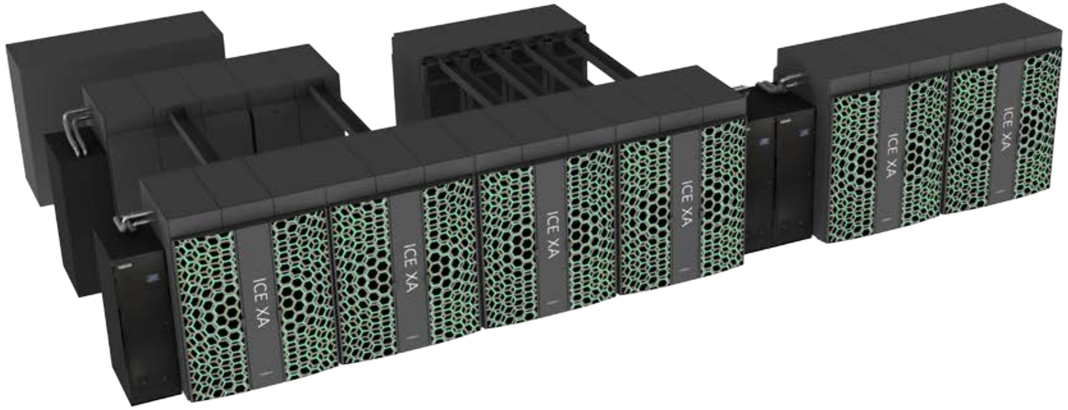
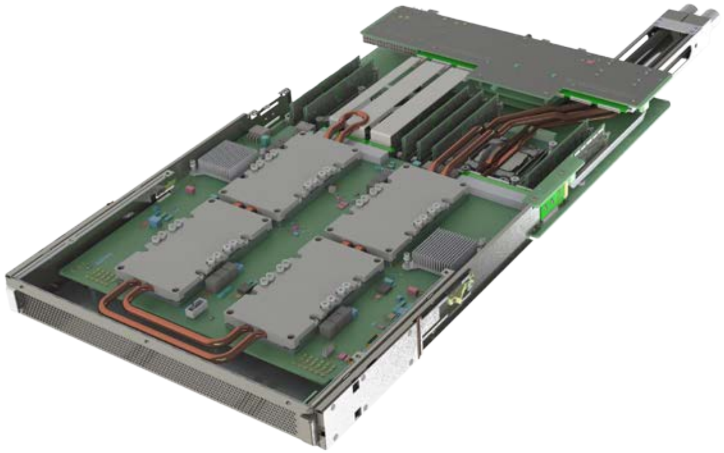
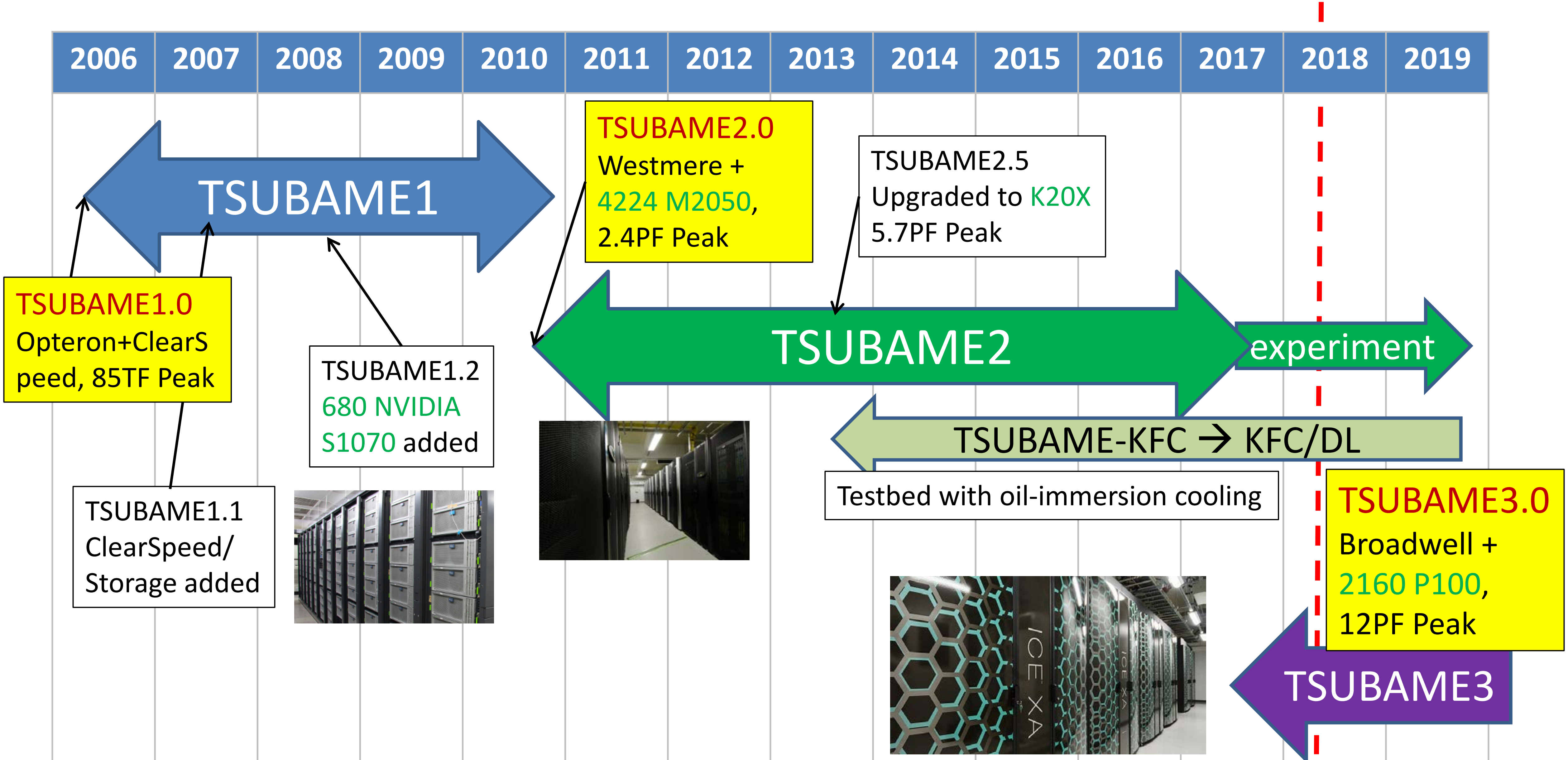
Current Status of TSUBAME3.0 Operation

Toshio Endo
Tokyo Institute of Technology





TSUBAME Series



Design Concepts of TSUBAME3.0

To inherit and improve advantages of TSUBAME1/2

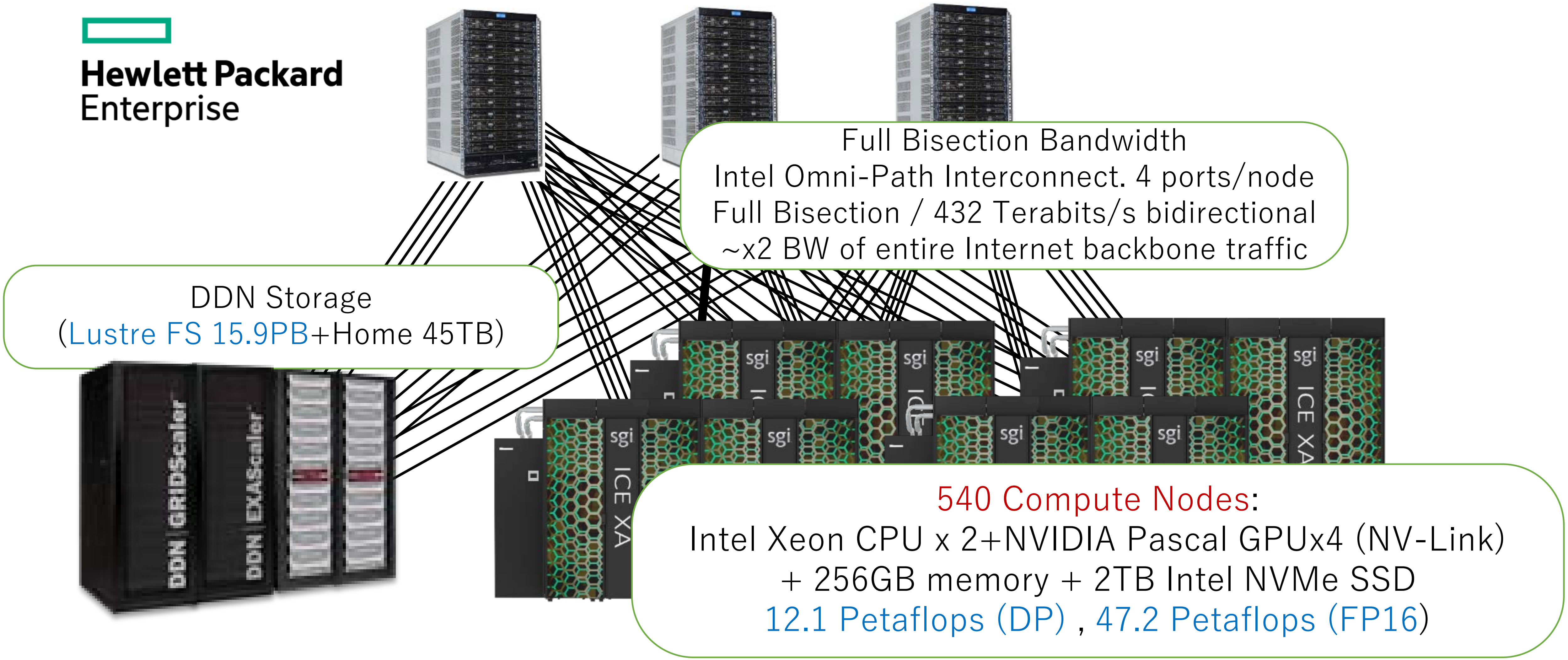
- ***<u>"Everybody's Supercomputer"</u>***:
  - Use commodity Intel/Linux to harness broad software assets
- ***<u>"Green Supercomputer":</u>***
  - Use GPGPUs for extremely higher Flops/Watt ratio
  - Modern energy-saving cooling facility → Warm water cooling in TSUBAME3.0

  Green500 World No.1 in June 2017
- ***<u>"Cloud Supercomputer":</u>***
  - VMs for flexible operation → Cgroups/container in TSUBAME3.0

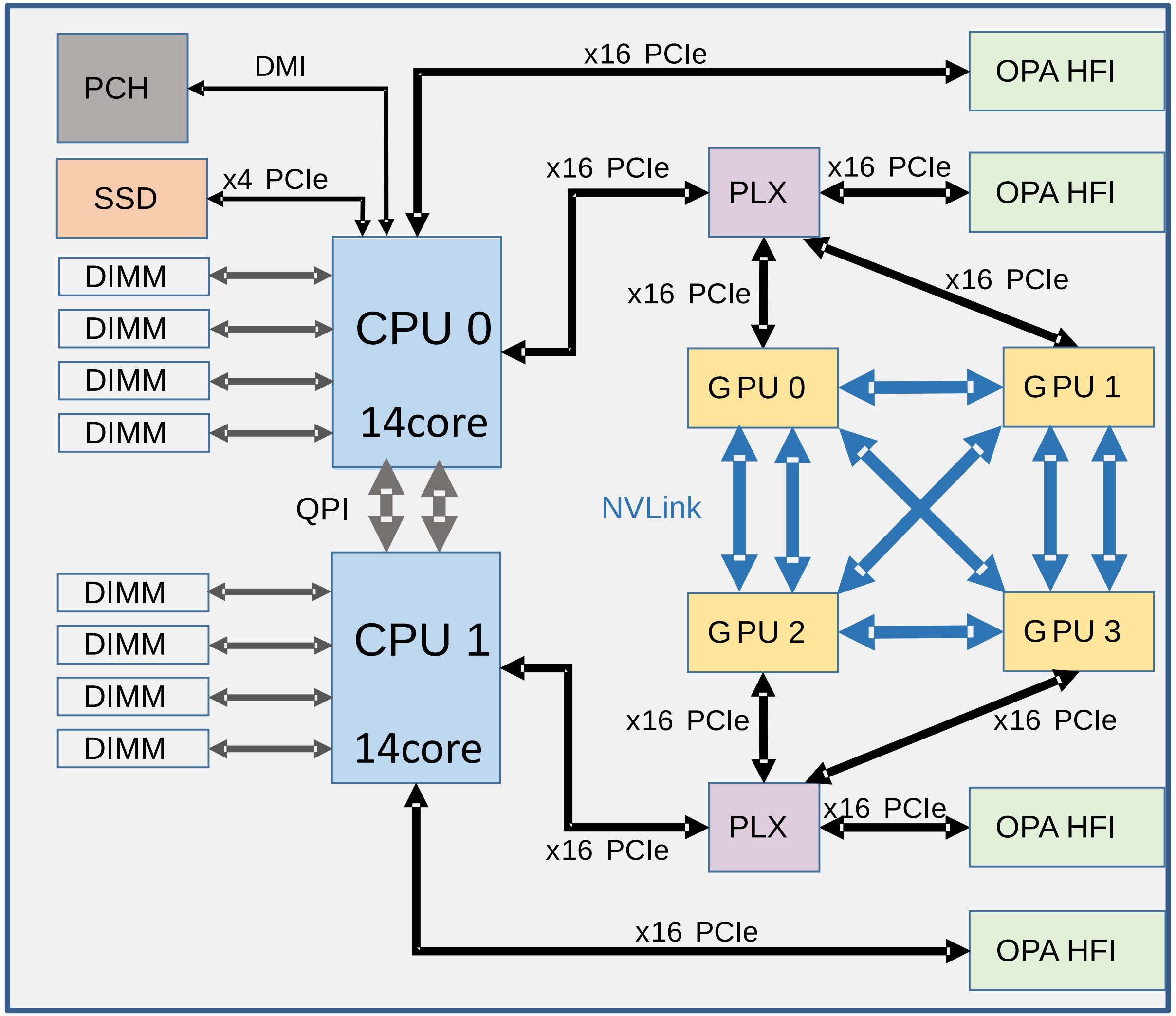
As a new concept in TSUBAME3.0

- ***<u>"Supercomputer for Convergence of BigData & HPC":</u>***
  - Bandwidth centric design (CPU⇔GPU, GPU⇔GPU, Node⇔Node…)
  - High FP16 performance for deep learning
  - Software stack for learning apps

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



TSUBAME3.0 Compute Node



x 540 nodes

Ultra high performance & bandwidth "Fat Node"

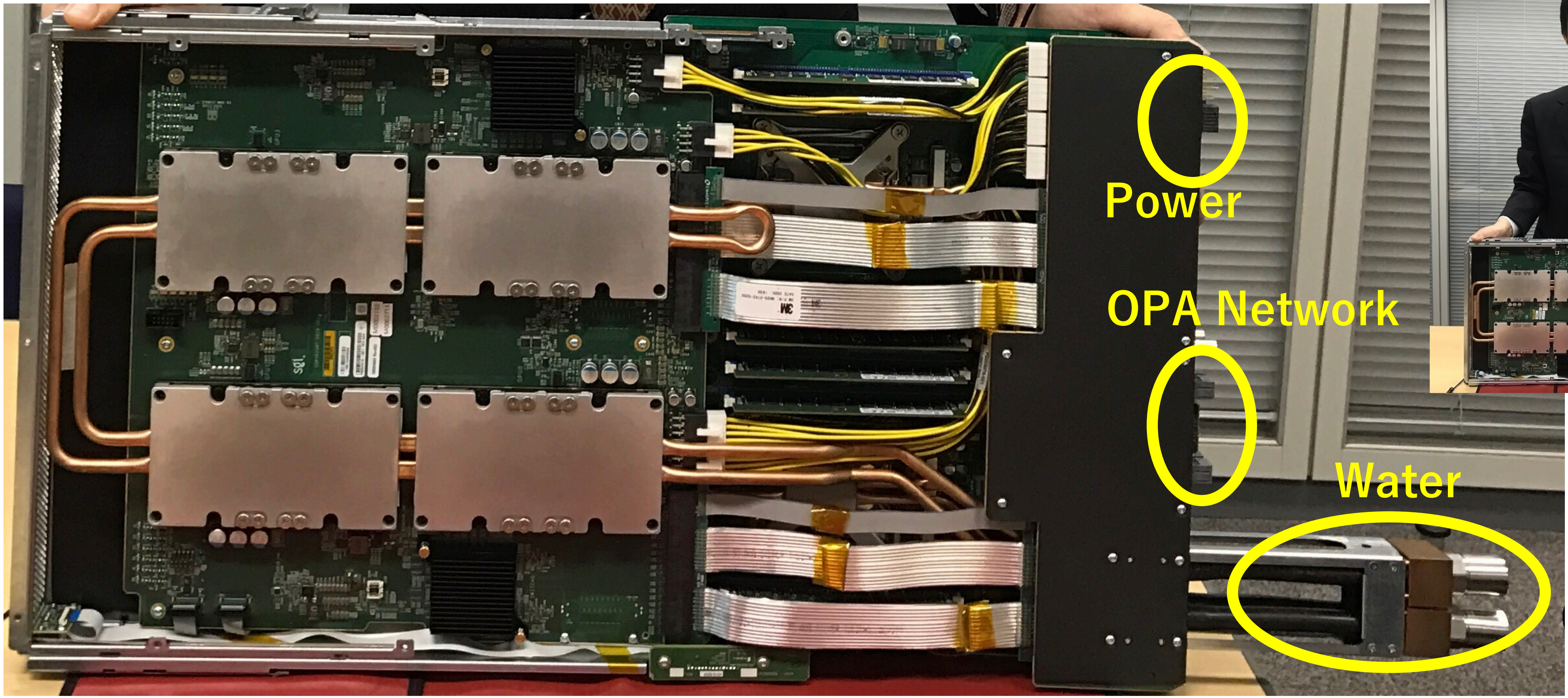
- High Performance:
  - 4 NVIDIA Pascal P100 (NVLink)
  - 2 Intel Broadwell 14-core Xeon
- High Network Bandwidth:
  - Intel Omnipath 100GBps x 4 = 400Gbps
- Memory Hierarchy for BD
  - 256GiB DDR4 memory
  - 2TB Intel NVMe SSD

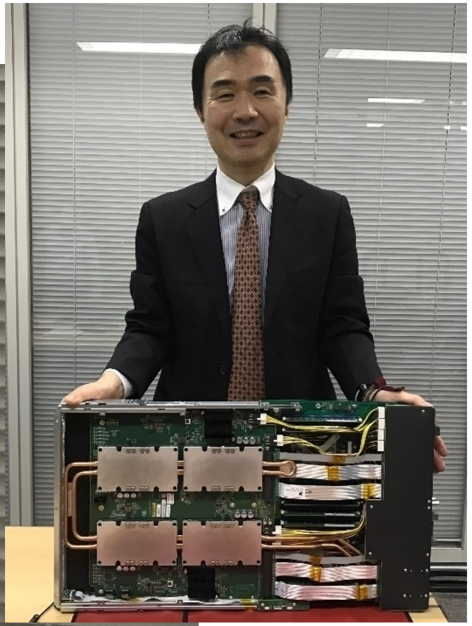
  → > 1PB & 1.5~2TB/s system total
- Ultra High Density, Hot Water Cooled Blades:
  - 36 blades / rack = 144 GPU + 72 CPU

  → 50-60KW / rack

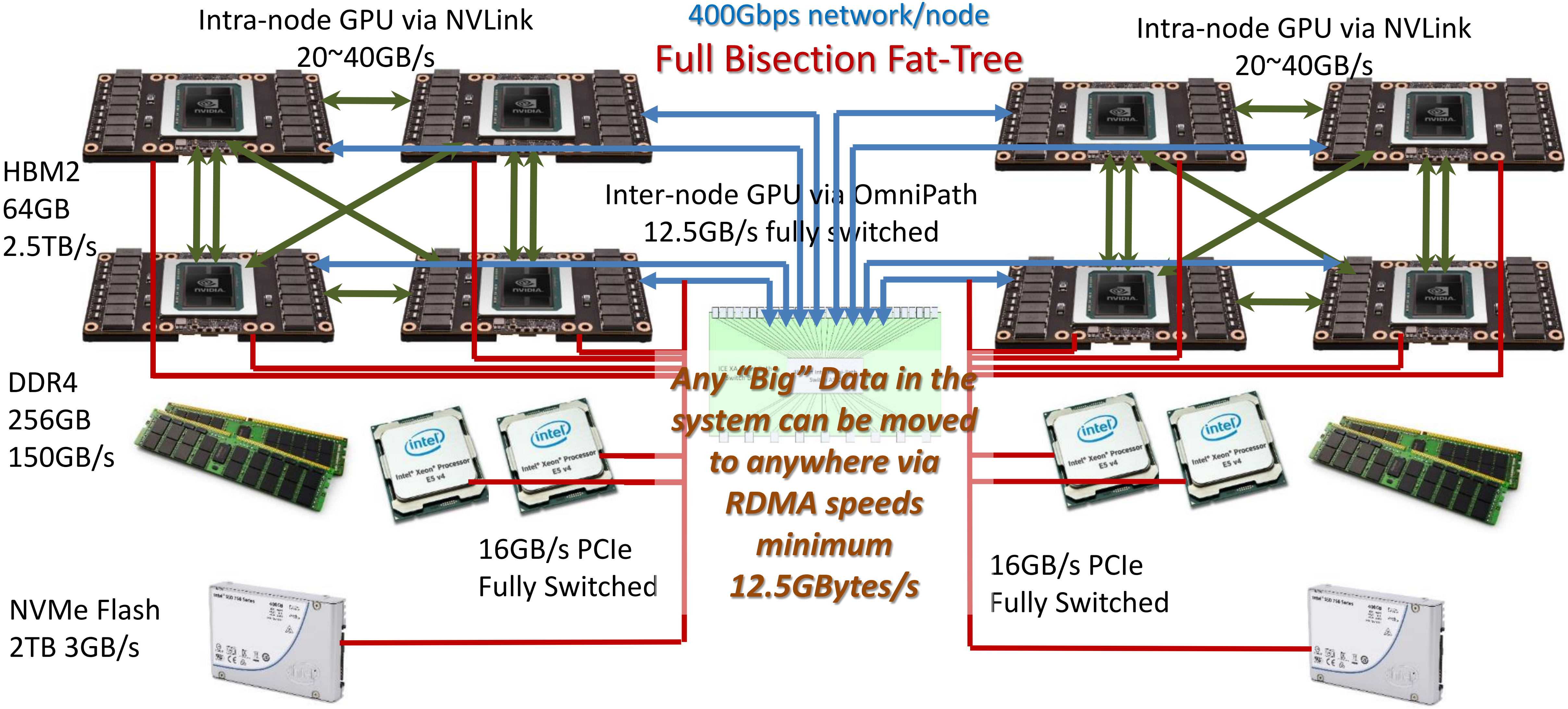
TSUBAME3.0 Node

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





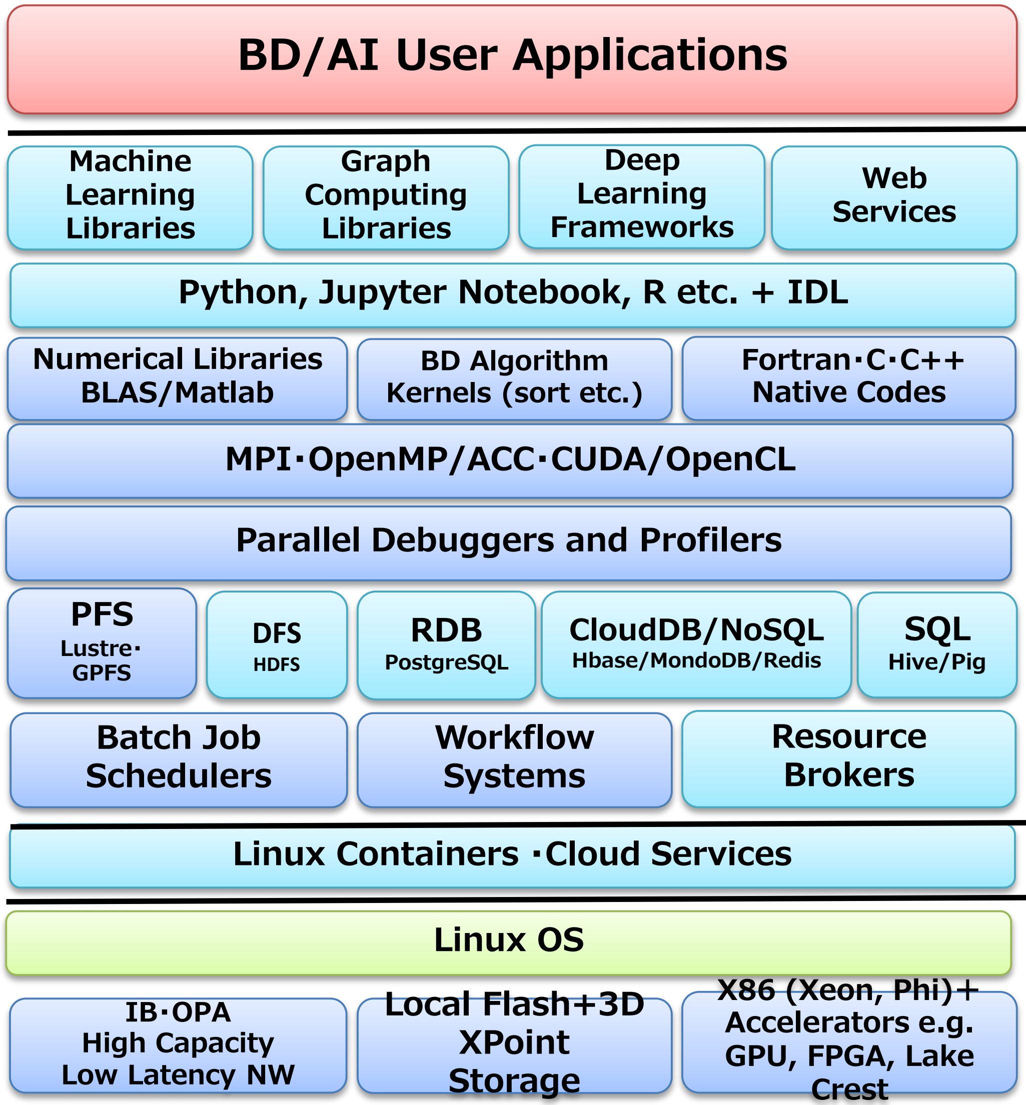
TSUBAME3: *A Massively BYTES Centric Architecture* for Converged BD/AI and HPC



~2.3 Terabytes/node Hierarchical Memory for Big Data / AI (c.f. K-compuer 16GB/node)

➔ *Over 1 Petabytes in TSUBAME3*

Goal: Software Stack towards BigData/AI System



Application

- ✓ Easy use of various ML/DL/Graph frameworks from Python, Jupyter Notebook, R, etc.
- ✓ Web-based applications and services provision

System Software

- ✓ HPC-oriented techniques for numerical libraries, BD Algorithm kernels, etc.
- ✓ Supporting long running jobs / workflow for DL
- ✓ Accelerated I/O and secure data access to large data sets
- ✓ User-customized environment based on Linux containers for easy deployment and reproducibility

OS

Hardware

- ✓ Modern supercomputing facilities based on commodity components

Current TSUBAME3.0 Software

System Software

- OS: SUSE Linex Enterprise Server (SLES)12SP2
- Job scheduler: Univa Grid Engine
- Container: Docker (plan)
- Shared file system: Lustre

Programming tools

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

Pre-installed packages/libs

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

Topics in Resource Management

- *Fee payment by TSUBAME point system*
- *Node partition*
  - TSUBAME3 nodes are rather “fat”
    - 2 CPUs (28 cores) + 4 GPUs
- *Advanced reservation*
  - cf) “I want to use 30 nodes during 13:00 to 18:00 tomorrow”
- *Container usage* (available soon)
  - cf) “I want to use XYZ framework ver 123.4 instead of default version”

TSUBAME Point System

- Each user group (TSUBAME group) buys "TSUBAME points" as pre-paid points
- Each TSUBAME group may be a laboratory, a joint-research group, etc.
  - A TSUBAME group is implemented as a Linux user group
  - A user may participate in several groups
- TSUBAME points are used for:

(1) Job execution

(2) Capacity of shared Lustre storage

- In job submission, the user specifies the group
  - % qsub –g TGA-XYZ ./job.sh

→ Points of TGA-XYZ group are consumed

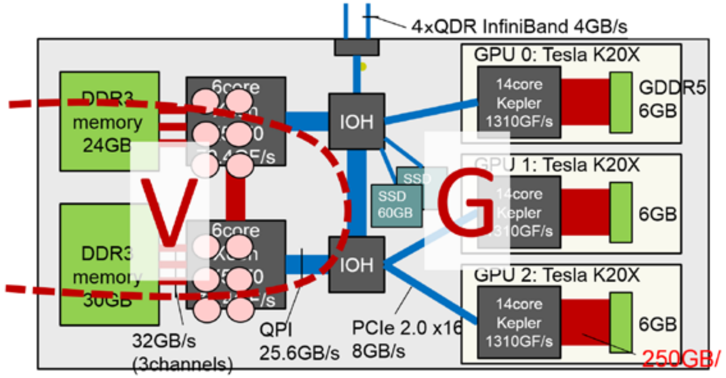
Approx. fee per [1 node x 1hour]
(= ~3600 points)

Group Attribute	Price in JPY (EUR)
Group inside Tokyo-Tech	¥25 (~€0.2)
Academic	¥100 (~€0.8)
Industry (results are public)	¥100 (~€0.8)
Industry (results are not public)	¥200 (~€1.6)

Node Partitioning (1): Motivation

- We should support jobs with various resource requests, while keeping resource utilization high:
  - CPU centric jobs, GPU centric jobs…
  - Some jobs require only 1 CPU core
- TSUBAME3 nodes are "fat", and partitioning is more important
  - 2 CPUs (28 cores) + 4 GPUs + 4 NICs
- On the other hand, "too flexible" partitions would introduce fragmentation of nodes

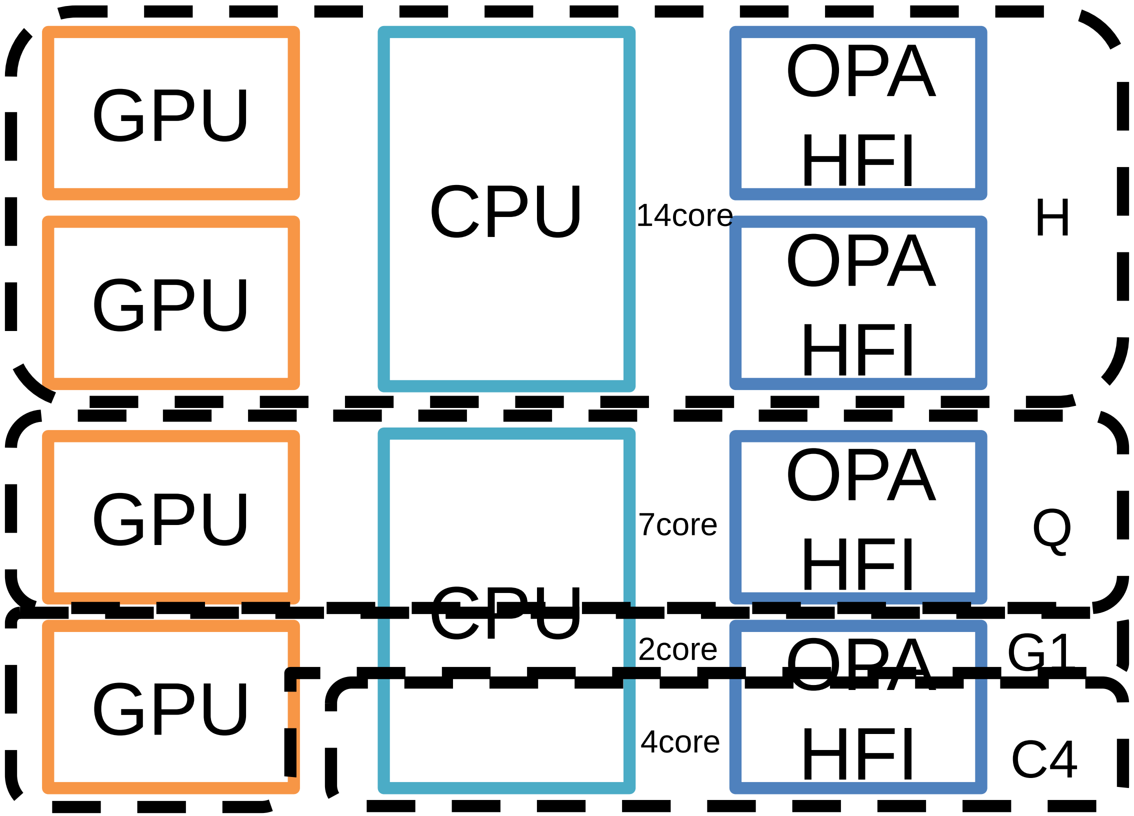
In TSUBAME2



- Partitioning was done statically ☹
  - 3 GPUs were not divided
- "V partition" was a VM
  - hard to use GPUs ☹
  - lower performance ☹

Node Partitioning (2)

- We define several “resource types”



- 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 (3)

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 should choose a resource type that is sufficient for job's resource requirement
  - cf) A job that uses 5 cores → type Q
  - cf) A job that uses 1 core + 100GB memory → type H
- An MPI execution consists of resources with uniform type
  - cf) 15 x Q → OK, 10 x C4 → OK, 1 x F + 5 x Q → NG
- 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

Advanced Reservation (Apr 2018--): Motivation

- Major part of nodes are used by batch jobs
- On the other hand, several users want to use nodes during a specific timeframe → advanced reservation (AR) facility
  - cf) "We want to reserve 30 nodes during 13:00 to 18:00 tomorrow"

→ Users can use the nodes like a "private cluster"

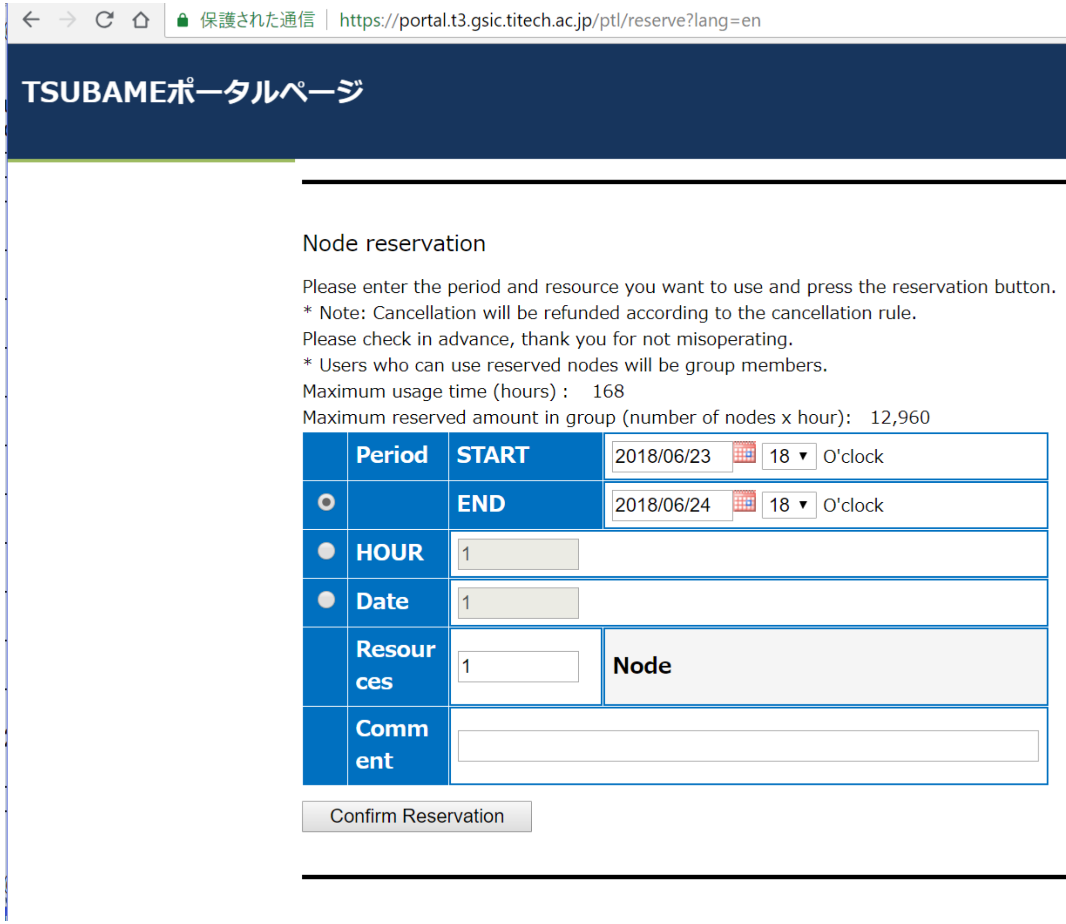
In TSUBAME2

- Node set for reservation was static ☹
- Duration of reservation was "daily basis", like hotels

Advanced Reservation (2)

- In TSUBAME3, AR is implemented seamlessly on UNIVA Grid Engine
- A user reserves nodes via Web interface
  - Currently, AR does not work wih node partition. User specifies # of physical nodes

→If the reservation succeeds
  - TSUBAME points of the group are consumed
  - The user obtains a “AR_ID”



Advanced Reservation (3)

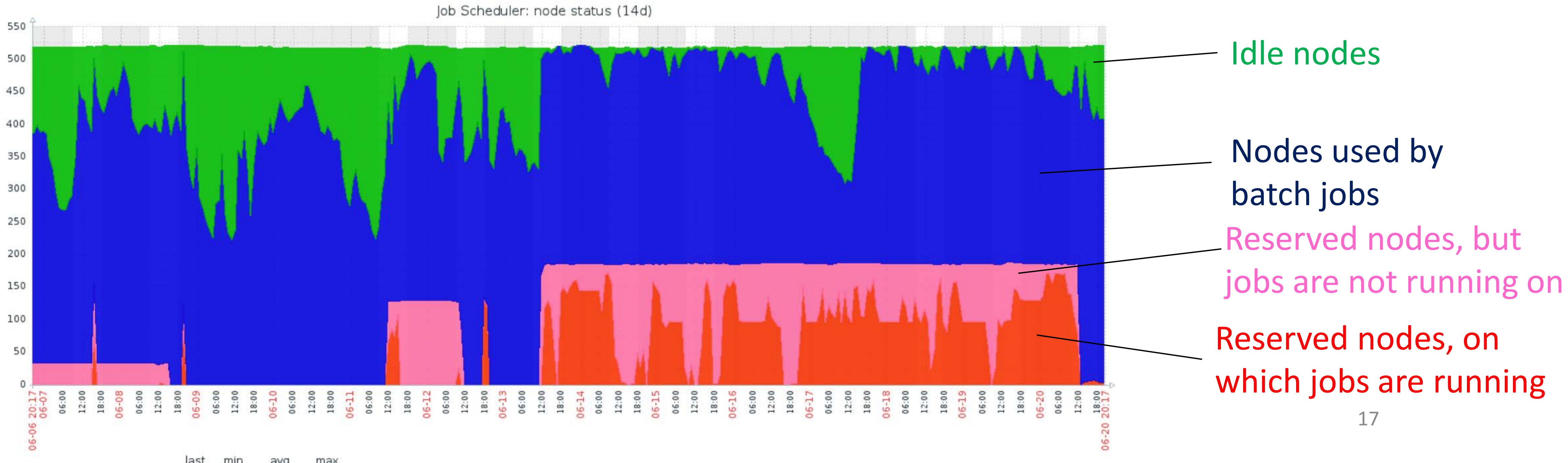
During the specified timeframe, the user can use nodes like a private cluster

Two methods for usage:

(1) Throwing jobs using scheduler

    % qsub –g [grp] –ar [AR_ID] ./job.sh

(2) ssh to reserved nodes → Interactive use is ok ☺



Container (1) Motivation

- TSUBAME3 has many pre-installed software packages
  - including DL frameworks such as Caffe, Tensorflow, Chainer…
- But upgrading of DL frameworks is so rapid
- Some users may want to use brand-new versions, while others use old versions
  - "pip --user install" everywhere? → Now this is happening on TSUBAME3
  - It is troublesome; also there are package dependencies

→We want to provide different images to such users!

However, we do not want to allow users to execute their own images, since they can become the root easily ☹

Container (2) Current Plan

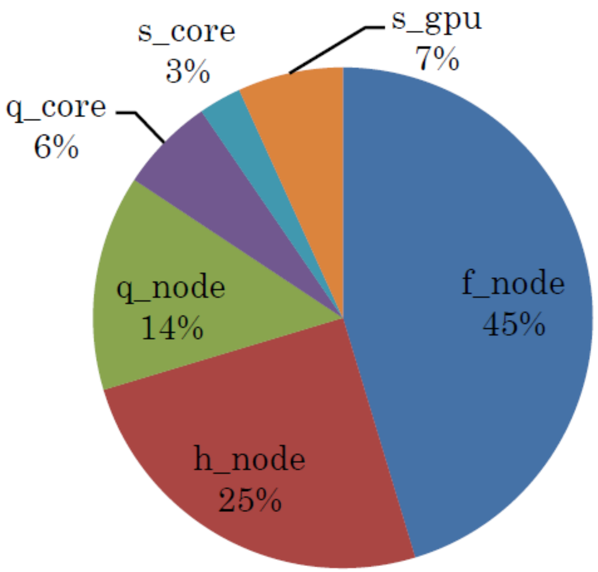
- We are going to use Docker in cooperative with UNIVA Grid Engine
  - The implementation is almost done, and now under testing
    - Cooperation with HPE and UNIVA
  - Will work with node partitioning
  - Will be started in 3Q 2018

- We will provide several “pre-defined” container images to users
- This is to reduce security risks, but it reduces flexibility for users

→ Singularity or Shifter improve the situation?

Statistics of TSUBAME3.0 (May 2018)

- # of users: 2178
  - Out of them, 567 users actually used the system
- Node usage: 75%
  - If a node is partially used, it is counted as “used nodes”
- # of jobs: 126,805



55% jobs used node partitioning

Summary

- Operation methods of TSUBAME3.0 have been designed through investigating existing issues in previous TSUBAME2
- Container support will be open soon
- Operations will be reconsidered continuously to improve usability, flexibility, resource utilization

http://www.t3.gsic.titech.ac.jp/en/