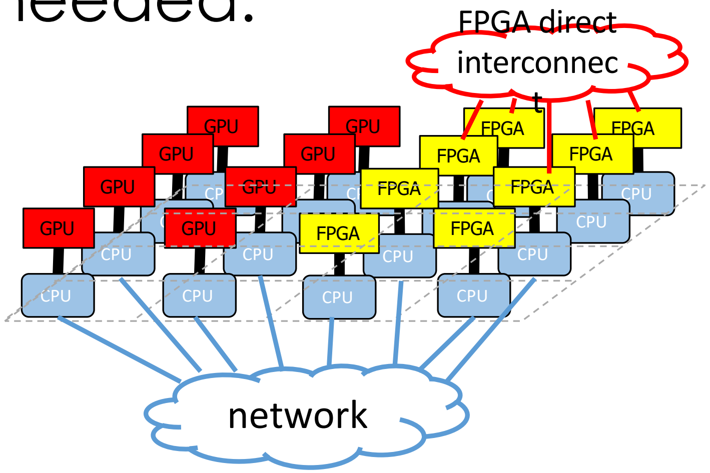


Almost the same performance !!

The performance of XACC is 100 - 104% of that of MPI+OpenACC, and 95 - 99% of that of MPI+CUDA

# Prospects

- FPGA-accelerated clusters have emerged as HPC platforms.
  - A collection of nodes has FPGA as an accelerator
  - FPGAs may have their own direct interconnect.
- More complicated programming is needed.
  - GPU & FPGA
  - Host CPU-GPU & FPGA
  - Multi-GPU & FPGA



# Cygnus: Multi-Hybrid Accelerated Cluster

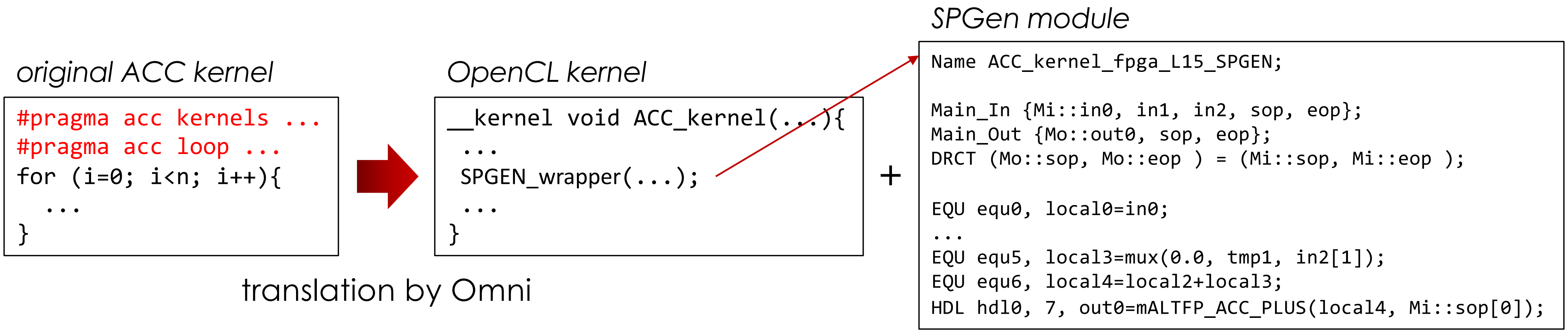
- New supercomputer @ CCS, U. Tsukuba
  - Operation started in April 2019
  - 2x Intel Xeon CPUs, 4x NVIDIA V100 GPUs, 2x Intel Stratix10 FPGAs
  - Deneb: 46 nodes
    - CPU + GPU
  - Albireo: 32 nodes
    - CPU + GPU + FPGA
    - 2D torus network for FPGAs
      - 100Gbps per link

Cygnus cluster @ CCS, U. Tsukuba

Intel Stratix10 FPGA + NVIDIA V100 GPU

# Ongoing Projects (1)

- High-level programming for FPGA
  - Omni translates an OpenACC kernel to an OpenCL kernel + an *SPGen* module.
    - SPGen is a framework for generating stream processors on FPGA, developed by Sano, RIKEN R-CCS.
  - OpenCL kernel for handling memory accesses and invoking the SPGen module
  - SPGen module for pipelined computation



*original ACC kernel*

```
#pragma acc kernels ...
#pragma acc loop ...
for (i=0; i<n; i++){
  ...
}
```

*OpenCL kernel*

```
__kernel void ACC_kernel(...){
 ...
 SPGEN_wrapper(...);
 ...
}
```

+

*SPGen module*

```
Name ACC_kernel_fpga_L15_SPGEN;

Main_In {Mi::in0, in1, in2, sop, eop};
Main_Out {Mo::out0, sop, eop};
DRCT (Mo::sop, Mo::eop ) = (Mi::sop, Mi::eop );

EQU equ0, local0=in0;
...
EQU equ5, local3=mux(0.0, tmp1, in2[1]);
EQU equ6, local4=local2+local3;
HDL hdl0, 7, out0=mALTFP_ACC_PLUS(local4, Mi::sop[0]);
```

translation by Omni

# Ongoing Projects (2)

- Unified programming model for GPU-FPGA heterogeneous systems, such as Cygnus
  - (described by Kobayashi-san)
  - OpenACC constructs are processed for GPU, or converted into OpenCL for FPGA, by Omni.

- Both of the two projects target FPGA and single-node execution.

- We plan to extend them to multi-node on the basis of the technologies derived from XACC.

# Summary

- A new programming language XcalableACC for accelerated clusters is proposed.
- XACC = XMP + OpenACC + XACC extensions
- The evaluation showed high performance and productivity of XACC.
- We are planning to apply XACC for FPGA clusters.
- (Offloading to FPGA in task-based programming)