An Extension of XcalableMP PGAS Lanaguage for Multi-node GPU Clusters

Jinpil Lee, Minh Tuan Tran, Tetsuya Odajima,
Taisuke Boku and Mitsuhisa Sato

University of Tsukuba

Presentation Overview

- Introduction
  - needs of programming model for GPU clusters
- Programming model for GPU clusters
  - XcalableMP: PGAS language for clusters
  - **XcalableMP-ACC**: GPGPU extension of XcalableMP
- Implementation of XcalableMP-ACC compiler
  - benchmark: N-Body problem solver
- Conclusion
  - future work

Background

- Accele
  - GPU
  - man
- GPGP
  - gene
  - NVI
    - p
    - b
    - ta

TOP500 List - June 2011 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

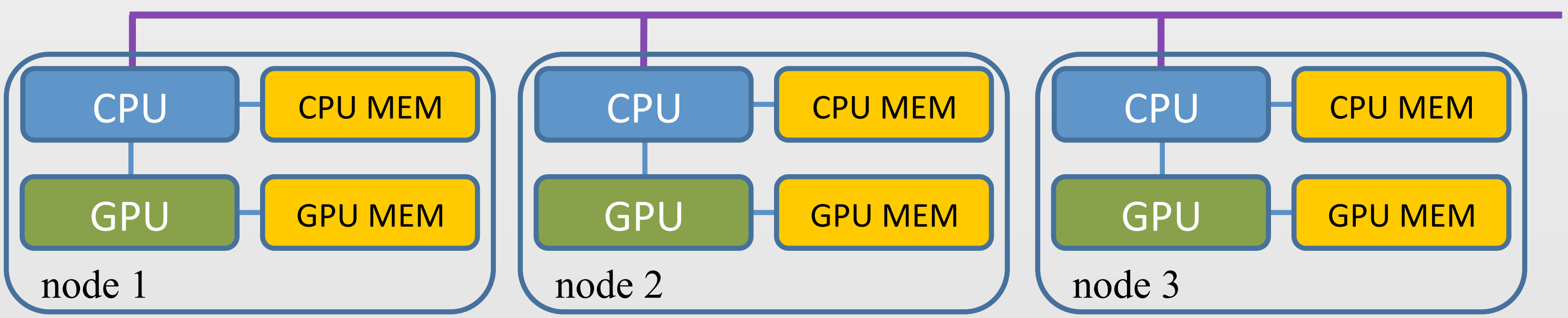
next

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	548352	8162.00	8773.63	9898.56
2	National Supercomputing Center in Tianjin China	Tianhe-1A - NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C / 2010 NUDT	186368	2566.00	4701.00	4040.00
3	DOE/SC/Oak Ridge National Laboratory United States	Jaguar - Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.60
4	National Supercomputing Centre in Shenzhen (NSCS) China	Nebulae - Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU / 2010 Dawning	120640	1271.00	2984.30	2580.00
5	GSIC Center, Tokyo Institute of Technology Japan	TSUBAME 2.0 - HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.61

- Clusters with accelerators
  - hot topic in High Performance Computing area
  - GPU clusters in TOP500

Objective

- Parallel programming with GPU clusters
  - hybrid parallel programming
    - inter-node parallelism + thread-level parallelism (on GPU)
    - data transfer (between nodes, CPU and GPU, and GPUs)
  - MPI + CUDA
- Problem: high programming cost
- XcalableMP-ACC
  - easy and highly productive programming model



XcalableMP (XMP)

- C & Fortran based language extension for clusters
- Spec proposed by XcalableMP Specification WG
- Execution model
  - SPMD（Single Program Multiple Data）
  - single thread per process
- Explicit parallelism
  - no virtual shared memory, automatic comm (RMA, sync)
- Directive-based programming model
  - OpenMP-like directives for distributed memory
  - incremental parallelization from the serial code

Sample Code of XMP

```
int A[YMAX][XMAX];
#pragma xmp nodes p(XPROCS, YPROCS)
#pragma xmp template t(0:XMAX-1, 0:YMAX-1)
#pragma xmp distribute t(BLOCK, BLOCK) onto p
#pragma xmp align A[i][j] with t(j, i)

int main(void) {
  int sum = 0;
#pragma xmp loop (i, j) on t(j, i)
  for (int i = 0; i < YMAX; i++) {
    for (int j = 0; j < XMAX; j++) {
      A[i][j] = func(i, j);
      sum += A[i][j];
    }
  }
#pragma xmp reduction(+:sum)
}
```

data distribution

work mapping

inter-node communication

Data Parallelization in XMP

- Data distribution and work sharing using *template*

0 100

double A[100];

#pragma xmp nodes p(4) — declare node set

#pragma xmp template t(0:99) — declare template

0 100

template t(0:99)

#pragma xmp distribute t(BLOCK) on p — distribute template : block distribution

0 25 50 75 100

p(1) **p(2)** **p(3)** **p(4)**

#pragma align A[i] with t(i) — distribute array : owner of t(i) has A[i]

0 25 50 75 100

A[] **p(1)** **p(2)** **p(3)** **p(4)**

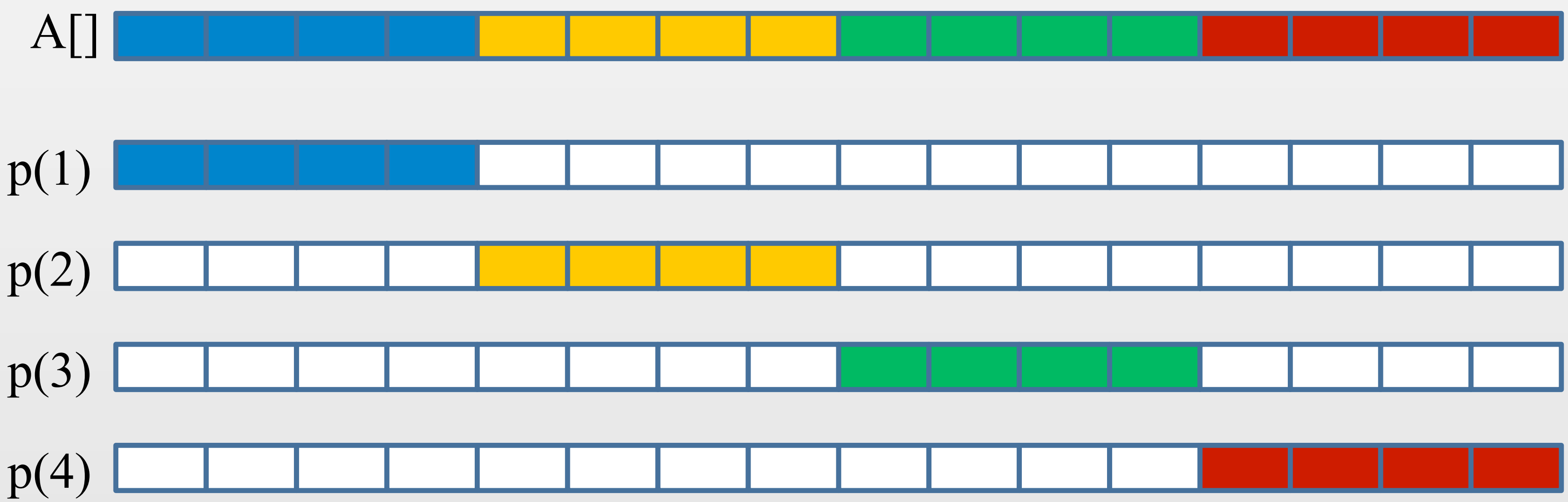
```
#pragma xmp loop on t(i)
 for (int i = 0; i < 100; i++)
  A[i] = func(. . .);
```

Inter-node Communication in XMP

- Broadcast, reduction, etc – typical inter-node comm
- Array synchronization with shadow region
  - shadow region: duplicated area for inter-node comm

#pragma xmp align A[i] with t(i) (distribute array)

#pragma xmp shadow A[*] (declare shadow region)



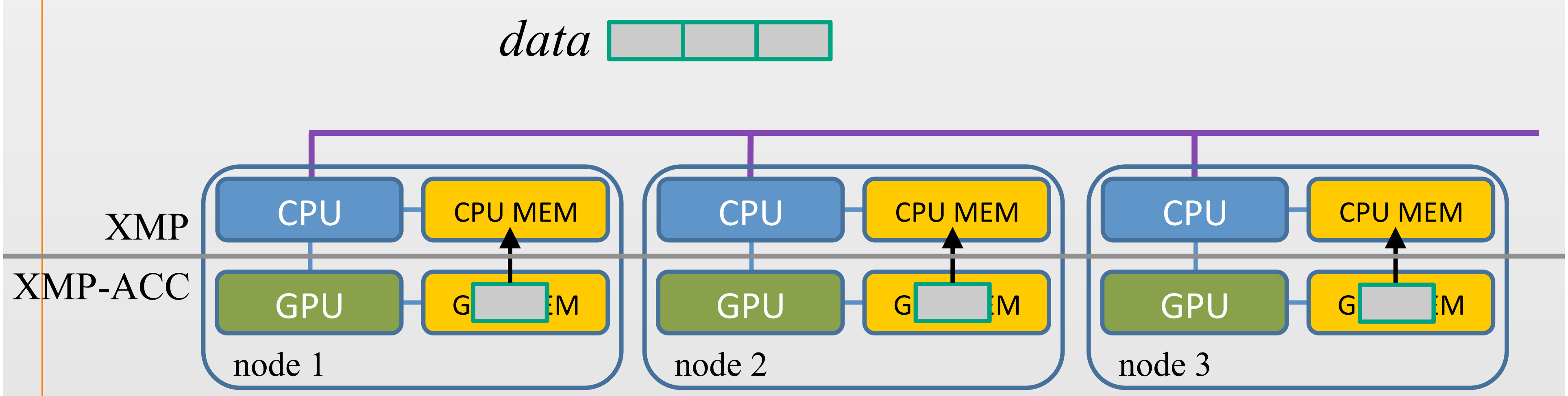
#pragma xmp reflect (A) (sync shadow region)

XcalableMP-ACC (XMP-ACC)

- GPGPU extension of XcalableMP
  - XcalableMP: PGAS programming language for clusters
  - new directives for GPGPU
  - describes data allocation, data transfer, task offloading
- Targetting mult-node GPUs
  - multi-node GPUs: GPUs within one node, GPU clusters
  - XMP supports for cluster computing + XMP-ACC
- Highly productive programming model
  - OpenMP-like directive-based programming model
  - small modification from the serial code

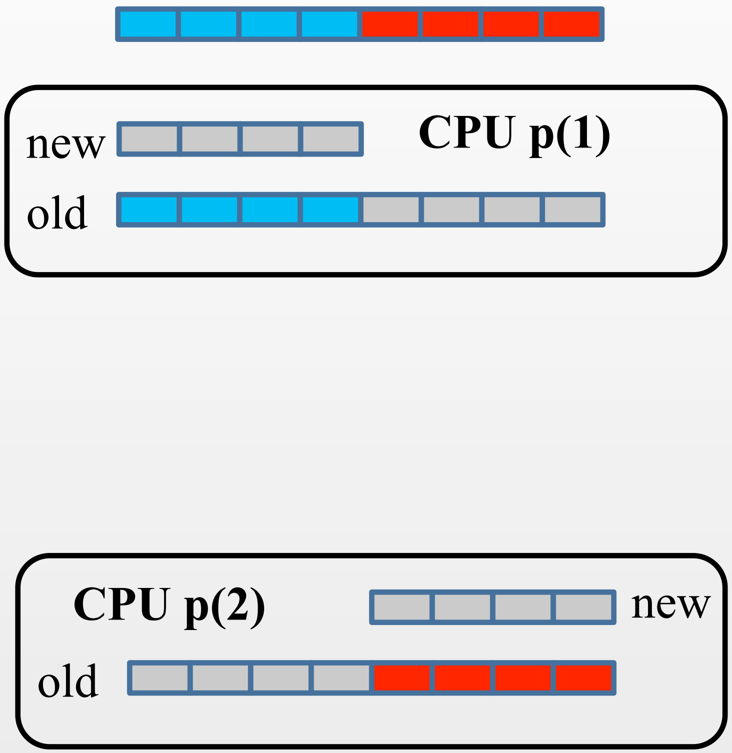
XMP-ACC Programming Model

- XMP directives + new directives for GPGPU
  - XMP directives: for cluster computing
    - data distribution, inter-node communication
  - XMP-ACC directives: for GPU computing
    - data replication on GPU memory
    - task offloading onto GPU
    - data transfer between CPU and GPU



XMP Code of N-Body

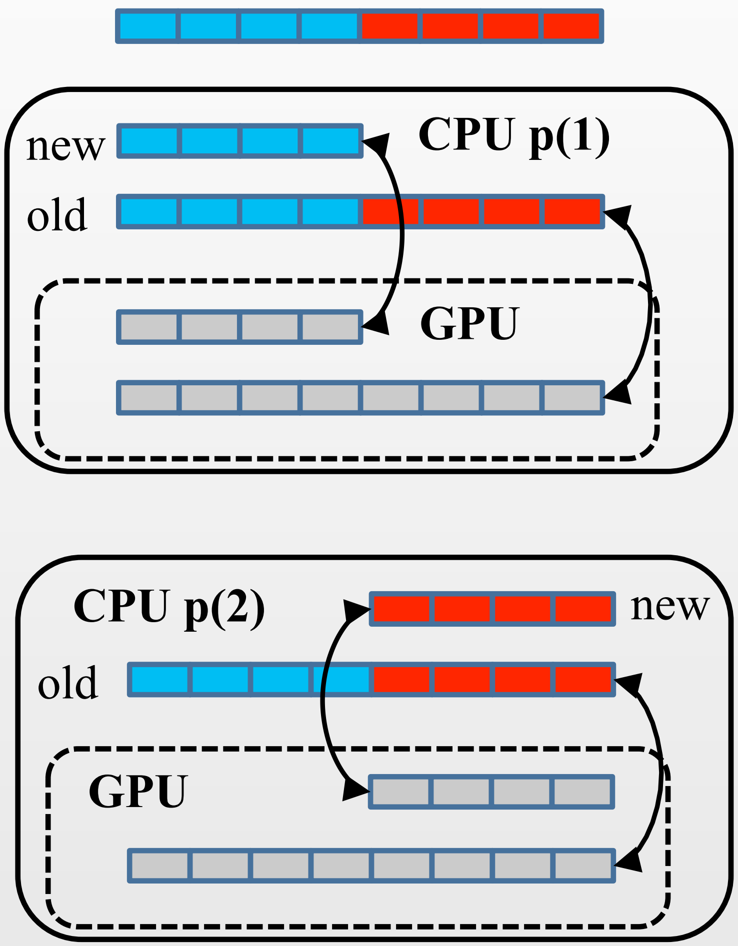
```
double old[N], new[N];
```




```
#pragma xmp nodes p(2)
#pragma xmp template t(0:N-1)
#pragma xmp distribute t(BLOCK) on p
#pragma xmp align [i] with t(i) :: old, new
#pragma xmp shadow [i] :: old, new
-------------------------------------------
#pragma xmp reflect (old)
#pragma xmp loop on t(i)
for (int i = 0; i < N; i++) {
  double f = 0;
  for (int j = 0; j < N; j++)
    f += calc_force(old[i], old[j]);
  new[i] = calc_position(f, old[i]);
}
```

Data Replication on GPU

double old[N], new[N];



- acc replicate directive

#pragma xmp acc replicate (*var-list*)
{
} → replication scope

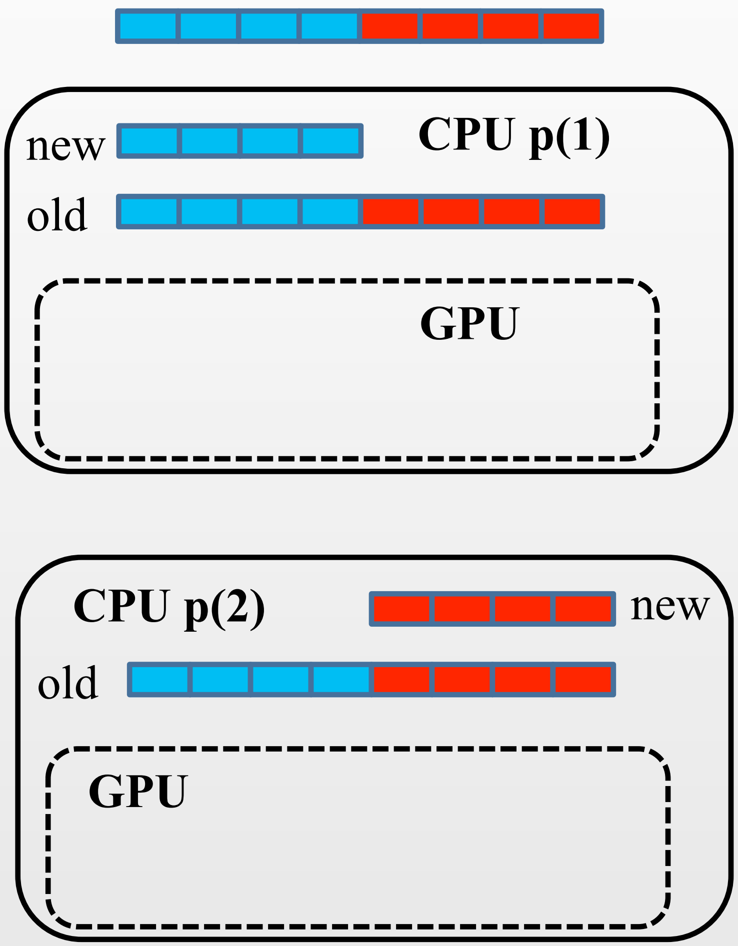
#pragma xmp reflect (old)
#pragma xmp acc replicate (old, new)

```
{
#pragma xmp loop on t(i)
  for (int i = 0; i < N; i++) {
    double f = 0;
    for (int j = 0; j < N; j++)
      f += calc_force(old[i], old[j]);
    new[i] = calc_position(f, old[i]);
  }
}
```

processed by CPU

Work Offloading onto GPU

double old[N], new[N];



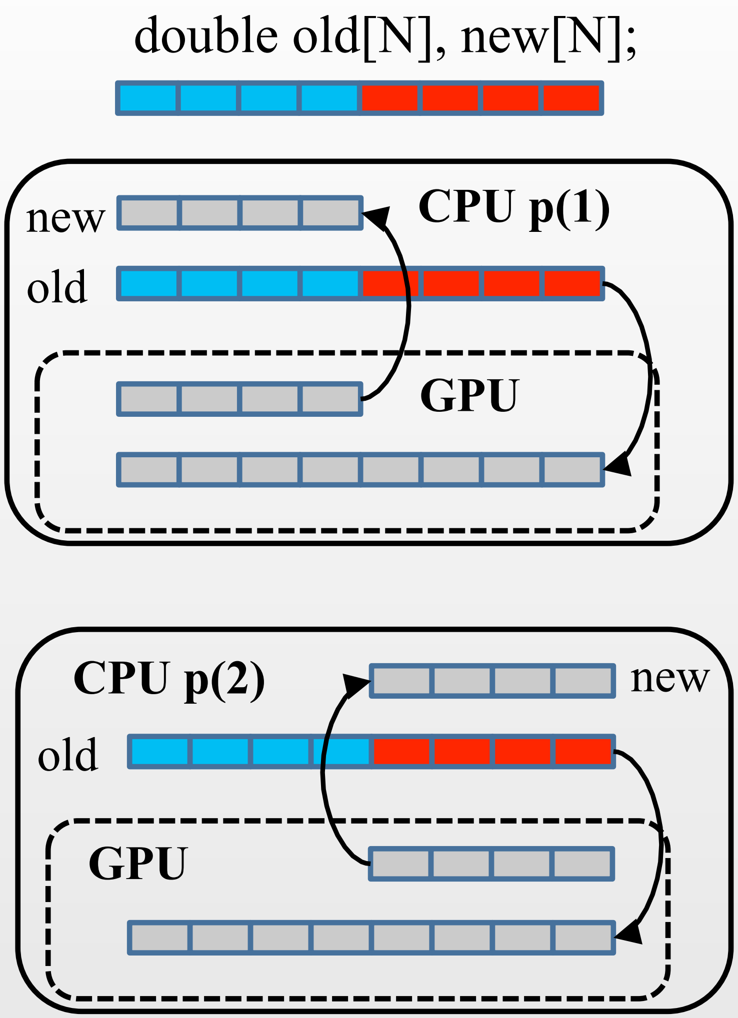
- extended loop directive

```
#pragma xmp loop on-ref acc {clause}
  for (int i = 0; i < N; i++) . . .
```

```
#pragma xmp reflect (old)
#pragma xmp acc replicate (old, new)
{
#pragma xmp loop on t(i) acc
  for (int i = 0; i < N; i++) {
    double f = 0;
    for (int j = 0; j < N; j++)
      f += calc_force(old[i], old[j]);
    new[i] = calc_position(f, old[i]);
  }
}
```

processed by CPU

Data Transfer between CPU and GPU



- acc replicate_sync directive

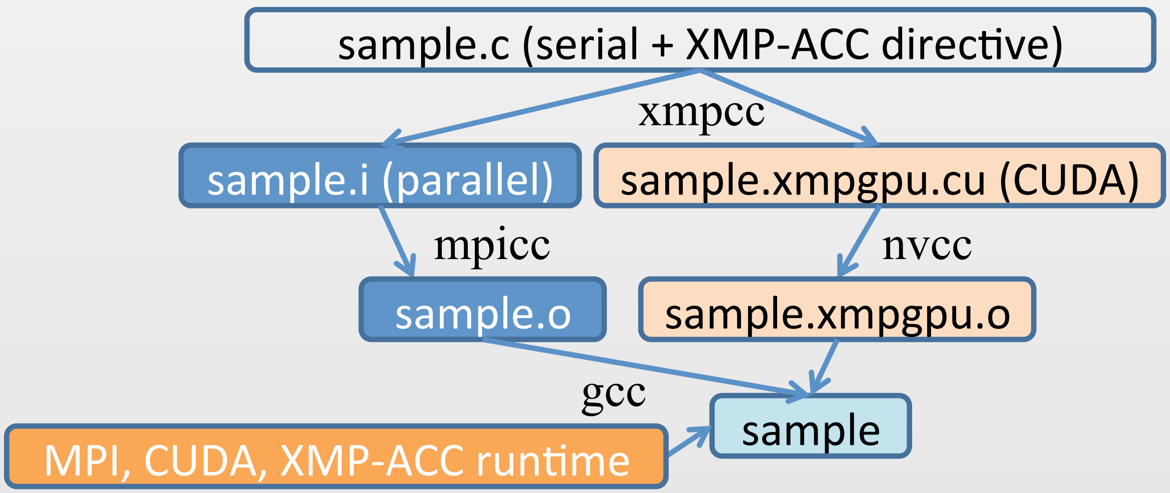
#pragma xmp acc replicate_sync *clause*

clause ::= in (*var-list*) | out (*var-list*)

```
#pragma xmp reflect (old)
#pragma xmp acc replicate (old, new)
{
#pragma xmp acc replicate_sync in (old)
#pragma xmp loop on t(i) acc
  for (int i = 0; i < N; i++) {
    double f = 0;
    for (int j = 0; j < N; j++)
      f += calc_force(old[i], old[j]);
    new[i] = calc_position(f, old[i]);
  }
#pragma xmp acc replicate_sync out (new)
}
```

Compiler Implementation

- XMP-ACC compiler
  - based on Omni XcalableMP Compiler
  - supports NVIDIA GPUs (CUDA architecture)
  - source-to-source translator
    - translates C code + XMP-ACC directives to MPI + CUDA code



Performance Evaluation

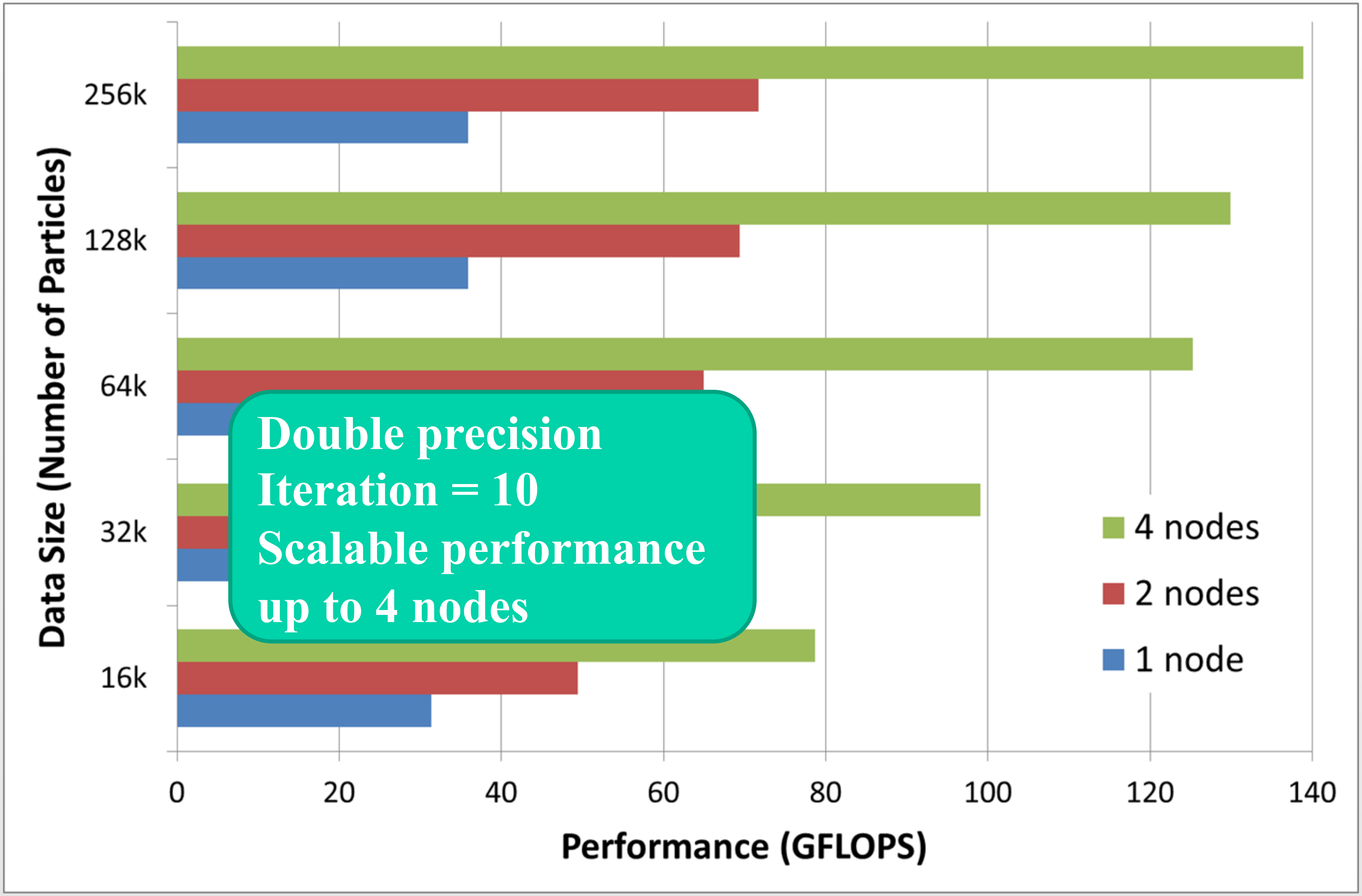
- Benchmark
  - N-Body problem solver
- Evaluation environment
  - Linux GPU cluster (1 ~ 4 nodes)
  - one XMP-ACC process to one node

GPU Cluster Node Configuration	
CPU	AMD Opteron Processor 6134 × 2 (8 cores × 2 sockets)
Memory	DDR3-1333 2GB × 2 (4GB)
GPU	NVIDIA Tesla C2050 (GDDR5 3GB)
Network	InfiniBand (4X QDR)
OS	Linux kernel 2.6.18 x86 64
MPI	OpenMPI 1.4.2
GPU Backend	NVIDIA CUDA Toolkit v3.2

XMP-ACC Code of N-Body

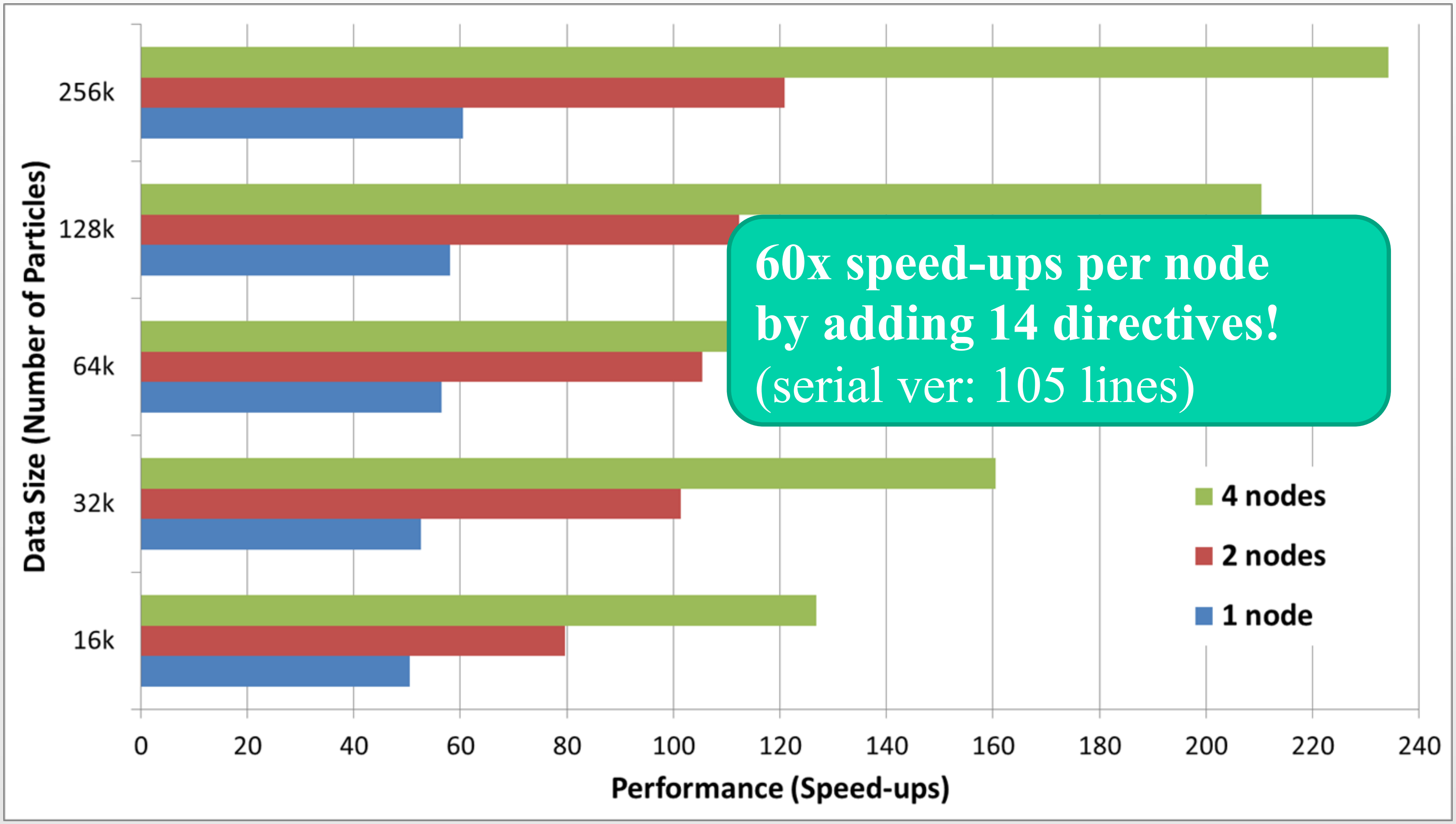
```
#pragma xmp align [i] with t(i) :: mass, velocity, position
#pragma xmp shadow [*] :: mass, velocity, position
#pragma xmp acc replicate (mass, velocity, position)
{
#pragma xmp acc replicate_sync in (mass, velocity)
  for (int t = 0; t < TIME_STEP; t++) {
#pragma xmp reflect (position)
#pragma xmp acc replicate_sync in (position)
#pragma xmp loop on t(i) acc
    for (int i = 0; i < N; i++)
      . . . // update velocity, position
#pragma xmp acc replicate_sync out (position)
  }
}
```

Performance of N-Body (GFLOPS)



Performance of N-Body (Speed-ups)

- compared with the serial version



Related Work

- OpenMPC, OMPCUDA
  - GPGPU extension of OpenMP
- PGI Accelerator Compilers, HMPP Workbench
  - provides directives for GPGPU
  - has the similar syntax to XMP-ACC
- Unified Parallel C
  - extends PGAS model for GPGPU
  - supports one-sided communication from/to GPU memory
  - requires significant modifications from the serial code

Conclusion

- XcalableMP-ACC
  - GPGPU extension of XcalableMP PGAS language
  - target: multi-node GPUs (GPU clusters)
  - highily productive programming model
    - OpenMP-like directives
    - describes task offloading, data replication, data transfer
- XcalableMP-ACC compiler implementation
  - achieved scalable performance in N-Body

Future Work

- Direct data communication between GPUs
  - shadow reflect among GPUs
  - lower programming cost & more efficient
- Asynchronous data transfer & kernel execution
- General programming model
  - to support other devices (AMD GPU, Intel MIC, etc ...)

```
#pragma xmp acc replicate_sync out (x)
#pragma xmp reduction (+:x)
#pragma xmp acc replicate sync in (x)

↓

#pragma xmp acc reduction (+:x)
```