AI Bridging Cloud Infrastructure (ABCI) and its Communication Performance

Shinichiro Takizawa

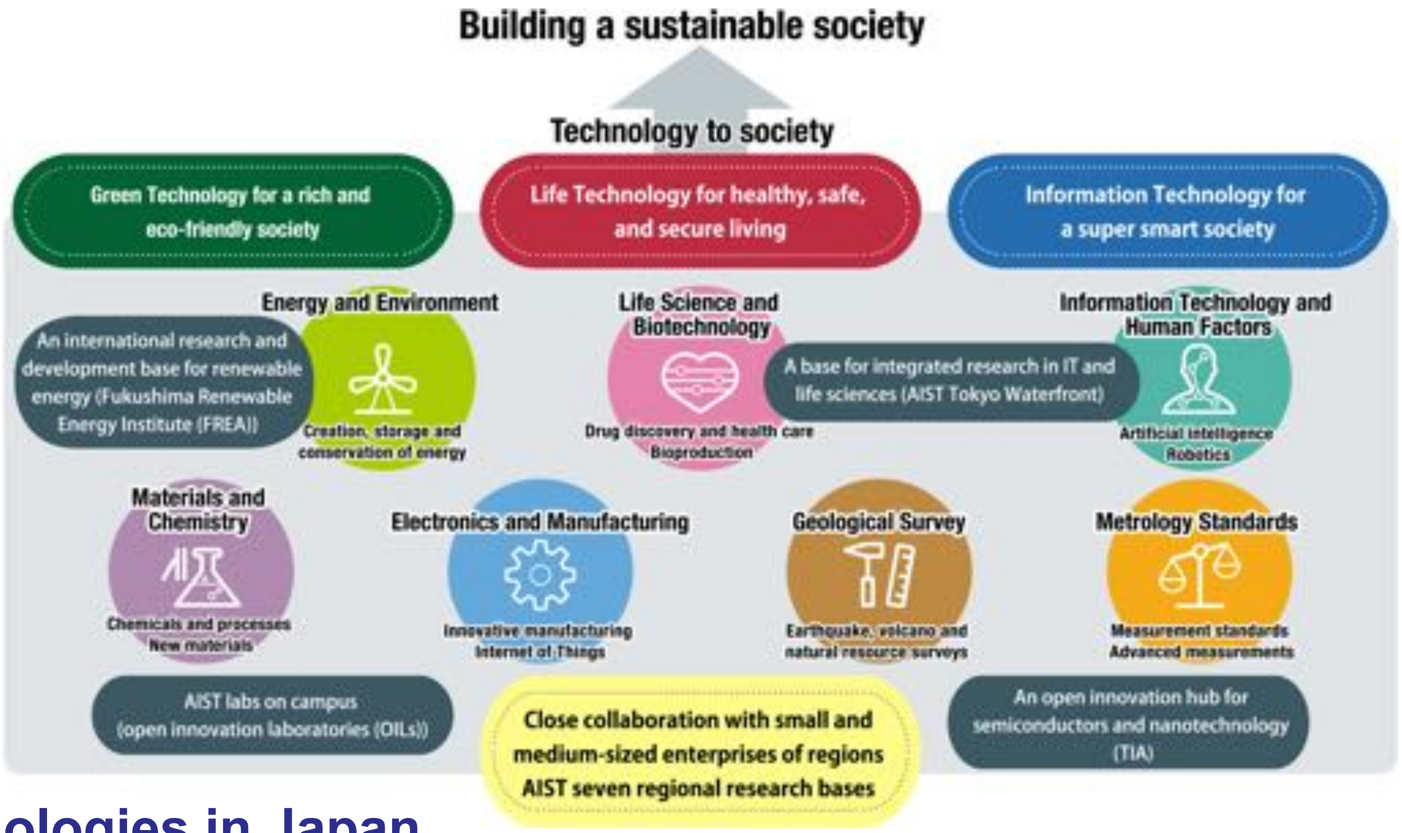
The National Institute of Advanced Industrial Science and Technology (AIST), Japan

Outline

- Introduction of AIST and ABCI
- ABCI Detail
  - Architecture, software stack, network topology
- MPI performance on ABCI
- A latest application on ABCI

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 built 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

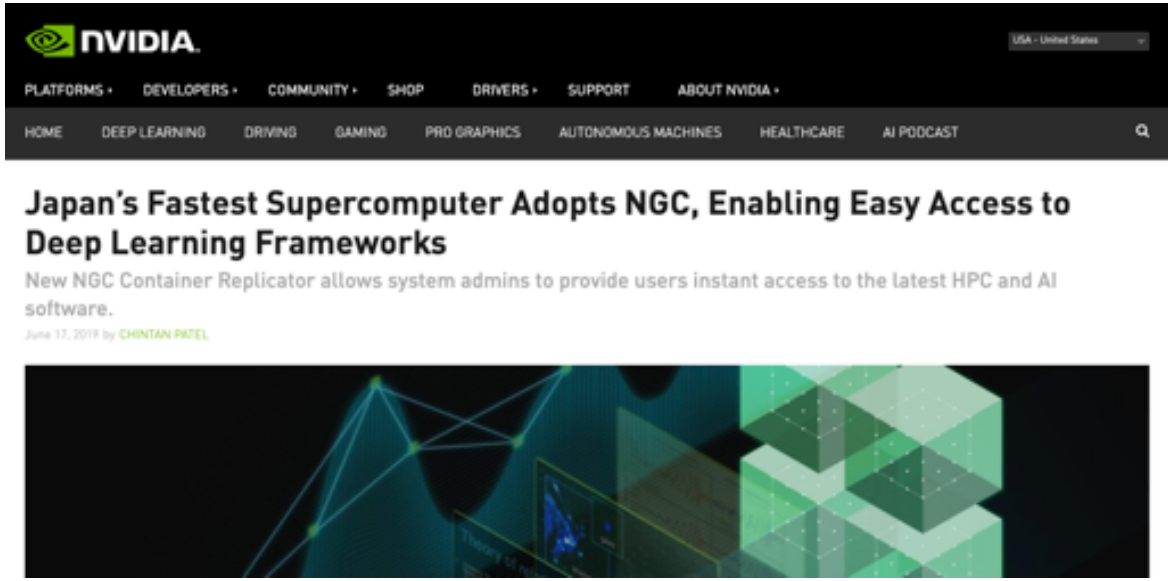
ABCI AI Bridging Cloud Infrastructure

- World Top-Level compute and data process capability
- **Open, Public, and Dedicated** infrastructure for AI & 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)

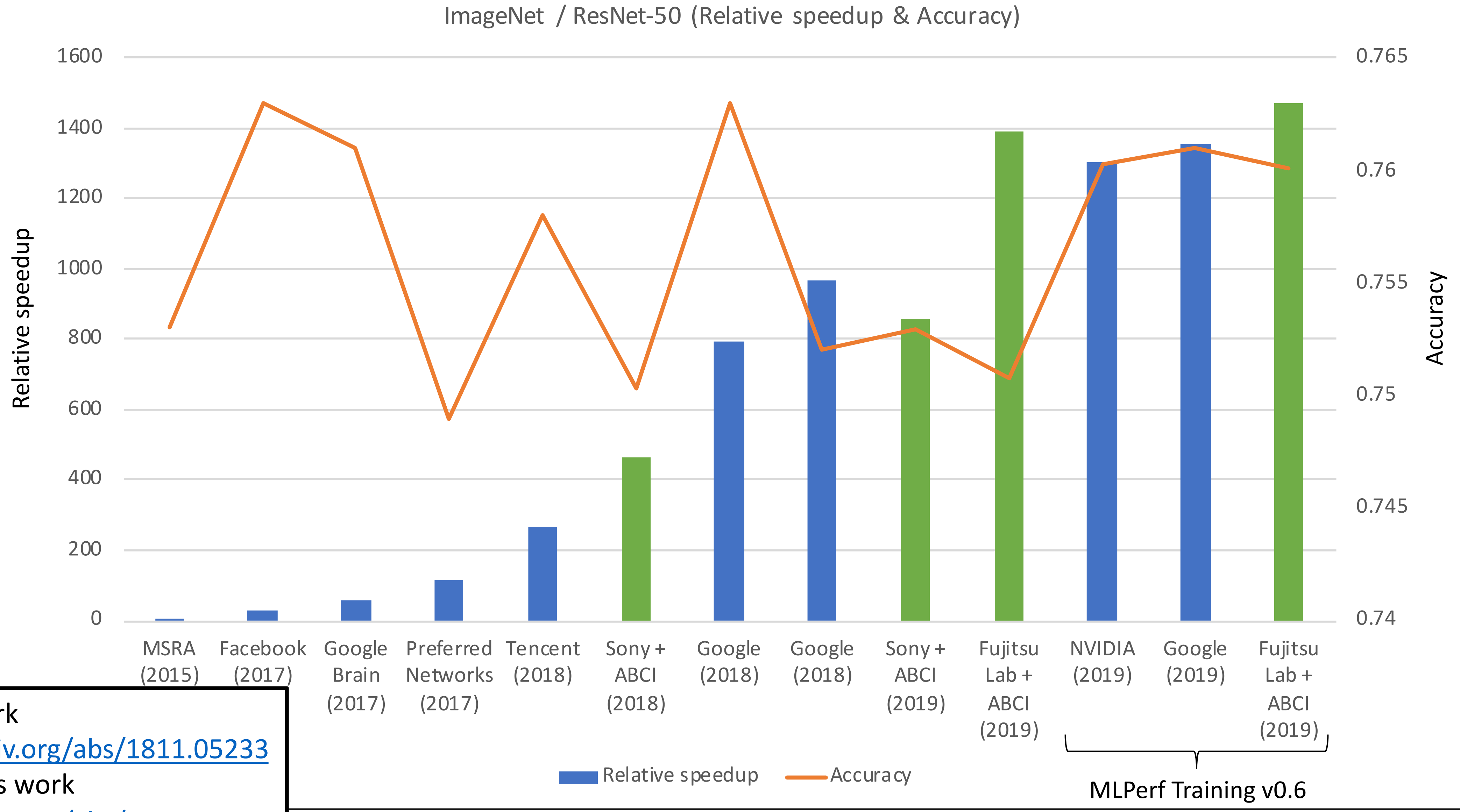
Some of 1 Year Achievements

- 100+ projects, 1000+ users
  - Academia, research institutes and companies
- Large Japanese companies start to use ABCI as their R&D platform
- ABCI supports NGC containers
- SONY and Fujitsu lab achieved good performance of ImageNet-1k classification on ABCI
- Two research papers that use ABCI were accepted by SC19



https://blogs.nvidia.com/blog/2019/06/17/abci-adopts-ngc/

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

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 (Preparing)

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

Gateway and Firewall

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

100Gbps

SINET5

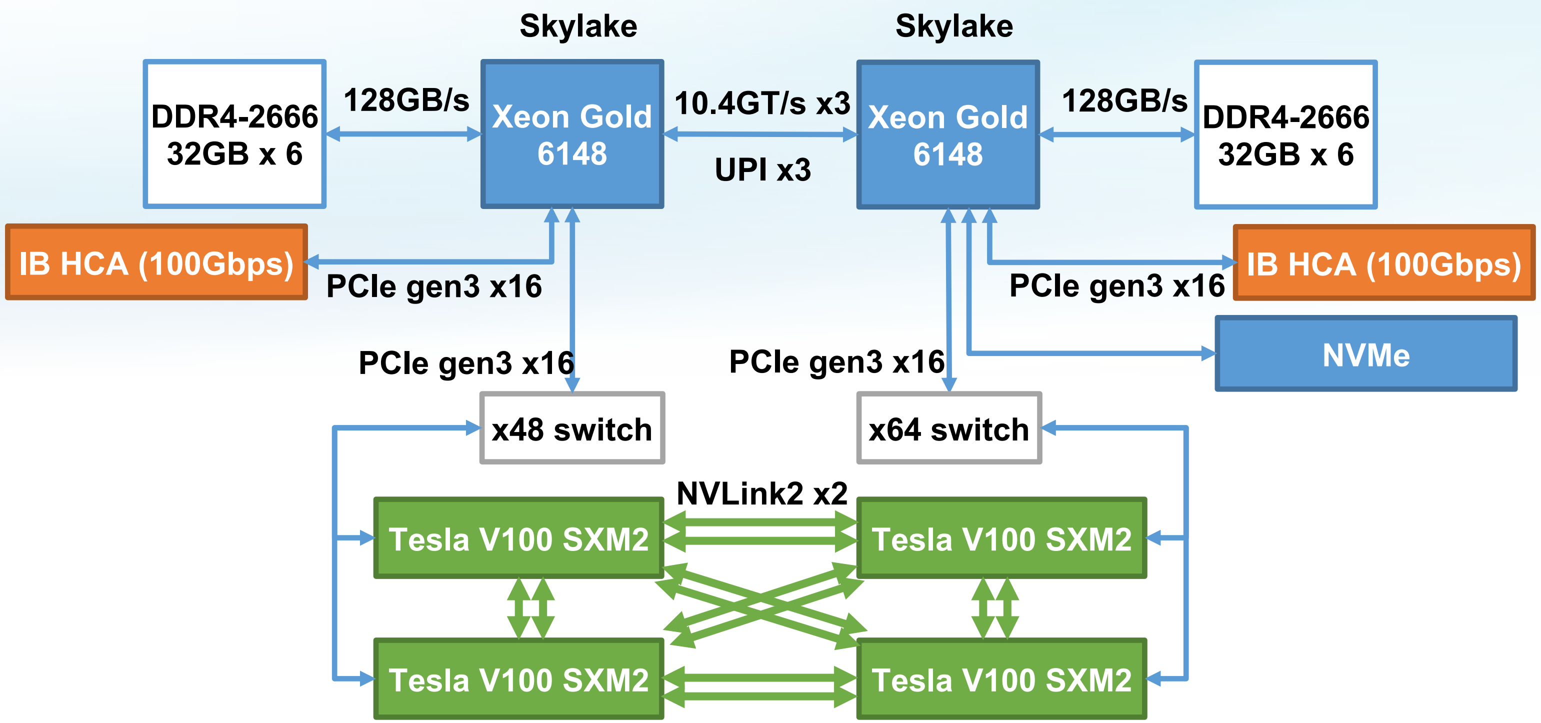
Service Network (10GbE)

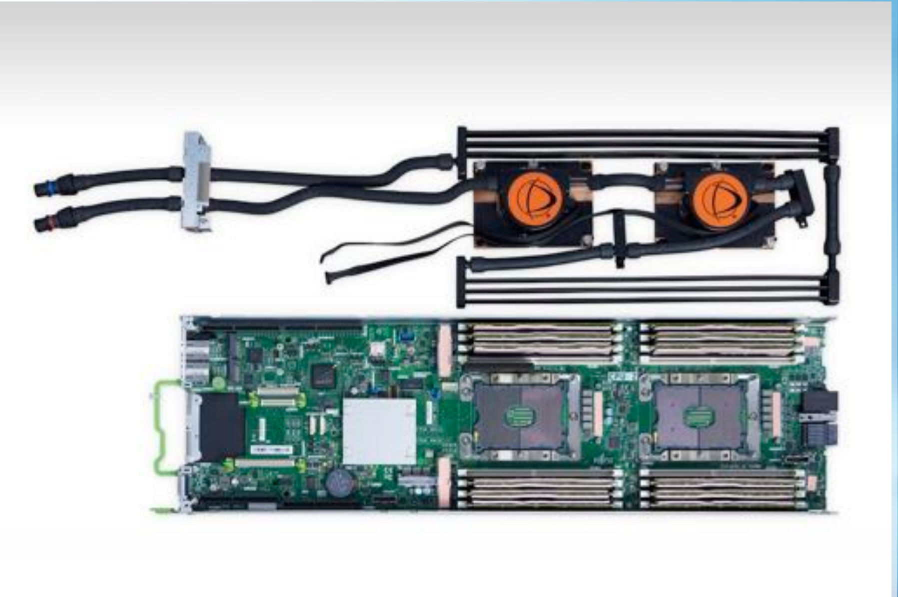
ABCI Software Stack

Operating System	RHEL / CentOS 7.6
Job Scheduler	Univa Grid Engine 8.6.3
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 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 cuDNN 5.1, 6.0, 7.0, 7.1, 7.2, 7.3, 7.4, 7.5 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

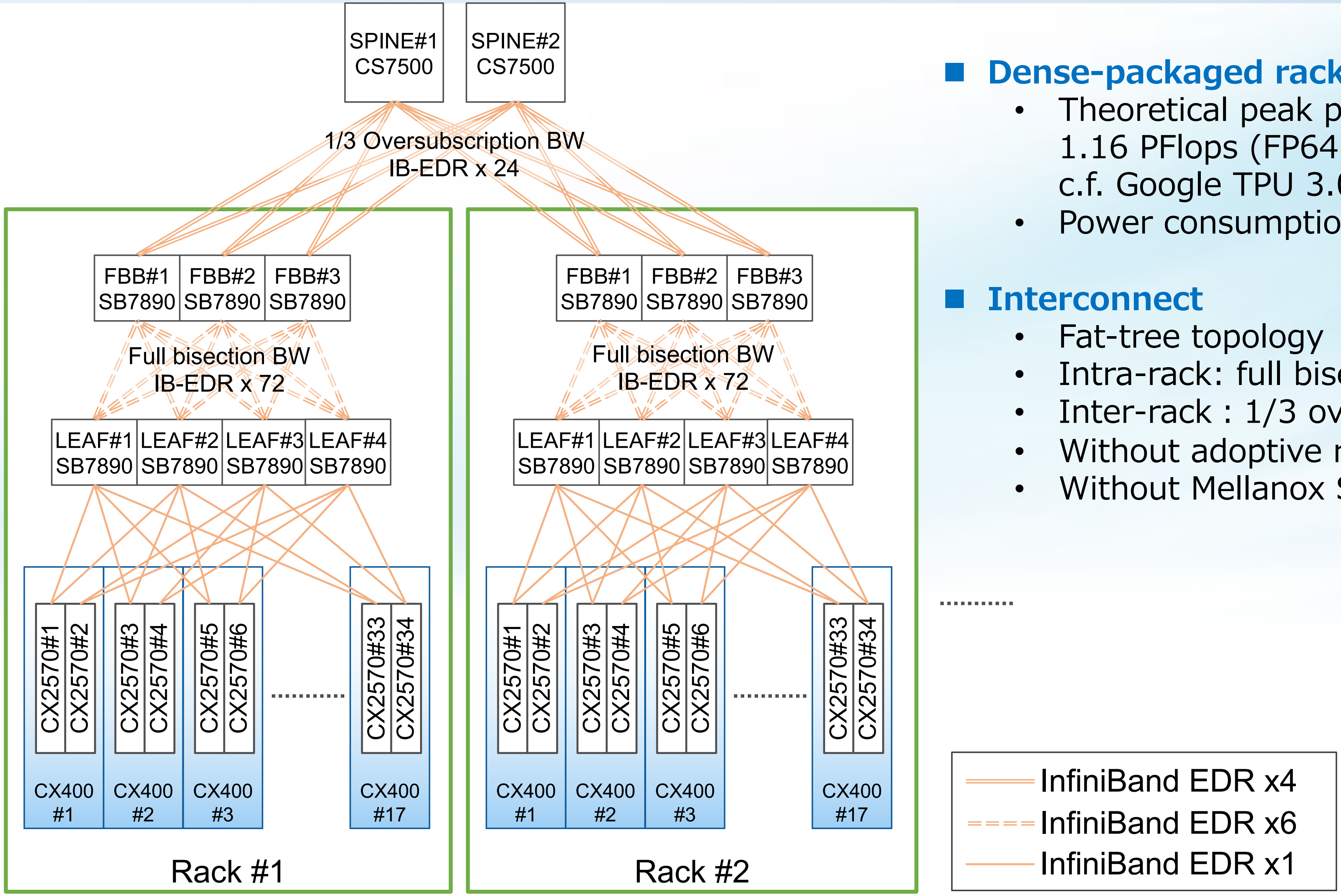
ABCI Compute Node

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



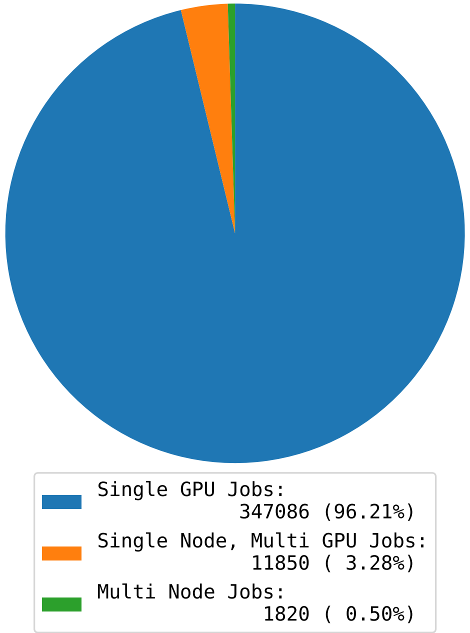
- **Dense-packaged rack: 34 nodes, 136 Tesla V100**
  - Theoretical peak performance per rack : 1.16 PFlops (FP64), 17 PFlops (FP16) c.f. Google TPU 3.0 Pod (>100PFlops/8racks)
  - Power consumption per rack: 67.33 kW
- **Interconnect**
  - Fat-tree topology
  - Intra-rack: full bisection BW
  - Inter-rack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

Grounds for the Interconnect Design

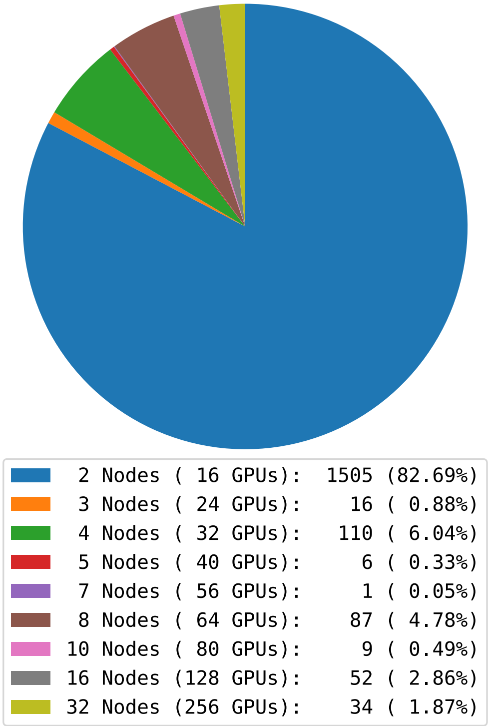
- 1/3 over-subscription network
  - A cost effective solution
    - Many DL training apps do not use large number of nodes
    - High bandwidth network is required by only a small set of nodes
- Without adoptive routing
  - InfiniBand is shared by compute and IO communication, and the delivery vendor suggested not to use adaptive routing on such a configuration
- Without Mellanox SHARP
  - We use EDR and switches which do not support large size message in SHARP

Distribution of Number of Used GPUs/Nodes in DL Jobs

Single GPU, Single Node and Multi Node jobs



 # Used Nodes in Multi Node Jobs



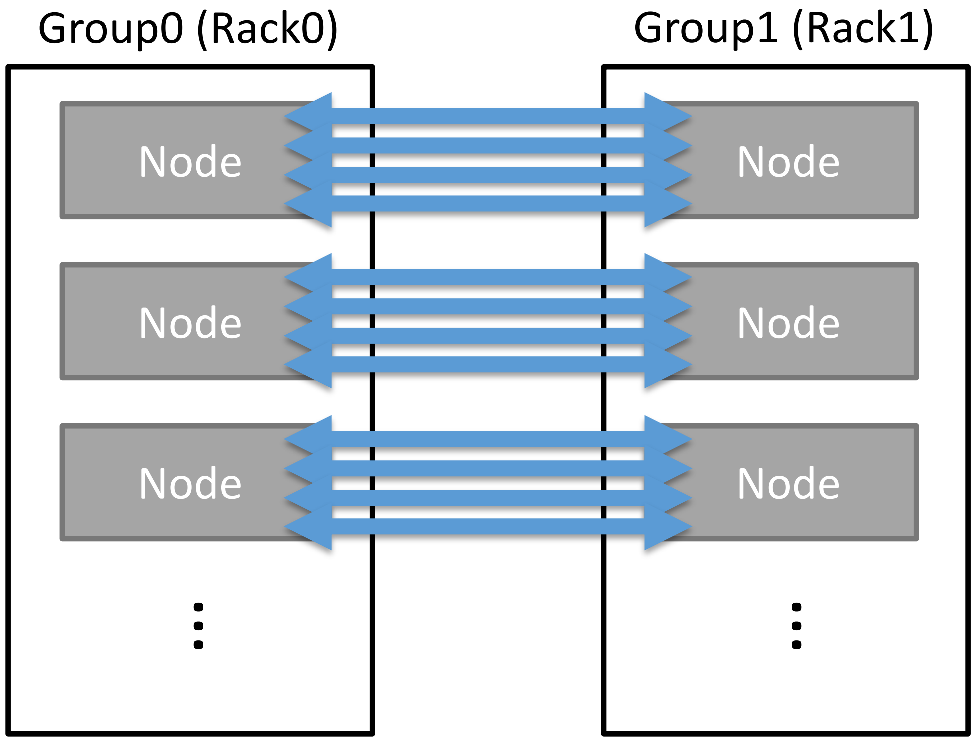
#Nodes: 32-38
#GPUs/Node: 8
#GPUs: 256-304

- Workload is collected from pre-ABCI system, AAIC
  - A system dedicated for AI research
- Single GPU Jobs are dominant and degree of parallelism is low
- ABCI is designed for a capacity computing system for AI

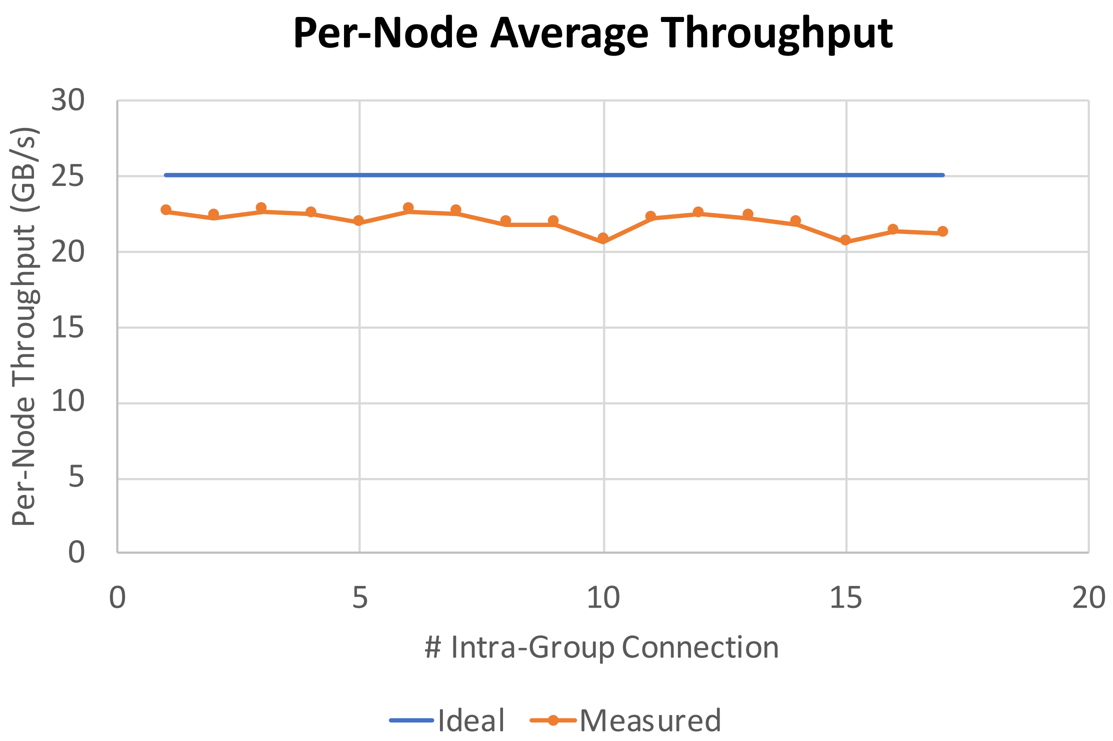
Workload is published under: https://github.com/aistairc/aaic-workload

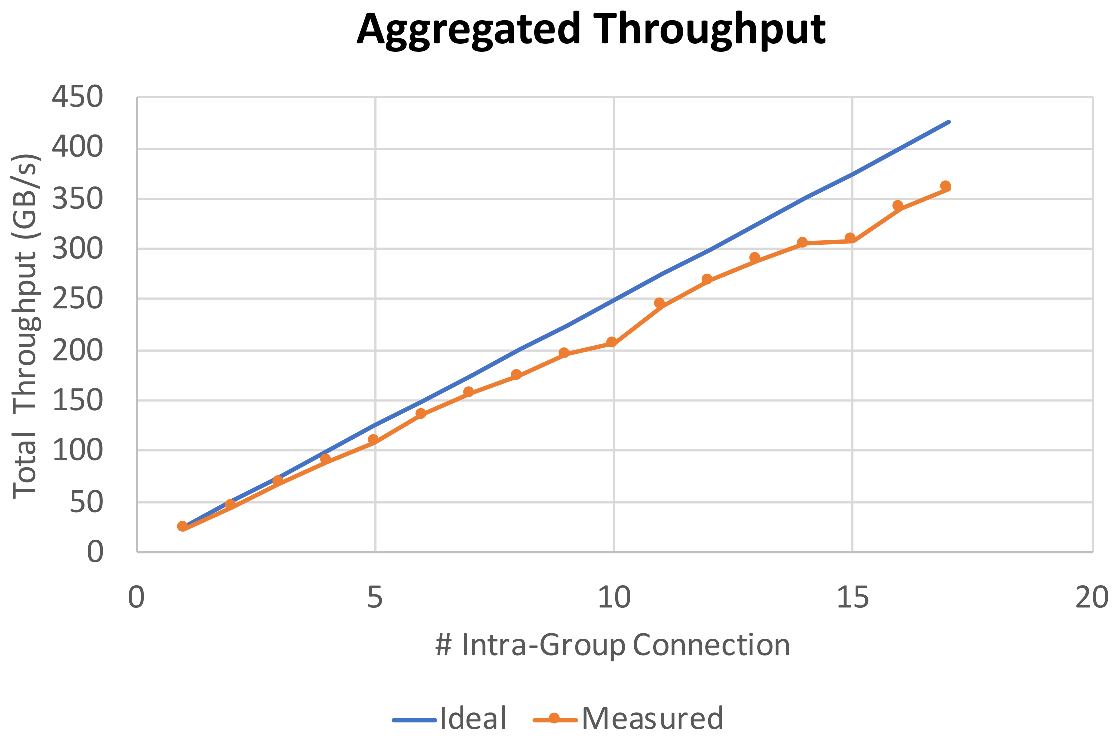
Performance Impact on MPI under 1/3 over-subscription without adoptive routing

- Measured P2P host-mem to host-mem transfer performance using OpenMPI 2.1.6
- Intra-rack: 17 - 17 nodes
- Inter-rack: 34 - 34 nodes
- Increase # of node pairs



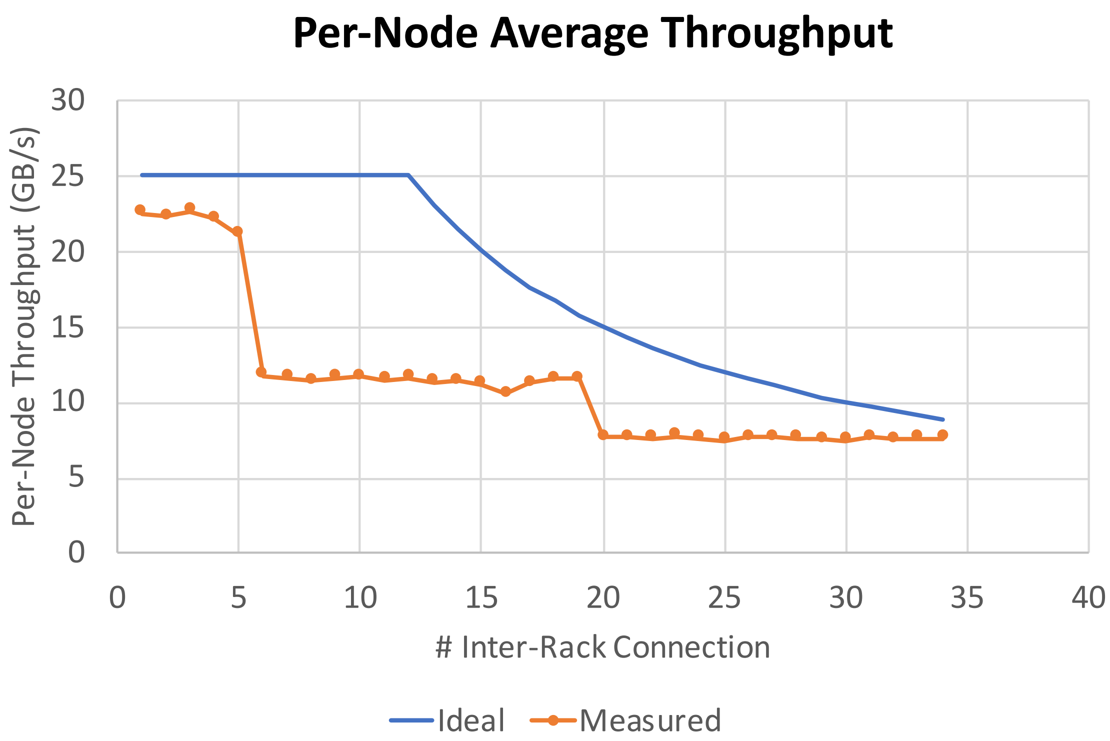
Intra-Rack P2P Performance

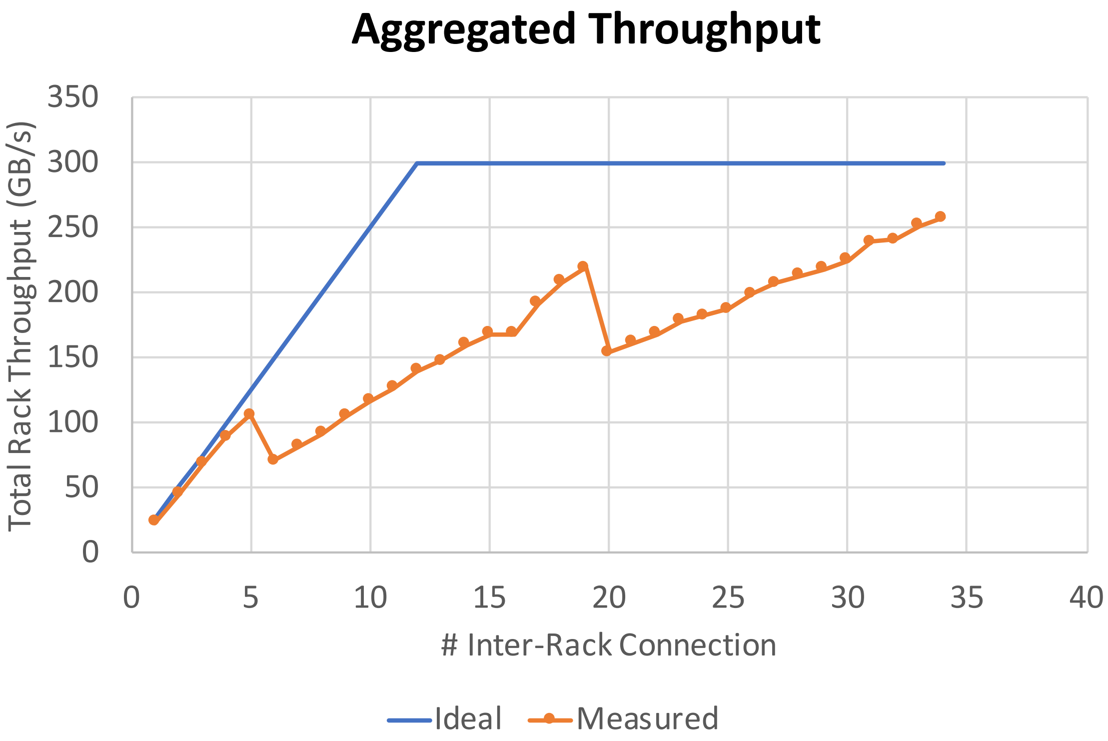




More than 80% of the theoretical performance is achieved.

Inter-Rack P2P Performance





Large performance degradation at 6 and 20 connections, and far from Ideal performance

Collective Comm. Performance on ABCI

- Measured performance of two collective comm. patterns
  - Allreduce: essential in distributed training of DL
  - Reduce: the most heavy comm. in our application
- Used OSU Micro-Benchmarks 5.6.1
  - NCCL Bench 2.0.0 is used for NCCL
  - Default parameters

Compiler	GCC 4.8.5
CUDA	10.0.130

Host to host Transfer

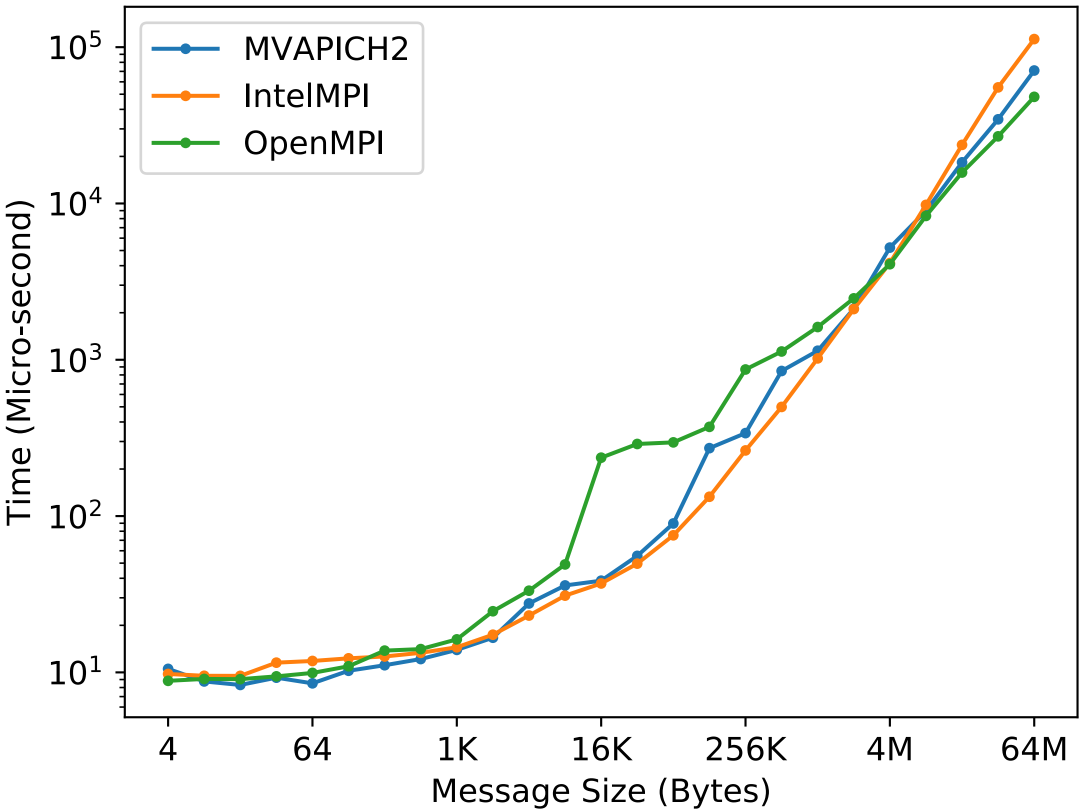
MPI	Version
MVAPICH2	MVAPICH2 2.3.1
IntelMPI	IntelMPI 2018 Update 2
OpenMPI	OpenMPI 3.1.3

GPU to GPU Transfer

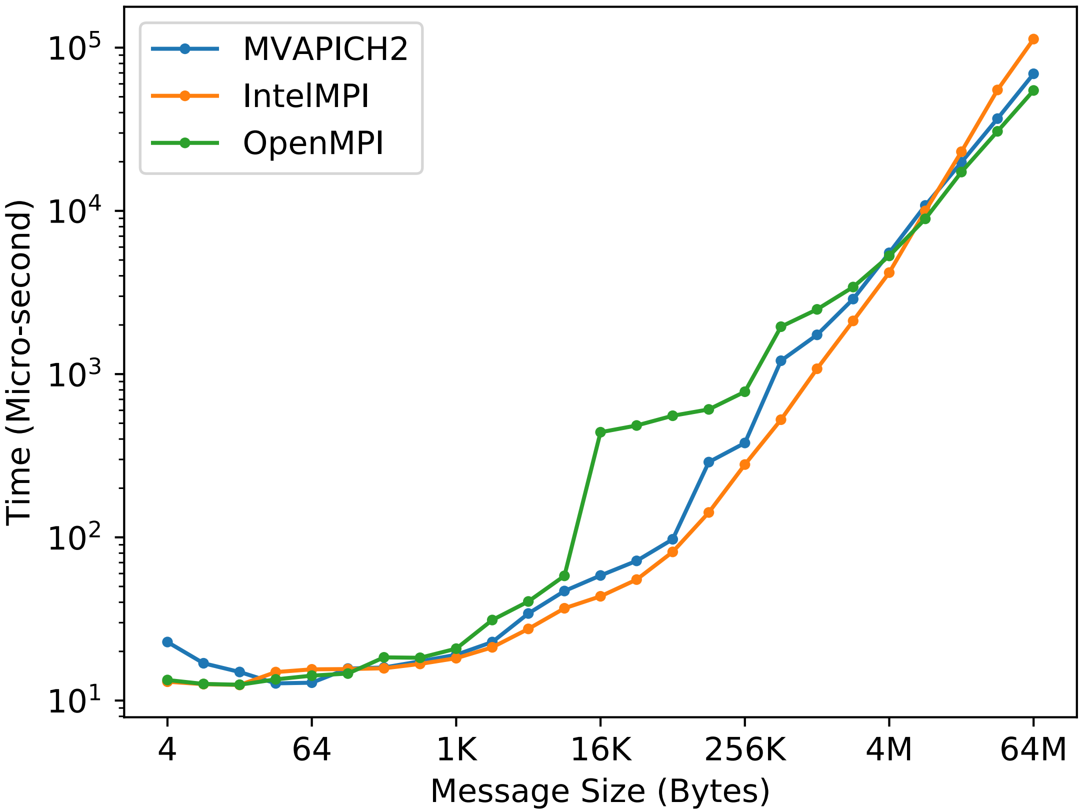
MPI, NCCL	Version
MVAPICH2	MVAPICH2 2.3.1
MVAPICH2-GDR	MVAPICH2-GDR 2.3.1, gdrcopy 1.2
OpenMPI	OpenMPI 3.1.3
NCCL	NCCL 2.4.7

H2H AllReduce

Intra-Rack (34 Nodes)

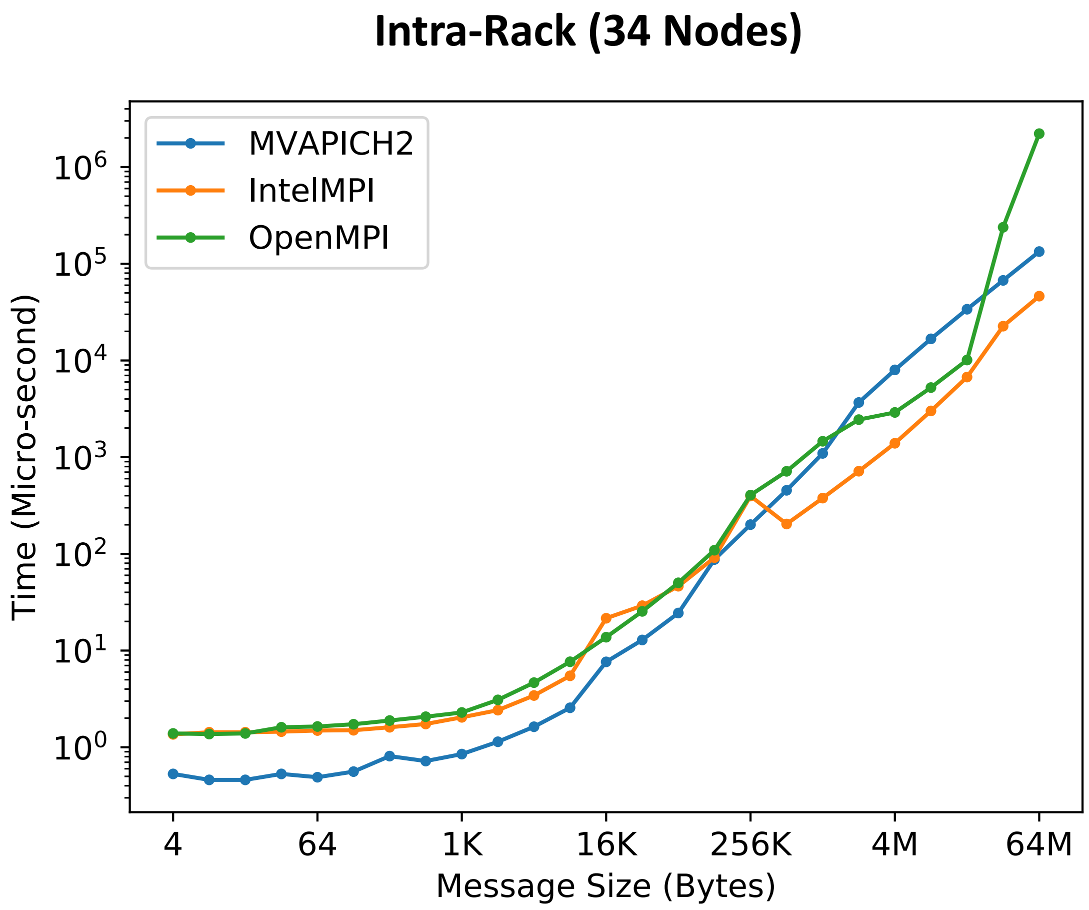


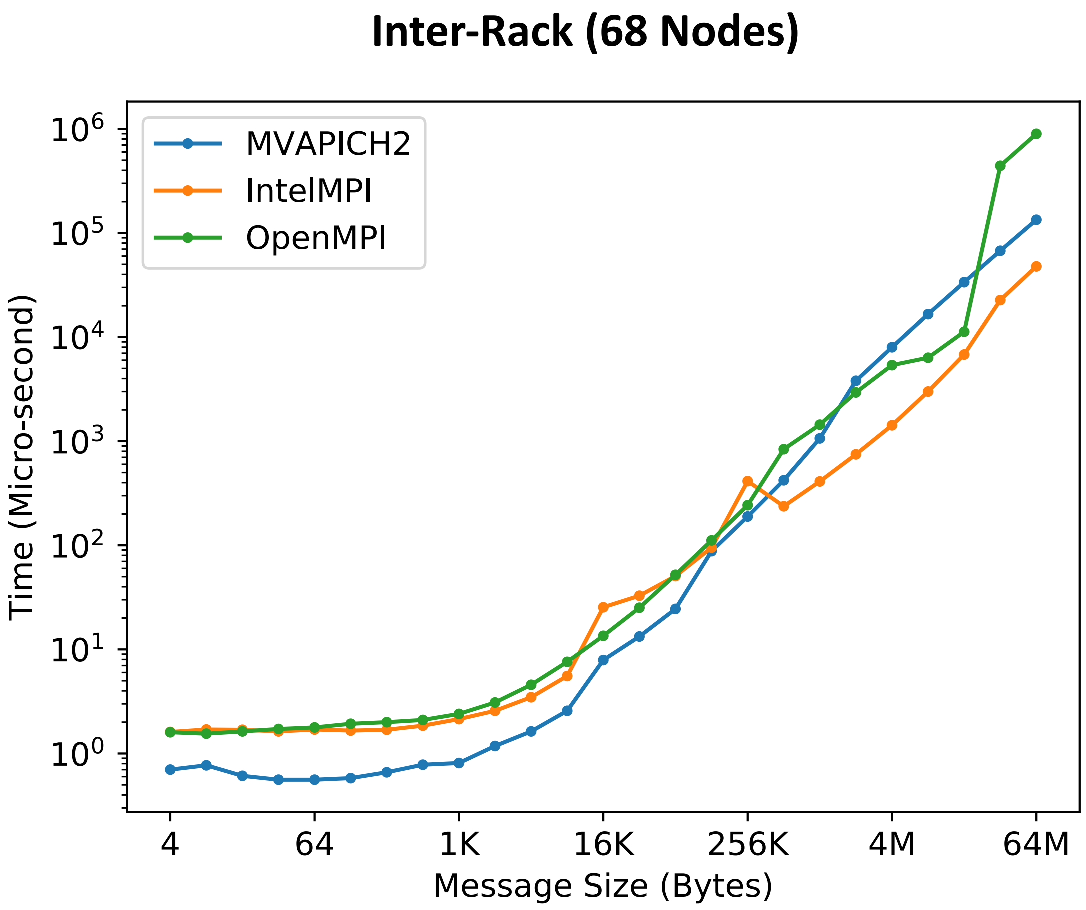
Inter-Rack (68 Nodes)



- MVAPICH2 is comparable to IntelMPI
- Performance do not suffer from 1/3 over-subscription inter-rack bandwidth

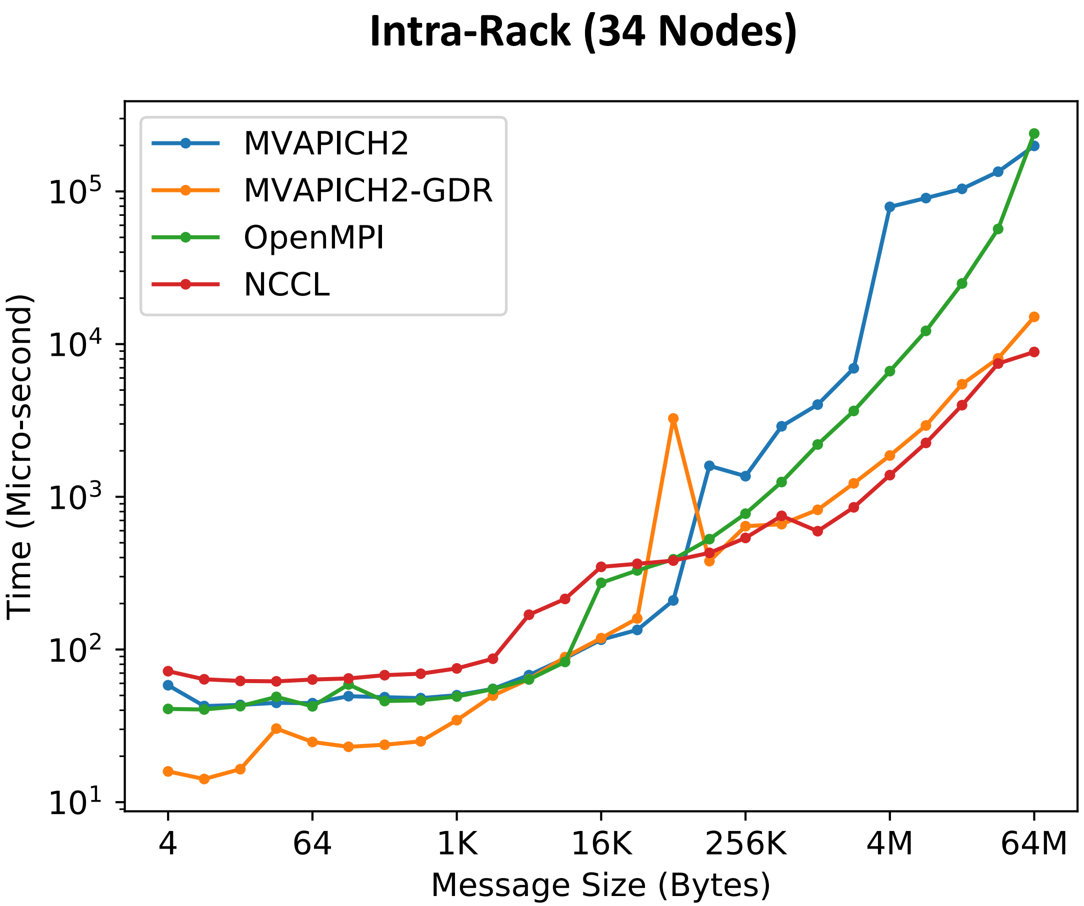
H2H Reduce

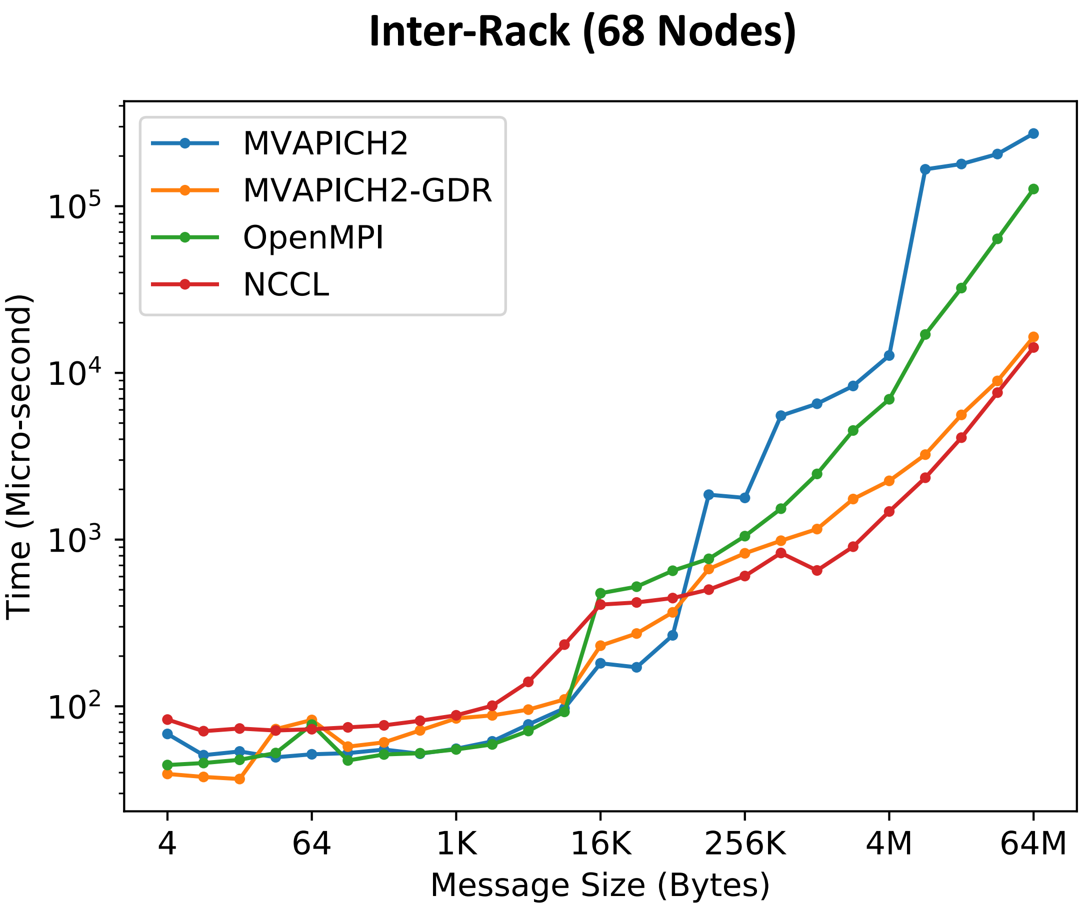




MVAPICH2 is better than others in small message, but IntelMPI outperforms MVAPICH in large message

D2D AllReduce

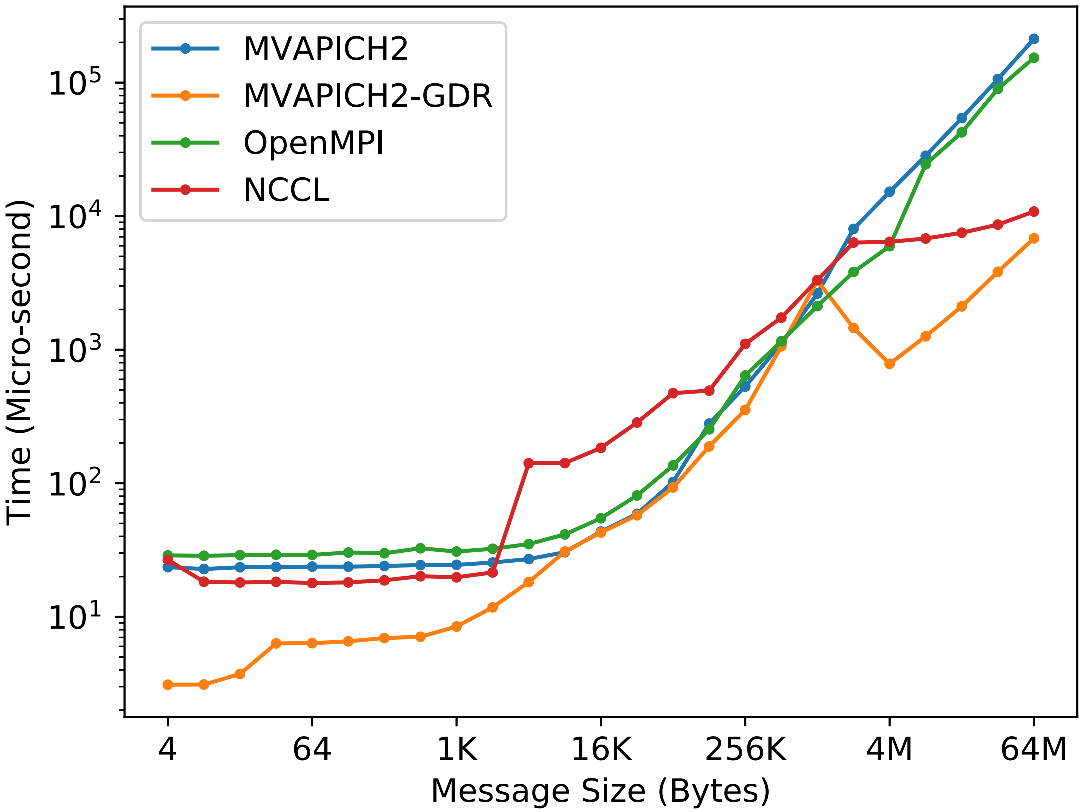




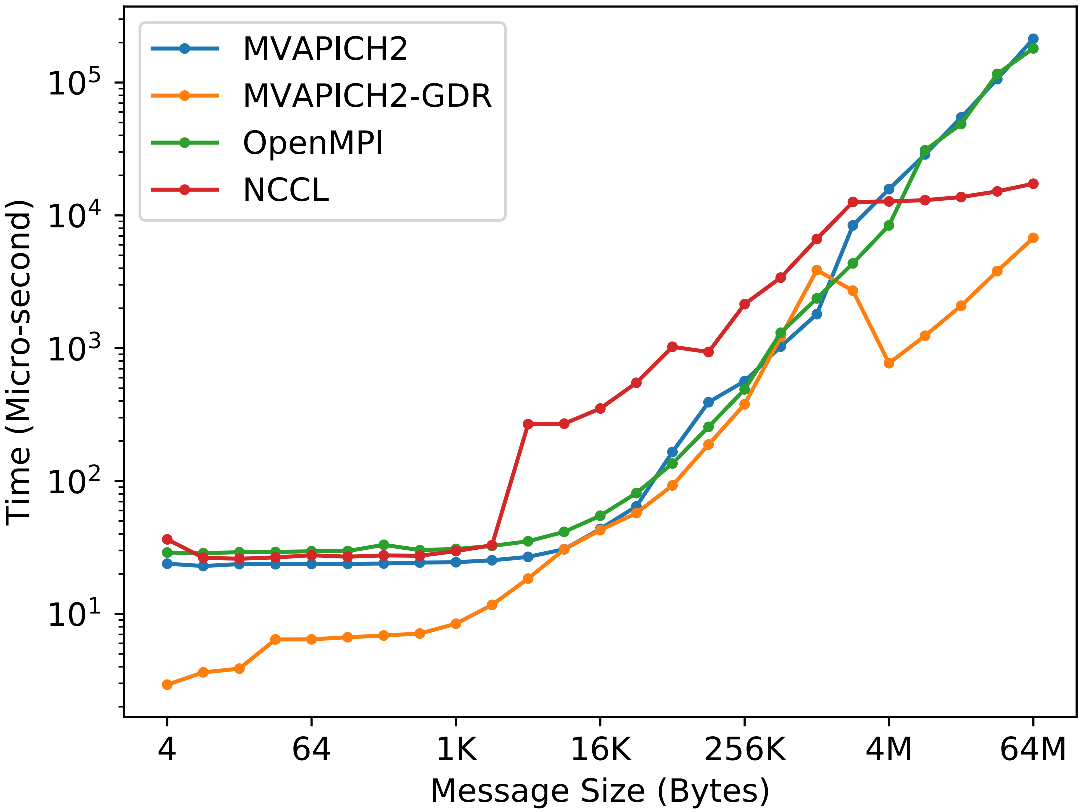
MVAPICH2-GDR and NCCL outperforms others in large messages

D2D Reduce

Intra-Rack (34 Nodes)



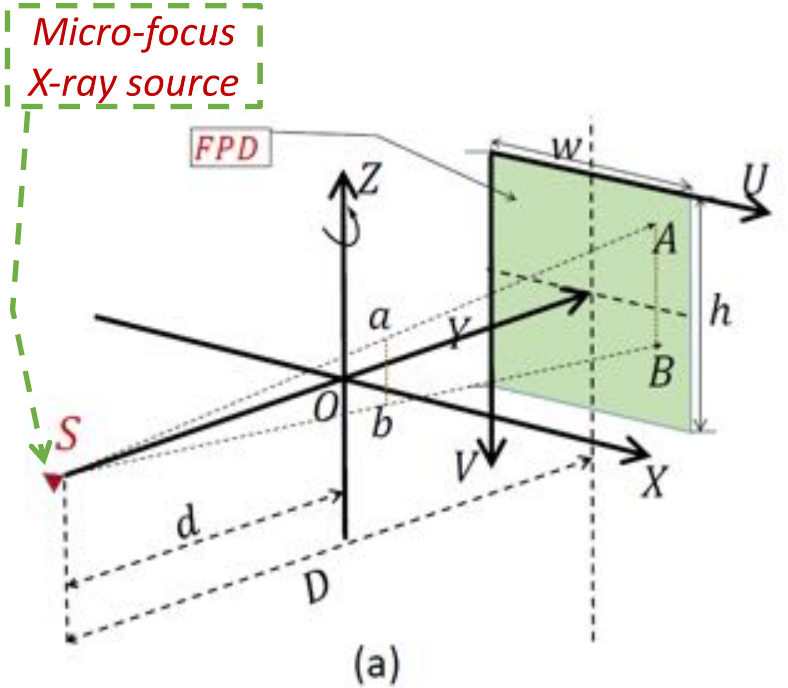
Inter-Rack (68 Nodes)



MVAPICH2-GDR achieves the best performance

Application Performance: iFDK

- 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.
- What we done
  - GPU kernel whose compute cost is 1/6 of the standard algorithm
  - Efficient FDK computation by overlapping CPU comp., GPU comp. and comm.
  - Distributed framework for high-resolution image reconstruction

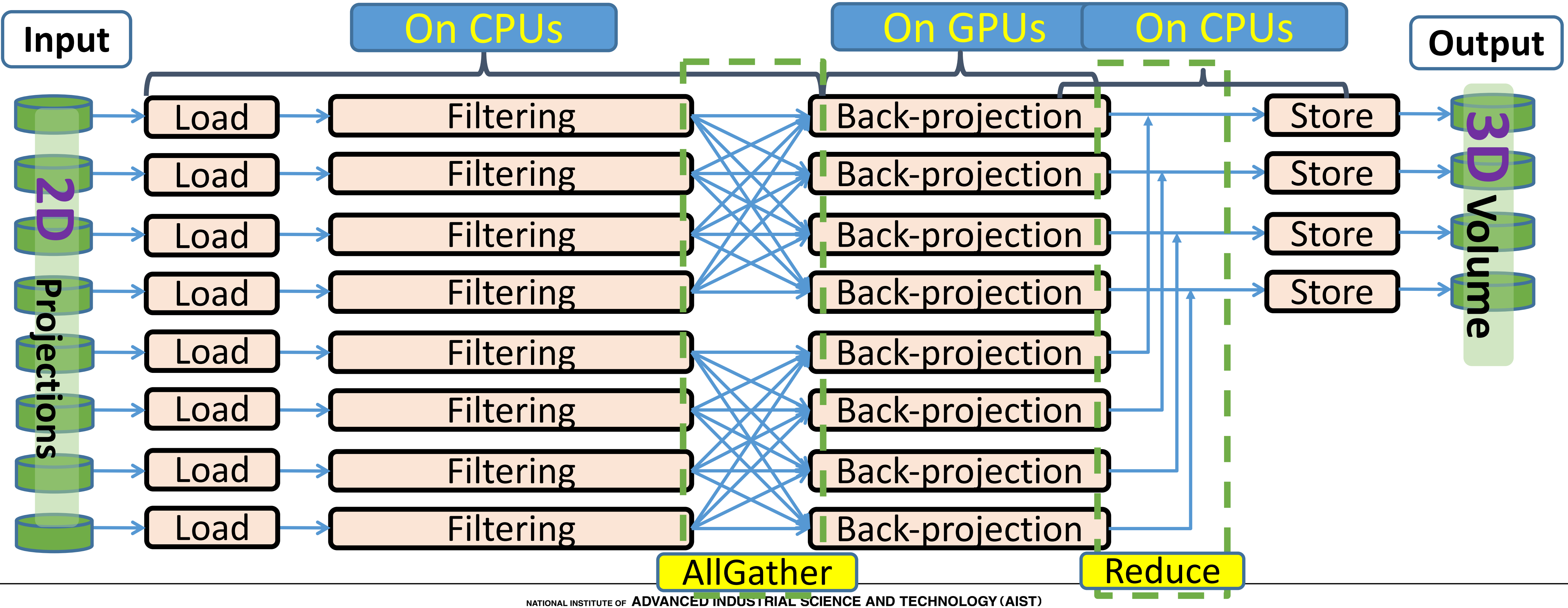


CBCT geometry and trajectory

Will be presented at SC19

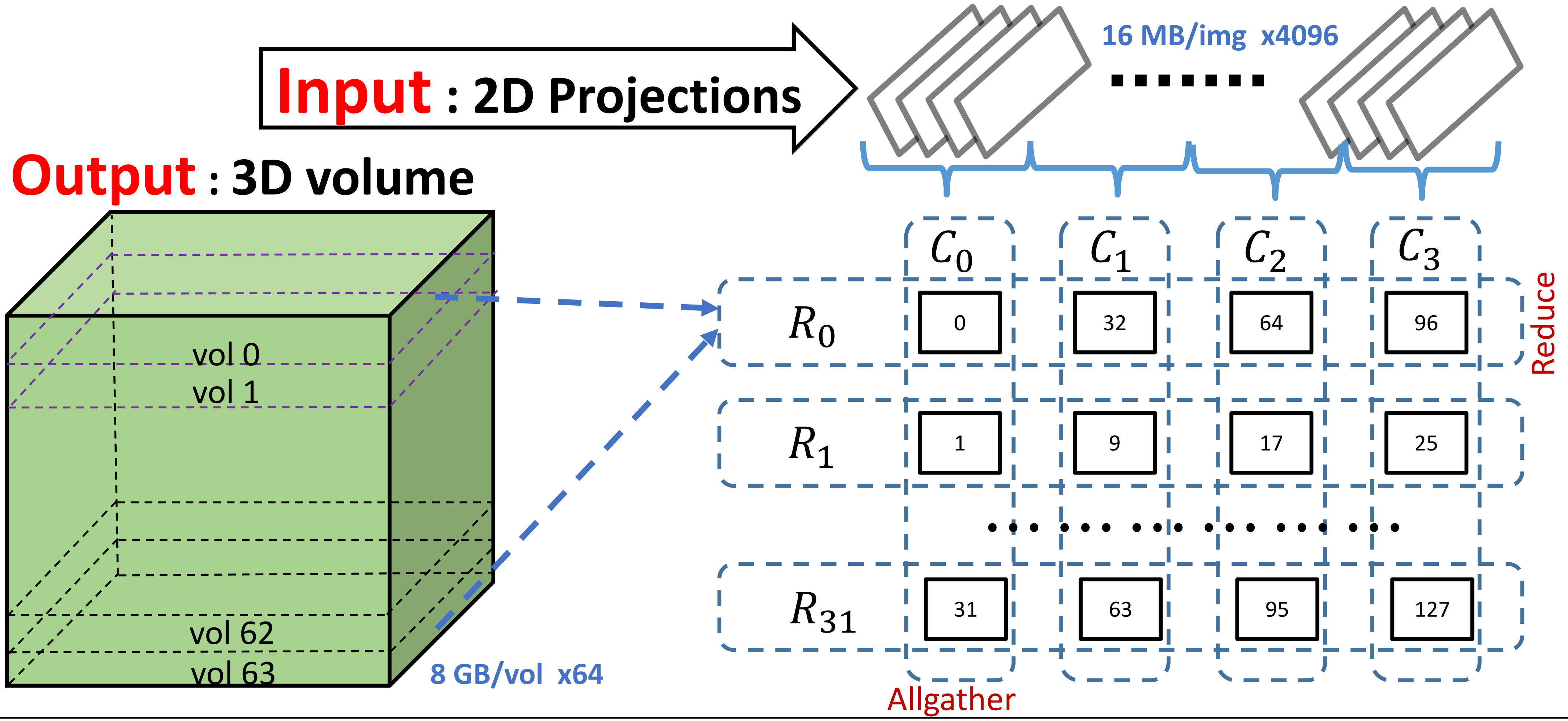
Peng Chen, Mohamed Wahib, Shinichiro Takizawa, Ryousei Takano, Satoshi Matsuoka. iFDK: A Scalable Framework for Instant High-resolution Image Reconstruction

Distributed Framework for High-resolution Image Reconstruction

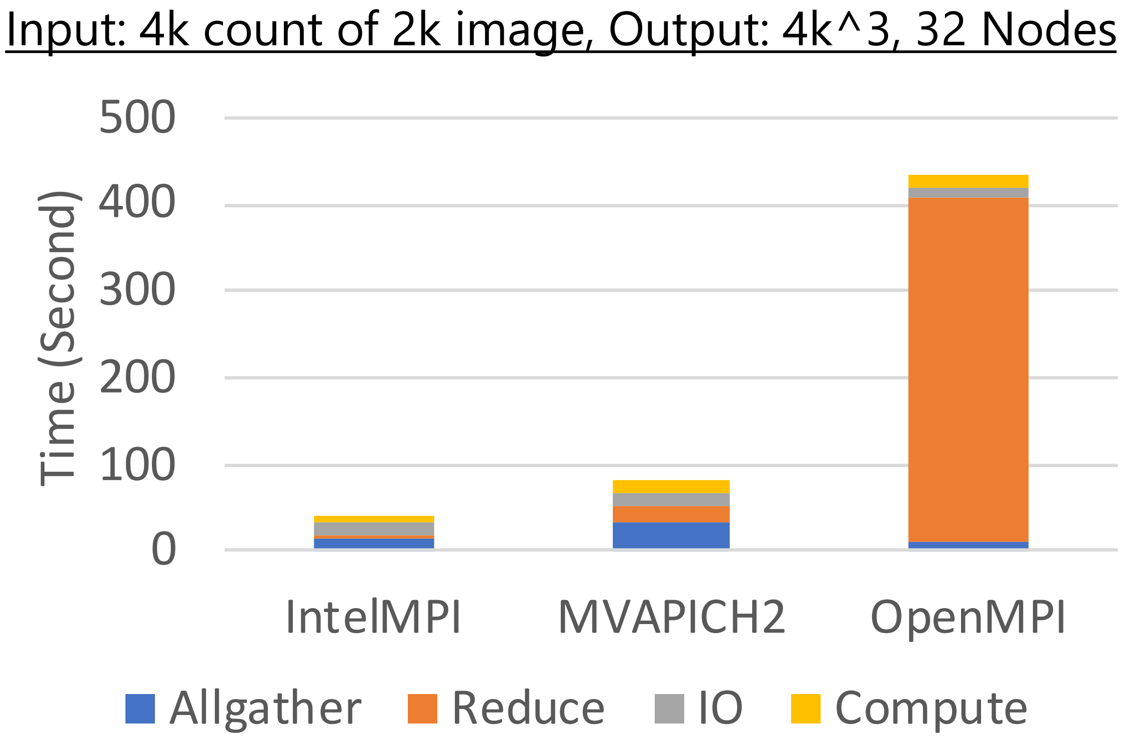


Example Case

Input: 4k count of 2k image, Output: 4k^3, 32 Nodes



Performance Result



- Large performance gap between MPI implementations.
  - Mainly by Reduction
  - Used IntelMPI in the paper
- Next step
  - Tune parameters for collectives
  - Use CUDA-aware MPI collectives

Summary

ABCI

https://abci.ai/

- Introduce ABCI, an open AI platform
  - Architecture, software stack and latest achievements
  - Detail network architecture
- MPI communication performance on ABCI
  - Effect of 1/3 inter-rack over-subscription bandwidth on P2P and collectives
  - An application performance example
- Future work
  - Evaluate the latest MVAPICH2 and MVAPICH2-GDR
  - Provide detail comm. performance results on ABCI to users