ABCI's Scheduling Design for Accommodating Various AI Jobs

Shinichiro Takizawa

The National Institute of Advanced Industrial Science and Technology (AIST), Japan

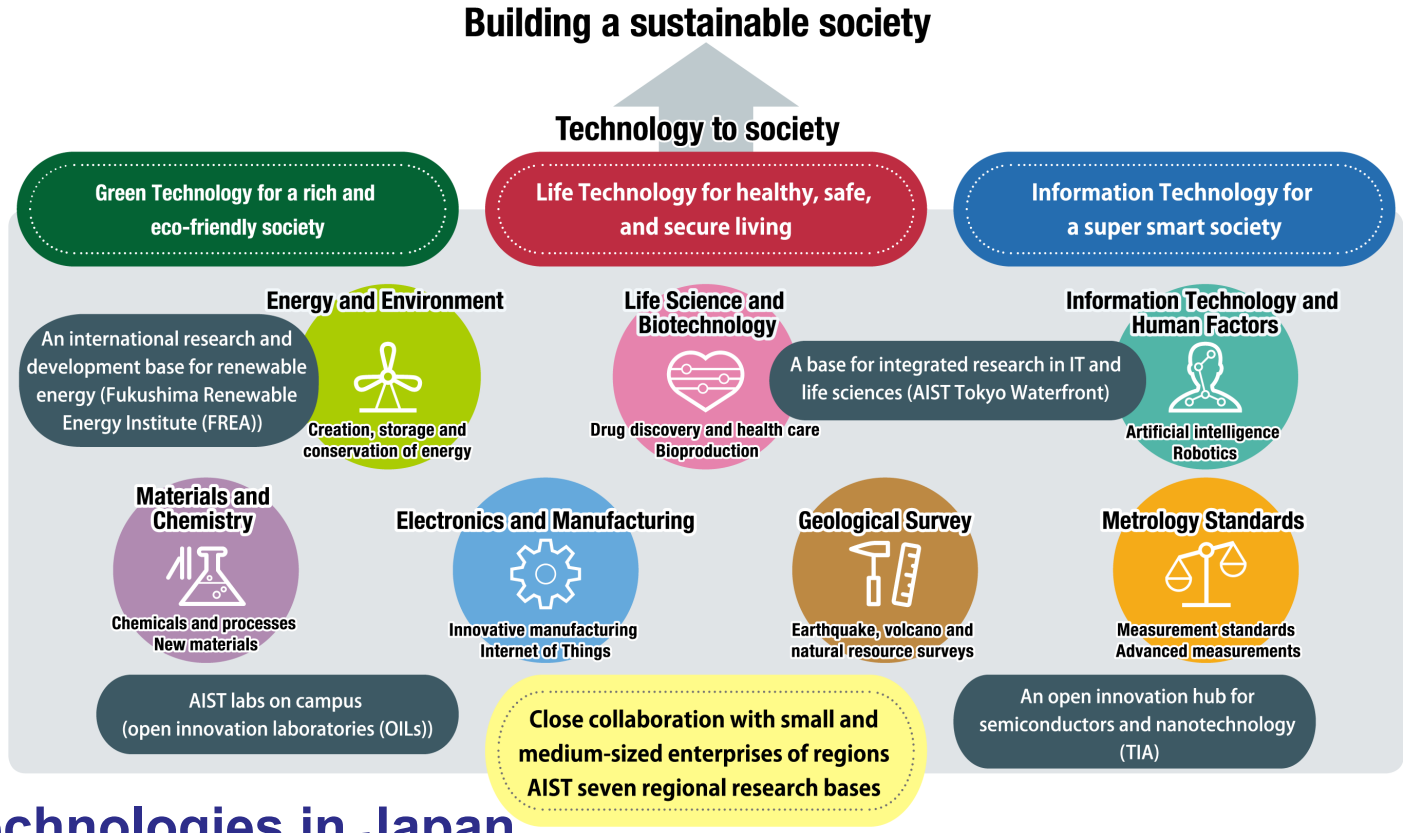
8th ADAC Workshop

Outline

- Introduction of AIST and ABCI
- ABCI system overview
  - Architecture, network topology, software stack
- ABCI scheduling system
  - Workload characteristics of our pre-ABCI system
  - Detail design of scheduling system

Introduction of AIST

- A research institute as a part of the Ministry of Economy, Trade and Industry (METI) of Japan
- Our mission
  - Integrate scientific and engineering knowledge to address industry and society needs
  - Bridge the gap between innovative technological seeds and commercialization



We develop ABCI for popularize AI technologies in Japan

ABCI: The World's First Large-Scale Open AI Infrastructure

東京大学 THE UNIVERSITY OF TOKYO AIST

Univ. Tokyo / Kashiwa II Campus

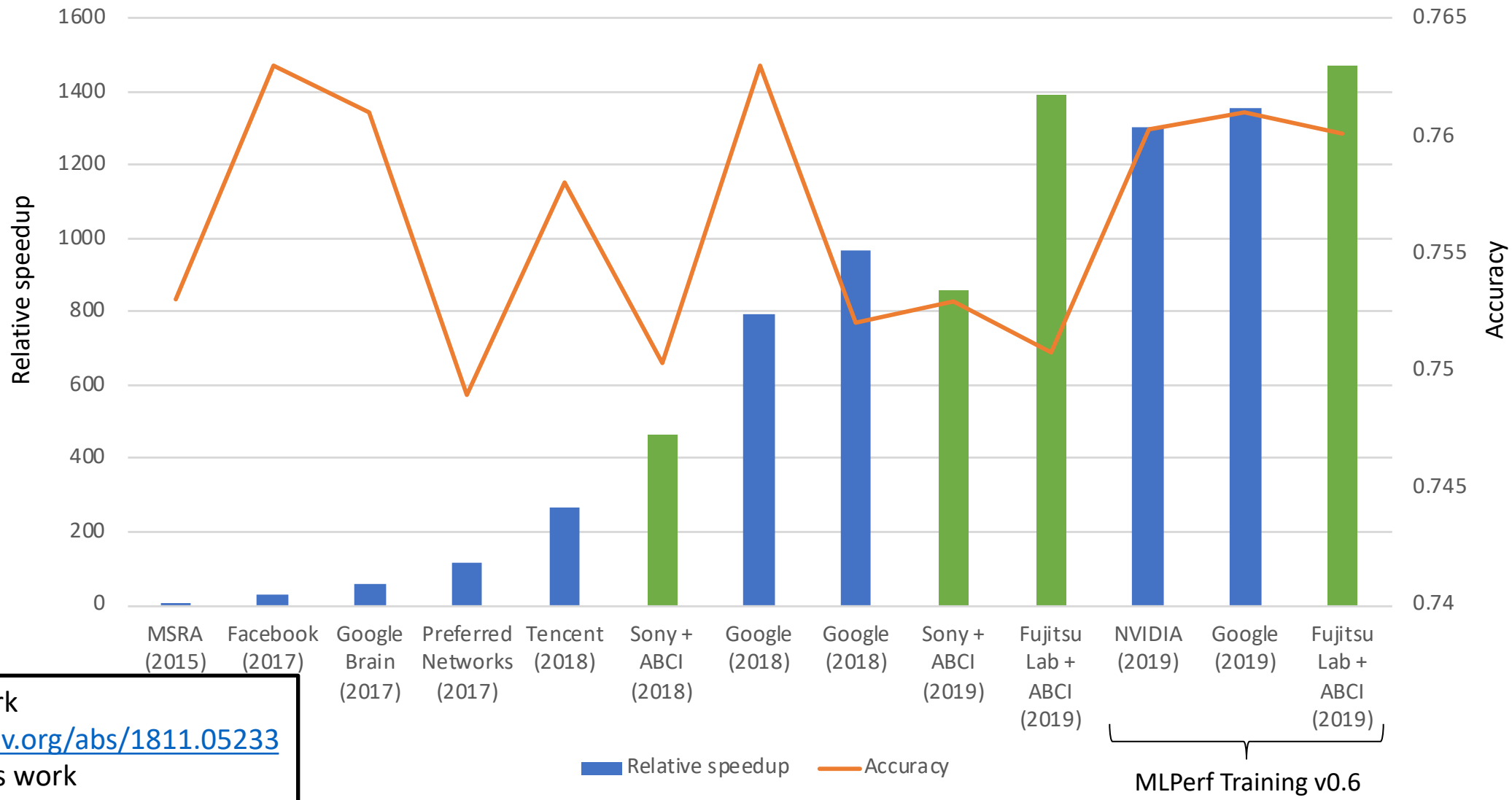
OPERATED SINCE AUGUST 1ST, 2018

ABCI AI Bridging Cloud Infrastructure

- World Top-Level compute and data process capability
- **Open, Public, and Dedicated** infrastructure for Al & Big Data Algorithms, Software, and Applications
- **Open Innovation Platform** to accelerate joint academic-industry R&D for AI

Peak Performance:
550 PFLOPS (FP16)
37 PFLOPS (FP64)
Effective Performance: (as of Jun 2019)
19.88 PFLOPS (#8 in TOP500)
14.423 GFLOPS/W (#3 in GREEN500)
508.85 TFLOPS (#5 in HPCG)
Power Usage: < 2.3 MW
Average PUE: < 1.1 (Estimated)

World's Highest Speed in ImageNet-1k Training

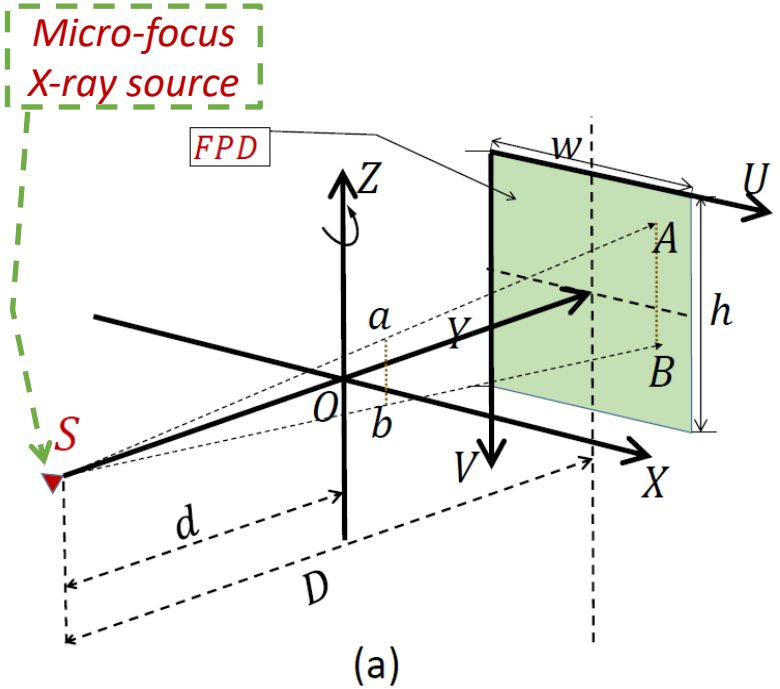


SONY's work
https://arxiv.org/abs/1811.05233
Fujitsu lab's work
https://arxiv.org/abs/1903.12650

iFDK: CT Image Reconstruction Framework on ABCI

SC19
Denver, CO | hpc is now.

- A high-speed and high-resolution CT image reconstruction framework running on GPU clusters
  - Create 3D images from multiple 2D x-ray images
  - Demands for high-resolution CT image: non-invasive inspection, reverse engineering, etc.
- Our Achievements
  - High speed FDK GPU kernel whose compute cost is 1/6 of the existing algorithms
  - Efficient FDK computation by overlapping CPU comp., GPU comp. and comm.
  - Distributed framework for high-resolution image reconstruction
  - Good scalability up to 2K GPUs



(a)
CBCT geometry and trajectory

Peng Chen, Mohamed Wahib, Shinichiro Takizawa, Ryousei Takano, Satoshi Matsuoka.
iFDK: A Scalable Framework for Instant High-resolution Image Reconstruction

ABCI SYSTEM OVERVIEW

High-Performance Computing System

550 PFlops(FP16), 37.2 PFlops(FP64)
476 TiB Memory, 1.74 PB NVMe SSD

Computing Nodes (w/ GPU) x 1088

GPU	NVIDIA Tesla V100 SXM2 x 4
CPU	Intel Xeon Gold 6148 (2.4GHz/20cores) x 2
Memory	384GiB
Local Storage	Intel SSD DC P4600 (NVMe) 1.6TB x 1
Interconnect	InfiniBand EDR x 2

Multi-platform Nodes (w/o GPU) x 10

- Intel Xeon Gold 6132 (2.6GHz/14cores) x 2
- 768GiB Memory, 3.8TB NVMe SSD, 1.5TB Intel Optane x2

Interactive Nodes x 4

Management and Gateway Nodes x 15

Large-scale Storage System

1 PB Lustre (Home Directory)

DDN SFA14KX (w/ SS9012 Enclosure x 10) x1
- 7.68TB SAS SSD x 185 for data
- 960GB SAS SSD x 13 for metadata

22 PB GPFS (Group Shared Directory, etc.)

DDN SFA14K (w/ SS8462 Enclosure x 10) x3
- 12TB 7.2Krpm NL-SAS HDD x 2400
- 3.84TB SAS SSD x 216

17 PB Object Storage (Scality RING)

HPE Apollo 4510 Gen10 x 24
- 12TB SATA HDD x 1440
- 3.2TB SSD x 24

Interconnect (InfiniBand EDR)

- Mellanox CS7500 x 2
- Mellanox SB7890 x 229

Service Network (10GbE)

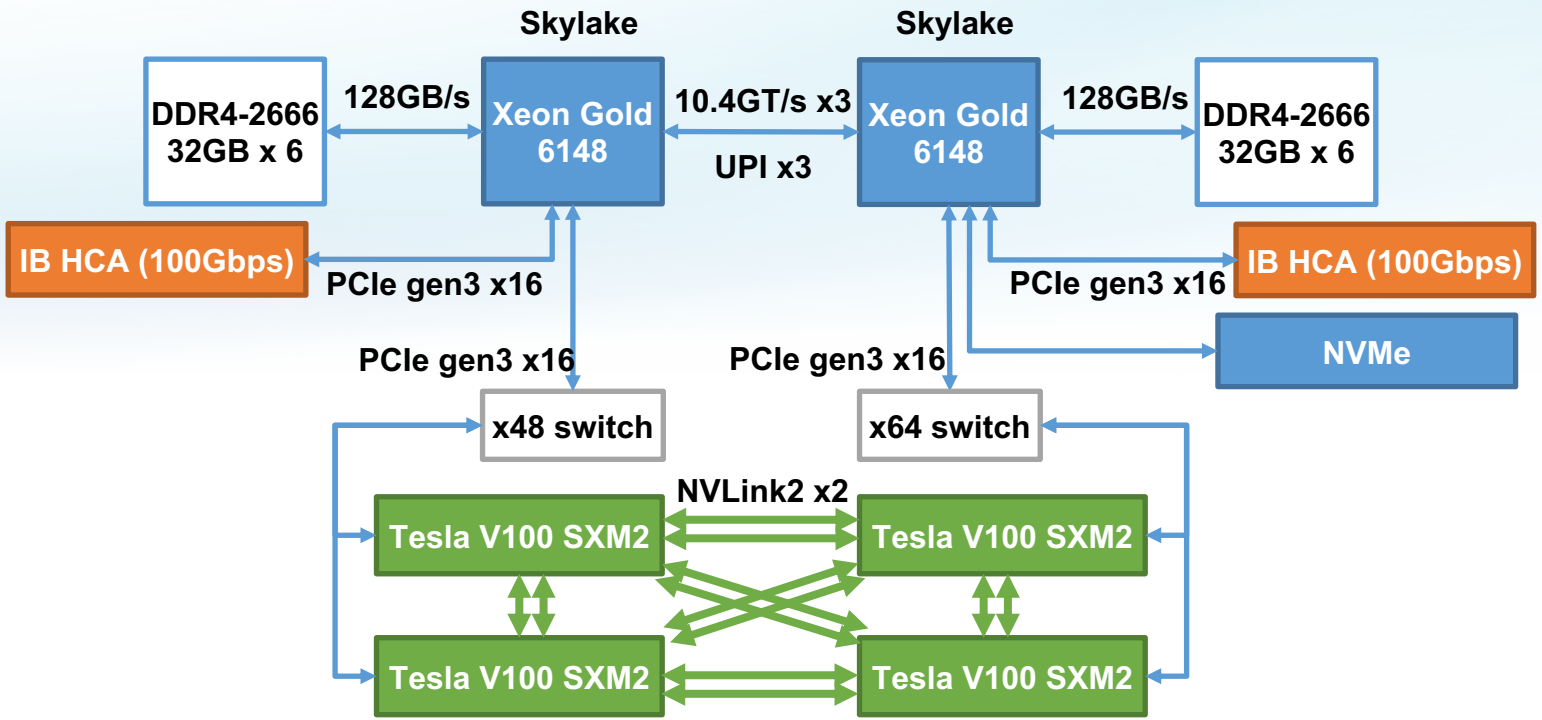
Gateway and Firewall

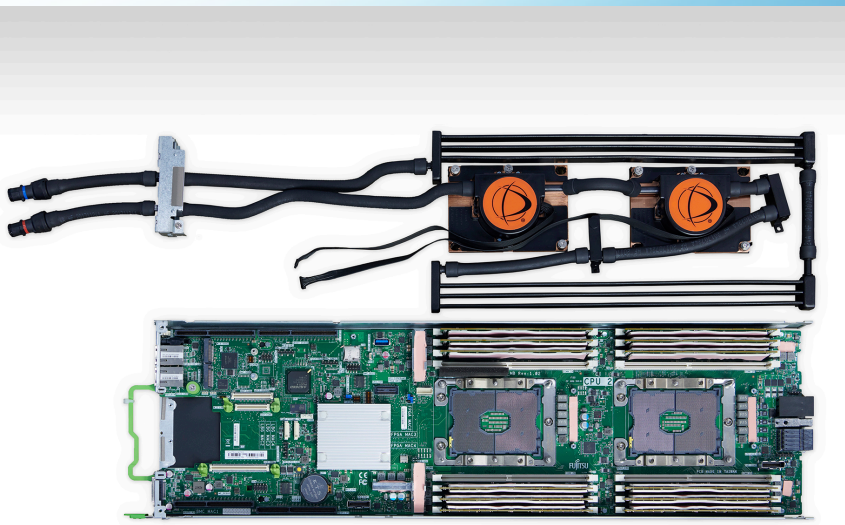
- Nexsus 3232C x2
- FortiGate 1500D x2
- FortiAnalyzer 400E x1

100Gbps SINET5

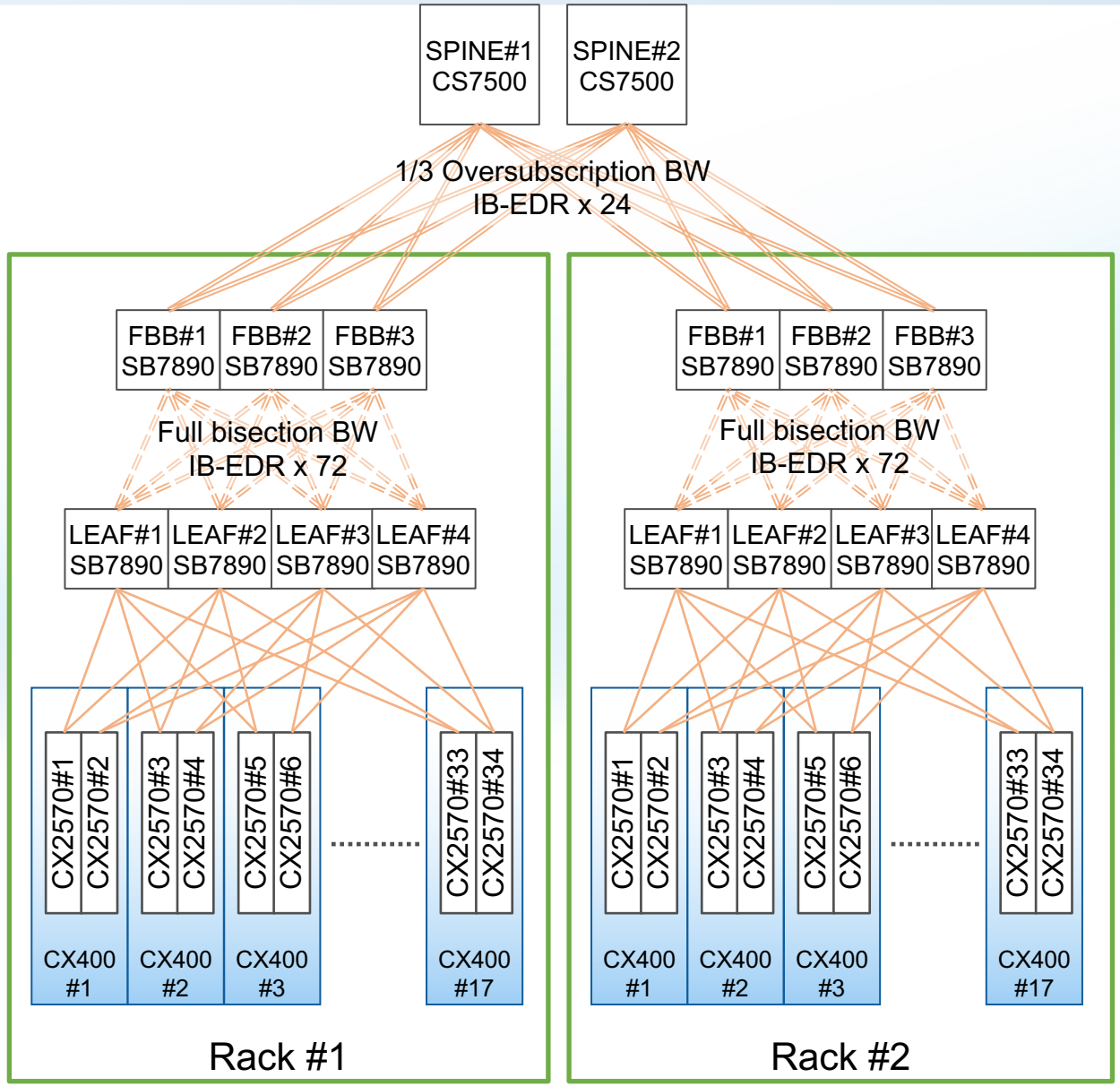
ABCI Compute Node

FUJITSU PRIMERGY Server (2 servers in 2U)	
CPU	Xeon Gold 6148 (27.5M Cache, 2.40 GHz, 20 Core) x2
GPU	NVIDIA Tesla V100 (SXM2) x4
Memory	384GiB DDR4 2666MHz RDIMM
Local Storage	1.6TB NVMe SSD (Intel SSD DC P4600 u.2) x1
Interconnect	InfiniBand EDR x2





ABCI Node Rack / Interconnect



- **Dense-packaged rack: 34 nodes, 136 Tesla V100**
  - Theoretical peak performance per rack : 1.16 PFlops (FP64), 17 PFlops (FP16) c.f. Google TPU 3.0 Pod (>100PFlops/8racks)
  - Power consumption per rack: 67.33 kW
- **Interconnect**
  - Fat-tree topology
  - Intra-rack: full bisection BW
  - Inter-rack : 1/3 over-subscription (2400/6800)
  - Without adoptive routing
  - Without Mellanox SHARP

32 Racks

InfiniBand EDR x4
InfiniBand EDR x6
InfiniBand EDR x1

ABCI Software Stack

Operating System	RHEL / CentOS 7.5
Job Scheduler	Univa Grid Engine 8.6.6
Container Engine	Docker 17.12.0 (Users can use only supported container images) Singularity 2.6.1 (Users can use any container images)
MPI	Intel MPI 2018.2.199 MVAPICH2 2.3rc2, 2.3 / MVAPICH2-GDR 2.3a, 2.3rc1, 2.3, 2.3.1, 2.3.2 OpenMPI 1.10.7, 2.1.3, 2.1.5, 2.1.6, 3.0.3, 3.1.0, 3.1.2, 3.1.3
Development tools	Intel Parallel Studio XE Cluster Edition 2017.8, 2018.2, 2018.3, 2019.3 PGI Professional Edition 17.10, 18.5, 18.10, 19.3 NVIDIA CUDA SDK 8.0, 9.0, 9.1, 9.2, 10.0, 10.1 cuDNN 5.1, 6.0, 7.0, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6 NCCL 1.3.5, 2.1, 2.2, 2.3, 2.4 Intel MKL 2017.8, 2018.2, 2018.3, 2019.3 GCC, Python, Ruby, R, OpenJDK, Go, Perl
Deep Learning	Caffe, Caffe2, TensorFlow, Theano, Torch, PyTorch, CNTK, MXnet, Chainer, Keras, etc. *(Frameworks provided by NVIDIA GPU Cloud can also be deployed)*
Big Data Processing	Hadoop, Spark

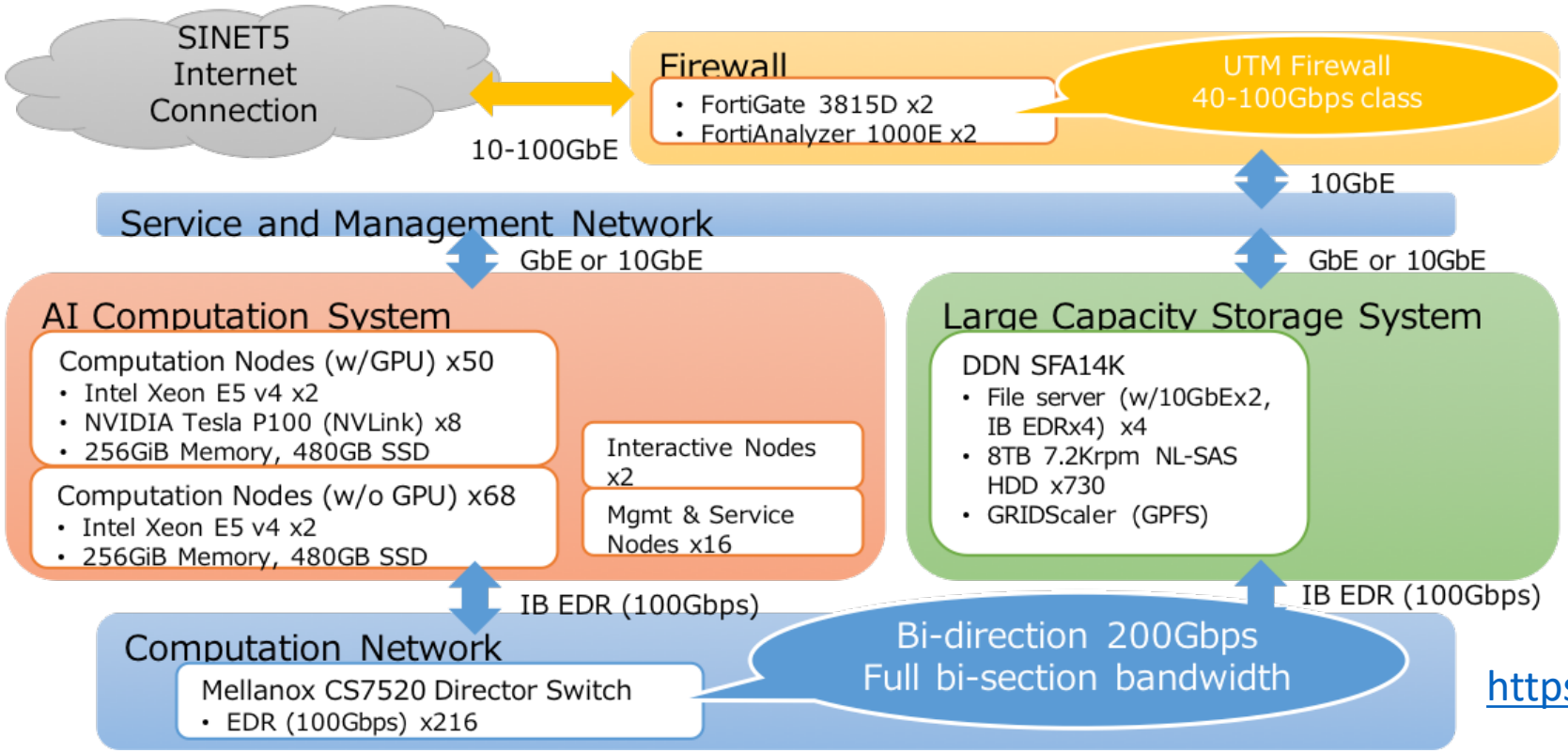
ABCI'S JOB SCHEDULING SYSTEM

Objective and Preparation

- Objective
  - Make ABCI an easy-to-use capacity computing system for AI R&D
    - Also need to be a capability computing system for grand challenge AI problems
- Preparation for designing the scheduling system
  - Operate a pre-ABCI GPU cluster for AI R&D more than two years (it's still in operation)
  - Collect job records and analyze them to understand how users use a GPU cluster for AI R&D
  - Use the obtained knowledges to design job scheduling system

AAIC and its Workload Logs

- **400x NVIDIA Tesla P100s** and **InfiniBand EDR** accelerate various AI workloads including ML (Machine Learning) and DL (Deep Learning).
- Advanced data analytics leveraged by **4PiB shared Big Data Storage** and **Apache Spark** w/ its ecosystem.

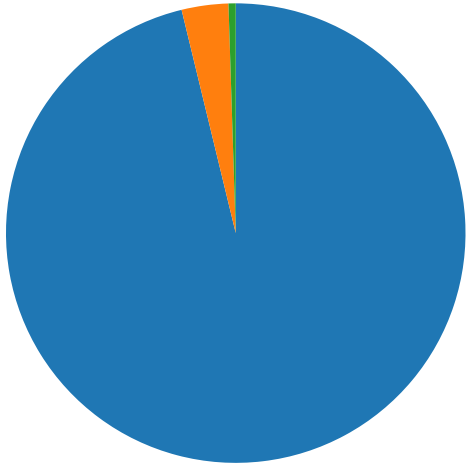


AAIC Workload Logs
(Jul. 14th, 2017 to Dec. 31st, 2018)

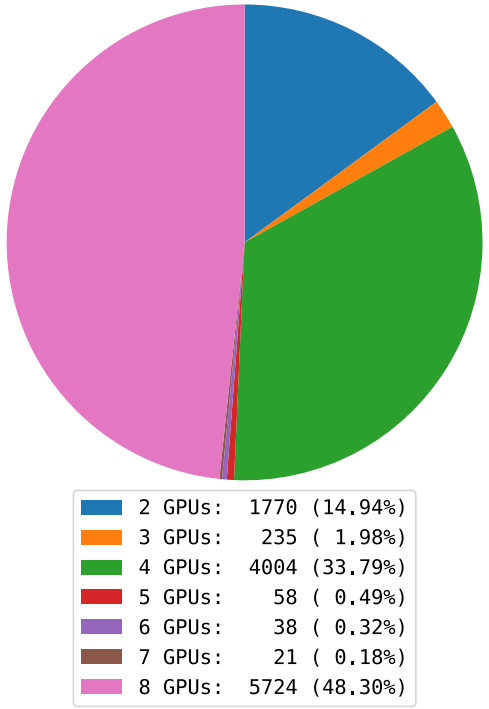
https://github.com/aistairc/aaic-workload

Distribution of Number of Used GPUs/Nodes in AAIC DL Jobs

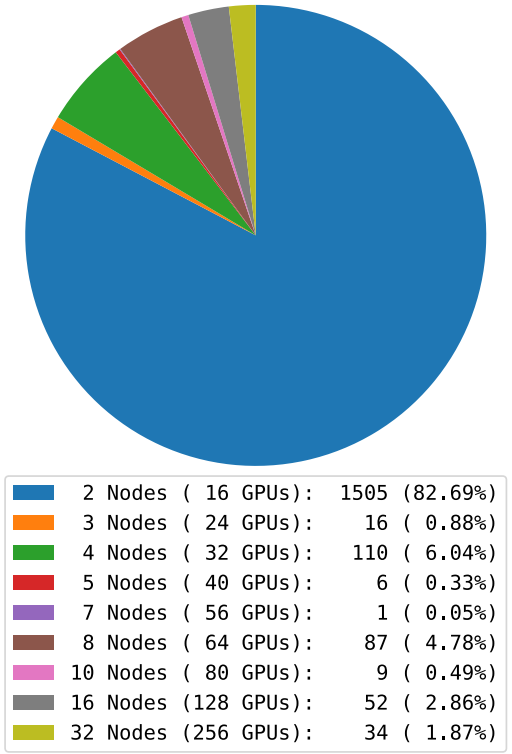
Single GPU, Single Node and Multi Node jobs



Used GPUs in Single Node Jobs



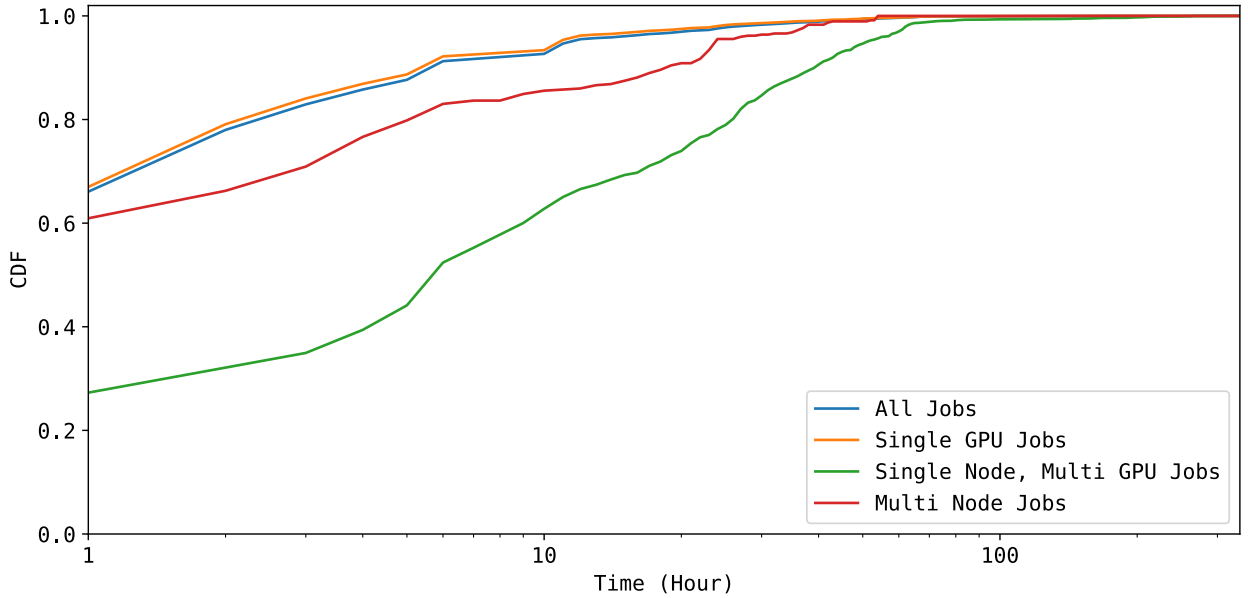
Used Nodes in Multi Node Jobs



Single GPU jobs are dominant

Walltime of AAIC DL Jobs

Cumulative Distribution of Walltime



- Some jobs run for two weeks
  - Some requested more than two weeks
- Walltime of single GPU jobs is shorter than that of multi GPU jobs

Statistics of Walltime

System, Job type	Average	STD	Median	Min	Max
All Jobs/AAIC	2.730	8.255	0.470	0.017	332.678
Single GPU Jobs/AAIC	2.461	7.429	0.460	0.017	332.678
Single Node, Multi GPU Jobs/AAIC	14.062	22.038	5.576	0.017	276.374
Multi Node Jobs/AAIC	4.642	9.640	0.187	0.017	53.798

Observations from AAIC AI Workload

- Most jobs use only one GPU
  - Although severs equipped with multiple GPUs are common (e.g. DGX-2)
- Small degree of parallelism are common in parallel jobs
- Walltime varies greatly depending on jobs
  - Many of them are less than one hour, but some requires more than a week
- Observations not explained today
  - Positive correlation between degree of parallelism and walltime
  - Low accuracy of users' requested walltime
  - Use of large array jobs
  - etc.

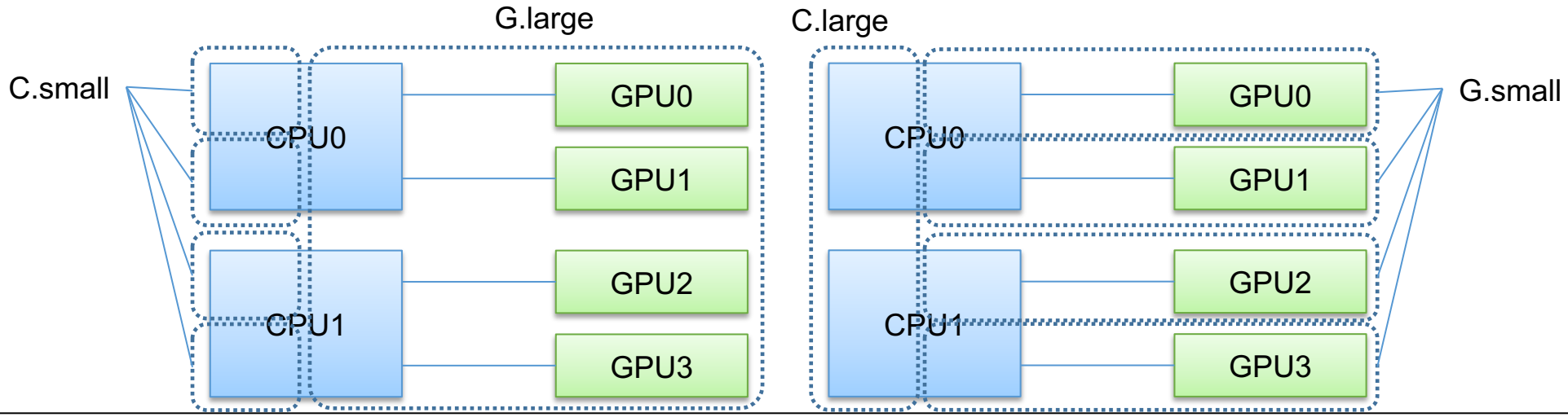
Design of ABCI's Job Scheduling System

- Basic scheduling design
  - Define various resource types by splitting compute nodes
  - Interactive use, batch jobs and node reservation services
  - Allow long maximum walltime and reservation duration
- Emulation of a rack-aware scheduling
  - Designed to achieve good performance for small jobs fit in a rack (34 nodes)
- Other parameters

Resource Types

Type Name	#CPU core Assigned / Total	#GPU Assigned / Total	Memory (GB) Assigned / Total	Local SSD (TB) Assigned / Total	Fee (Relative to F)
F (Full node)	40 / 40	4 / 4	360 / 384	1.4 / 1.6	1.0
G.large	20 / 40	4 / 4	240 / 384	0.7 / 1.6	0.9
G.small	5 / 40	1 / 4	60 / 384	0.175 / 1.6	0.3
C.large	20 / 40	0 / 4	120 / 384	0.7 / 1.6	0.6
C.small	5 / 40	0 / 4	30 / 384	0.175 / 1.6	0.2

Resource types are defined by cgroups feature included in UGE



Job Execution Services

Normal Job Execution Services

Name of Service	Description	Maximum #Nodes	Minimum Walltime	Maximum Walltime	Maximum #Nodes x Walltime
Spot	Batch job (e.g. qsub) Charges users for nodes x time (in seconds)	512	1 S	72 H	2304 NxH
On-demand	Interactive job (e.g. qrsh) Charges users for nodes x time (in seconds)	32	1 S	12 H	12 NxH

Node Reservation Services

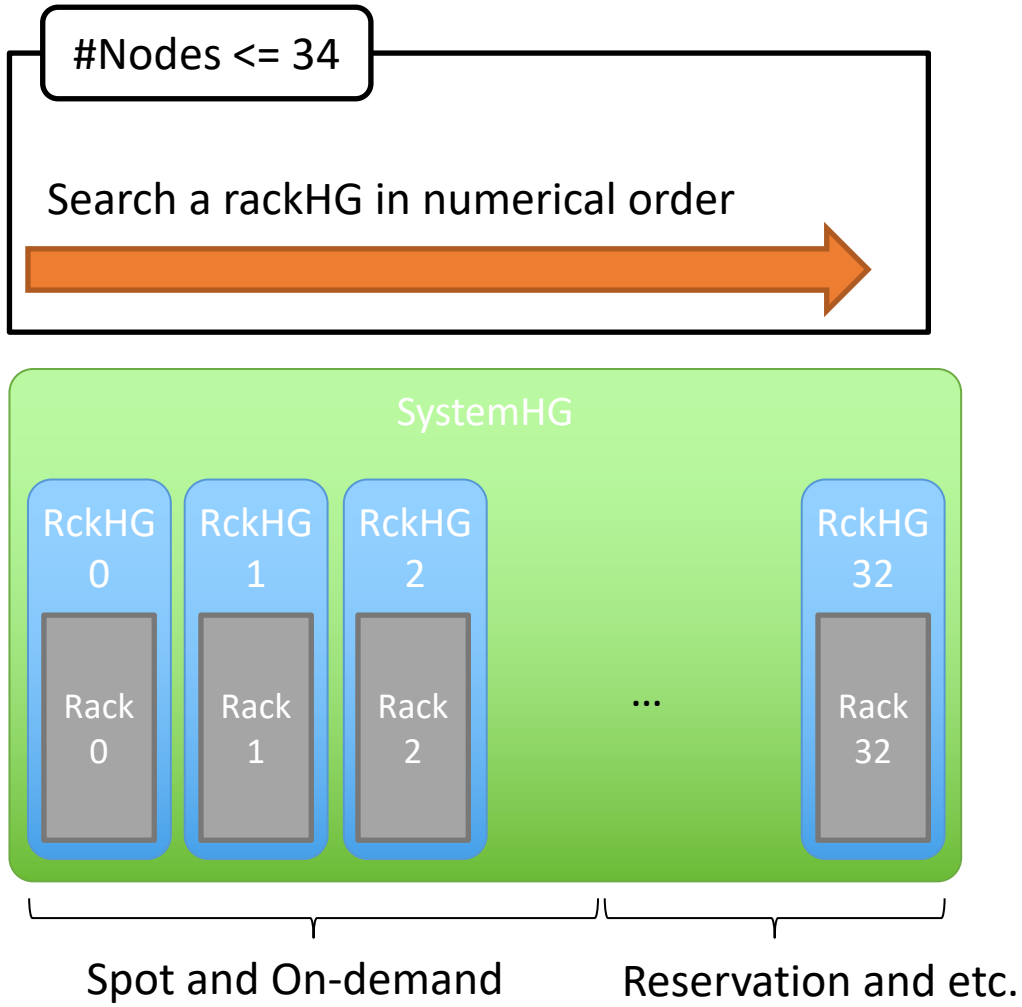
Name of Service	Description	Maximum #Nodes	Minimum Resv. Time	Maximum Resv. Time	Maximum #Nodes x Resv. Time
Reserved	Node reservation Charges users for nodes x reserved_time (in days)	32	1 D	30 D	12288 NxH

Detail job scheduling description: https://docs.abci.ai/en/03/

Emulation of Rack-aware Scheduling (1/2)

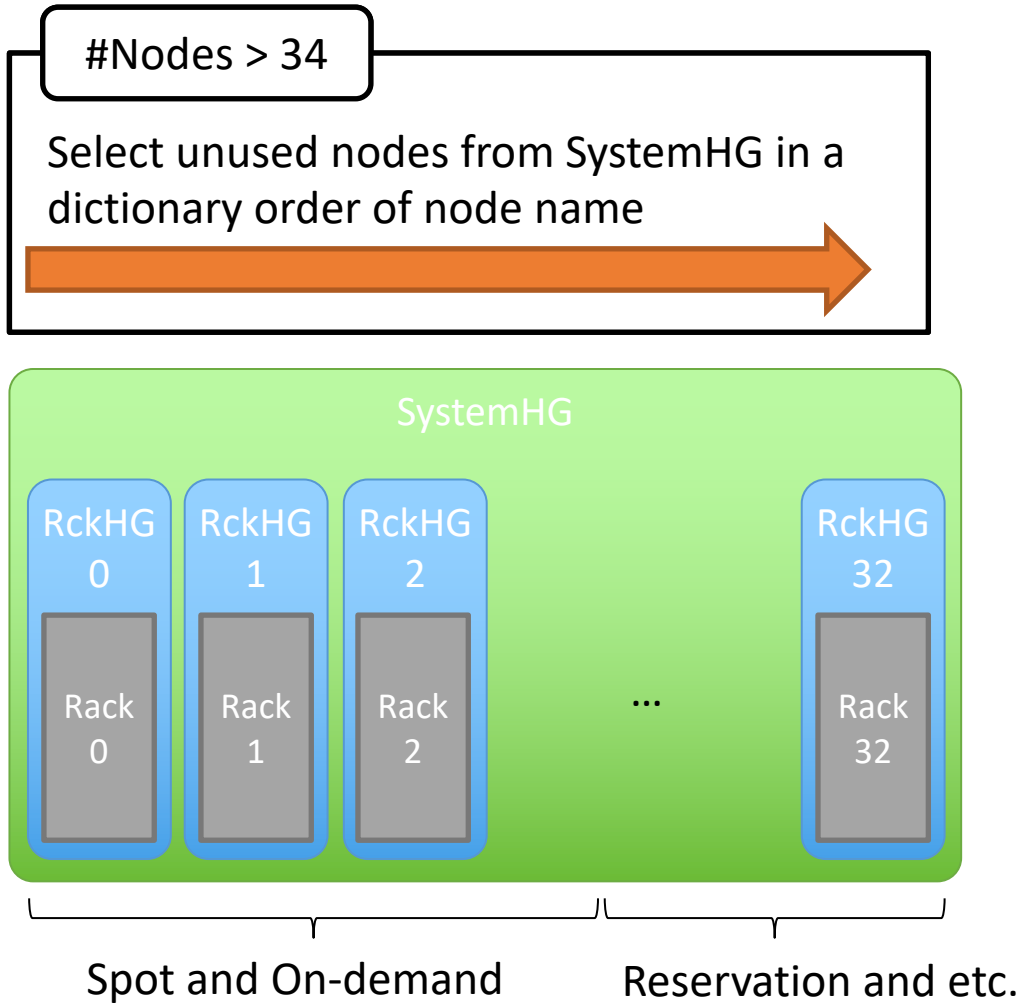
- Limitation
  - ABCI does not have full bisection-bandwidth network
  - UGE does not consider network topology in scheduling
- Objective
  - Provide the maximum network performance for small jobs
    - Small jobs : Use less than 34 nodes (within 1 rack)
  - Keep wait time for both small and large jobs as small as possible

Emulation of Rack-aware Scheduling (2/2)



- Single Queue
- Two types of host group under the queue
  - RackHG: host group for each rack
  - SystemHG: unique host group for all nodes
- Jobs using <=34 nodes
  - Find a rackHG that has enough nodes and run the job in the rackHG
- Jobs using >34 nodes
  - Select nodes from systemHG
- Different resource pools for different kind of services
  - (1)Spot and On-demand and (2)Others

Emulation of Rack-aware Scheduling (2/2)



- Single Queue
- Two types of host group under the queue
  - RackHG: host group for each rack
  - SystemHG: unique host group for all nodes
- Jobs using <=34 nodes
  - Find a rackHG that has enough nodes and run the job in the rackHG
- Jobs using >34 nodes
  - Select nodes from systemHG
- Different resource pools for different kind of services
  - (1)Spot and On-demand and (2)Others

Other parameters

- Enable backfill and priority option
- Limitation on selecting resource types for job
  - Using multiple resources are allowed only when F resource type is used
    - i.e. Can not use multiple G.[small | large], C.[small | large] in a job
  - A job can use only one resource type
- Limit the maximum number of running jobs per user: 200
  - Prevent resource occupation by a specific user
  - No restriction on number of job submission

Sample Job Script

```
#! /bin/sh
#$ -l rt_F=32
#$ -l h_rt=1:00:00
#$ -N testjob
#$ -l USE_BEEOND=1
#$ -v BEEOND_METADATA_SERVER=4
#$ -v BEEOND_STORAGE_SERVER=16
#$ -cwd


source /etc/profile.d/modules.sh
module load openmpi/3.1.3

# Stage-in data to Local SSD
cp -r $HOME/something0 $SGE_LOCALDIR
# Stage-in data to BeeGFS
cp -r $HOME/something1 /beeond

mpirun -np 32 ...
...
```

Type of resource and its amount

Walltime for the job

Create a temporal distributed filesystem for the job using BeeGFS on Demand

Local NVMe SSD mount point

BeeGFS mount point

Job Submission Command

```
$ qsub -g gaa50004 sample_job.sh
```

Budget code

Summary

https://abci.ai/

- Introduce ABCI, an open AI platform
  - Architecture, network topology and software stack
  - AI workload analysis results of pre-ABCI system
  - Detail of scheduling system design
- Future work
  - Introduce resource types for memory intensive jobs
    - Preparing several 2.8TB memory (2TB by Intel Optane) nodes
  - Collect job records executed on ABCI and analyze them
  - Consider supporting multiple [C | G].[small | large] resource job
    - Combination of them