Practical Parallel Computing (実践的並列コンピューティング)

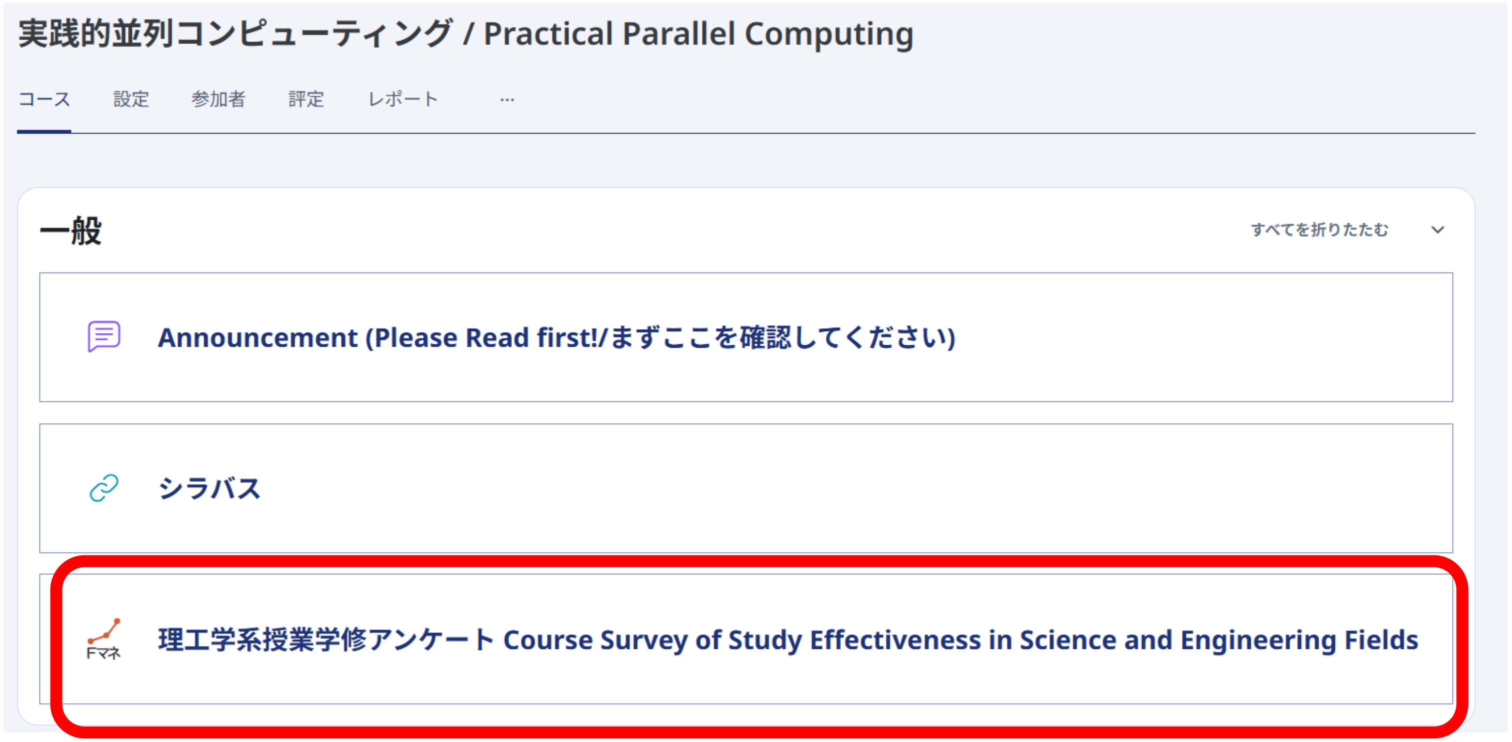
2025 Class No.15 (Optional)
Hybrid Programming &
TSUBAME4.0

Toshio Endo
endo@scrc.iir.isct.ac.jp

Please Take Part in Couse Survey 授業学修アンケートに回答ください

- Please log-in LMS and go to lecture page

➔ 「理工学系授業学修アンケート Course Survey of Study Effectiveness」



実践的並列コンピューティング / Practical Parallel Computing

コース　設定　参加者　評定　レポート　…

一般

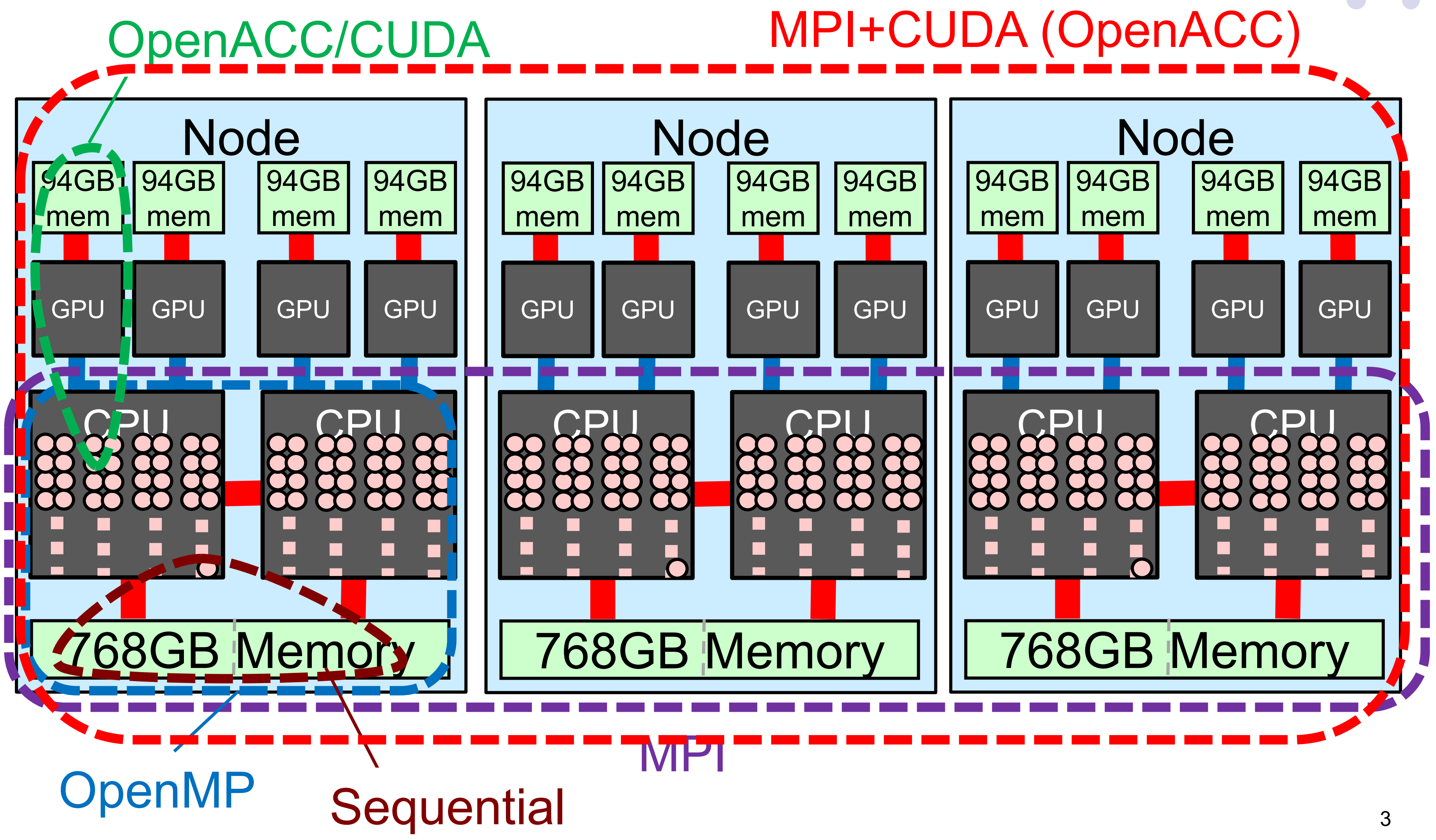
すべてを折りたたむ

Announcement (Please Read first!/まずここを確認してください)

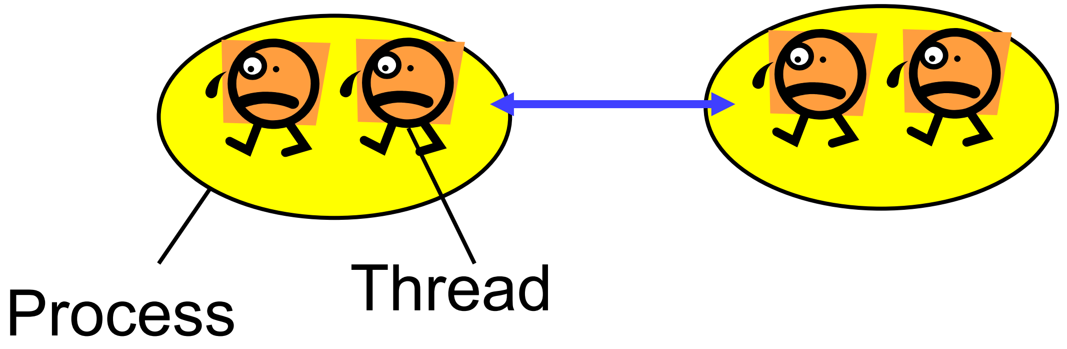
シラバス

理工学系授業学修アンケート Course Survey of Study Effectiveness in Science and Engineering Fields

Parallel Programming Methods on TSUBAME



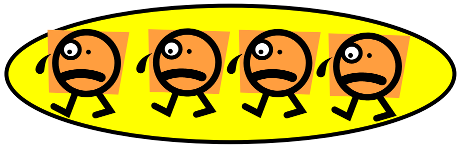
Hybrid Programming with MPI+OpenMP



MPI+OpenMP

OpenMP:

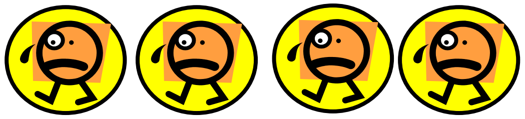
1process has multiple threads



➔ Only 1 node can be used

MPI :

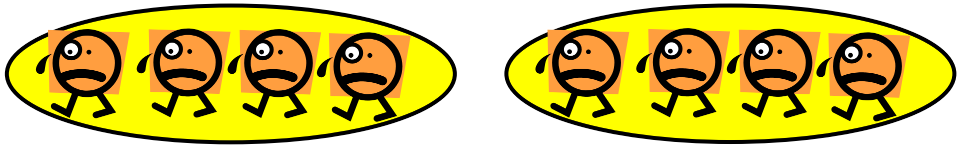
Multiple processes are used
(usually) Each has 1 thread



➔ Multiple nodes can be used

MPI +OpenMP:

Multiple processes are used
Each has multiple threads



➔ Multiple nodes can be used
Communication cost may be smaller than pure MPI

Sample: ppcomp-ex/mpi/mm-mpi-omp/

Compiling mm-mpi-omp Sample

[make sure that you are at a interactive node (rXn11)]

```
module load intel-mpi
```
[Do once after login]

[please go to your ppcomp-ex directory]

```
cd mpi/mm-mpi-omp
make
```
[An executable file "mm" is created]

```
export OMP_NUM_THREADS=8
mpiexec -n 2 ./mm 2000 2000 2000
```

Number of threads per process

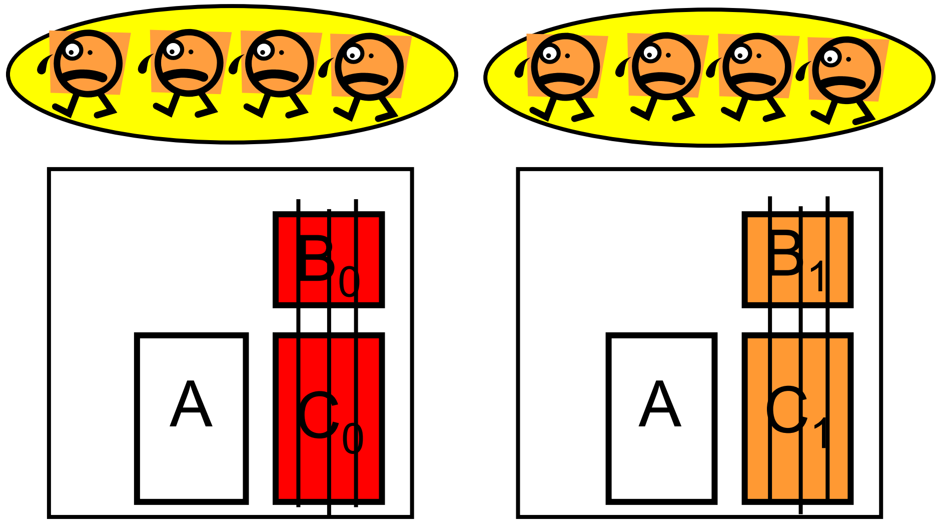
Number of processes

Messages between Processes

- Basically MPI communication should be out of OpenMP parallel region
- If you want to do communication freely, please try google MPI_Init_thread()

Data Distribution in mm-mpi-omp

- Each process has whole copy of A and divided B/C
- Threads in a process computes a part of partial C



There are 2 copies of A, not 8

Job Submission of mm-mpi-omp

In job.sh sample, 2 cpu_16 node partitions are allocated

job.sh

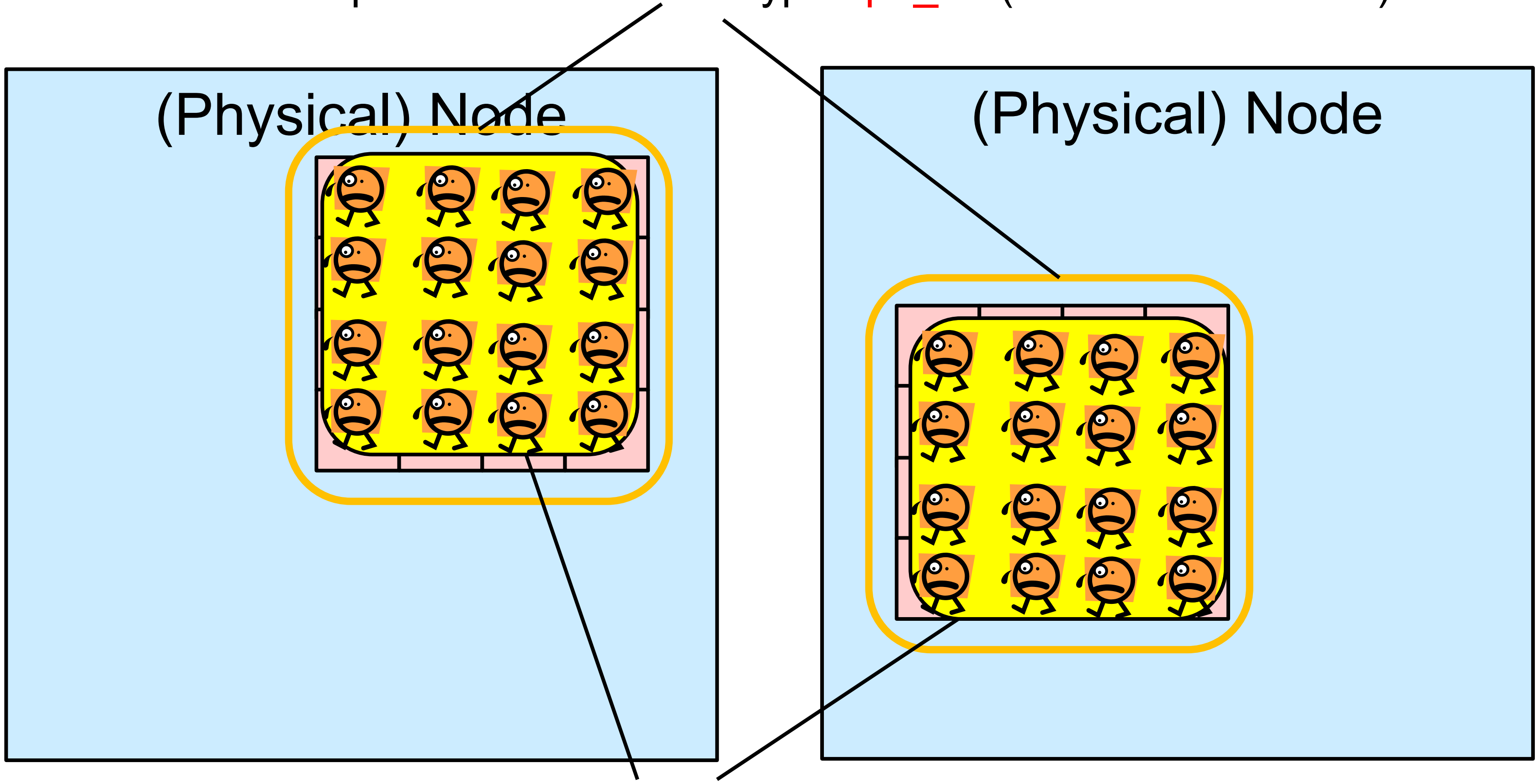
```
#!/bin/sh
#$ -cwd
#$ -l cpu_16=2
#$ -l h_rt=0:10:00

module load intel-mpi

export OMP_NUM_THREADS=16
mpiexec -n 2 -ppn 1 ./mm 2000 2000 2000
```

16 threads per process

2 processes, 1 processes per node

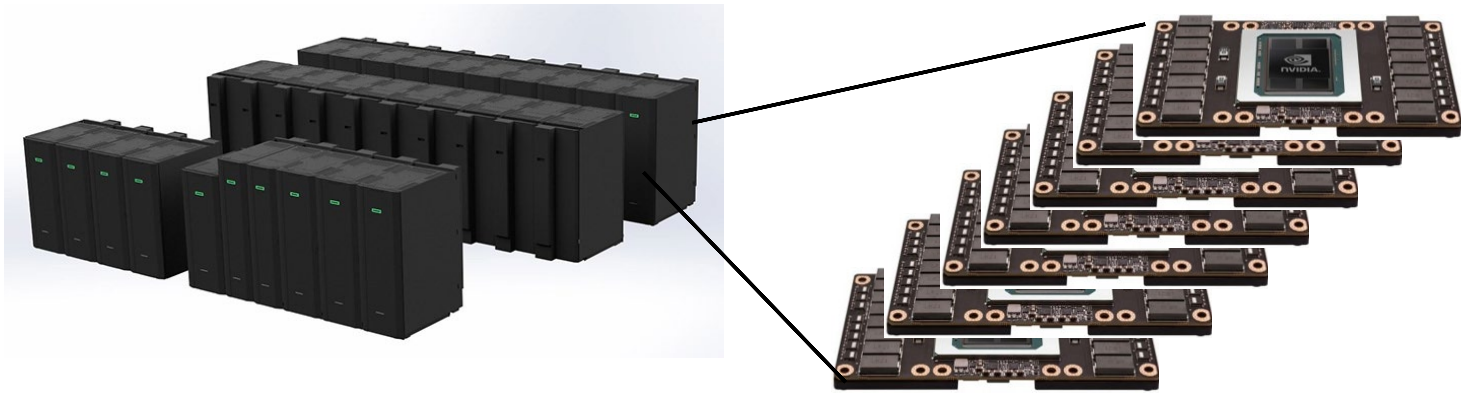
What Happens on TSUBAME with mm-mpi-omp/job.sh

(1) -l cpu_16=2 ➔ Job scheduler allocates 2 node partitions. Each has type cpu_16 (with 16 CPU cores)



(2) -n 2 -ppn 1 ➔ mpiexec invokes 2 process, 1 processes per node (partition)

Using Multiple GPUs with MPI+CUDA



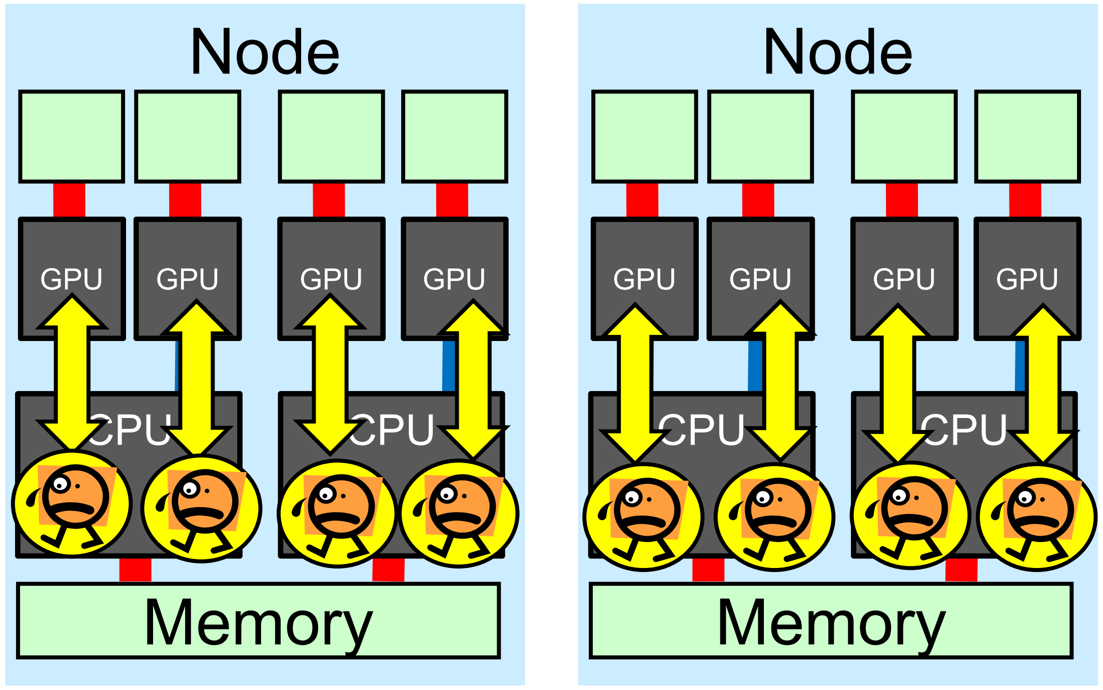
Methods to Use Multiple GPUs

- GPUs on multiple nodes
  - MPI + CUDA
    - 1 process uses 1 GPU (mpi/mm-mpi-cuda sample)
- GPUs on a single node

  (Up to 4 GPUs on a TSUBAME4.0 node_f)
  - MPI+CUDA
  - OpenMP + CUDA
    - 1 thread uses 1 GPU
  - 1 thread switches multiple GPUs
    - cudaSetDevice() is called many times

Using Multiple GPUs with MPI

- Basic idea:
  - (1) Start processes on multiple nodes by MPI
  - (2) Each process uses its local GPU by CUDA



Sample: ppcomp-ex/mpi/mm-mpi-cuda/

Compiling mm-mpi-cuda Sample

```
[make sure that you are at a interactive node (rXn11) ]
module load cuda
module load intel-mpi    [Do once after login]
[please go to your ppcomp-ex directory]
cd mpi/mm-mpi-cuda
make
[An executable file "mm" is created]
```

In this Makefile,

- nvcc is used as the compiler
- mpicxx is used as the linker, with CUDA libraries

This Makefile is for current TSUBAME, so you will need to modify it for other systems

Executing mm-mpi-cuda

- Interactive use is only for one node

→ To use multiple nodes, job submission is required

qsub job2q.sh ➔ node_q (1GPU) x 2 are used ➔ 2GPUs in total

qsub job2f.sh ➔ node_f (4GPU) x 2 are used ➔ 8 GPUs in total

job2f.sh

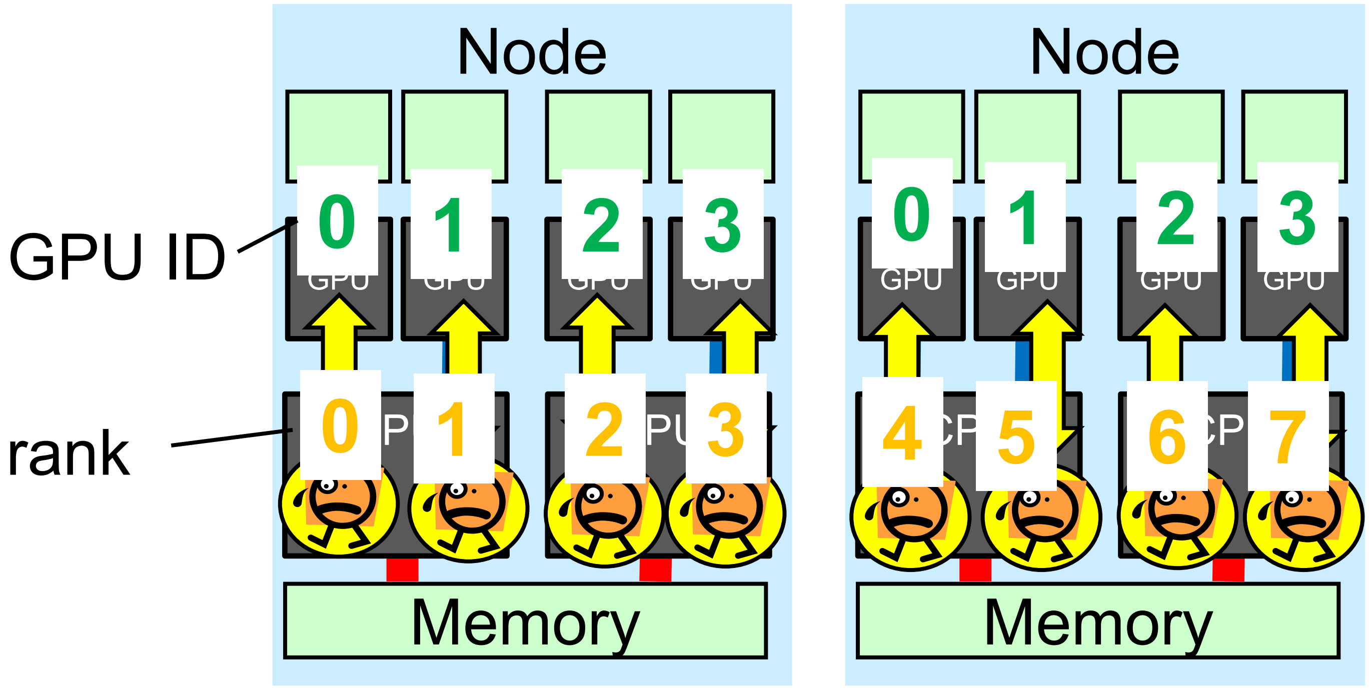
```
#!/bin/sh
#$ -cwd
#$ -l node_f=2
#$ -l h_rt=0:10:00

module load cuda intel-mpi

mpiexec -n 8 -ppn 4 ./mm 10000 10000 10000
```

8 processes, 4 processes per node

TSUBAME is crowded and jobs with node_f may require long waiting time...

Using Multiple GPUs per node (1)

- In case of "node_f=2", each node has 4 GPU
  - In default, all processes use "GPU 0" on the node → slow ☹
- Each process should determine GPU ID by (rank%4)
  - Number of GPU per node is obtained with cudaGetDeviceCount()



Using Multiple GPUs per Node (2)

- If each node has multiple GPUs, each process should use distinct GPUs

mm-mpi-cuda/mm.cu

```
        :
int ndev;
cudaGetDeviceCount(&ndev);
cudaSetDevice(rank % ndev);
// Hereafter, GPU (rank%ndev) is used
```

Number of GPUs per node
➔ 1 on node_q, node_o...
➔ 2 on node_h
➔ 4 on node_f

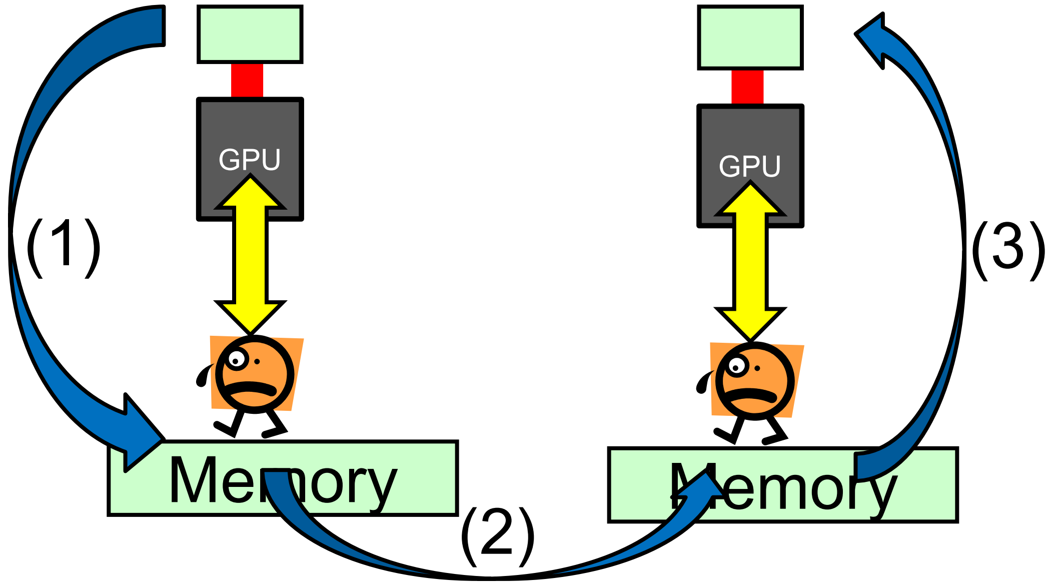
Data Transfer (1)

- mm sample does not use communication
- If we want to do, the basic method is

  (1) Copy data on GPU memory to CPU (cudaMemcpy)

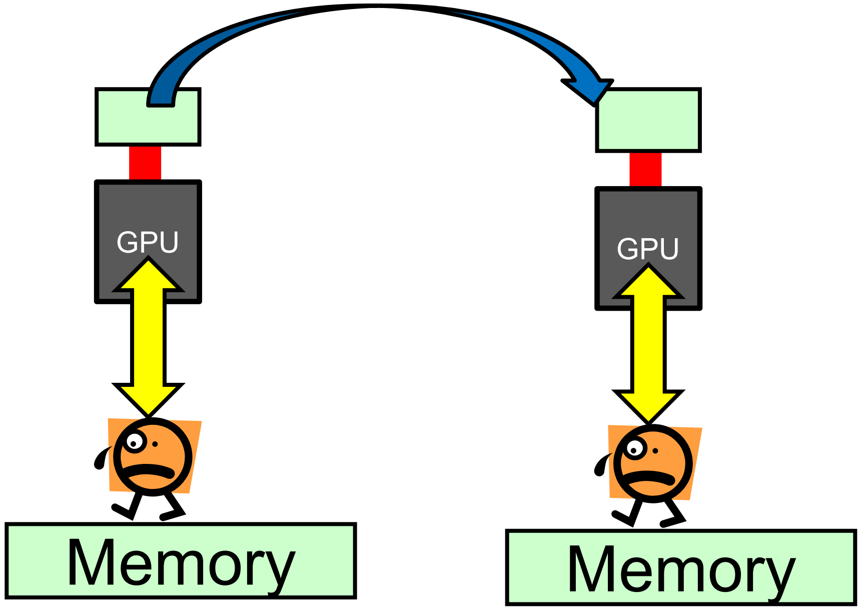
  (2) Transfer between processes (MPI_Send/MPI_Recv)

  (3) Copy data on CPU memory to GPU (cudaMemcpy)



Data Transfer (2)

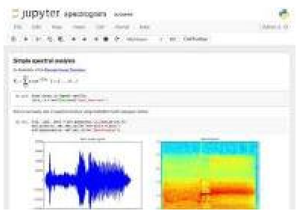
- Recent MPI supports GPU direct
- For direct communication on GPU memory
  - MPI_Send(DP, …) and MPI_Recv(DP, ….) can use pointers on device memory

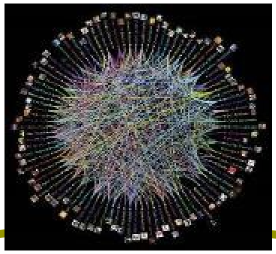


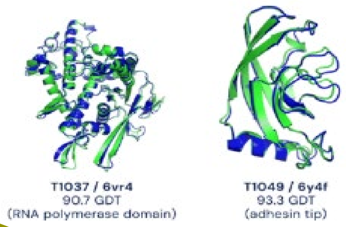
TSUBAME4.0 Supercomputer

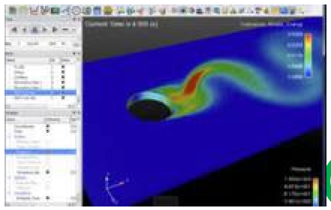
TSUBAME4.0 at Science Tokyo

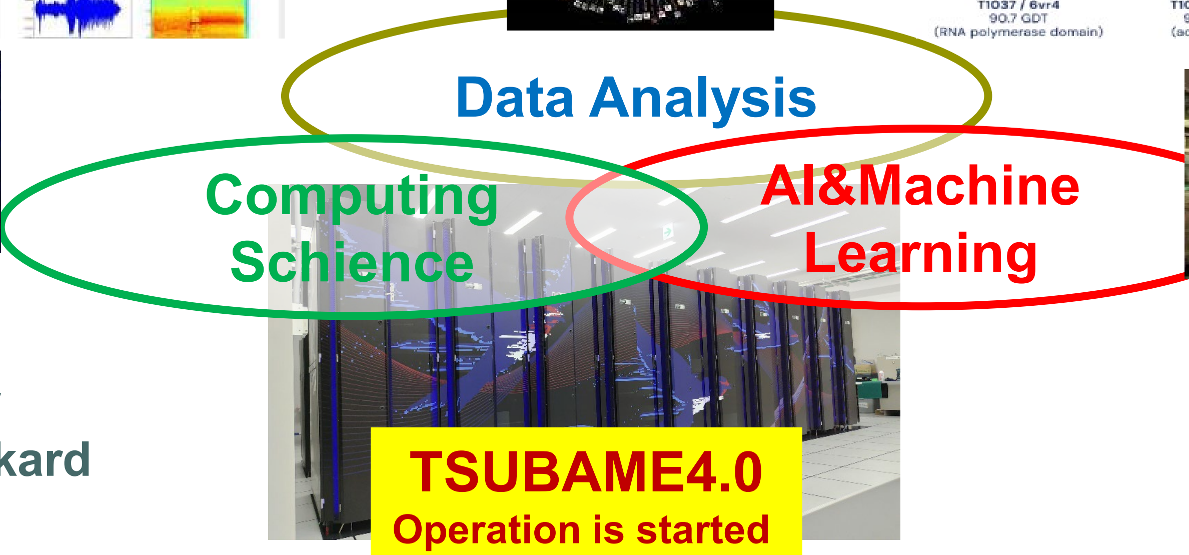
Supercomputer for Everybody for collaboration of Computing Science / Data Analysis / AI&Machine Learning

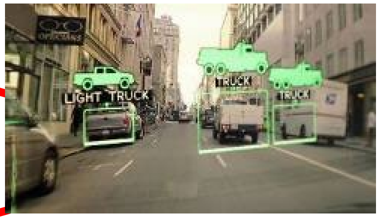












Integrated by
Hewlett Packard Enterprise

TSUBAME4.0
Operation is started in Apr 2024 !

TSUBAME4.0 Overview

Computing Nodes:
240 HPE Cray XD665 Nodes
(4x H100 + 2x 96-core EPYC)
Total computation speed:
- **66.8 PFlops** (FP64)
- **952 PFlops** (FP16 for AI)

Storage:
HPE Cray ClusterStor E1000

Total capacity:
- 44 PByte (Hard disk part)
- 327 TByte (SSD part)

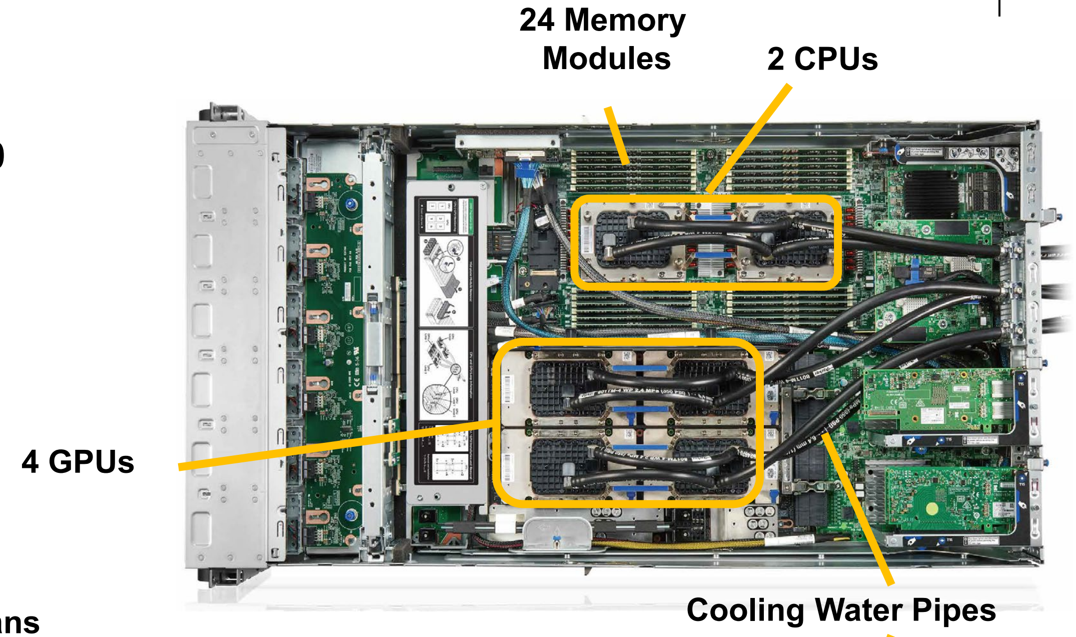
System in 30 racks
- Compute: 23 racks
- Storage&mgmt.: 7 racks

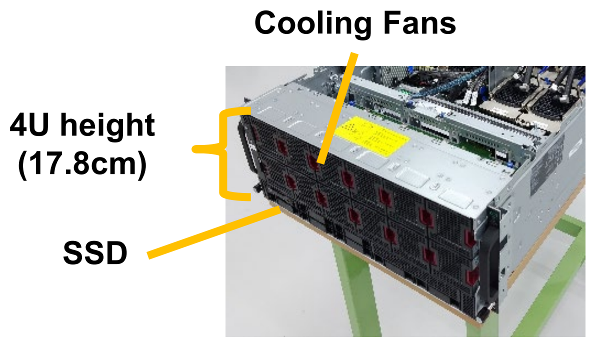
Installed in Suzukakedai campus, Science Tokyo

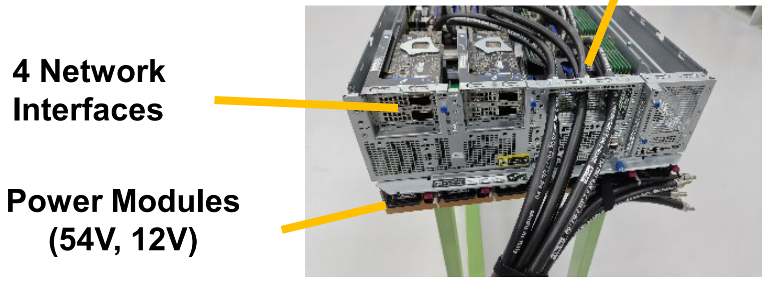
Integrated by HPE

TSUBAME4.0 Node – HPE Cray XD665 4U Server

CPU: 2x AMD EPYC 9654
96 cores, 2.4~3.55GHz
Memory: 24 x 32GiB DDR5-4800
768 GiB in total
GPU: 4x NVIDIA H100
SXM5 94GB HBM2e
Network: 4x InfiniBand NDR200
SSD: 1.92TB NVMe







TSUBAME4.0 Node Specifications

	TSUBAME3.0	TSUBAME4.0
CPU	Intel Xeon 2680v4 ×2	**AMD EPYC 9654 ×2**
• Clock, #cores	2.4GHz, 28 cores(=14×2)	**2.4GHz, 192 cores (=96×2)**
Main Memory	DDR3-2400 4ch×2	**DDR5-4800 12ch×2**
• Size	256GiB	**768GiB**
Network	OmniPath 100Gbps×4	**InfiniBand NDR 200Gbps×4**
OS	SUSE Linux Enterprise 12	**RedHat Enterprise Linux 8**
GPU	NVIDIA P100 SXM×4	**NVIDIA H100 SXM5 94GB HBM2e ×4 (*1)**
Specs per GPU:		
• Speed (FP64)	5.3TFlops	**66.9TFlops (Matrix (*2)), 33.4TFlops(Vector)**
• Mem Size	16GB	**94GB**
• Mem Speed	0.73TB/s	**2.39TB/s**

*1 H100 customized variant (memory size and speed differ from normal H100)
*2 Matrix performance is speed with Tensor Core units

Dynamic Node Partitioning

- Each TSUBAME node is fairly big (called fat-node), and each can be partitioned for each job
  - Several resource types are pre-defined
  - “0.5 GPU” means GPU partitioning with NVIDIA MIG facility

Resource type	CPU cores	Memory(GB)	GPUs
node_f	192	768	4
node_h	96	384	2
node_q	48	192	1
node_o	24	96	0.5
gpu_1	8	96	1
gpu_h	4	48	0.5
cpu_160	160	368	0
cpu_80	80	184	0
cpu_40	40	92	0
cpu_16	16	36.8	0
cpu_8	8	18.4	0
cpu_4	4	9.2	0

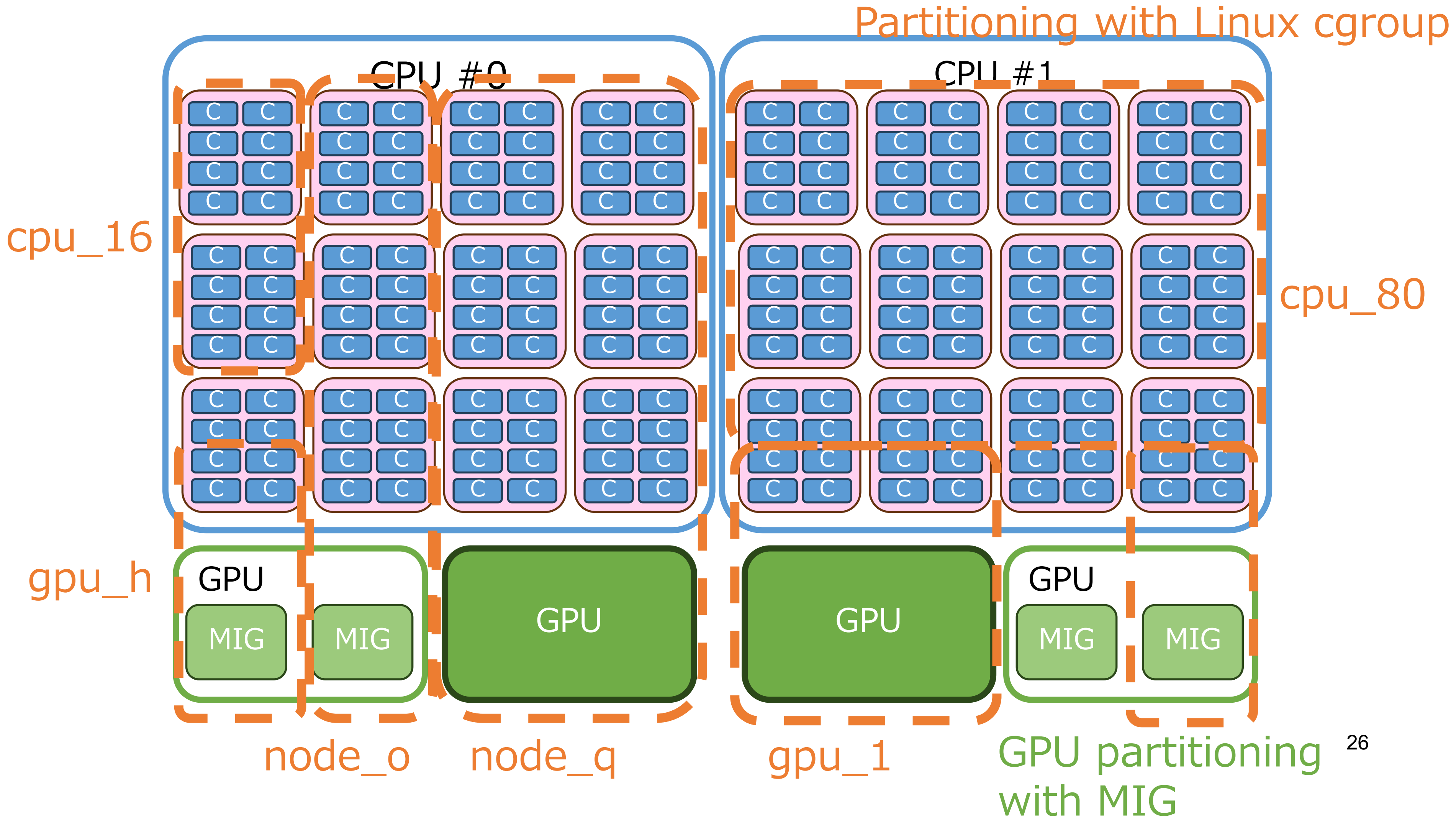
🡄 largest

🡄 same size with a partition of interactive node

🡄 smallest

Partitioning 192 Cores (2CPU) & 4GPU

To minimize interference among node partitions, topology of CPU/GPU connection, chiplets in EPYC CPUs, NUMA topology are considered



TSUBAME4.0 System Performance with 240 Nodes

	TSUBAME3.0(2017-2024)	TSUBAME4.0
Computational Performance		
• FP64 (double precision)	12PFlops	**66.8PFlops (5.5x) (Matrix operation)**
		34.7PFlops (2.8x) (Vector operation)
• AI Performance	47PFlops (FP16)	**952PFlops (20x) (FP16 Matrix)** **1900PFlops (FP8 Matrix)**
GPU Memory Bandwidth	1.56 PB/s	**3.07 PB/s (1.97x)**
Number of Nodes	540 Nodes (homogeneous config)	**240 nodes** **(homogeneous config)**
GPUs	2160 NVIDIA P100	**960 NVIDIA H100**
Cooling / Inlet Water Temperature	Free Cooling with Cooling Tower 32℃	**Chiller** **20℃**
Power Consumption (incl. cooling)	1080kW (Spec. value) 400~600kW(Operation)	**1820kW (Spec. value)** **450~800kW(Expected. Op.)**

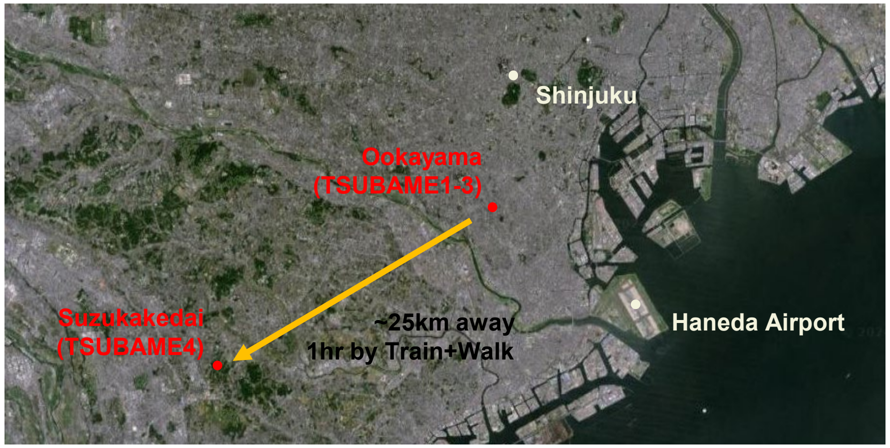
TSUBAME4.0 in World Supercomputer Ranking (as of Nov 2024)

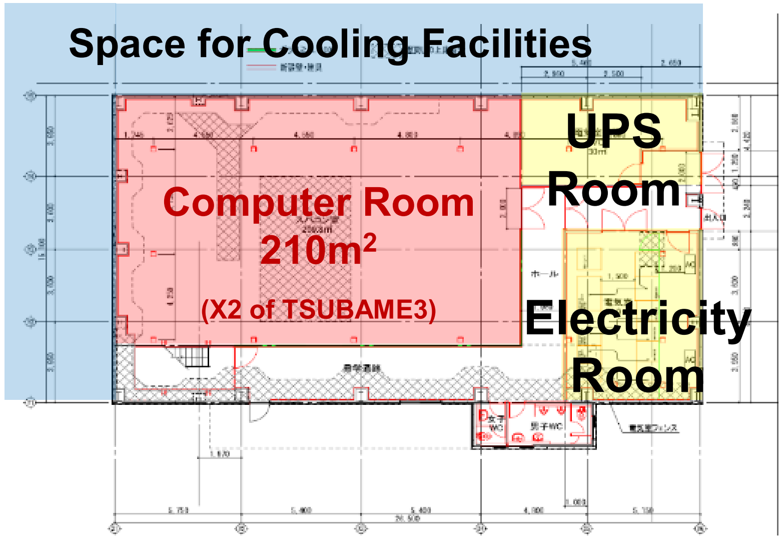
Ranking Name	Score	Rank
Top500	39.62 PFlops (59% of FP64 peak)	No. 36
Green500	48.565 GFlops/W	No. 30
HPCG	353.06 TFlops	No. 40
Graph500/BFS	5361.83 GTEPS	No. 16
GreenGraph500 BFS Big Data	13.60 MTEPS/W	No. 23
HPL-MxP	641.09 PFlops FP8 is used (34% of FP8 peak)	No. 6

Parallel benchmark programs were executed using the whole system
Such programs use MPI, CUDA, OpenMP...

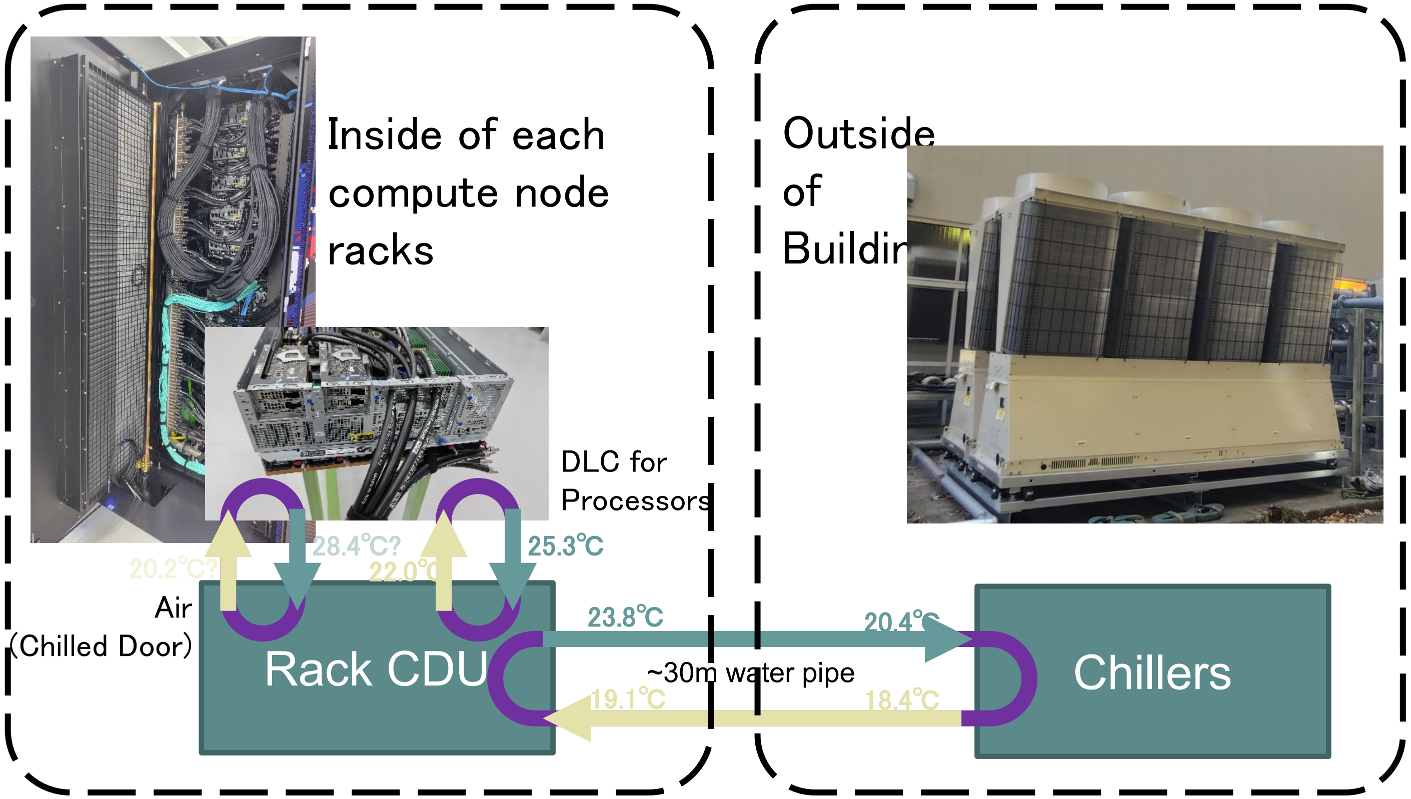
Data Center for TSUBAME4.0

- Renovated old experimental factory for power plant research in Suzukake-dai
- TSUBAME1, 2, 3 were in Ookayama





TSUBAME4 Cooling



- System is cooled by chillers
- GPUs/CPUs are cooled by water, other parts are cooled by air
  - 80~90% by water
  - Warm air is cooled by rack doors, with water pipes

- Thank you for participating in practical parallel computing

Today, we will go to the TSUBAME tour