HA-PACS: A new challenge for Accelerated Computational Sciences

Taisuke Boku

Center for Computational Sciences

University of Tsukuba

taisuke@cs.tsukuba.ac.jp

Three Supercomputers at CCS (as in 2011)

PACS-CS
2560 nodes
2560 cores
14.4TFLOPS
(July 2006)
"B/W-aware Thin Node System"

FIRST
256 nodes
512 cores
+ BladeGRAPE
3.5TFLOPS
+ 35TFLOPS
(March 2006)
"Hybrid Cluster for Astrophysics"

T2K-Tsukuba
648 nodes
10368 cores
95 TFLOPS
(June 2008)
"General-purpose Fat Node System"

Project Plan of HA-PACS

- PACS-CS (not T2K) was shut down on Sept./2011
- New powerful system needed to be deployed around the end of 2011
  => HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences)
- Target performance range is up to 1PFLOPS with “GPU acceleration”
  - The target application fields will be partially limited
  - Current target: QCD, Astro, QM/MM (quantum mechanics / molecular mechanics, for life science)

Dual system plan

- HA-PACS is not only a “GPU-accelerated PC cluster”
- Two parts
  - HA-PACS *base cluster*:
    - for development of GPU-accelerated code for target fields, and performing product-run of them
  - HA-PACS/*TCA*: (TCA = Tightly Coupled Accelerators)
    - for elementary research on new technology for accelerated computing
    - Our original communication system based on PCI-Express named “PEARL”, and a prototype communication chip named “PEACH2” (=> Talk by Prof. Kodama)

GPU Computing: current trend of HPC

- GPU clusters in TOP500 on Nov. 2011
  - 2nd 天河Tienha-1A (Rpeak=4.70 PFLOPS, Rmax=2.57PFLOPS)
  - 4th 星雲Nebulae (Rpeak=2.98 PFLOPS, Rmax=1.27PFLOPS)
  - 5th TSUBAME2.0 (Rpeak=2.29 PFLOPS, Rmax=1.19PFLOPS)
  - (1st K Computer Rpeak=11.28 PFLOPS, Rmax=10.5PFLOPS)
- Features
  - high peak performance / cost ratio
  - high peak performance / power ratio
  - large scale applications with GPU acceleration don't run yet in daily production on GPU cluster

⇒ **Our First target is to develop large scale applications accelerated by GPU in real computational sciences**

Problems of GPU Cluster

- **Problems of GPGPU for HPC**
  - Data I/O performance limitation
    - Ex) GPGPU: PCIe gen2 x16
    - Peak Performance: 8GB/s (I/O) ⇔ 665 GFLOPS (NVIDIA M2090)
  - Memory size limitation
    - Ex) M2090: 6GByte vs CPU: 4 – 128 GByte
  - Communication between accelerators: no direct path (external)
    ⇒ communication latency via CPU becomes large
    - Ex) GPGPU:
      GPU mem ⇒ CPU mem ⇒ (MPI) ⇒ CPU mem ⇒ GPU mem
- **Researches for direct communication between GPUs are required**

Our second target is developing a direct communication system between external GPUs for a feasibility study for future accelerated computing

Project Formation

- HA-PACS (Highly Accelerated Parallel Advanced system for Computational Sciences)
  - Apr. 2011 – Mar. 2014, 3-year project (the system will be maintain until Mar. 2016)
  - Project Office for Exascale Computational Sciences (Leader: Prof. M. Umemura)
    - Develop large scale GPU applications : 14 members

    Elementary Particle Physics, Astrophysics, Bioscience, Nuclear Physics, Quantum Matter Physics, Global Environmental Science, Computational Informatics, High Performance Computing Systems
  - Project Office for Exascale Computing System Development（Leader: Prof. T. Boku）
    - Develop two types of GPU cluster systems: 15 members (including 4 members from outside universities)

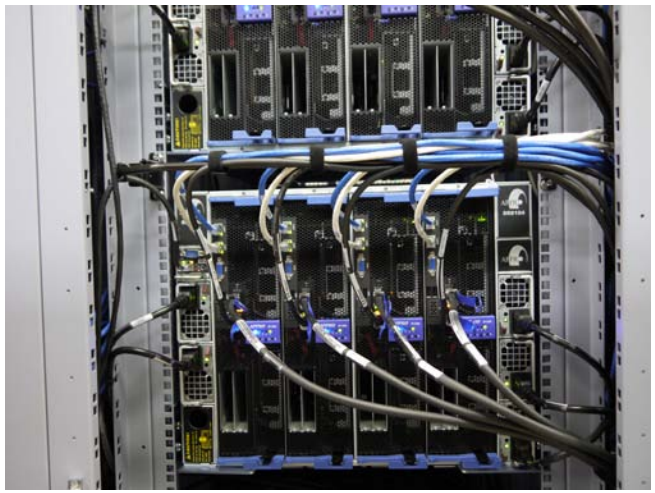
HA-PACS base cluster (Feb. 2012)

HA-PACS base cluster

Front view

Side view

HA-PACS base cluster



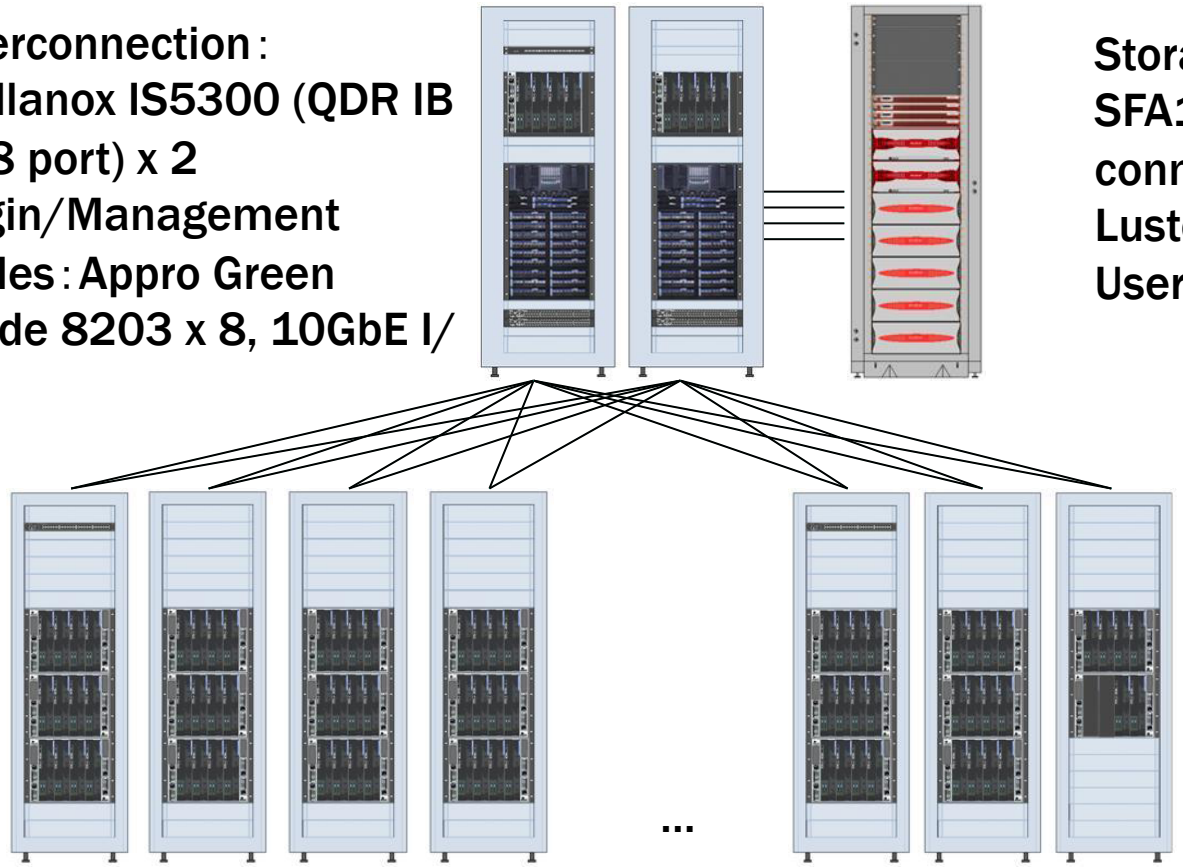
Rear view of one blade chassis with 4 blades

Front view of 3 blade chassis

Rear view of Infiniband switch and cables
(yellow=fibre, black=copper)

HA-PACS: Base Cluster Unit

Interconnection : Mellanox IS5300 (QDR IB 288 port) x 2
Login/Management nodes : Appro Green Blade 8203 x 8, 10GbE I/F

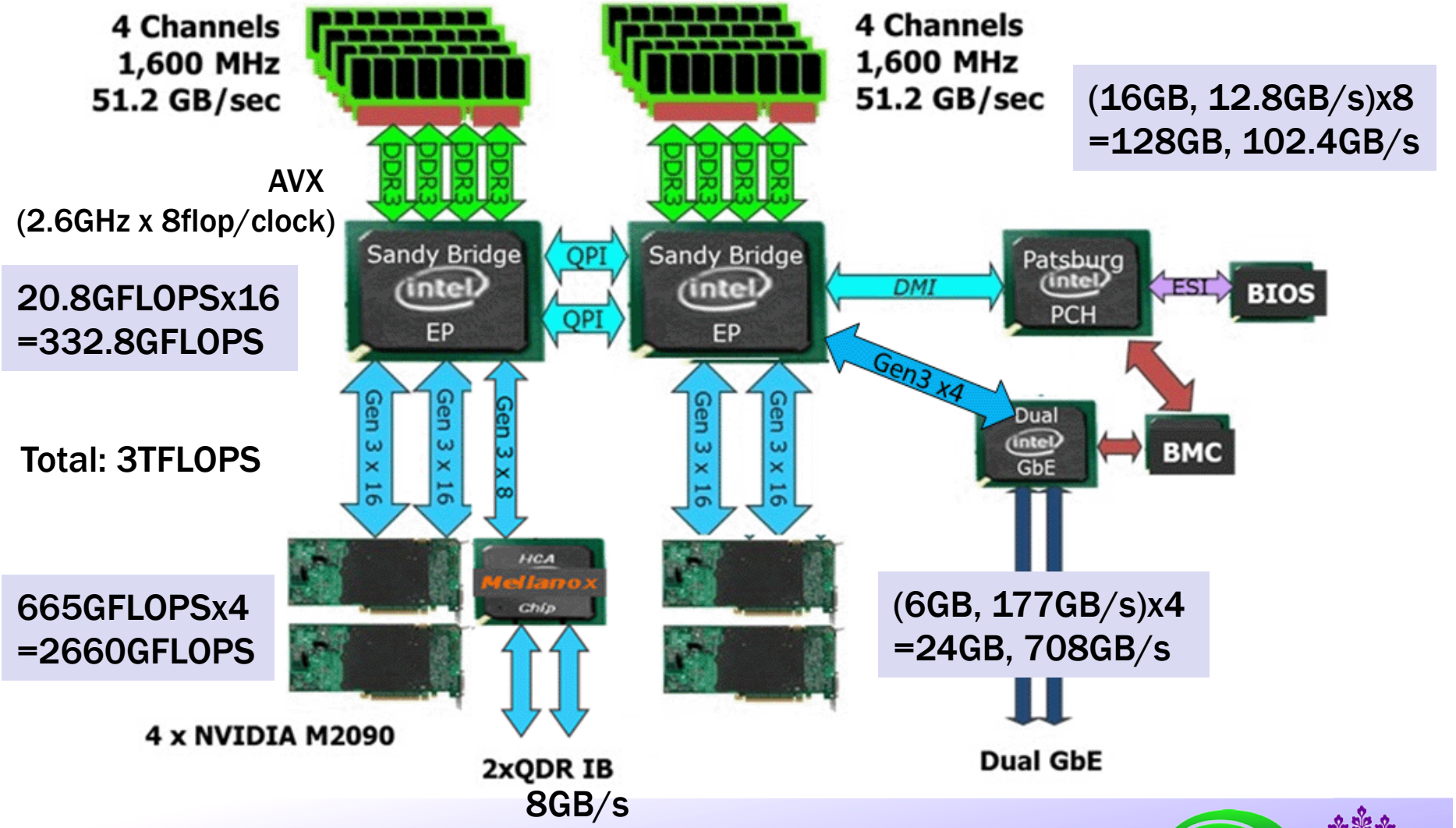


Storage : DDN SFA10000, connecting QDR IB, Luster File system, User area: 504TB

Computation node : Appro Green Blade 8204 (8U enc. 4 node) 268 node (67 enc./23 rack), 802TFLOPS

Total 26 racks

HA-PACS: base cluster (computation node)



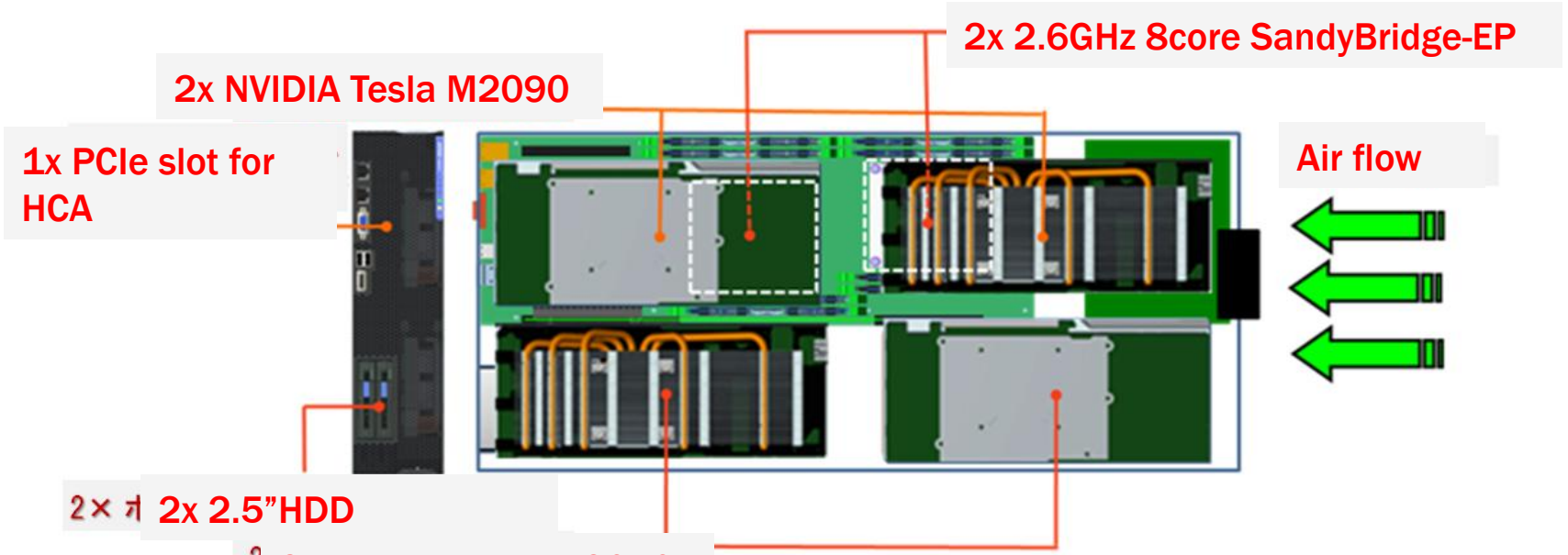
HA-PACS: base cluster unit（CPU）

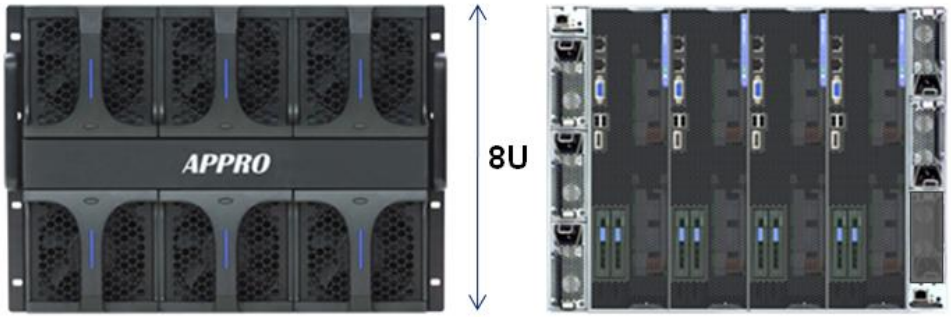
- **Intel Xeon E5 (SandyBridge-EP) x 2**
  - 8 cores/socket (16 cores/node) with 2.6 GHz
  - AVX (256-bit SIMD) on each core
    ⇒ peak perf./socket = 2.6 x 4 x 2 = 166.4 GFLOPS
    ⇒ pek perf./node = 332.8 GFLOPS
  - Each socket supports up to 40 lanes of PCIe gen3
    ⇒ great performance to connect multiple GPUs without I/O performance bottleneck
    ⇒ current NVIDIA M2090 supports just PCIe gen2, but net generation (Kepler) will support PCIe gen3
  - M2090 x4 can be connected to 2 SandyBridge-EP still remaining PCIe gen3 x8 x2
    ⇒ Infiniband QDR x 2

HA-PACS: base cluster unit（GPU）

- **NVIDIA M2090 x 4**
  - Number of processor core: 512
  - Processor core clock: 1.3 GHz
  - DP 665 GFLOPS, SP 1331GFLOPS
  - PCI Express gen2 ×16 system interface
  - Board power dissipation: <= 225 W
  - Memory clock: 1.85 GHz, size: 6GB with ECC, 177GB/s
  - Shared/L1 Cache: 64KB, L2 Cache: 768KB
  - With 4 GPUs = 2.66 TFLOPS/node

HA-PACS: base cluster unit（blade node）



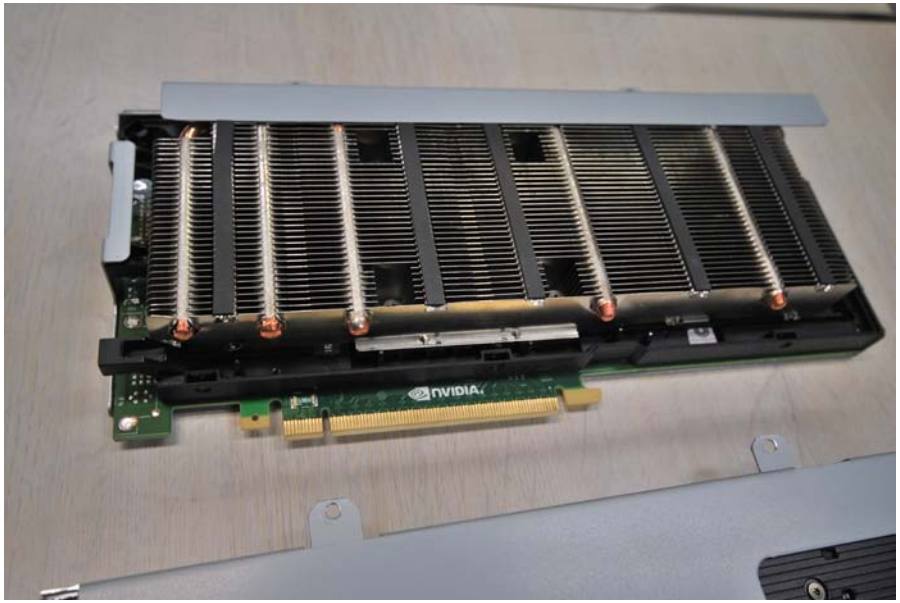


Front view

Rear view

Power Supply Unit and Fan
- 8U enclosure
- 4 nodes
- 3 PSU(Hot Swappable)
- 6 Fans(Hot Swappable)

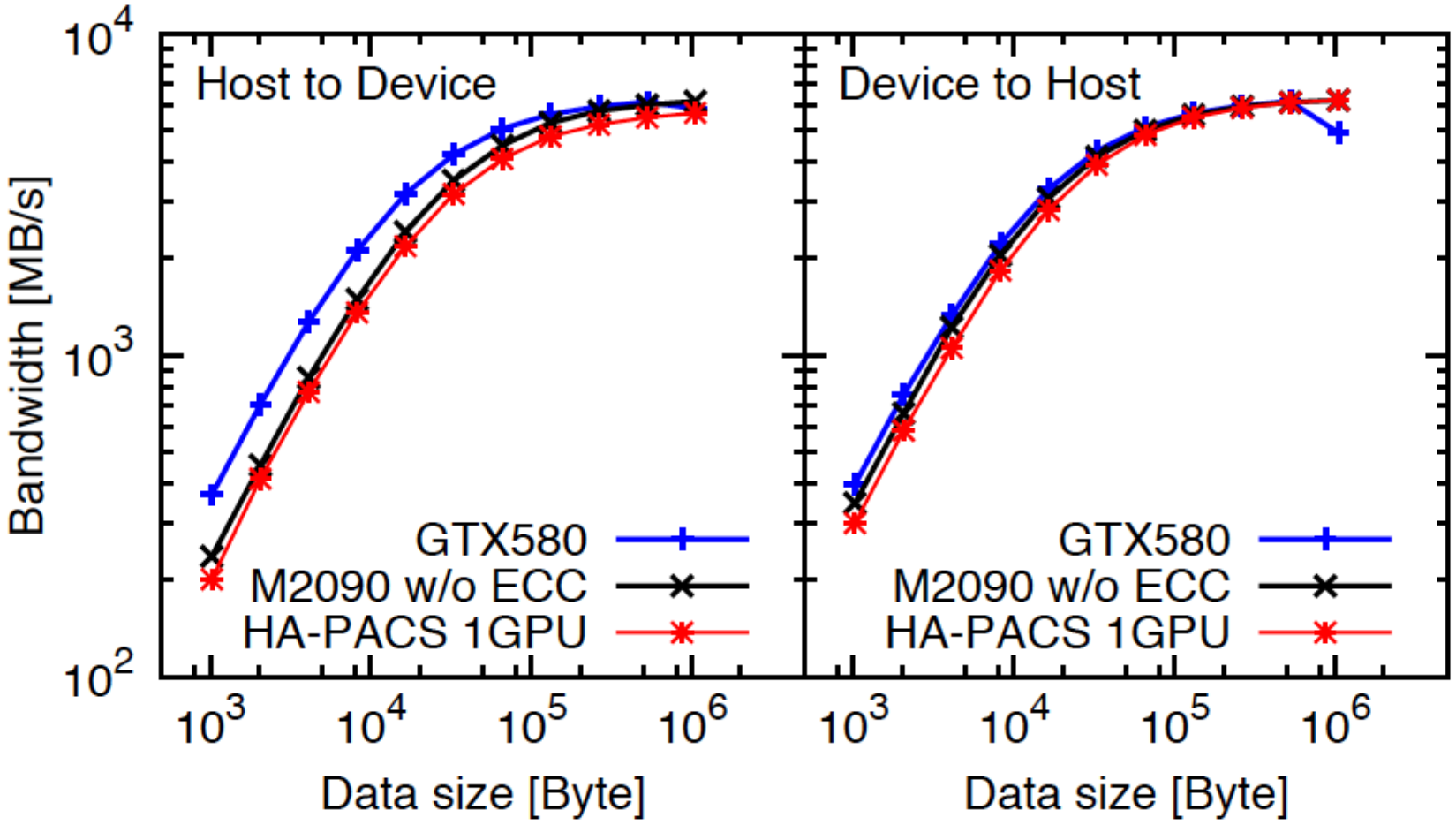
HA-PACS: base cluster unit（blade node）



Basic performance data

- MPI pingpong
  - 6.4 GB/s ($N_{1/2}$= 8KB)
  - with dual rail Infiniband QDR (Mellanox ConnectX-3)
  - actually FDR for HCA and QDR for switch
- PCIe benchmark (Device -> Host memory copy), aggregated perf. for 4 GPUs simultaneously
  - 24 GB/s ($N_{1/2}$= 20KB)
  - PCIe gen2 x16 x4, theoretical peak = 8 GB/s x4 = 32 GB/s
- Stream (memory)
  - 74.6 GB/s
  - theoretical peak = 102.4 GB/s

PCIe Host:Device communication performance



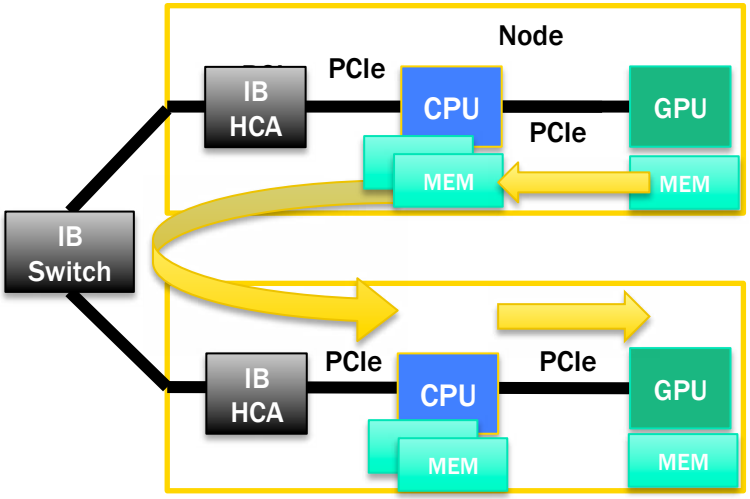
Slower start on Host->Device compared with Device->Host

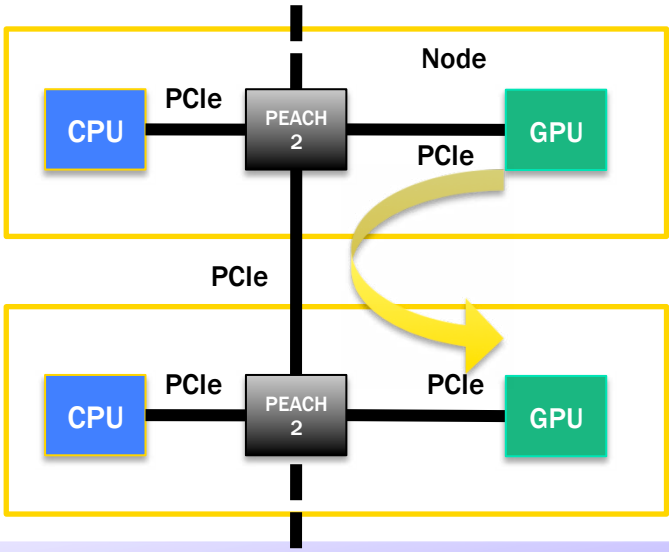
HA-PACS: base cluster system summary

- 268 nodes are connected by 2 IB switches
- CPU: 89TFLOPS + GPU: 713TFLOPS = total 802TFLOPS
- CPU: Memory size 34TByte, Bandwidth 27TByte/sec,
  GPU: Memory size 6.4TByte, Bandwidth 190TByte/sec
- Bisection bandwidth of IB: 2.1TByte/sec
- Storage User Area 504TByte
- Power Consumption max. 408kW (monitoring available)
- 26 racks (5.5m x 10m including maintenance area)
- Operation started: Feb. 2012

HA-PACS/TCA (Tightly Coupled Accelerator)

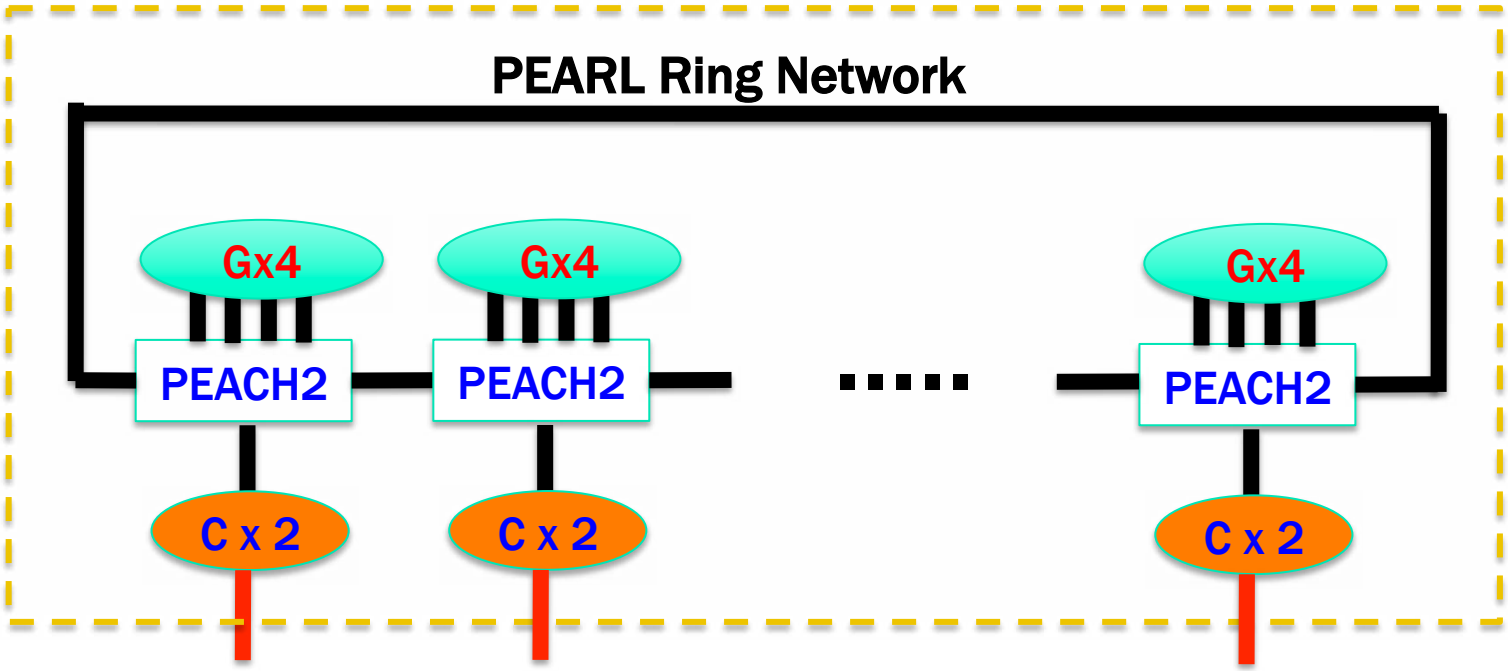
- **True GPU-direct**
  - current GPU clusters require 3-hop communication (3-5 times memory copy)
  - For strong scaling, Inter-GPU direct communication protocol is needed for lower latency and higher throughput
- Enhanced version of PEACH
  ⇒ ***PEACH2***
  - x4 lanes -> x8 lanes
  - hardwired on main data path and PCIe interface fabric





HA-PACS/TCA

Node Cluster = NC

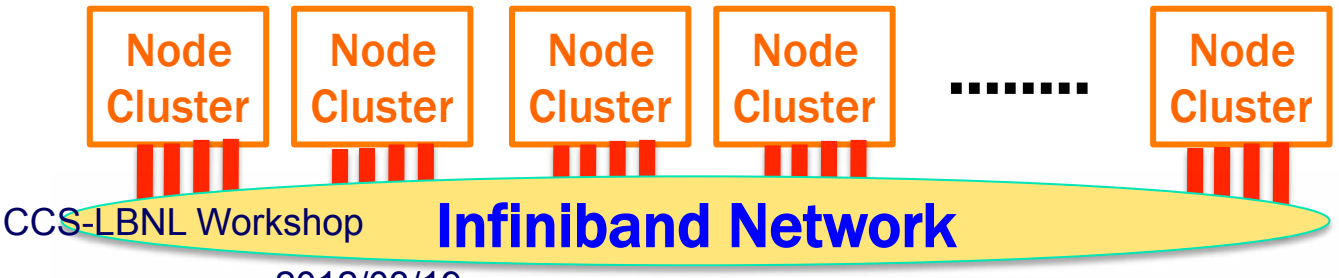


Node Cluster with 16 nodes
- GPUx64 (G)
- CPUx32 (C)
- GPU comm with PCIe
- IB link / node
- CPU: Xeon E5
- GPU: Kepler

Infiniband Link

- High speed GPU-GPU comm. by PEACH within NC (PCI-E gen2x8 = 5GB/s/link)
- Infiniband QDR (x2) for NC-NC comm. (4GB/s/link)

4 NC with 16 nodes, or 8 NC with 8 nodes = 360 TFLOPS extension to base cluster

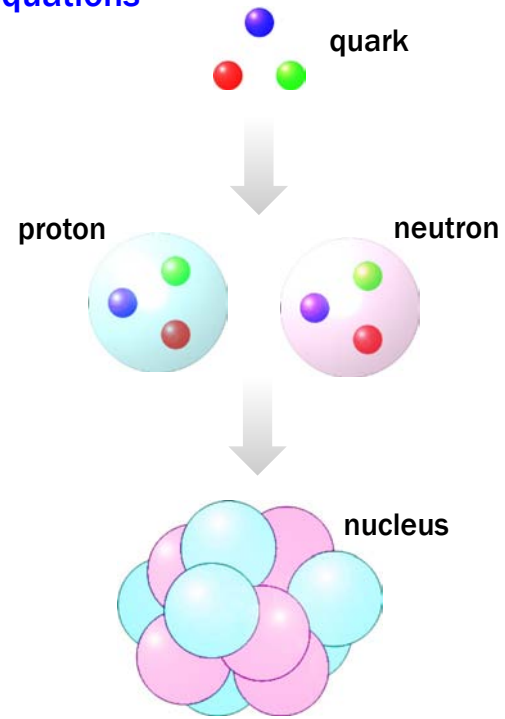


HA-PACS Application (1) : Elementary Particle Physics

Multi-scale physics

Investigate hierarchical properties via direct construction of nuclei in lattice QCD

GPU to solve large sparse linear systems of equations

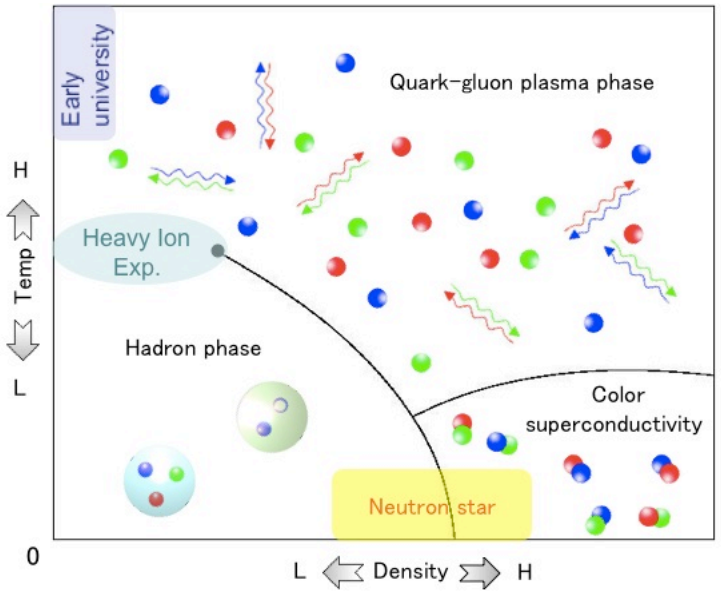


Finite temperature and density

Phase analysis of QCD at finite temperature and density

GPU to perform matrix-matrix product of dense matrices

Expected QCD phase diagram



HA-PACS Applications (2) : Astrophysics

(A) Collisional N-body Simulation

Globular Clusters

- Formation of the most primordial objects formed more than 10 giga years.
- Fossil object as a clue to investigate the primordial universe

Massive Black Holes in Galaxies

- Understanding of the formation of massive black holes in galaxies
- Numerical simulations of complicated gravitational interactions between stars and multiple black holes in galaxy centers.

➢ Direct (brute force) calculations of acceleration and jerks are required to achieve the required numerical accuracy

➢ Computations of the accelerations of particles and their time derivatives (jerks) are time consuming.

➢ Accelerations and jerks are computed on GPU

(B) Radiation Transfer

First Stars and Re-ionization of the Universe

- Understanding of the formation of the first stars in the universe and the succeeded re-ionization of the universe.

Accretion Disks around Black Holes

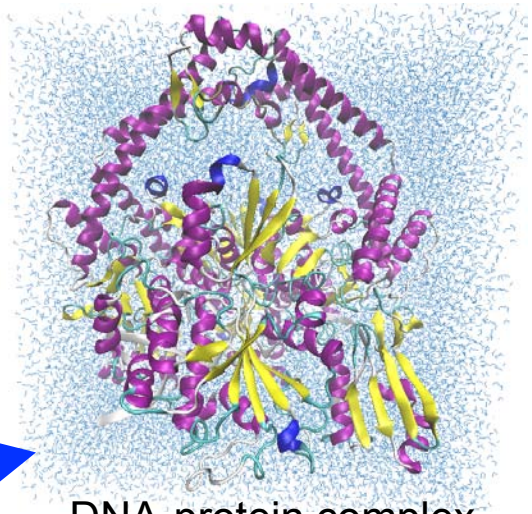
- Study of the high temperature regions around black holes

➢ Calculation of the physical effects of photons emitted by stars and galaxies onto the surrounding matter.

➢ So far, poorly investigated due to its huge amount of computational cost, though it is of critical importance in the formation of stars and galaxies.

➢ Computations of the radiation intensity and the resulting chemical reactions based on the ray-tracing methods can be highly accelerated with GPUs owing to its high concurrency.

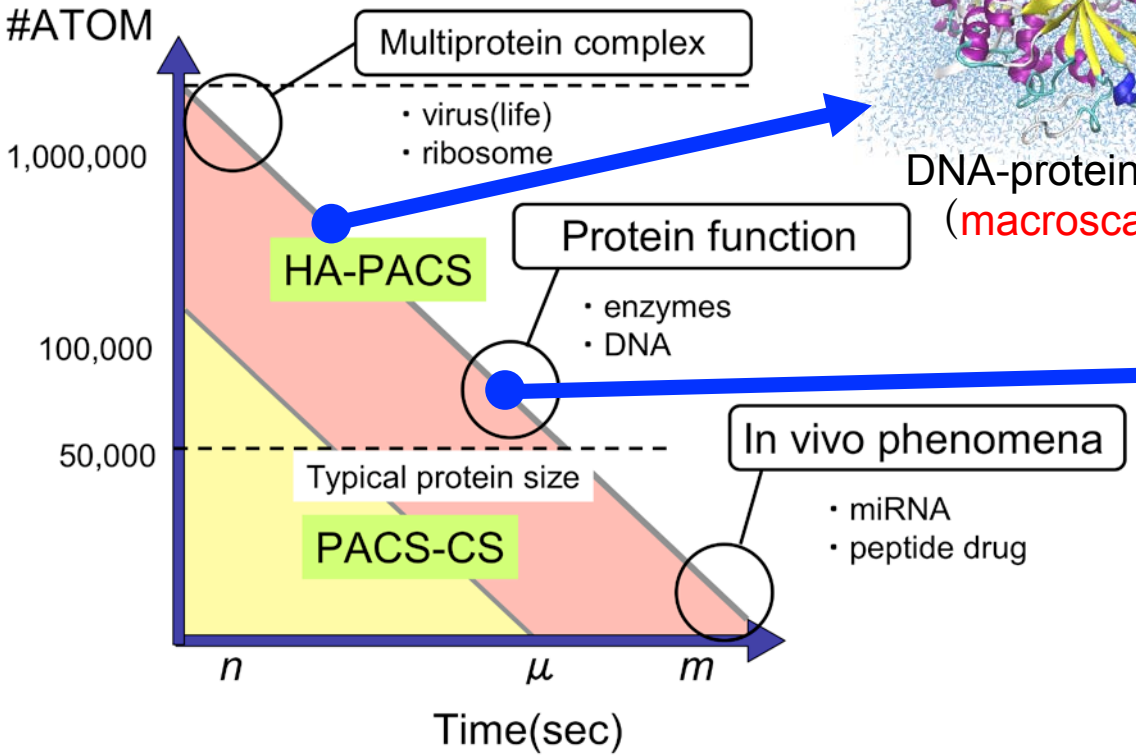
HA-PACS Application (3)：Bioscience



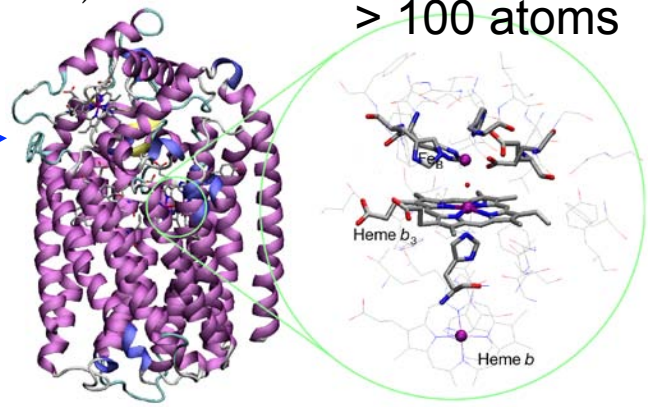
DNA-protein complex
（macroscale MD）

GPU acceleration

- Direct coulmb
（Gromacs, NAMD, Amber)
-2 electron integral



QM region
> 100 atoms



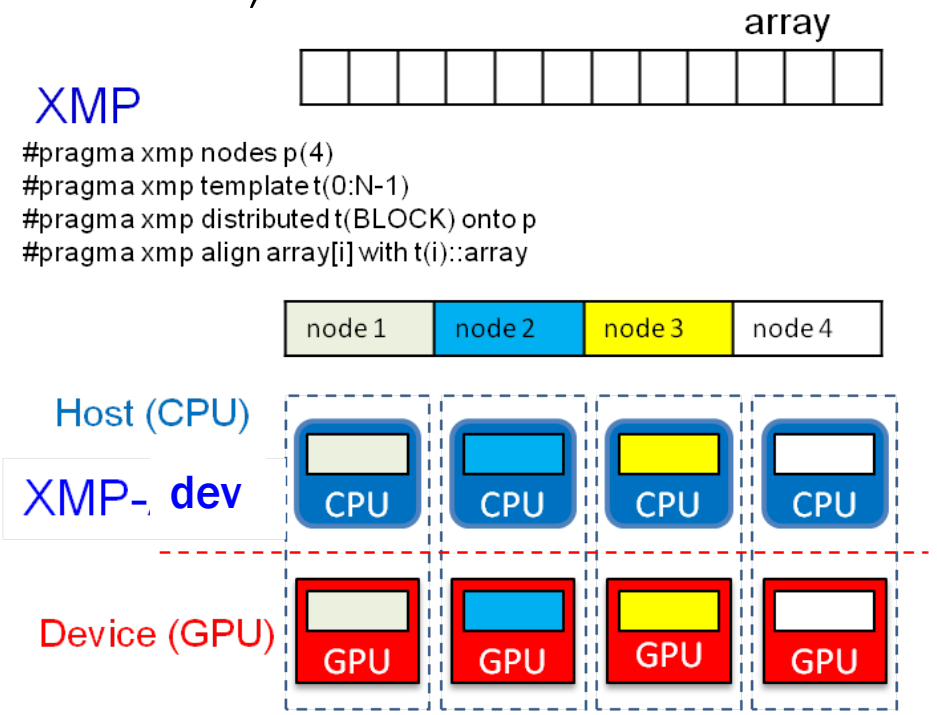
Reaction mechanisms
（QM/MM-MD）

HA-PACS Application (4)

- **Other advanced researches on HPC Division in CCS**
  - **XcalableMP-dev (XMP-dev) for easy and simple programming language to support distributed memory & GPU accelerated computing for large scale computational sciences**
  - **G8 NuFuSE (Nuclear Fusion Simulation for Exascale) project platform for porting Plasma Simulation Code with GPU technology**
  - **Climate simulation especially for LES (Large Eddy Simulation) for cloud-level resolution on city-model size simulation**
  - **Any other collaboration …**

XMP-dev: language for cluster with accelerators

- According to extended directives of XMP, on distributed local array, XMP-dev performs:
  - transfer to GPU memory
  - computation on GPU
  - transfer from GPU memory



- node-to-node GPU comm. is performed through CPU memory
  - A part of distributed array on CPU is also mapped on GPU memory (replicate) and they are transferred to/from CPU memory（copy-in, copy-out）-> parallel processing among GPU nodes

XMP-dev extended directives

- dev replicate directive: data allocation/free

```
#pragma xmp dev replicate (list)
  compound-statement
```

- dev replicate_sync directive: data copy between CPU
  - in clause: host -> GPU, out clause: host -> GPU

```
#pragma xmp dev replicate_sync clause
  clause ::= in (list) | out (list)
```

- dev loop directive: loop process distribution to GPU
  - partial code is converted to GPU kernel function on compilation, and executed by GPU

```
#pragma xmp dev loop [(list)] on onref [dev clause]
  loop-statement
  dev clause ::= private(list) | firstprivate (list) |shared (list) num_threads(x, y[,z])
```

- GPU computing primitives are implemented with directives
- currently, assuming each node has only one GPU device
  -> “node” in XMP corresponds to “MPI process”

Sample code of XMP-dev extension

XMP

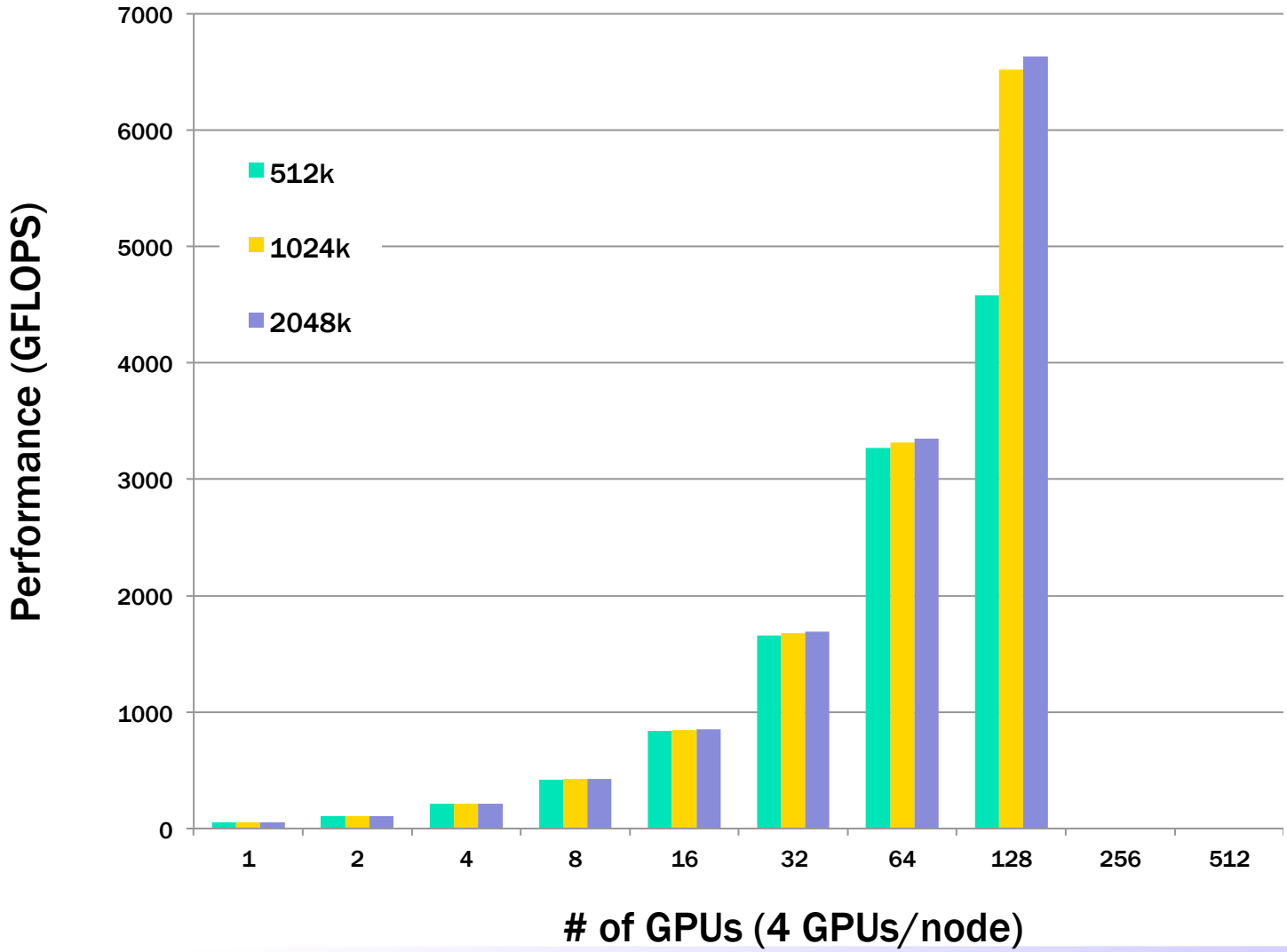
```
int a[N], b[N];
#pragma xmp align [i] with t(i) :: a, b
#pragma xmp shadow a[*]
void main(void) { . . .
int i, j;
#pragma xmp loop on t(i)
for (i = 0; i < N; i++) {
   b[i] = 0; a[i] = i;
#pragma xmp reflect a
#pragma xmp loop on t(i)
for (i = 0; i < N; i++) {
   b[i] = 0;
   for (j = 0; j < N; j++) {
      b[i] += a[j] ;
   }
}
} // main()
```

- Getting summation of an array
- Loop execution part is compiled by CUDA compiler

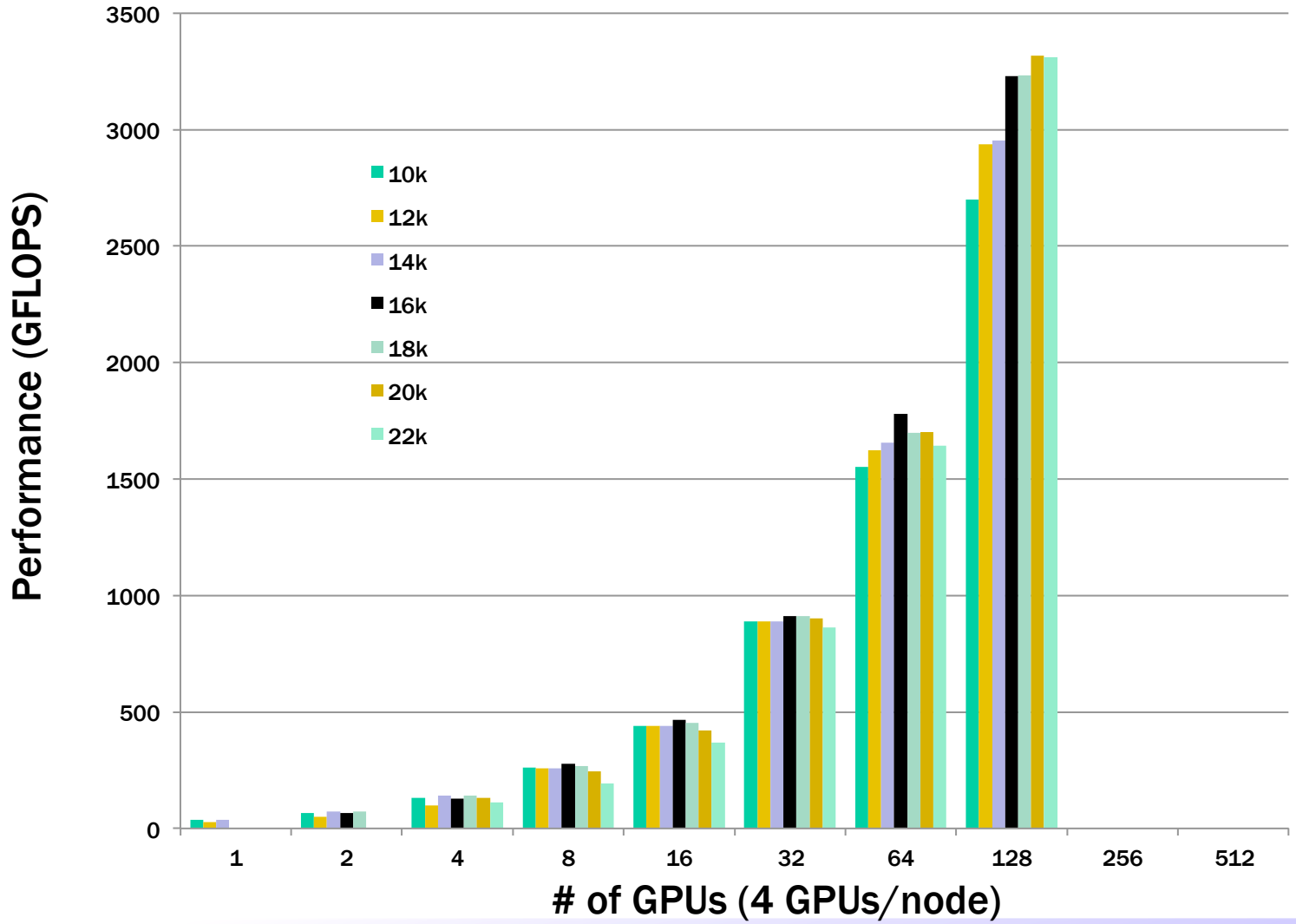
XMP-ACC

```
int a[N], b[N];
#pragma xmp align [i] with t(i) :: a, b
#pragma xmp shadow a[*]
void main(void) {
int i, j;
#pragma xmp loop on t(i)                    ┐
for (i = 0; i < N; i++) {                   │ Execution
   b[i] = 0; a[i] = i;                      │ Host (CPU)
#pragma xmp reflect a                       ┘
// GPU memory malloc
#pragma xmp dev replicate (a, b)
{
// Data copy CPU → GPU
#pragma xmp dev replicate_sync in (a)
#pragma xmp dev loop on t(i)                ┐
for (i = 0; i < N; i++) {                   │ Execution
    for (j = 0; j < N; j++) {               │ Device (GPU)
      b[i] += a[j] ;                        │
   }                                        │
}                                           ┘
// Data copy GPU → CPU
#pragma xmp dev replicate_sync out (b)
} // acc replicate
} // main()
```

Preliminary evaluation: N-body (strong scaling)



Preliminary evaluation: Matrix-Multiply (strong scaling)

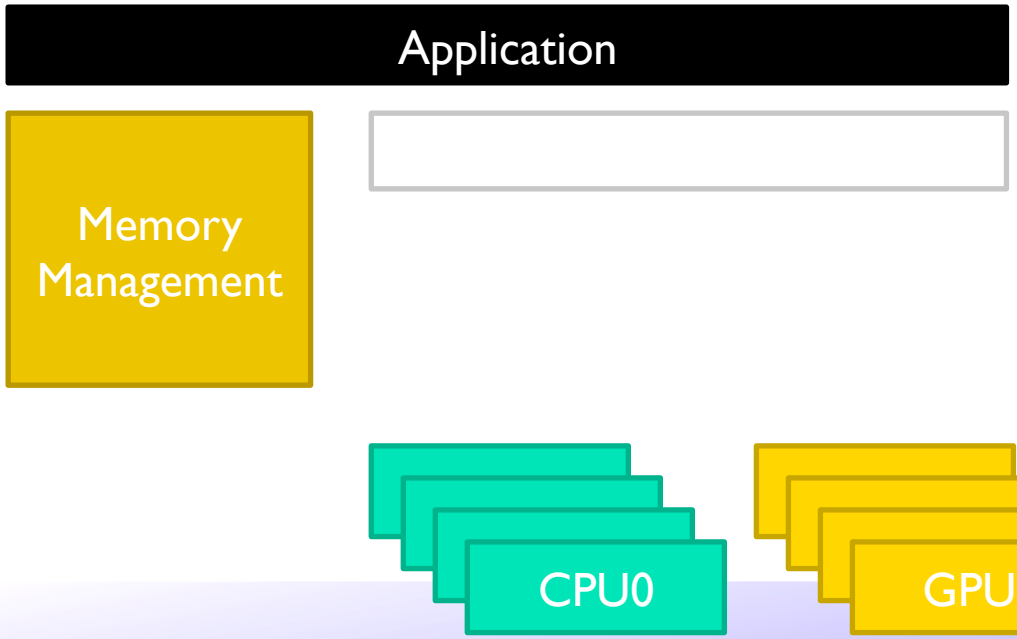


XMP-dev(StarPU)

- Coordinating multi-core CPU and GPUs in a node for hybrid work sharing for large scale GPU clusters
- Issues
  - Load balancing among CPU and GPU computational resources
  - Data distribution among multiple address spaces
  - Sub-task scheduling
  - Data distribution among nodes
- Solution
  - Introducing multi-core/GPUs hybrid task scheduling system with data management to XMP-dev
    ⇒ StarPU (by INRIA Bordeaux)

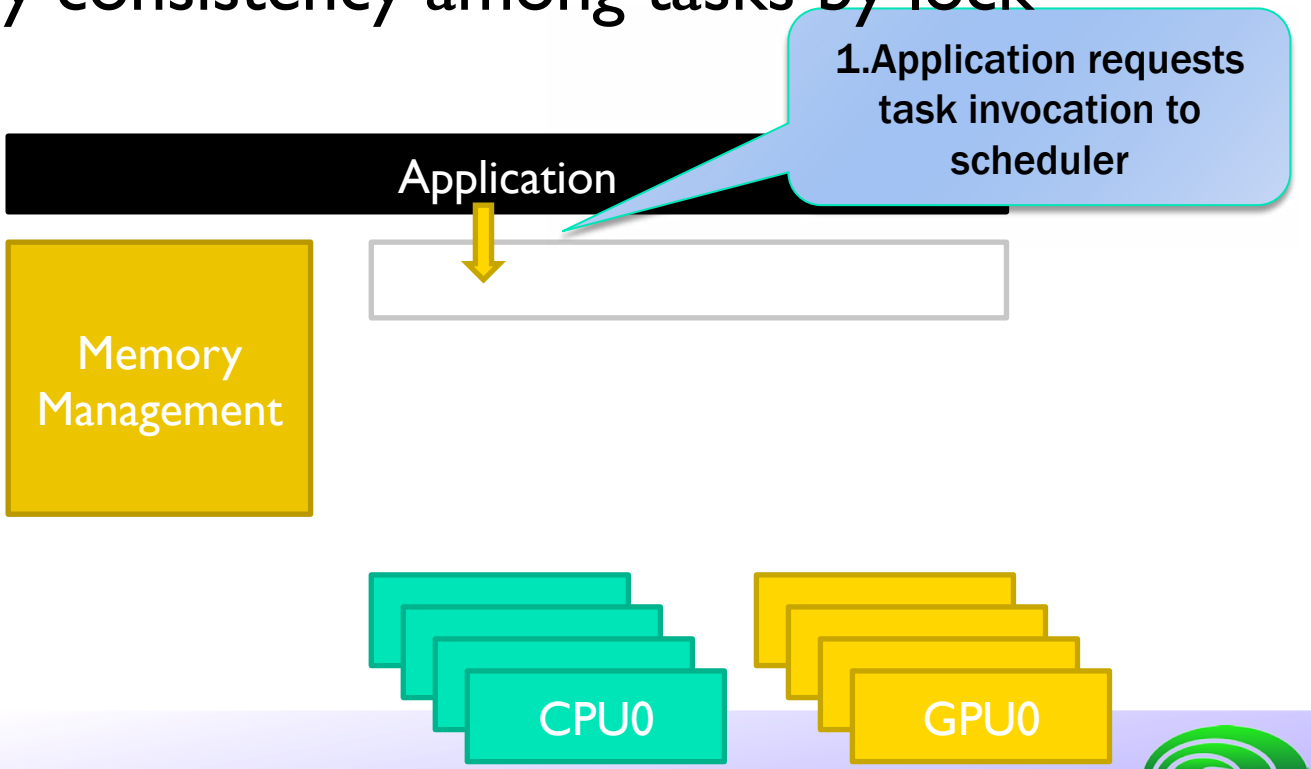
StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



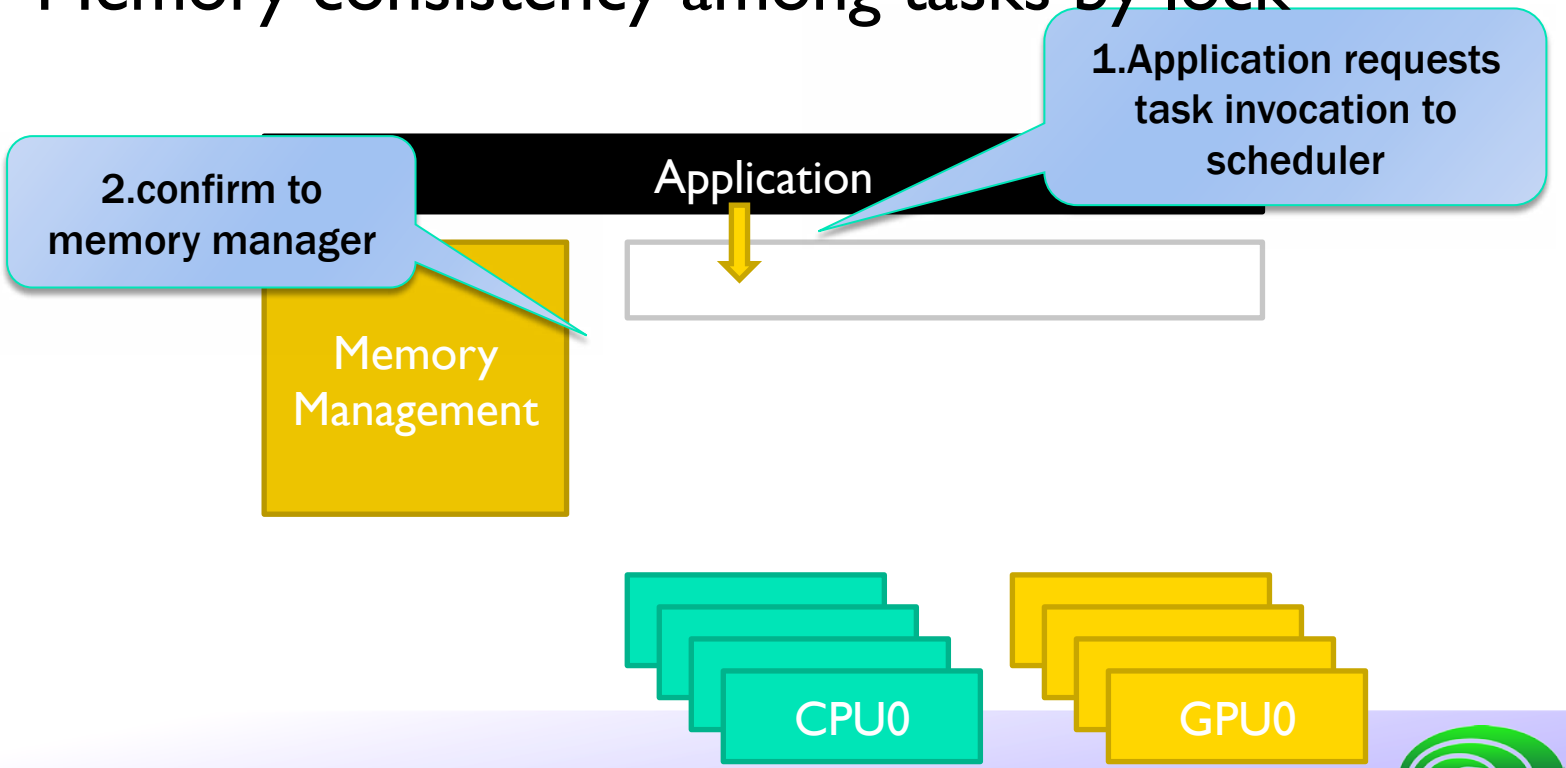
StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



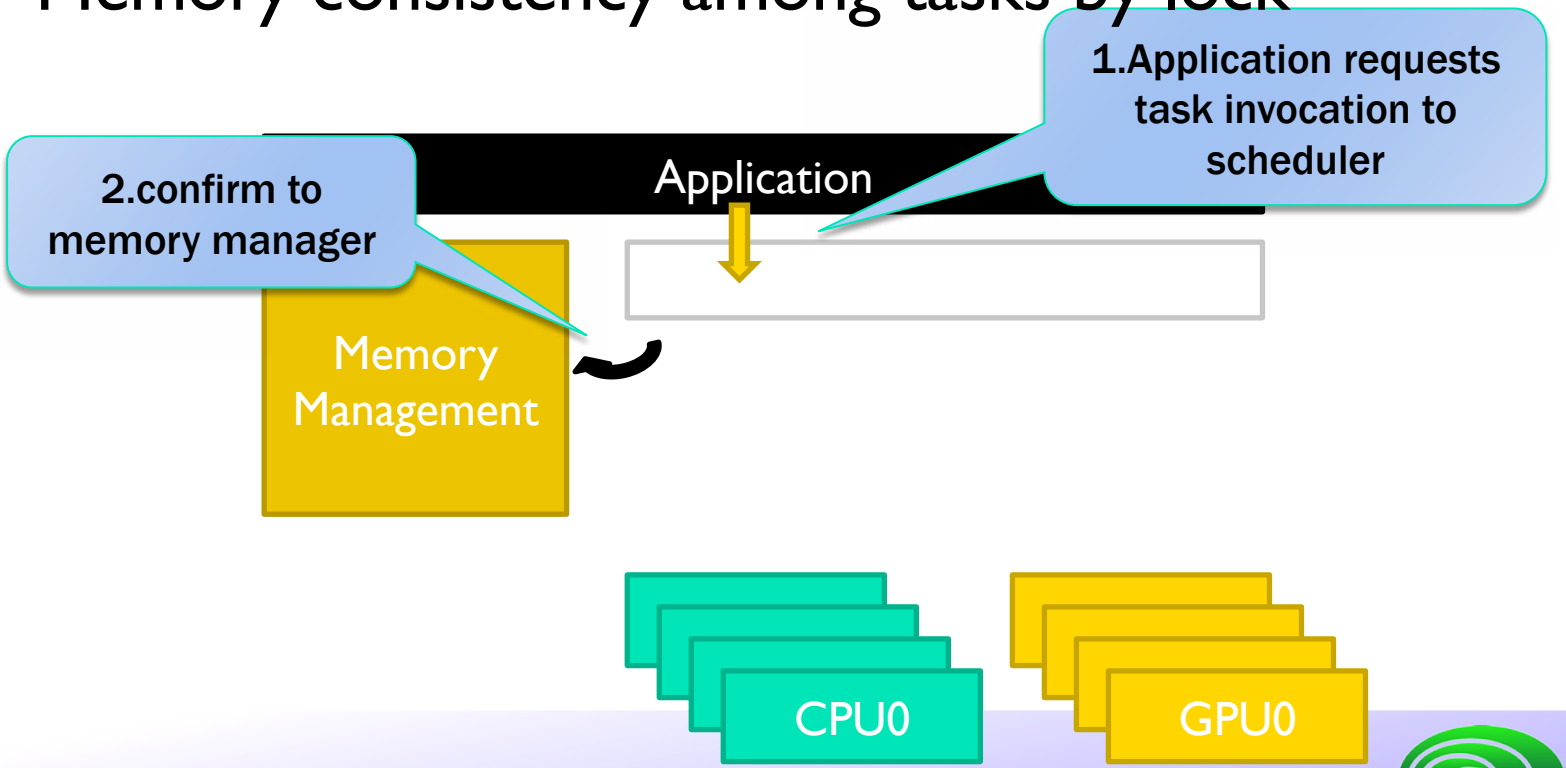
StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



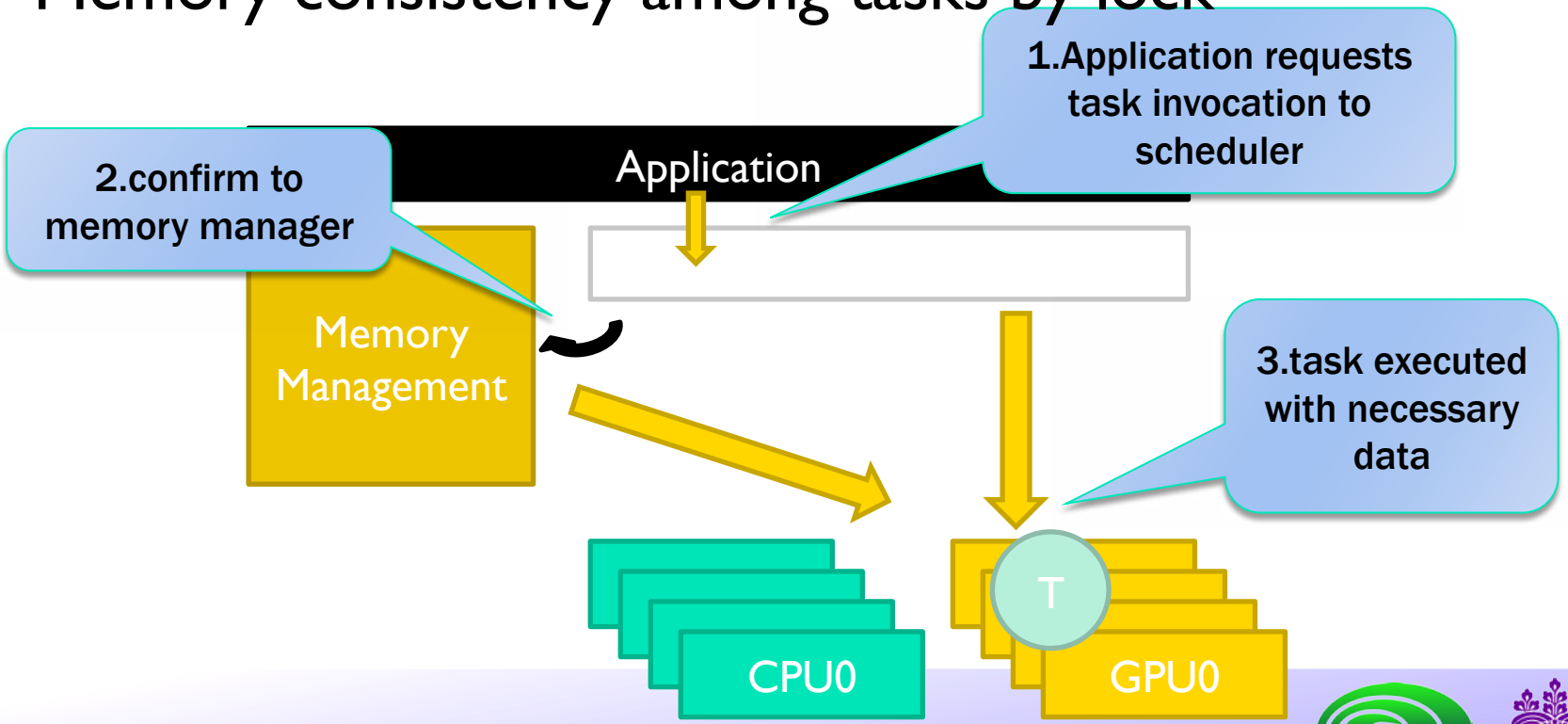
StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



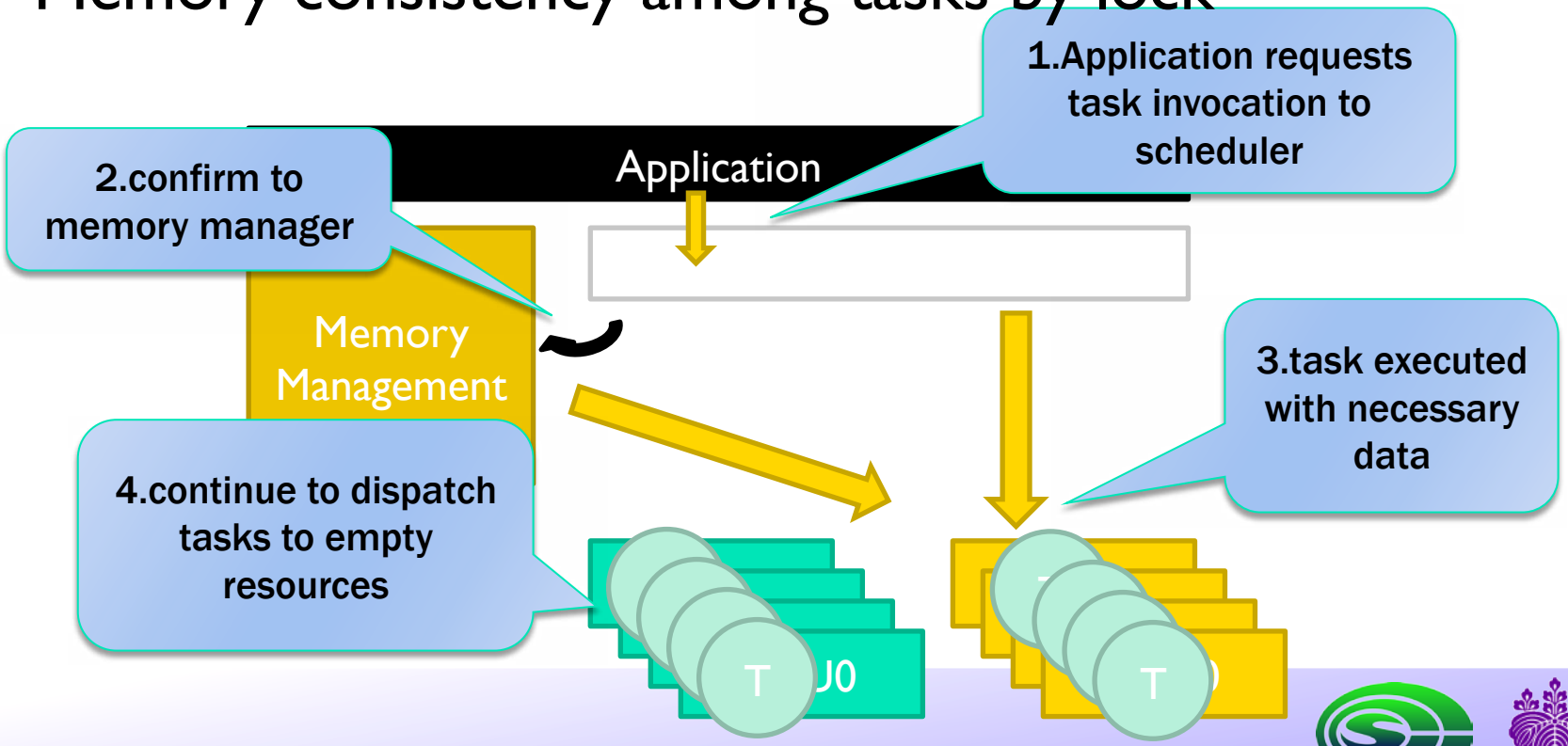
StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



StarPU

- Developed by INRIA Bordeaux
- Data distribution in a node, dynamic scheduling of tasks on CPU cores and GPUs
- Memory consistency among tasks by lock



XMP-dev(StarPU) strategy

- **Basic construction of XMP-dev -> StarPU**
  - Memory mapping of host CPU to GPU/CPU-core
    - “#pragma XMP dev replicate”
      ⇒ “partiton” and “register” of StarPU to allocate copy image of host memory to device
  - Memory synchronization between host CPU and GPU
    - “#pragma XMP dev replicate_sync in”
      ⇒ dispatching partitioned area of array when task is invoked
    - “#pragma XMP dev replicate_sync out”
      ⇒ RW variables are automatically retrieved at task finishing
  - Loop iteration dispatching to GPU/CPU-core
    - “#pragma XMP dev loop”
      ⇒ task creation and dispatch

Summary

- HA-PACS consists of two elements: HA-PACS base cluster for application development and HA-PACS/TCA for elementary study for advanced technology on direct communication among accelerating devices (GPUs)
- HA-PACS base cluster started its operation from Feb. 2012 with 802 TFLOPS peak performance
  (Linpack performance will come on June 2012, also expecting good score on Green500)
- FPGA implementation of PEACH2 will be finished for the prototype version on Mar. 2012 and enhanced for final version in following 6 months
- HA-PACS/TCA with at least 200 TFLOPS additional performance will be installed around Mar. 2013