XcalableACC: Extension of XcalableMP PGAS Language using OpenACC for Accelerator Clusters

Masahiro Nakao,[†] Hitoshi Murai,[†] Takenori Shimosaka[†]
Akihiro Tabuchi,[‡] Toshihiro Hanawa,[§] Yuetsu Kodama[‡*]
Taisuke Boku,[‡*] Mitsuhisa Sato[†‡*]

† RIKEN Advanced Institute for Computational Science, Japan
‡ Graduate School of Systems and Information Engineering, University of Tsukuba
§ Information Technology Center, The University of Tokyo
* Center for Computational Sciences, University of Tsukuba

1st WACCPD, November 17th, 2014

Background

Accelerator cluster

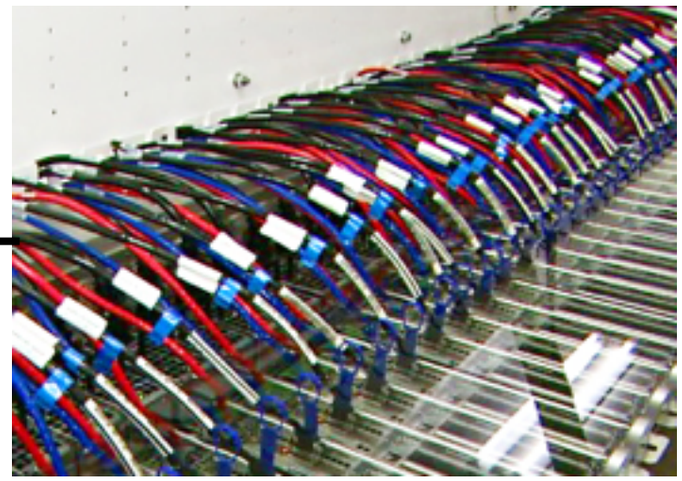
- High performance
- High efficiency power consumption

The Green500 List

Listed below are the June 2014 The Green500's energy-efficient supercomputers ranked from 1 to 10.

(June, 2014)

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	4,389.82	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR, NVIDIA K20x	34.58
2	3,631.70	Cambridge University	Wilkes - Dell T620 Cluster, Intel Xeon E5-2630v2 6C 2.600GHz, Infiniband FDR, NVIDIA K20	52.62
3	3,517.84	Center for Computational Sciences, University of Tsukuba	HA-PACS TCA - Cray 3623G4-SM Cluster, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband QDR, NVIDIA K20x	78.77
4	3,459.46	SURFsara	Cartesius Accelerator Island - Bullx B515 cluster, Intel Xeon E5-2450v2 8C 2.5GHz, InfiniBand 4× FDR, Nvidia K40m	44.40
5	3,185.91	Swiss National Supercomputing Centre (CSCS)	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Level 3 measurement data available	1,753.66
6	3,131.06	ROMEO HPC Center - Champagne-Ardenne	romeo - Bull R421-E3 Cluster, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR, NVIDIA K20x	81.41
7	3,019.72	CSIRO	CSIRO GPU Cluster - Nitro G16 3GPU, Xeon E5-2650 8C 2GHz, Infiniband FDR, Nvidia K20m	86.20
8	2,951.95	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.5 - Cluster Platform SL390s G7, Xeon X5670 6C 2.93GHz, Infiniband QDR, NVIDIA K20x	927.86
9	2,813.14	Exploration & Production - Eni S.p.A.	HPC2 - iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, NVIDIA K20x	1,067.49
10	2,678.41	Financial Institution	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	54.60



Tsubame KFC, Tokyo Tech
http://www.titech.ac.jp

HA-PACS, Univ. of Tsukuba
(Our Cluster)
http://www.ccs.tsukuba.ac.jp

Problems (1/2)

1. Complex programming on accelerator clusters

   OpenACC and **Message Passing Interface (MPI)**

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
double a[100];
#pragma acc data copy(a)
{
#pragma acc parallel loop
  for(i=0;i<100;i++)
    a[i] = i;
}
if(rank == 0)
  MPI_Send(a, 100, MPI_DOUBLE, 1, tag,
           MPI_COMM_WORLD);
else
  MPI_Recv(a, 100, MPI_DOUBLE, 0, tag,
           MPI_COMM_WORLD, &status);
```

OpenACC has a good productivity.
But MPI is often difficult.

Must control local data by using primitive MPI functions

(While search arguments …)

Solutions (1/2)

1. Usage of XcalableMP (XMP)

Directive-based language extensions of Fortran and C language for clusters

MPI

```
int a[MAX];
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
dx=MAX/size;
llimit=rank*dx;
if(rank !=(size-1)) ulimit=llimit*dx;
else ulimit = MAX;

temp_res=0;
for(i=llimit;i<ulimit;i++){
  a[i]=func(i);
  tmp_res += a[i];
}
MPI_Allreduce(&temp_res, &res, 1, ..);
```

XMP

```
int a[MAX];
#pragma xmp nodes p(*)
#pragma xmp template t(1:MAX)
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i] with t(i)

res=0;
#pragma xmp loop t(i) reduction(+:res)
for(i=0;i<MAX;i++){
  a[i]=func(i);
  res += a[i];
}
```

XcalableACC (XACC): Extension of XMP using OpenACC for accelerator clusters

Problems (2/2)

2. Large communication latency among accelerators on different nodes

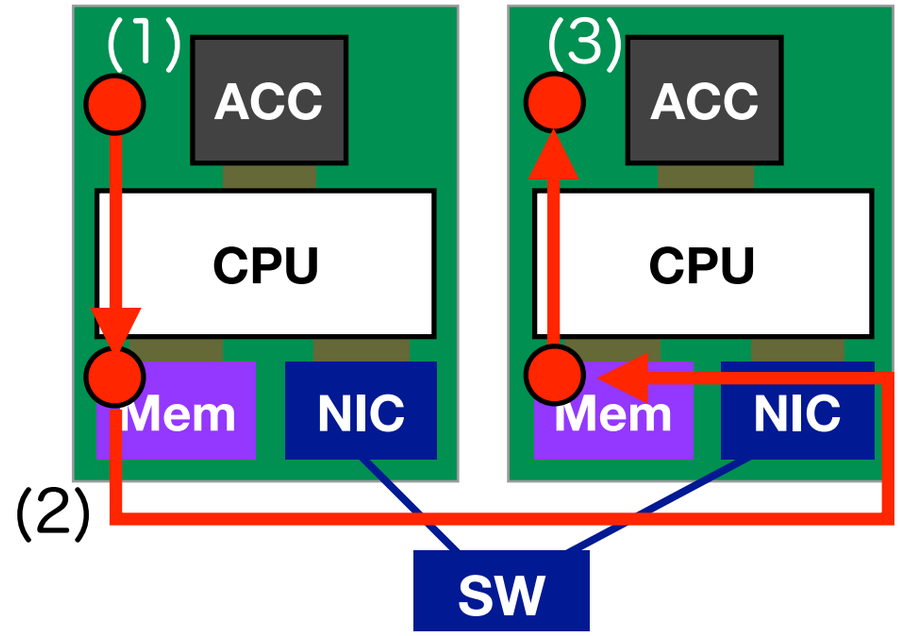
ACC. = Accelerator **Mem.** = Host memory

(1) Copy data from **ACC.** to **Mem.** by OpenACC

(2) Send data from **Mem.** to **Mem.** by MPI

(3) Copy data from **Mem.** to **ACC.** by OpenACC

Twice Mem. copies occur !!

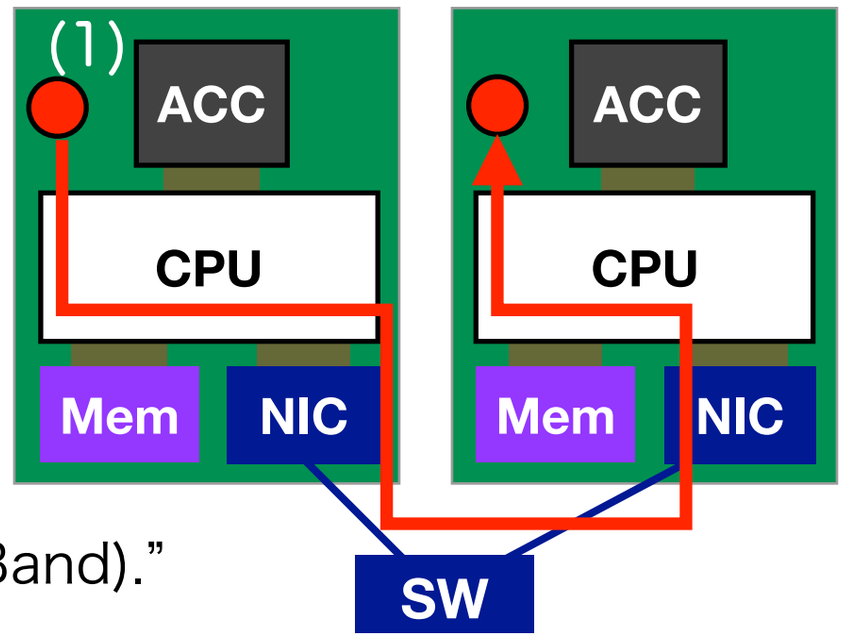


To overcome

GPUDirect does NOT require **Mem.** copy.

- Faster than the above method
- But accelerator applications want smaller latency !!

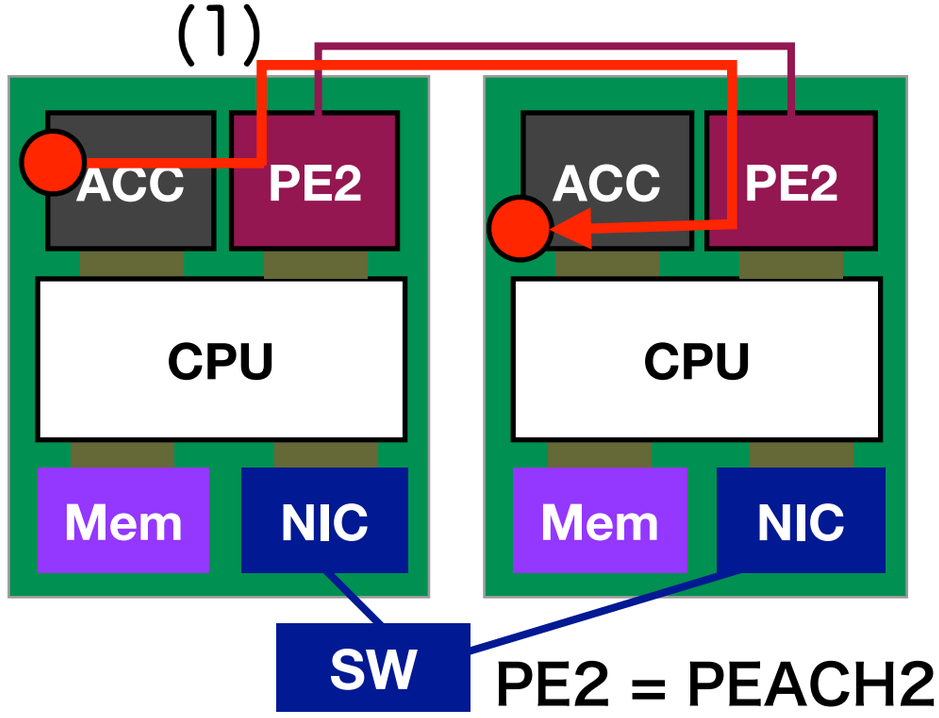
  GPUDirect still requires time for “MPI software stack”, and “Protocol conversion (e.g. PCIe to InfiniBand).”



Solutions (2/2)

2. Usage of Tightly Coupled Accelerators (TCA) [1]
   - Communication architecture based on PCIe technology
   - Developed by HA-PACS Project in Univ, of Tsukuba, Japan
   - Nodes are connected using PCIe external cable through **PEACH2,** which is a TCA interface Board

   - Direct, low latency data transfers among accelerator memories
     - No host memory copies
     - No MPI software stack
     - No protocol conversions



[1] Toshihiro Hanawa et al. "Tightly Coupled Accelerators Architecture for Minimizing Communication Latency among Accelerators," in IPDPSW '13 Proceedings of the 2013

Solutions (2/2)

TCA provides Hardware-level APIs to deal with itself.

It is difficult !!

TCA requires a programming interface.

We also develop XACC as a TCA programming interface.

Objectives

(1) Provide a design for a directive-based language for accelerator clusters and evaluates its effectiveness

(2) Discuss our experiences implementing directives for data transfer among accelerators

Outline

- Overview of XMP and XACC programming models
- Implementation of XACC compiler
- Evaluation of productivity and performance of XACC programming model

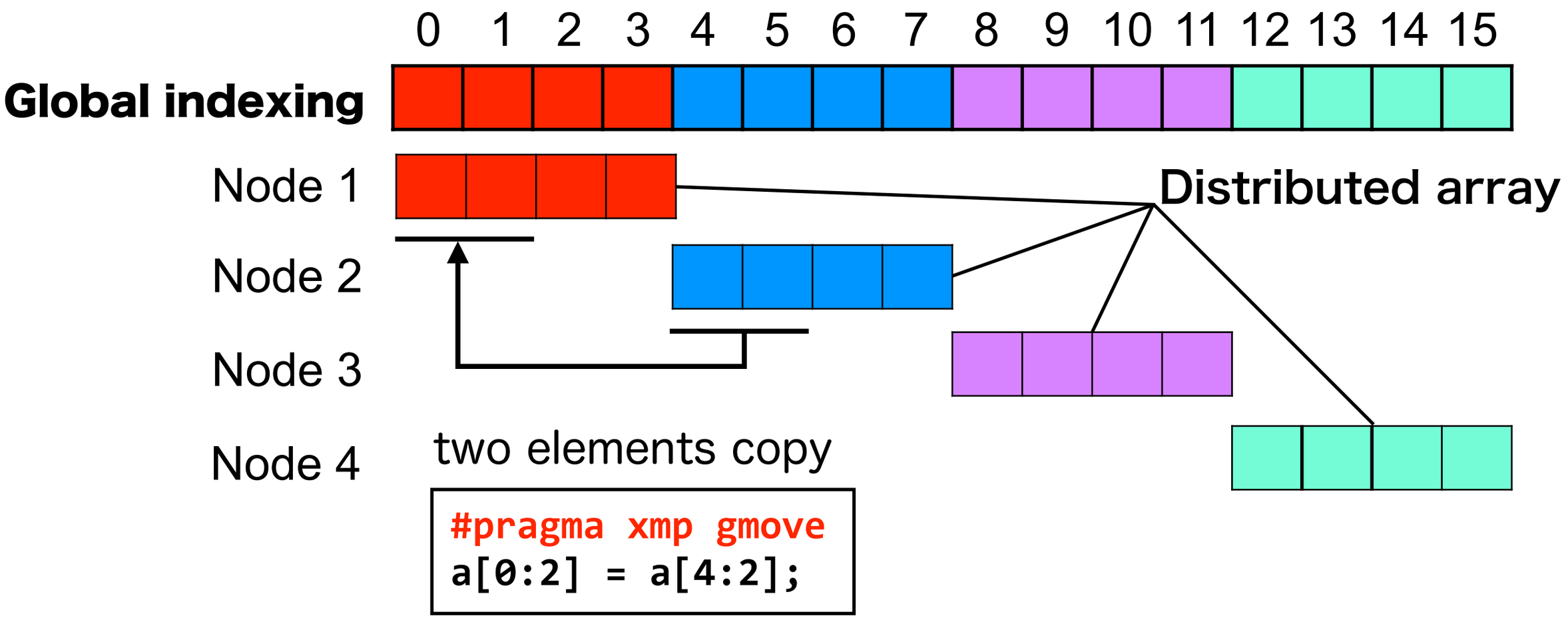
Outline

- Overview of XMP and XACC programming models
- Implementation of XACC compiler
- Evaluation of productivity and performance of XACC programming model

XMP Global-view memory model

PGAS (Partitioned Global Address Space) is a memory region can be accessed from other processes

In XMP, to improve productivity, programmers can use **“Global indexing.”**

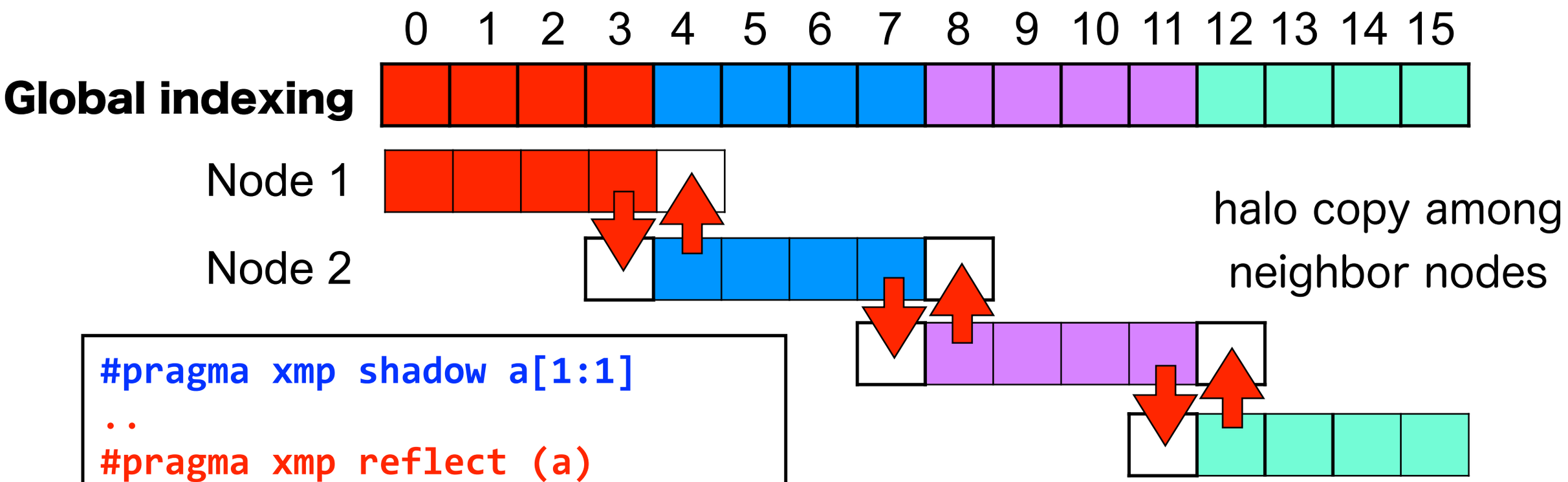


Maintain a sequential code image, as OpenACC and OpenMP

XMP Global-view memory model

PGAS (Partitioned Global Address Space) is a memory region can be accessed from other processes

In XMP, to improve productivity, programmers can use **“Global indexing.”**




```
#pragma xmp shadow a[1:1]
..
#pragma xmp reflect (a)
#pragma xmp loop on t(i)
for(..){
  .. = array[i-1] + array[i+1];
```

For Stencil Application

Maintain a sequential code image, as OpenACC and OpenMP

Reason for extending XMP to use OpenACC

Basically, programmers can use OpenACC directives in XMP source code.

Problems:

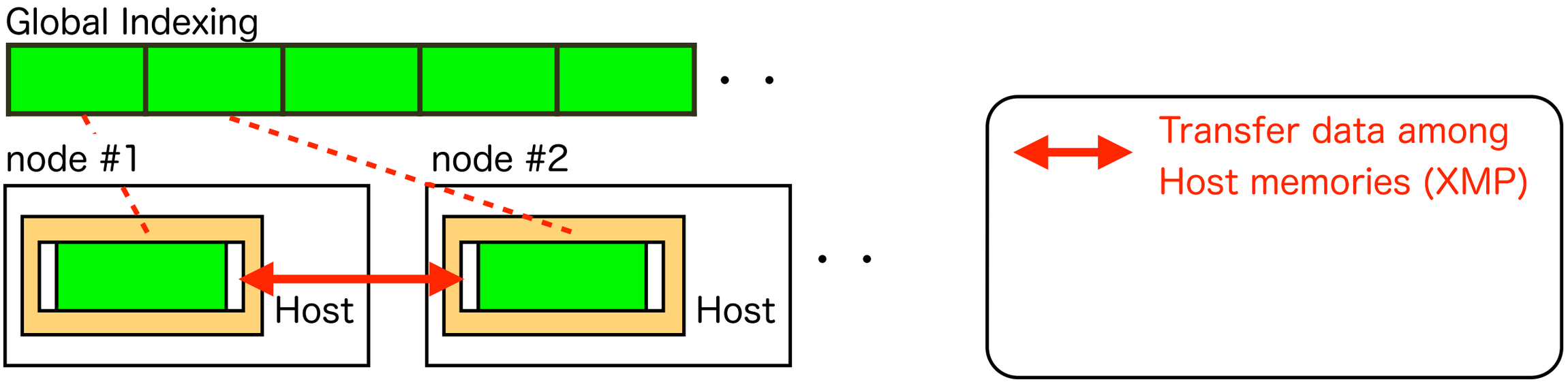
1. **XMP** does NOT support XMP distributed array in OpenACC directives
   Local OpenACC directive, data movement between CPU and accelerator, cannot be applied to XMP distributed array
2. **XMP** does NOT express transferring data among accelerators directly
   Need to express data movement **ACC.** -> **MEM.** -> **MEM. -> ACC.**

Solutions: **Development of XACC for accelerator clusters**

1. **XACC** supports XMP distributed array in OpenACC directives
   Programmer can use both XMP and OpenACC directives **seamlessly**
2. **XACC** can express transferring data among accelerators directly
   To implement it , **TCA** is used as a data transfer architecture

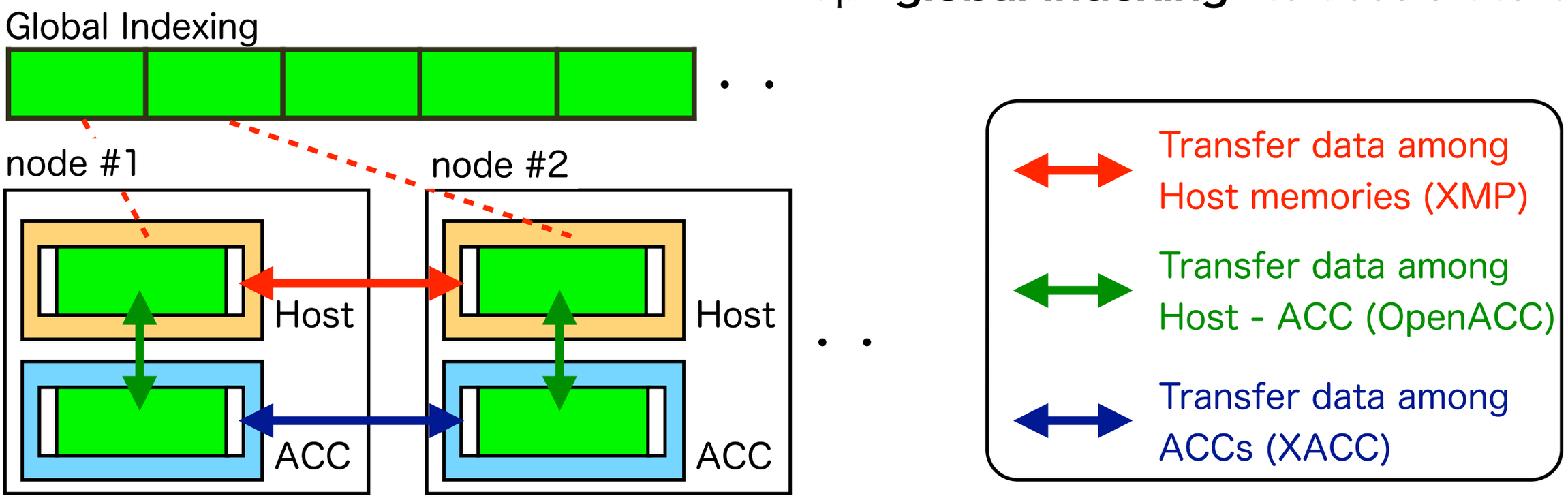
Difference XMP and XACC memory models

- **XMP** memory model



- **XACC** memory model

Map **“global Indexing”** to accelerators



XACC code example

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];
#pragma xmp nodes p(x, y)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align [j][i] with t(i,j) :: u, uu
#pragma xmp shadow uu[1:1][1:1]
…
#pragma acc data copy(u) copyin(uu)
{
  for(k=0; k<MAX_ITER; k++){
#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

#pragma xmp reflect (uu) acc

#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
} // end data
```

Laplace's equation

Data Distribution and Halo

Transfer XMP distributed arrays to accelerator

OpenACC directive parallelizes the loop statement parallelized by XMP directive

Exchange halo region of uu[][]

When "acc" clause is specified in XMP communication directive, data on accelerator is transferred.

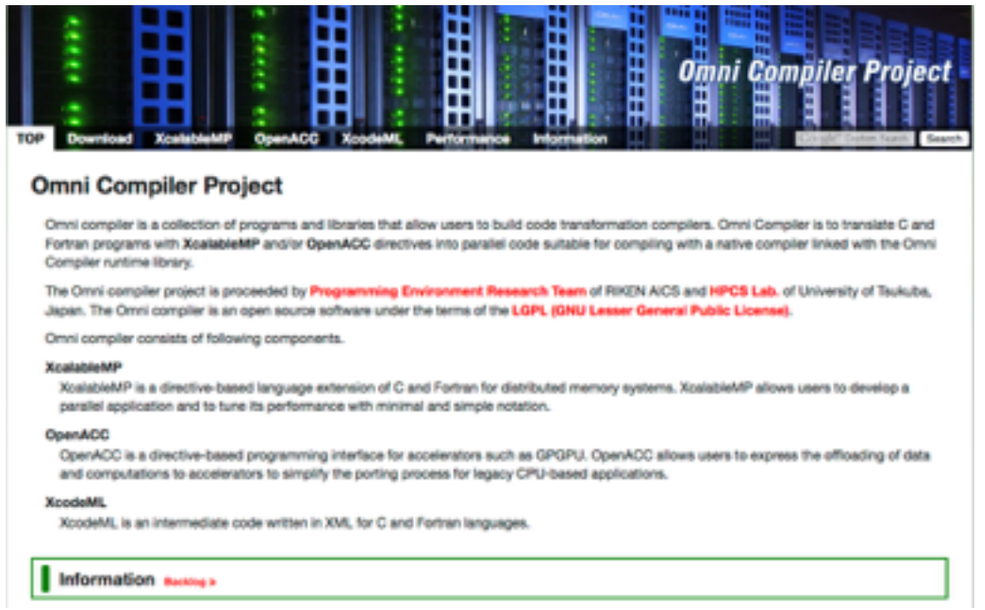
Outline

- Overview of XMP and XACC programming models
- Implementation of XACC compiler
- Evaluation of productivity and performance of XACC programming model

Implementation of XACC compiler

Extension of the Omni XMP Compiler

- Free software *http://omni-compiler.org*
- source-to-source compiler
- PGI, Cray, Omni OpenACC compilers are available as a backend compiler



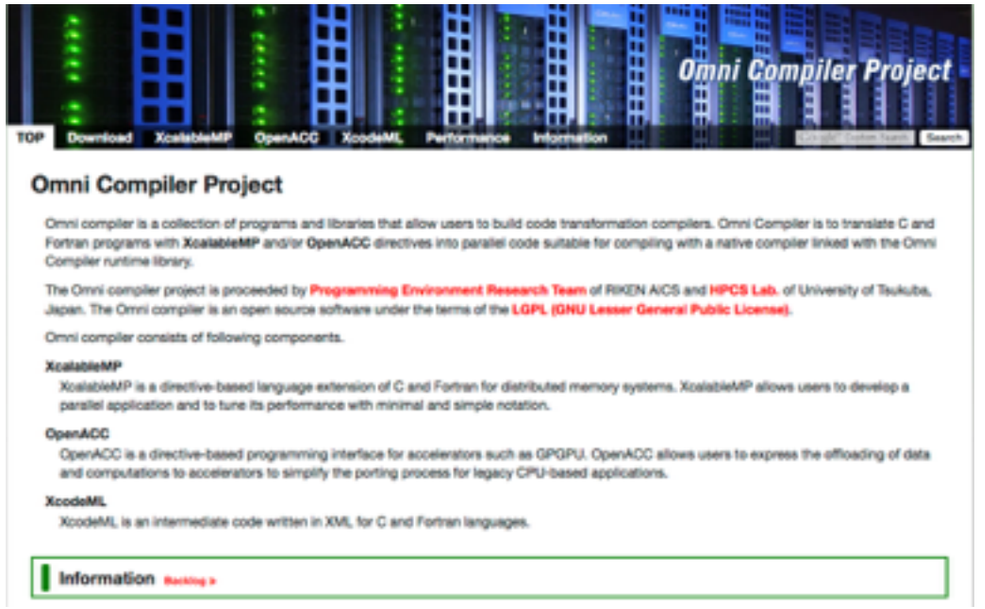
Development points:

1. Specify an XMP distributed array in OpenACC directives
2. Mix XMP loop directive and OpenACC loop directive
3. Transfer data among accelerators

Implementation of XACC compiler

Extension of the Omni XMP Compiler

- Free software *http://omni-compiler.org*
- source-to-source compiler
- PGI, Cray, Omni OpenACC compilers are available as a backend compiler



Development points:

1. Specify an XMP distributed array in OpenACC directives
2. Mix XMP loop directive and OpenACC loop directive
3. Transfer data among accelerators

Please refer to our paper

Transfer data among accelerators

How to transfer data among accelerators in the XACC compiler

1. DMA of PEACH2 : sophisticated DMA functions
   - **Internal memory mode**
   - **Host memory mode**
2. GPUDirect
3. CUDA + MPI : **ACC.** -> **MEM.** -> **MEM. -> ACC.**

The maximum number of registrable regions is 1,024

Problem:

In preliminary evaluation, depending on data size and hardware, **“GPUDirect”** may be faster than **“DMA of PEACH2.”**

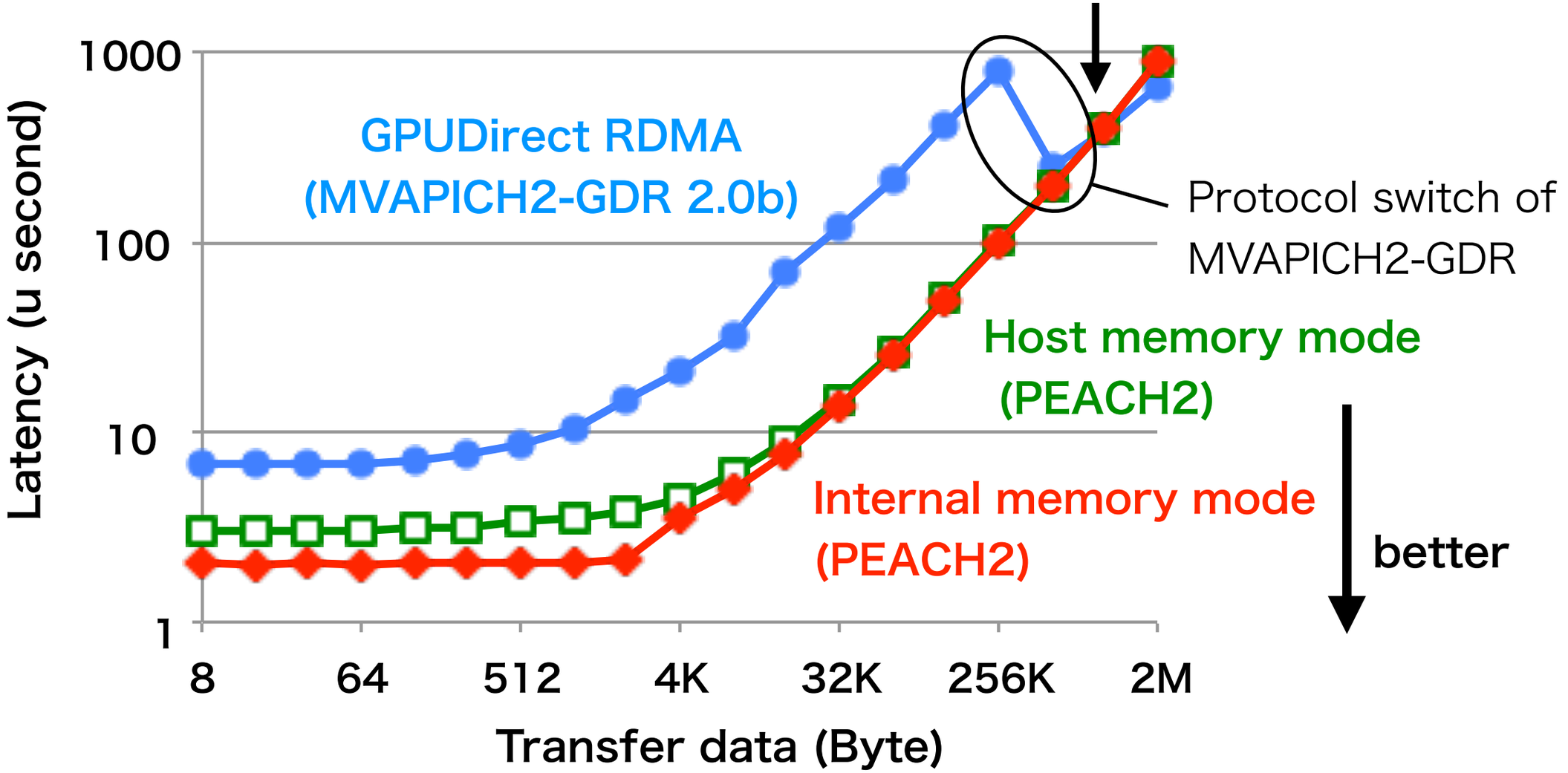
Solution:

Implement of flow of switching communication method

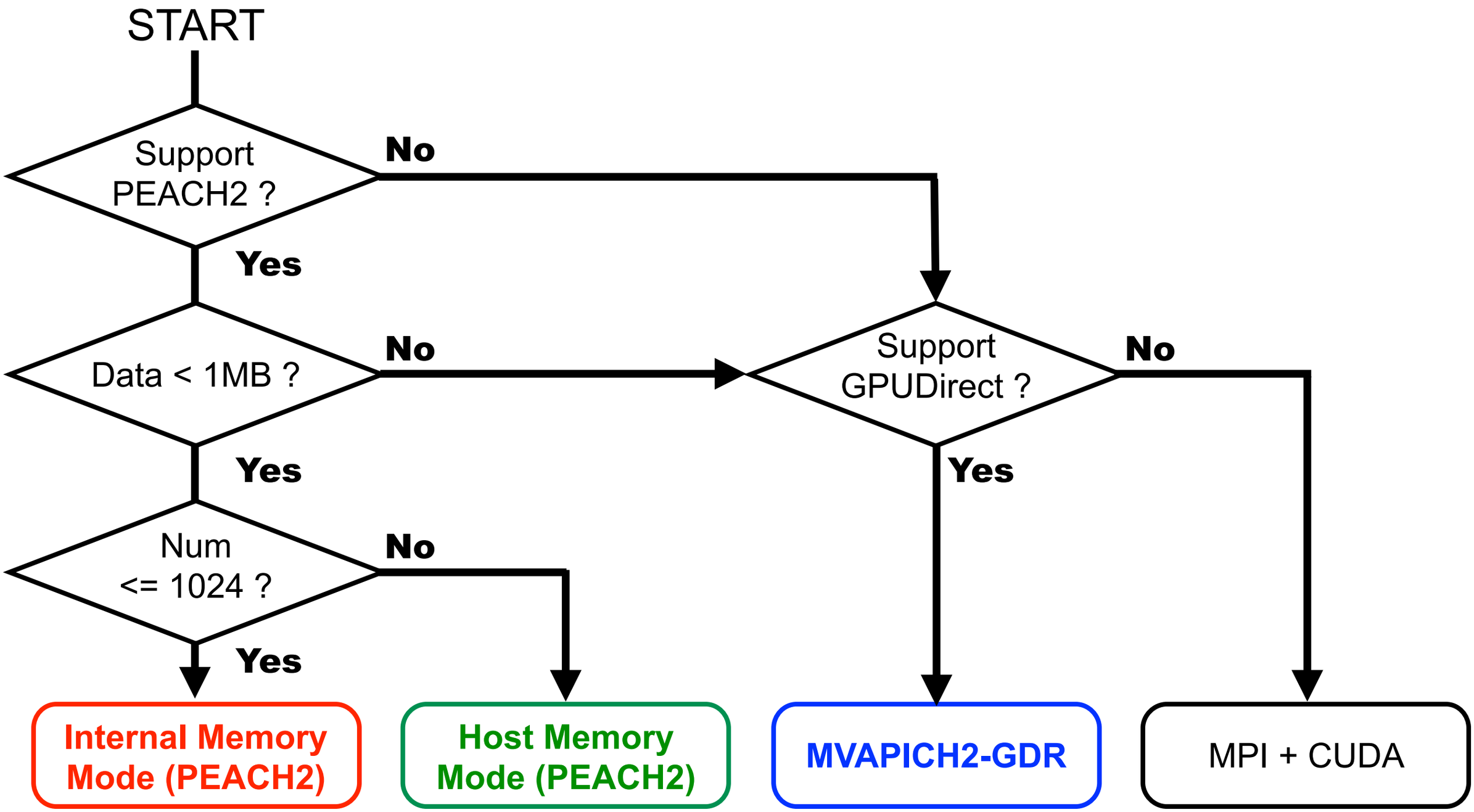
Preliminary Evaluation

- GPUDirect uses **InfiniBand 4xQDR x 2rails**, the bandwidth is **8GB/s**
- DMA of PEACH2 uses **PCIe Gen.2 x 8links**, the bandwidth is **4GB/s**

Transferring more than 1M byte, the performance of GPUDirect is better than that of PEACH2 in PCIe Gen2



Flow of communication method



Outline

- Overview of XMP and XACC programming models
- Implementation of XACC compiler
- Evaluation of productivity and performance of XACC programming model

Target Benchmark

- Himeno Benchmark
  - Stencil application of Incompressible fluid analysis code
  - Solving the Poisson's equation
  - ***http://accc.riken.jp/2444.htm*** : Sequential/MPI versions

```
float p[MIMAX][MJMAX][MKMAX];
// Define distributed array and halo

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

Transfer XMP distributed array to accelerator

Exchange halo region

Parallelize loop statement

Productivity

For comparison purposes, we also implemented **HIMENO Benchmark using OpenACC based on MPI HIMENO Benchmark (OpenACC+MPI HIMENO)**

- To parallelize loop statements, we use OpenACC **data** and **loop** directives
- To use GPUDirect, we insert OpenACC **host_data** directive before MPI functions

Source Lines of Code

	Breakdown			Total
	XMP	OpenACC	Others	
OpenACC + MPI	-	15	473	488
XACC	28	9	176	213

50% reduction

- **OpenACC+MPI HIMENO Benchmark** requires numerous lines to calculate indexes for a loop statement, and to transfer the halo region
- In XACC, we only add directives to the sequential HIMENO Benchmark

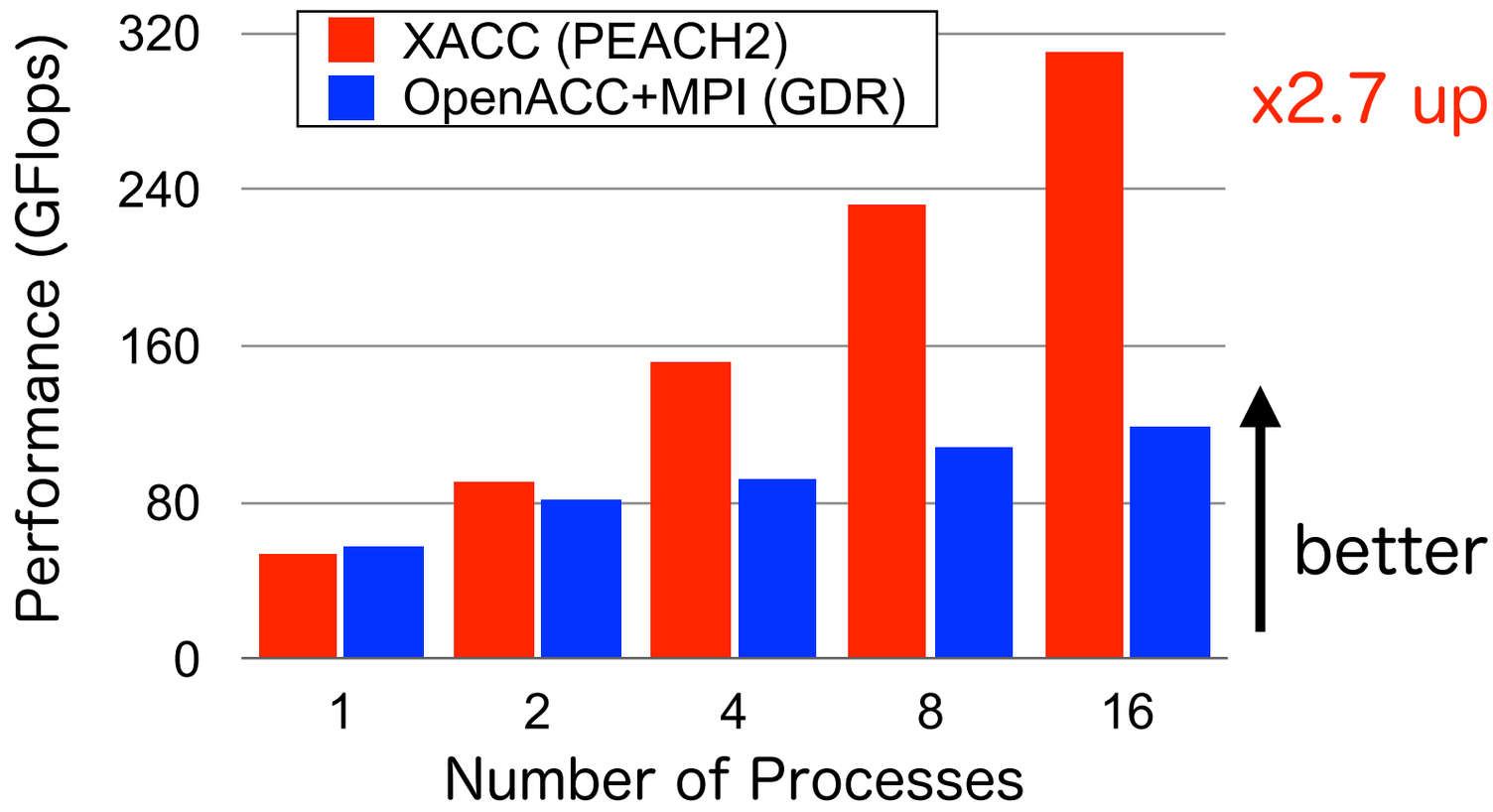
XACC HIMENO Benchmark maintains a sequential source code image

Performance

HA-PACS, Univ. of Tsukuba

NVIDIA K20X
InfiniBand 4xQDR x 2rails
PCIe Gen2 x8 for PEACH2
MVAPICH-GDR2.0b
gcc-4.7, CUDA6.0, Omni OpenACC Compiler 0.9b

Array size 128x128x256 : Strong scaling, 1GPU/procs



- In XACC, all communication uses the **Internal memory mode of PEACH2**
- **Internal memory mode of PEACH2** vs. **GPUDirect RDMA over InfiniBand**

Note that performance of OpenACC using PEACH2 is the almost the same that of XACC.

Conclusion

- New programming model for accelerator clusters, called **XACC**
  - Programmer can use both XMP and OpenACC directives seamlessly
  - Programmer can express transferring data among accelerators directly
- Two Objectives

  (1) Provide a design for a directive-based language for accelerator clusters and evaluates its effectiveness

  → Through implementing the HIMENO Benchmark,
  XACC has a good productivity and good performance using TCA

  (2) Discuss our experiences implementing directives for data transfer among accelerators via TCA

  → Automatically select appropriate communication method

Future works

- Development of mini-applications and real world applications
- Usage of both TCA and normal interconnect (e.g. InfiniBand)
- Extension of OpenACC to use multiple accelerators
  - Need to specify a device number by **acc_set_device_num()** before each statement to use multiple accelerators
  - To use multiple accelerators seamlessly similar to XMP directives

```
float a[N], b[N];
#pragma acc device d(*)
#pragma acc declare device_resident(a) layout([block]) ¥
                                    shadow([1:1]) on_device(d)
..
#pragma acc reflect (a)

#pragma acc parallel loop layout(a[i]) on_device(d)
for (int j = 1; j < 99; j++)
  b[i] = a[i-1] + a[i+1];
```

Also show SC poster