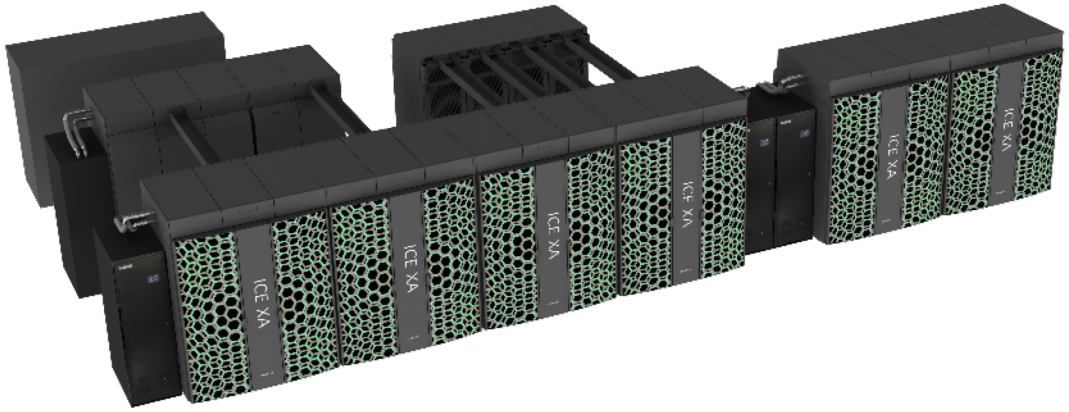
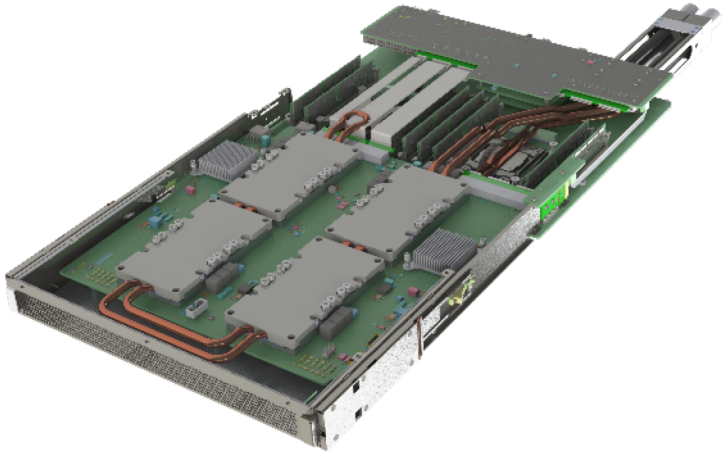
Current Status of TSUBAME3.0 Operation (Mar 2019)

Toshio Endo

Tokyo Institute of Technology





TSUBAME Series

2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

TSUBAME1

TSUBAME1.0
Opteron+ClearSpeed, 85TF Peak

TSUBAME1.1
ClearSpeed/ Storage added

TSUBAME1.2
680 NVIDIA S1070 added

TSUBAME2.0
Westmere + 4224 M2050, 2.4PF Peak

TSUBAME2.5
Upgraded to K20X
5.7PF Peak

TSUBAME2

experiment

TSUBAME-KFC → KFC/DL

Testbed with oil-immersion cooling

TSUBAME3.0
Broadwell + 2160 P100, 12PF Peak

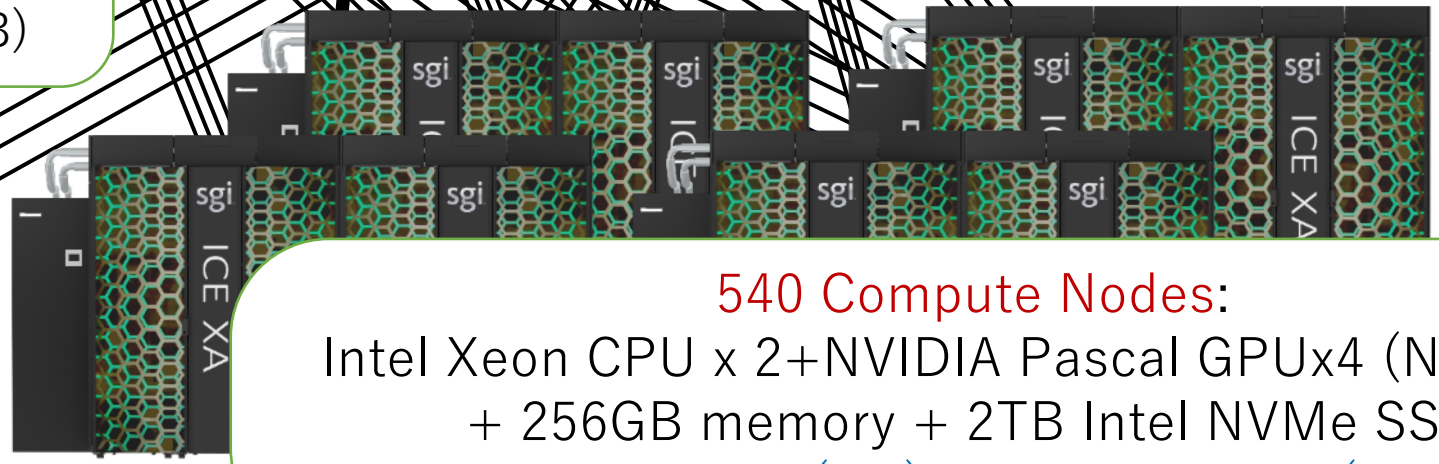
TSUBAME3

Overview of TSUBAME3.0
BYTES-centric Architecture, Scalability to all 2160 GPUs, all nodes, the entire memory hiearchy

Hewlett Packard Enterprise

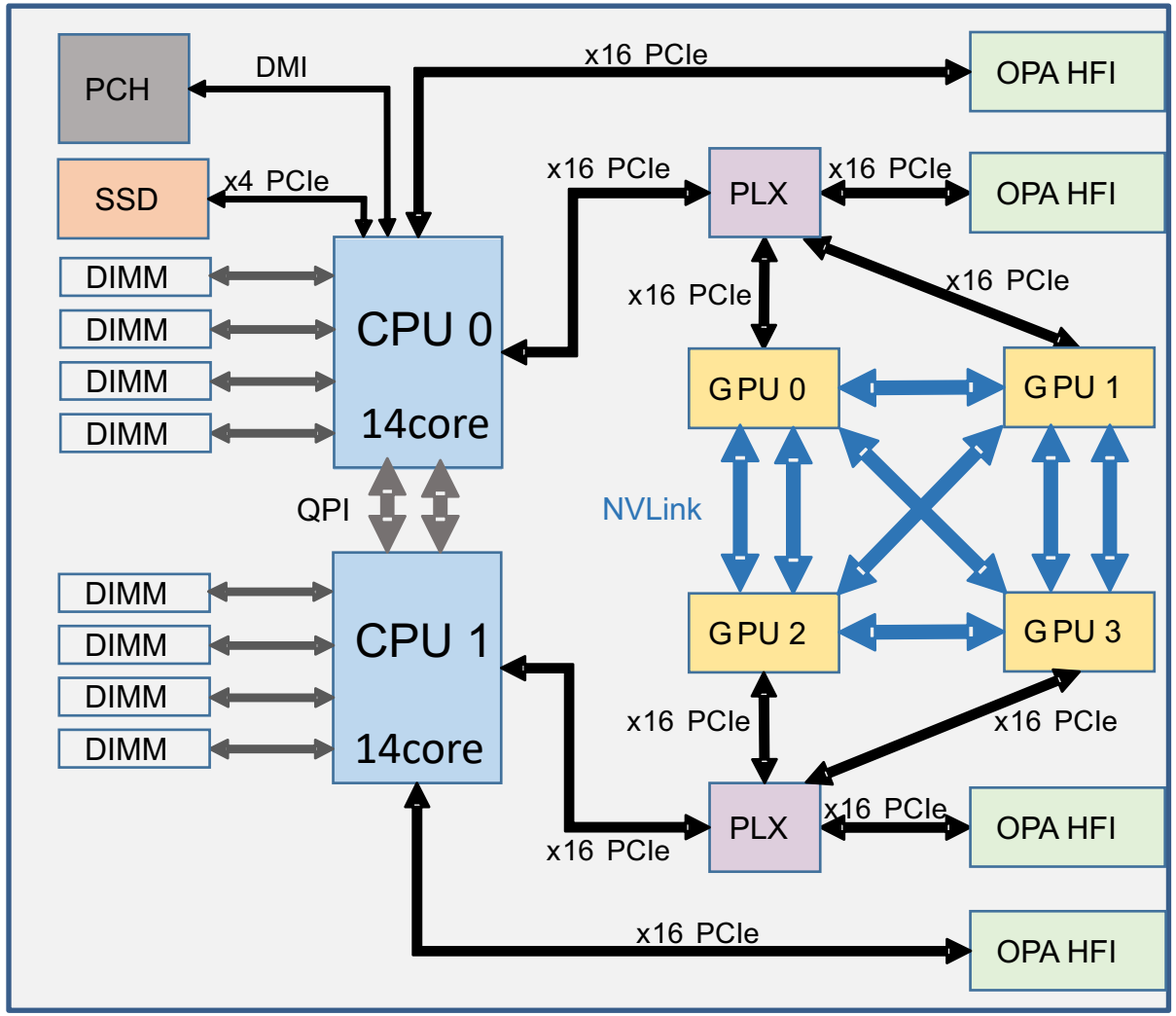
Full Bisection Bandwidth
Intel Omni-Path Interconnect. 4 ports/node
Full Bisection / 432 Terabits/s bidirectional
~x2 BW of entire Internet backbone traffic

DDN Storage
(Lustre FS 15.9PB+Home 45TB)



540 Compute Nodes:
Intel Xeon CPU x 2+NVIDIA Pascal GPUx4 (NV-Link)
+ 256GB memory + 2TB Intel NVMe SSD
12.1 Petaflops (DP) , 47.2 Petaflops (FP16)

TSUBAME3.0 Compute Node



x 540 nodes

Ultra high performance & bandwidth "Fat Node"

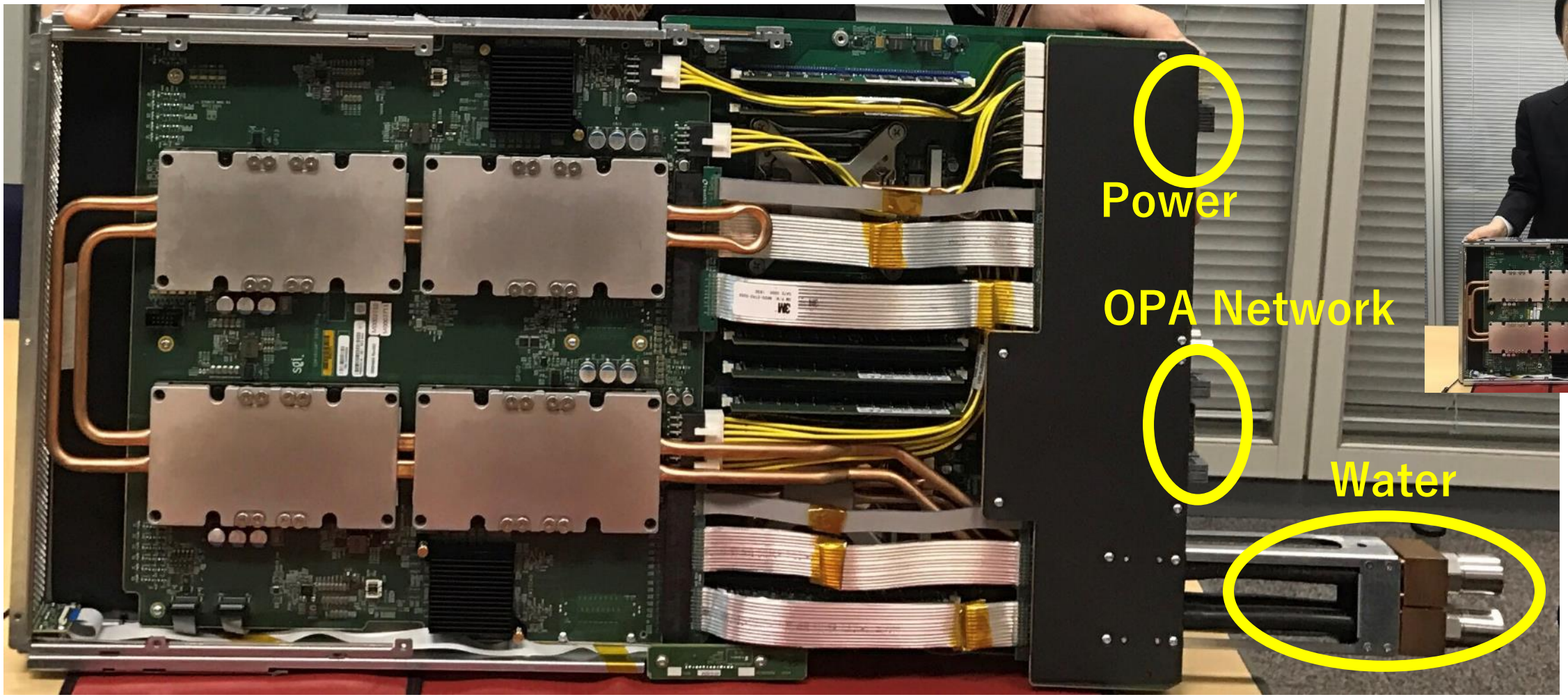
- High Performance:
  - 4 NVIDIA Pascal P100 (NVLink)
  - 2 Intel Broadwell 14-core Xeon
- High Network Bandwidth:
  - Intel Omnipath 100GBps x 4 = 400Gbps
- Memory Hierarchy for BD
  - 256GiB DDR4 memory
  - 2TB Intel NVMe SSD

  → > 1PB & 1.5~2TB/s system total
- Ultra High Density, Hot Water Cooled Blades:
  - 36 blades / rack = 144 GPU + 72 CPU

  → 50-60KW / rack

TSUBAME3.0 Node

- No exterior cable mess (power, NW, water)
- Plan to become a future HPE product



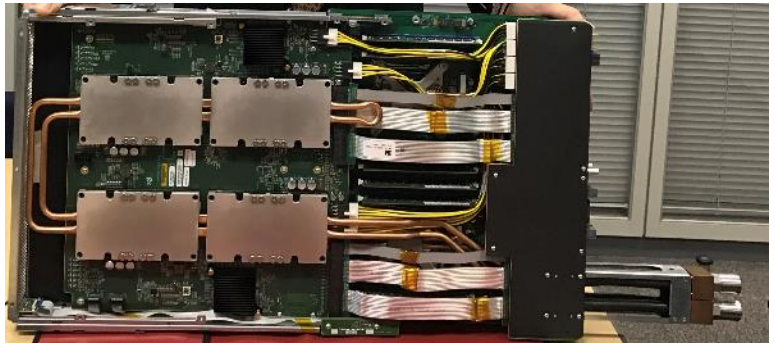
Main Topics

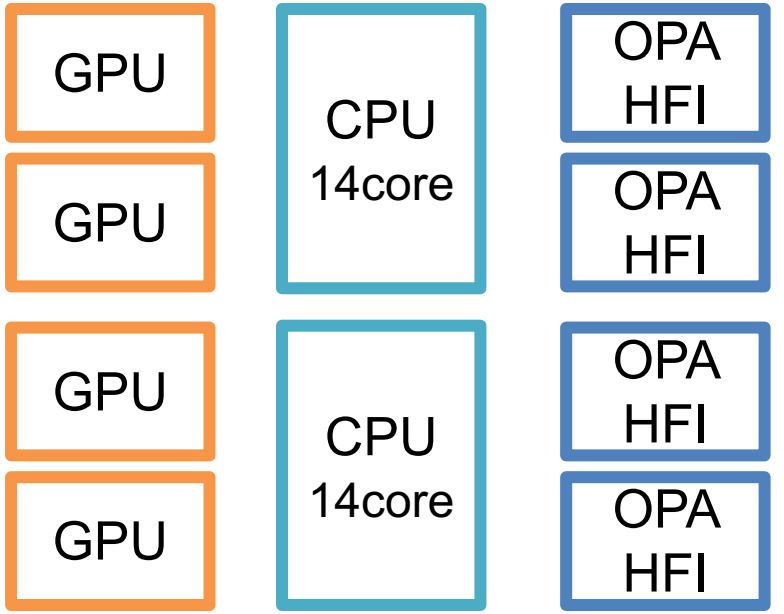
- *Efficient node usage*
  - TSUBAME3 nodes are rather “fat” → Node Partitioning
  - Usage statistics (New)
- *Software deployment for wider usage*
  - Statistics of users behavior
    - Collecting “module” command usage
  - Containers (New)
    - cf) “I want to use XYZ framework ver 123.4 instead of default version”
    - Now users can use Singularity
- *Hierarchical storage*
  - Lustre + temporal local file system on NVMe SSDs
  - BeeGFS for multi-node jobs

PART 1: NODE PARTITIONING

Node Partitioning: Motivation

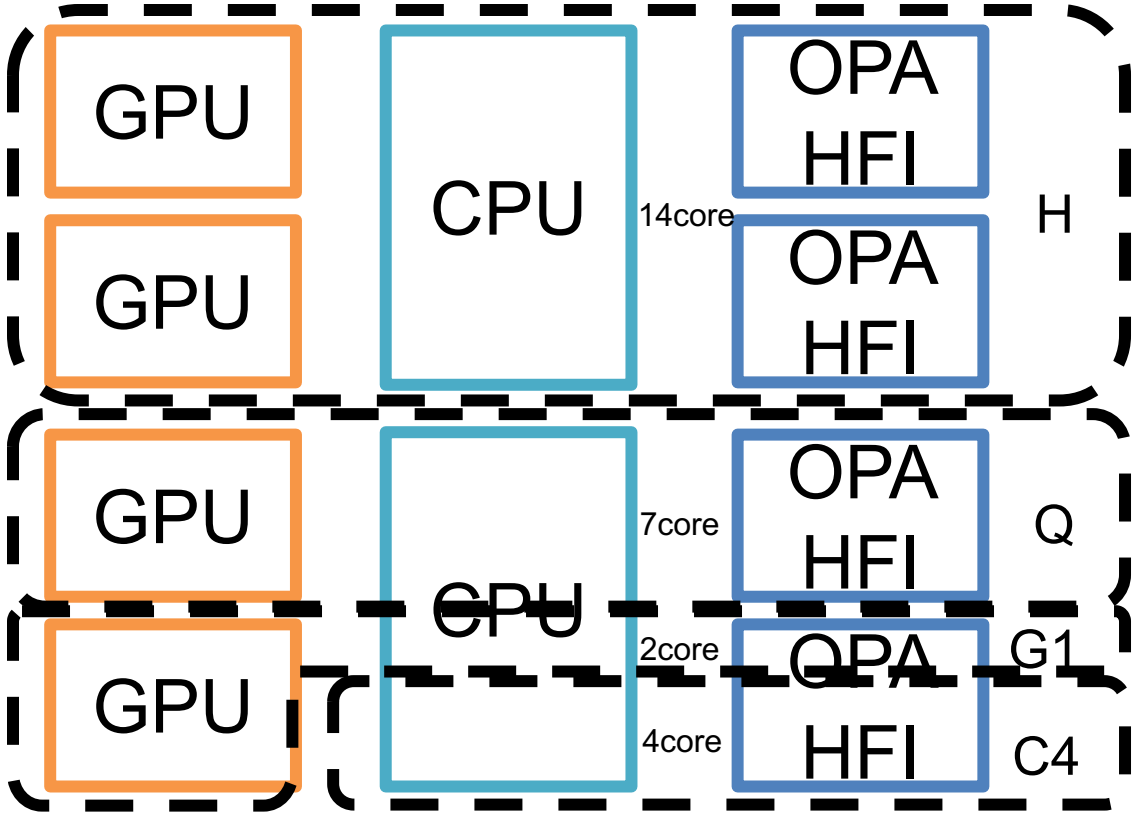
- We should support jobs with various resource requests
  - CPU centric jobs, GPU centric jobs…
  - Some users do “parameter survey” with plenty of 1-core jobs !
- Each TSUBAME3 node is “fat”, and partitioning is more important for efficient resource usage
  - 2 CPUs (28 cores) + 4 GPUs + 4 NICs
- On the other hand, “too flexible” partitions would introduce fragmentation of nodes





Node Partitioning: Overview

- We define several “resource types” on TSUBAME3
  - Like “instance types” on AWS



- F: A node without partitioning
- H: ½ node
- Q: ¼ node
- C1: 1 CPU Core
- C4: 4 CPU Core
- G1: 1 GPU + 2 CPU Core

Node Partitioning: Details

type	Resource type Name	Physical CPU cores	Memory (GB)	GPUs
F	f_node	28	240	4
H	h_node	14	120	2
Q	q_node	7	60	1
C1	s_core	1	7.5	0
C4	q_core	4	30	0
G1	s_gpu	2	15	1

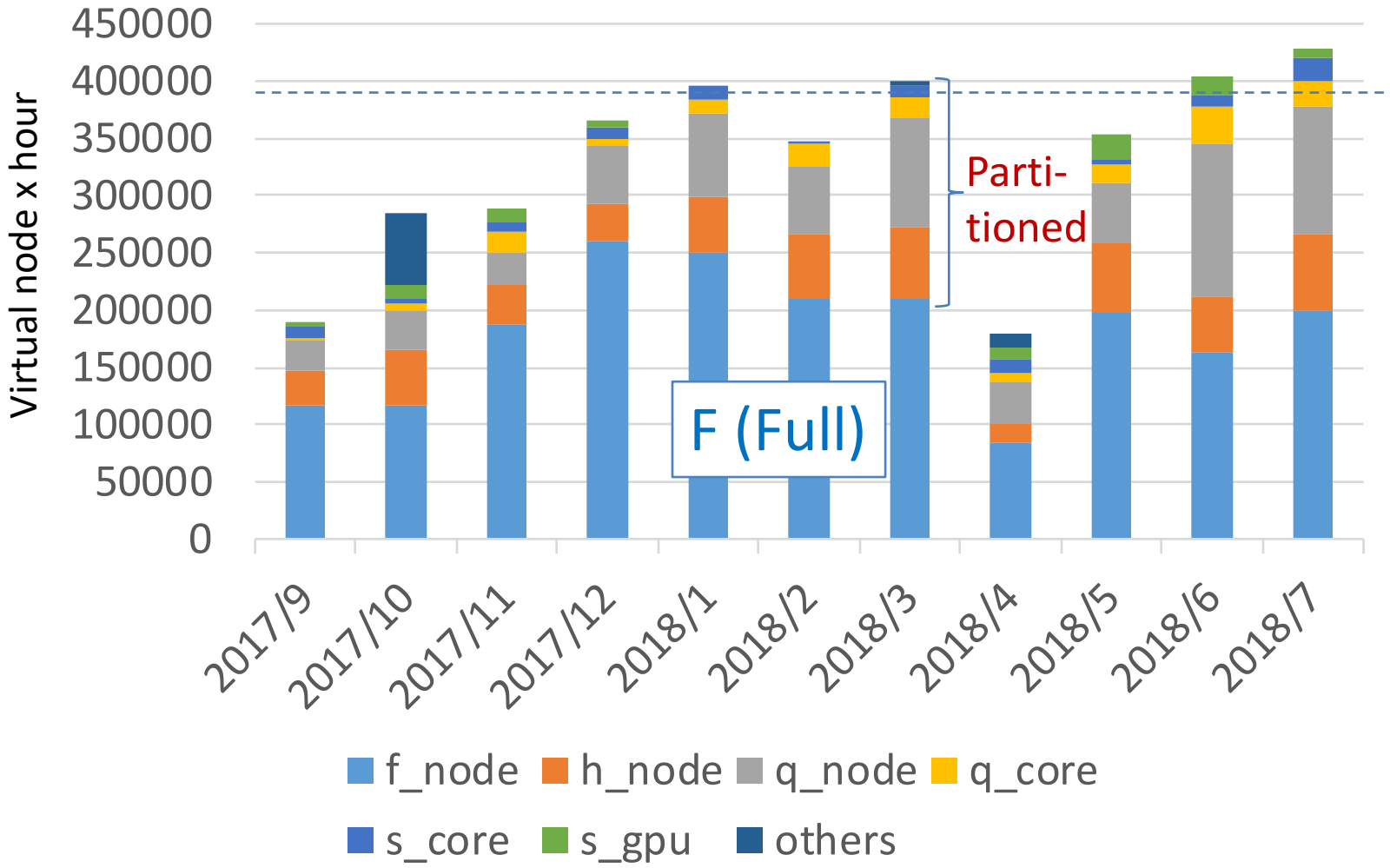
- Users choose “resource type” × “# of nodes” in job submissions
  - cf) 15 x Q, 10 x C4

- Implementation is done with cgroups in cooperation with UNIVA Grid Engine

  →CPU cores, GPU, memory, HCA are assigned to each resource correctly

  →Currently, a partition can see processes on other partitions ☹. Better isolation will be achieved with containers

Node Partitioning: Statistics



Accumulated (virtual nodes x hour) are shown
→It can exceed 388,800
= 540 nodes x 24h x 30d

In FY 2017, F (Full) occupied 60%
In FY 2018, F (Full) occupied 47%

Users tend to use node partition intensively after experiences

PART 2: USERS' APPLICATION DEPLOYMENT AND STATISTICS

How Users Deploy/Select Software Packages on TSUBAME3

1) The ‘module’ command for pre-installed packages

Programming tools

- Compilers: gcc, Intel, PGI
- MPI: OpenMPI, Intel, SGI MPT
- CUDA, JAVA, Python, R, MATLAB…

Pre-installed applications/libraries

- Caffe, Tensorflow, Chainer…
- HDF5, OpenFoam, cuDNN …
- Gromacs, ABAQUS, AMBER, ANSYS, Gaussian, LS-DYNA, NASTRAN…

2) Using Singularity containers (Since Aug 2018)

A simpler way to deploy non-supported packages

3) Manual installation by themselves

Issues to Capture Users Behavior on TSUBAME

- Capturing users behavior is important to improve the system operation
- However, especially for internal users, we have very limited information on their usage
  - Tokyo Tech members (staffs, students) can apply TSUBAME accounts easily from the web. No research reports currently
  - On the other hand, we collect research reports from non Tokyo Tech users

→ We started to count invocations of “module” command

“Modules” usage of TSUBAME3 (1)

- Stats on compute nodes Nov 10, 2018 – Mar 12, 2019
- Only modules pre-defined by system operators

Applications/DL frameworks

module name	# of unique users	# of loads
gromacs/2016.3	19	255008
amber/16_cuda	9	11859
chainer/4.3.0	7	10385
abaqus/2017	56	8787
imagemagick/7.0.6	4	7908
tensorflow/1.9.0	22	6902
gaussian16/B01	59	6641
gamess/apr202017r1	4	5035
amber/16up12_cuda	7	4888
namd/2.13	5	4171
amber/18up5	12	3892
gaussian16/A03	42	3143
amber/16up10_cuda	7	3005
amber/16	5	2494

⋮

- Currently, most popular package is “gromacs”
- Traditional HPC packages (MD/MO…) are always important
- DL workload is increasing

“Modules” usage of TSUBAME3 (2)

- Stats on compute nodes Nov 10, 2018 – Mar 12, 2019
- Only modules pre-defined by system operators

Compilers/system software/libs

module name	# of unique users	# of loads
cuda/8.0.61	373	1111742
intel/18.0.1.163	288	754674
openmpi/2.1.2	216	372239
intel-mpi/18.1.163	122	371581
python/3.6.5	59	250273
fftw/3.3.8	25	171597
cuda/9.1.85	60	125892
intel/17.0.5.239	23	98316
hdf5/1.10.1	69	98023
intel-mpi/17.4.239	6	93669
cudnn/6.0	57	76501
nccl/1.3.4	59	75541
vtk/6.1.0	42	72447
python-extension/2.7	41	72360
cudnn/7.0	27	57514
intel/17.0.4.196	63	55969
intel-mpi/17.3.196	49	55796

Note:

- The numbers may be larger than actual usage for module dependencies
  - Ex) “cuda” module is required to load “openmpi” module

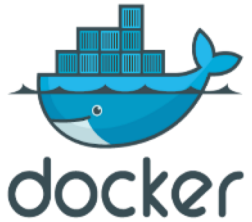
→ We should improve package configurations to reduce needless dependency

Motivation for Containers

- TSUBAME3 has many pre-installed software packages, but upgrading of DL frameworks is so rapid
- Some users may want to use the latest versions, while others use old versions
  - "pip --user install" everywhere?
  - It is troublesome for users; also there are package dependencies
  - It is not good for efficient storage usage

→We want to make such users use different images easier

Restriction: We CANNOT allow users to execute root on TSUBAME3



?

Container Operation on TSUBAME3

We are providing “singularity” container for users

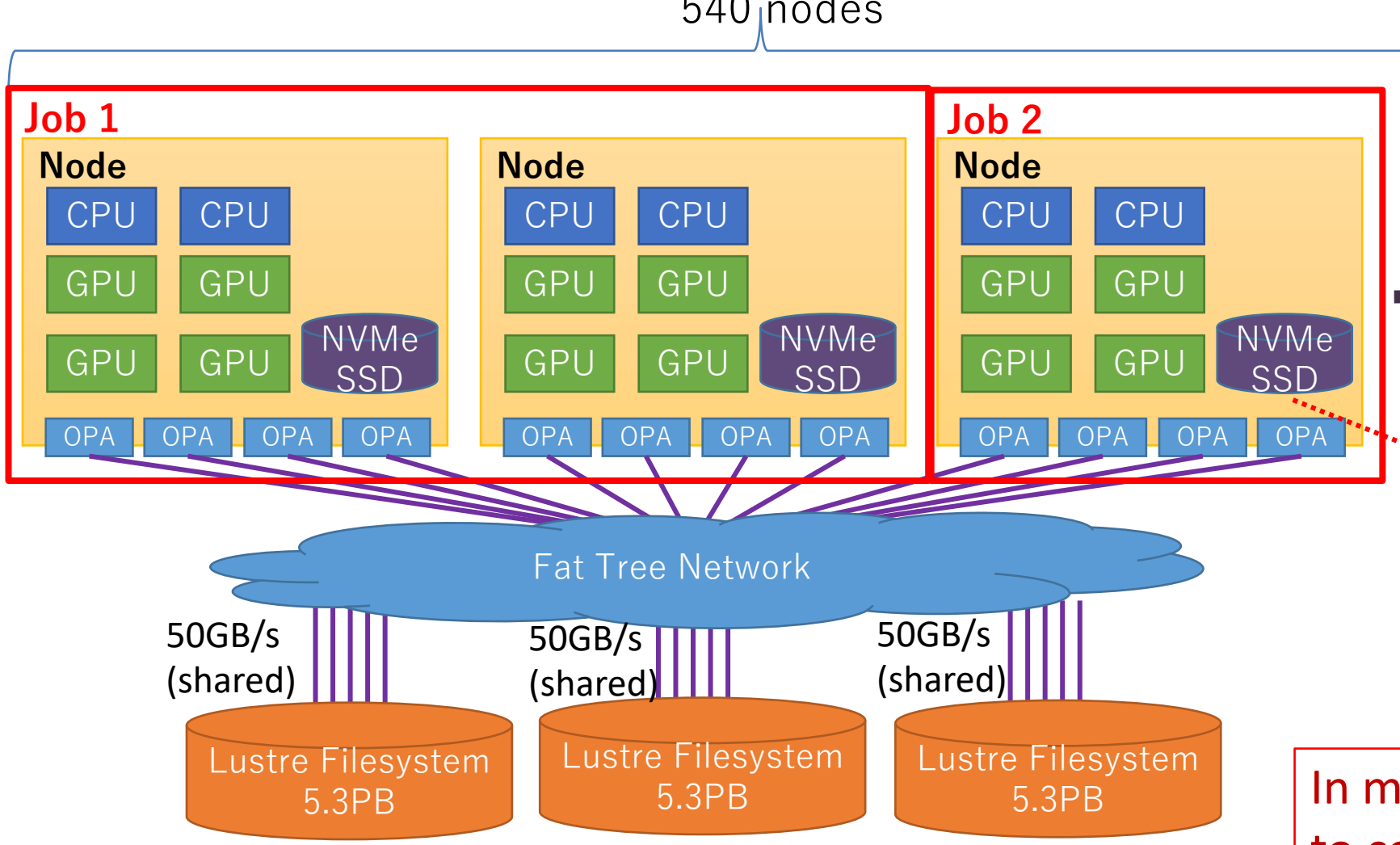
- It works with the UNIVA Grid Engine scheduler
  - It is realized in cooperation with HPE and UNIVA
- It works with node partitioning
- Currently users have to prepare their container images
  1) Download from web (ex: docker hub)
  2) Make own images on their local machines
  3) Several pre-defined images (planning)

A comment from a bio-informatics user:
“Software deployment by container technology is mandatory for us, since we daily use container images from BioContainers.pro. ☺”

A new issue for administrators:
We can capture the usage of “singularity” module, but we do NOT know what packages are used inside the containers ☹

PART 3: OPERATION OF HIERARCHICAL STORAGE WITH LUSTRE + SSDS

TSUBAME3 Hierarchical Storage



The largest storage is shared Lustre filesystems (5.3PB x 3)

Users can also use node local SSDs for temporary data

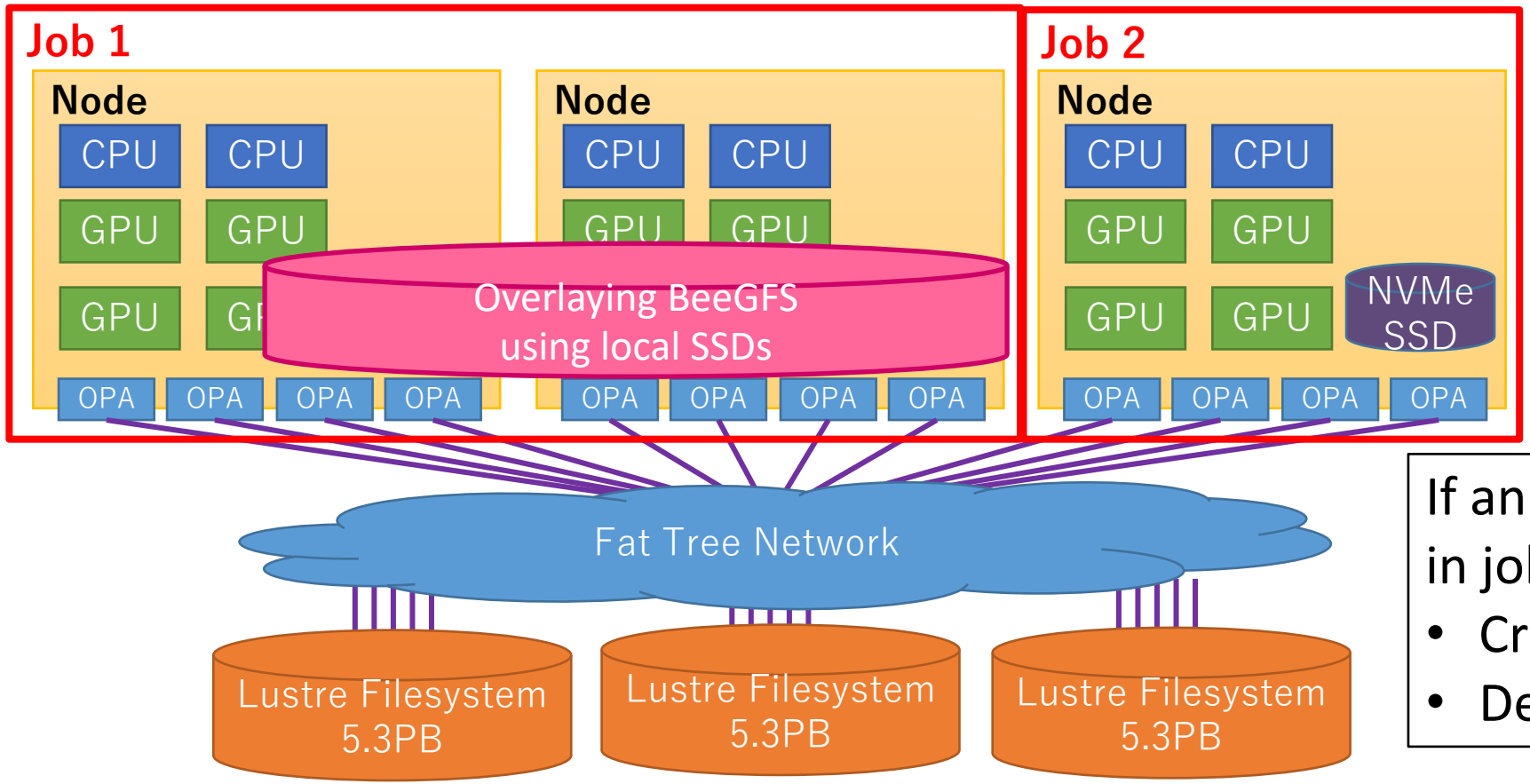
- Size 2TB
- Read 2.8GB/s
- Write 2.0GB/s

→ More stable performance

In multi-node jobs, users have to consider node affinity

BeeOND Service

- Users can use BeeOND (BeeGFS On Demand)
- Nodes in a single job can use a single temporary FS



If an user sets “BeeOND” flag in job submission, a BeeGFS is

- Created in job prologue
- Destroyed in job epilogue

Current BeeOND Operation

- Currently only jobs with F resource type (no partition) can use BeeOND

An example of job script

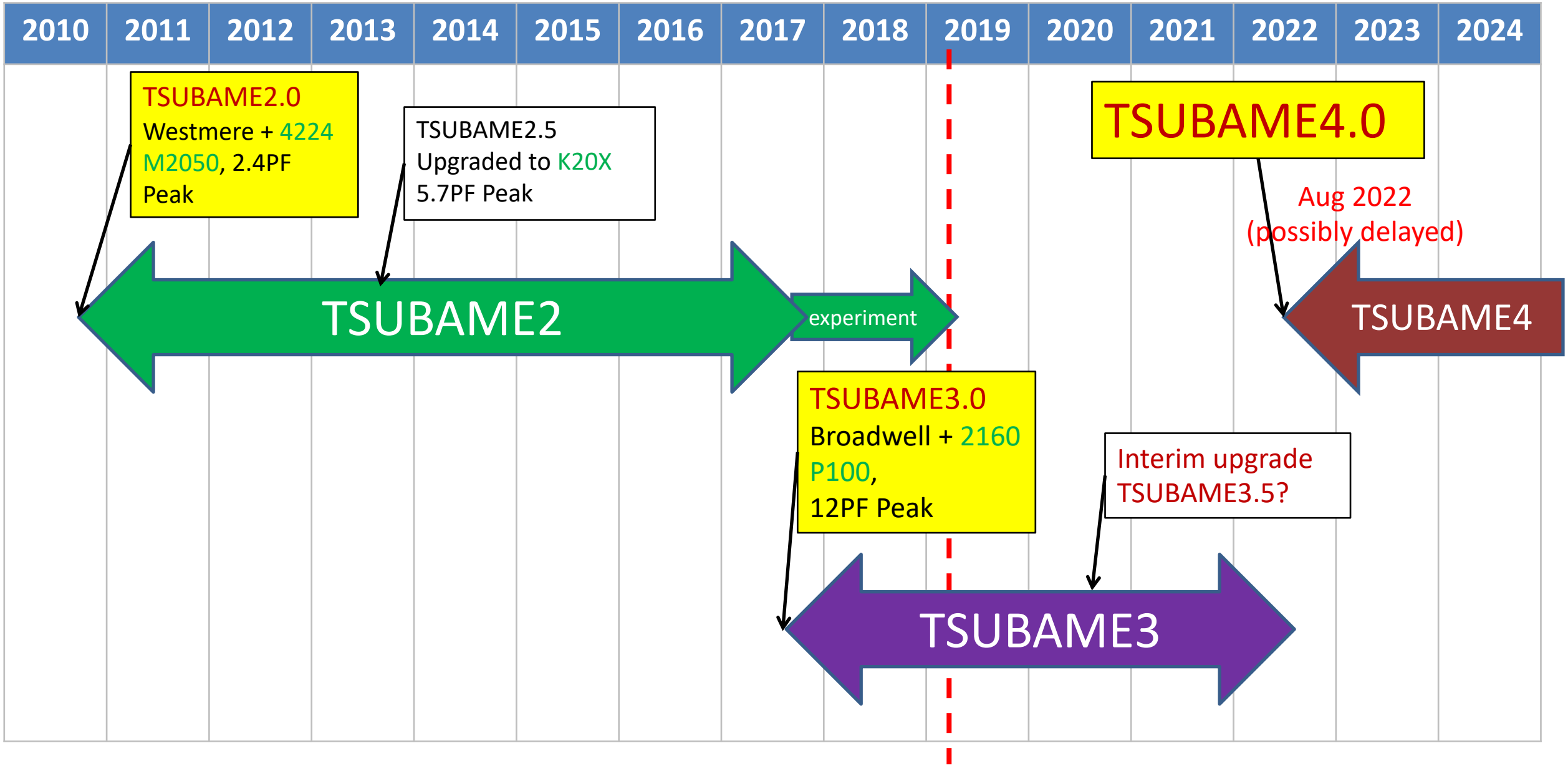
```
#!/bin/sh
#$ -cwd
#$ -l f_node=4
#$ -l h_rt=1:00:00
#$ -N flatmpi
#$ -v USE_BEEOND=1
. /etc/profile.d/modules.sh
module load cuda
module load intel
module load intel-mpi
mpiexec.hydra -ppn 8 -n 32 ./a.out
```

Usage: Dec 2018 -- Feb 2019

#nodes	# of jobs with BeeOND
1	560
2	388
3 -- 4	146
5 -- 8	59
9 -- 16	43
17 -- 32	3
33 -- 64	7
Total	1206

# of all F-type jobs (Dec 18-Feb19): 149,281

Schedule towards Next-Gen TSUBAME



Thank you for your attention

Toshio Endo
endo@is.titech.ac.jp