# High Performance Big Data Systems for Extreme-scale Data Science on Fugaku

International Workshop on the Integration of Simulation + Data + Learning: Towards Society5.0 by h3-Open-BDEC
16:00 – 16:40, November 30th, 2021

Kento Sato, RIKEN R-CCS

HIGPH PERFORMANCE
BIG DATA
RESEARCH TEAM

# Mission:
# Convergence of AI, Big Data and HPC

| HPC for AI/BD | AI/BD for HPC |
| --- | --- |
| Research and software development for accelerating AI/Big data workloads and applications on HPC systems (i.e., large-scale systems) | Research and software development for accelerating HPC workloads and applications by using Big Data/AI techniques |

**Fundamental R&D in HPC**

# Research projects and collaborations

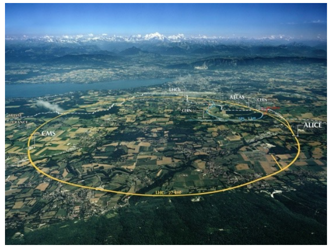
HIGPH PERFORMANCE
BIG DATA
RESEARCH TEAM

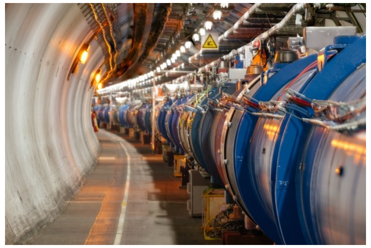
- Fundamental R&D in HPC
  - Reproducibility in MPI/OpenMP applications by record-and-replay techniques
  - Design space exploration for the next-gen supercomputers (Jens Domke, Matsuoka team, AIST)
  - Auto-detection of checkpoint variables (Nanchang Hangkong University, PNNL)
  - ABFT for tensor operations in deep learning framework (Nanchang Hangkong University, PNNL)
  - Failure analysis on Fugaku (Shoji, Yamamoto, Northeastern Univ.)
  - Benchmarking and Performance analysis of big data applications on NVDIMM (Andres Rubio Proano, FSU)
  - I/O optimization for 2D/3D sub-tiling of MPI-IO on a near-node local storage architecture (KTH)
- HPC for AI/BD
  - Data platform for Fugaku and RSC facilities, SPring-8 and SACLA (Matsuda, Kaneyama, Harada, Shoji +RSC)
  - DL4Fugaku: Deep learning framework tuning on Fugaku (Matsuoka team, Imamura team, Fujitsu)
  - Storage performance analysis and storage design exploration for deep learning (Takaaki Fukai)
- AI/BD for HPC
  - Big data compression with AI techniques (FSU)

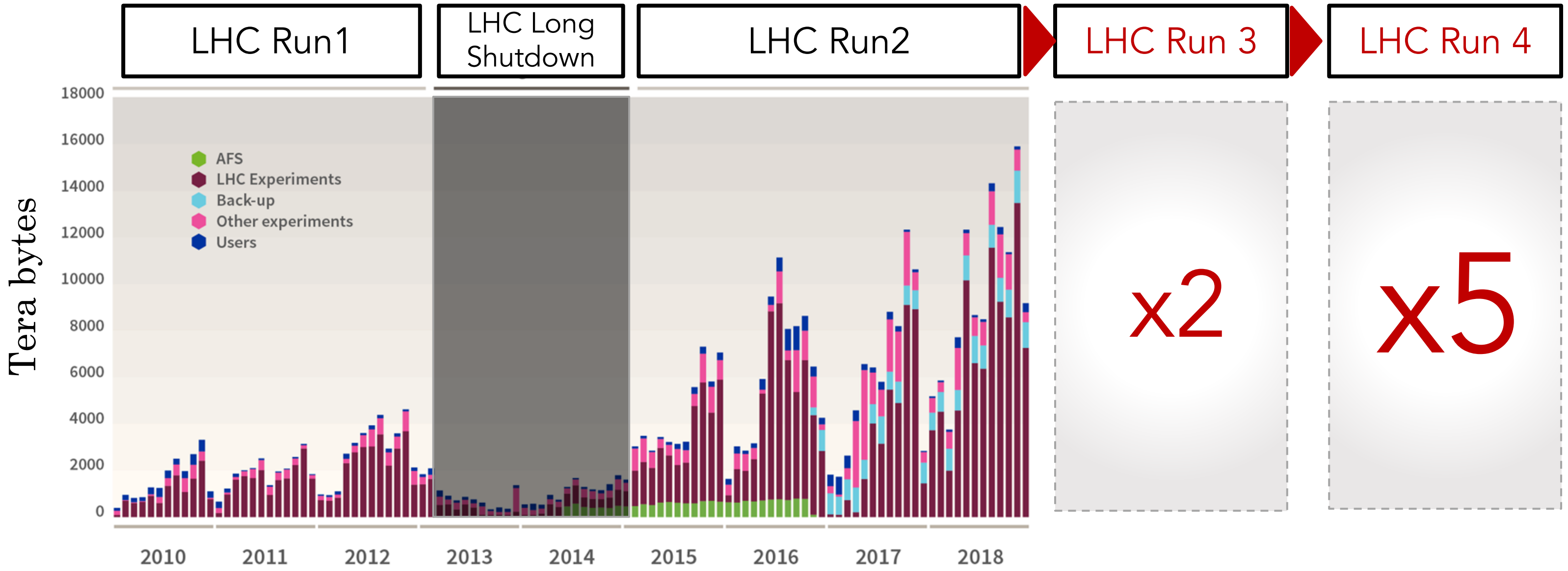
# Big Data Generation and Transfer

**Generation**: Scientific big data is generated every day all over the world

- LHC (Large Hadron Collider) in CERN generated about 88PB of data in 2018 [1]
  - *"Data archival is expected to be two-times higher during Run 3 and five-times higher or more during Run 4 (foreseen for 2026 to 2029). "*








[1] Esra Ozcesmeci, "LHC: pushing computing to the limits", https://home.cern/news/news/computing/lhc-pushing-computing-limits March 1st, 2019

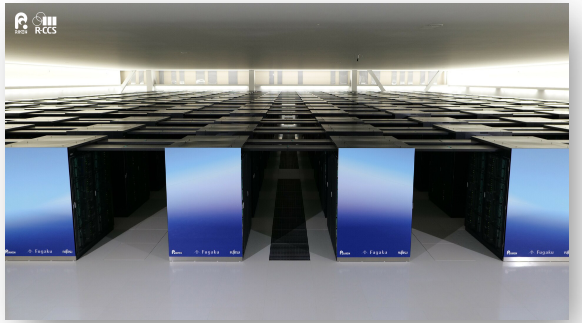
# Big Data Generation and Transfer (Cont'd)

<u>Transfer</u>: Data transfer is an essential part of data analytics

- Generated data from sensors must be transferred to internal computers for the analysis
- In some case, the facilities needs to transfer the data to external collaborators via WAN
  – e.g. ) In LHC, 830 PB of data and 1.1 billion files were transferred all over the world [1]

Sensors

Big data transfer



Internal/External Computers

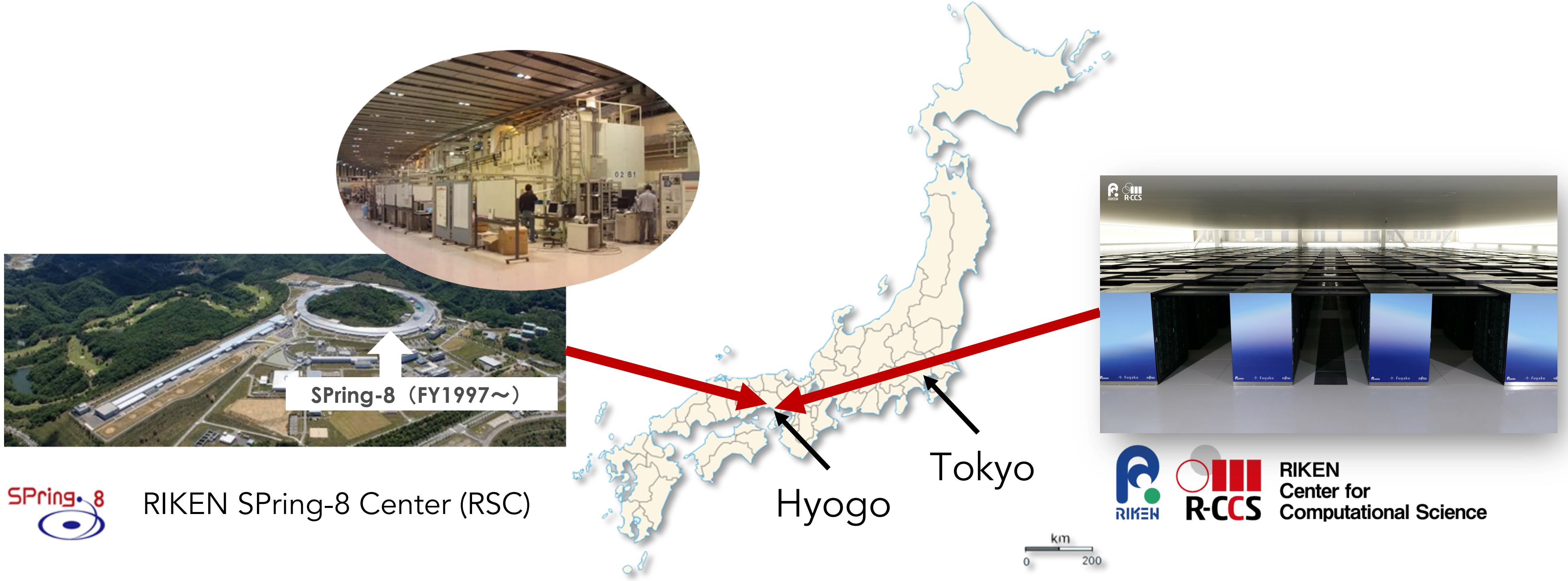
**Efficient data transfer and its management is important in big data analysis**

[1] Esra Ozcesmeci, "LHC: pushing computing to the limits", https://home.cern/news/news/computing/lhc-pushing-computing-limits March 1st, 2019

# SPring-8

RIKEN has SPring-8 large synchrotron radiation facility

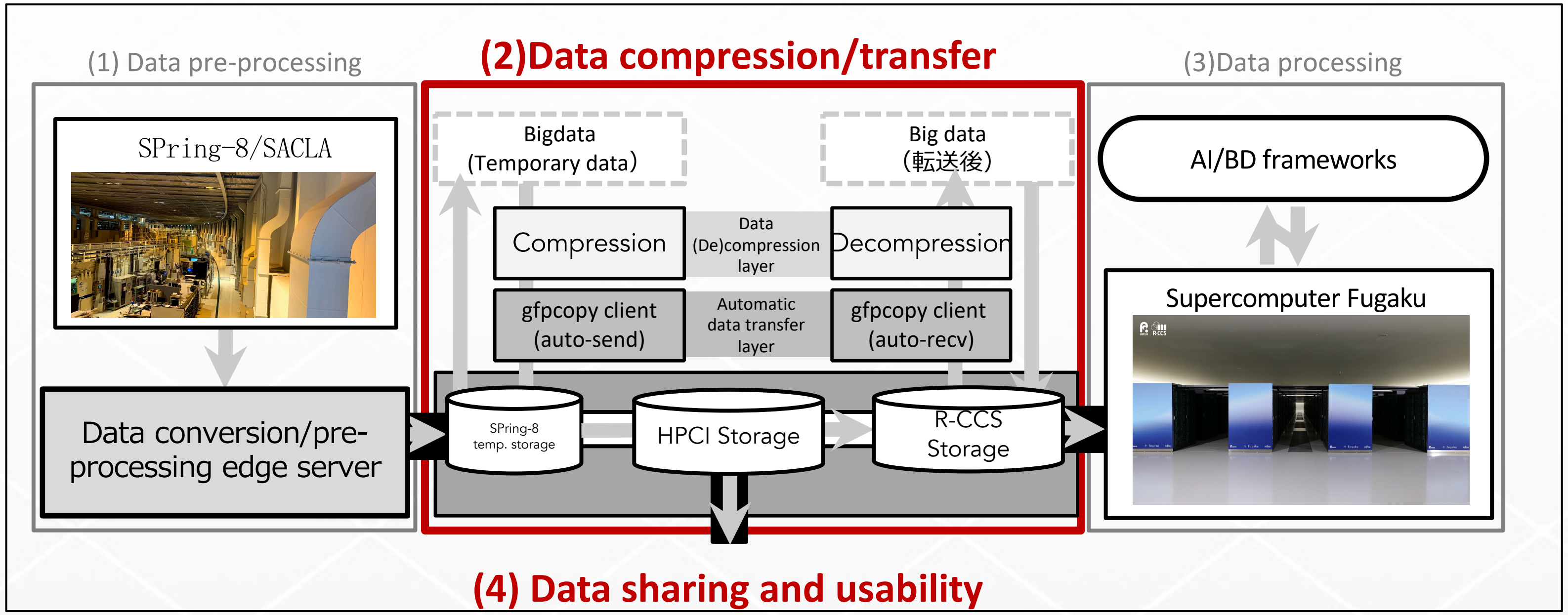
- Opened in 1997 in Harima, located in the same Hyogo prefecture as R-CCS
- Managed by RIKEN, with Japan synchrotron radiation research institute (JASRI)
- SPring-8 stands for Super Photon ring-8 GeV
  - 8 GeV (giga electron volts) is the energy of electron beam circulation in the storage ring
- Generates PB-order of big data



# Research and development of an infrastructure for collecting, analyzing and utilizing big data in large-scale research facilities (Fugaku/SPring-8/SACLA)

Project Leader: Kento Sato

## [Overview]




Project team members

| Members |
|---|
| Kento Sato, R-CCS |
| Fumiyoshi Shoji, R-CCS |
| Motohiko Matsuda, R-CCS |
| Kaneyama Hidetomo, R-CCS |
| Hiroshi Harada, R-CCS |
| Jorji Nonaka, R-CCS |
| Kentaro Sano, R-CCS |
| Masaaki Kondo, R-CCS |
| Tomohiro Ueno, R-CCS |
| Takaki Hatsui, RSC |
| Yasumasa Joti, RSC |

## [Objective]

- The Objective of this project is to establish a "big data infrastructure" that enables data collection, analysis, and utilization between SPring-8/SACLA and Fugaku. We are working on following sub-proejcts:
  - (1) Data pre-processing infrastructure: To efficiently store experimental data obtained from sensors, we perform data conversion and pre-processing at the hardware level using FPGA
  - (2) Data compression and transfer infrastructure: We develop data compression and transfer infrastructure
  - (3) Data analysis infrastructure: We will build an infrastructure (workflow tools and deep learning framework) to efficiently analyze the data in HPC systems
  - (4) Data utilization infrastructure: We will build a data utilization infrastructure to make use of the collected primary data and analysis results (e.g., Single sign-on authentication, GakuNin RDM etc.)

# Data transfer service in SACLA

- We started data transfer service from SACLA to HPCI shared storage
  - To facilitates data analitycs in HPCI systems inlucidng Fugaku
- We are planing to expand the service to SPring-8 synchortron radiation facility and enhance the usability (Common authentication scheme, GakuNin RDM etc.)

SACLA HPC:
Data Transfer Service to HPCI Shared Storage

SACLA(XFEL)

ホーム | 組織概要 | SACLAとは | ビームライン | トピックス | 研究成果 | イベント

HOME » 利用者のみなさまへ » 技術情報 » ユーティリティ

技術情報 Technical Information

HPC (High Performance Computing)

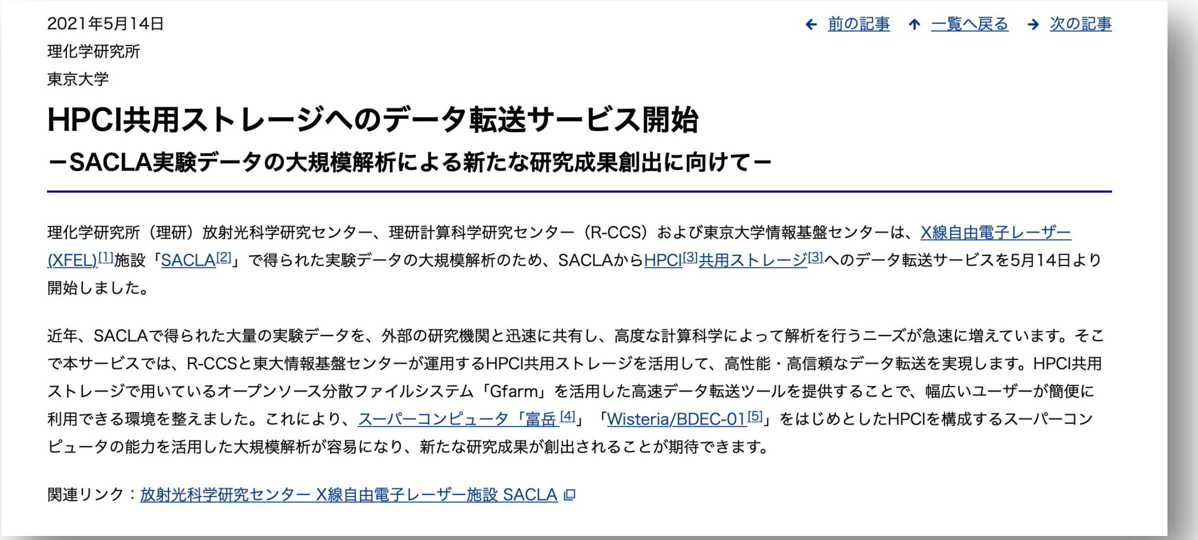
## SACLA HPC - HPCI共用ストレージへのデータ転送サービスについて

近年、SACLAで得られた大量の実験データを外部の研究機関と迅速に共有し高度な計算科学によって解析を行うニーズが急速に増えてきています。SACLAでは、そのような大規模データ解析のために、SACLAからHPCI共用ストレージへのデータ転送サービスを開始しました。 これにより、「富岳」「Wisteria/BDEC-01」をはじめとしたHPCIを構成するスーパーコンピュータの能力を活用した大規模解析が容易になり、新たな研究成果が創出されることが期待できます

技術情報: MPCCD, DAQ, HPC

Source (May 14, 2021): http://xfel.riken.jp/users/bml09-1.html

Data Transfer Service to HPCI Shared Storage
Toward the creation of innovative achievement through SACLA



2021年5月14日
理化学研究所
東京大学

← 前の記事 ↑ 一覧へ戻る → 次の記事

## HPCI共用ストレージへのデータ転送サービス開始

### －SACLA実験データの大規模解析による新たな研究成果創出に向けて－

理化学研究所（理研）放射光科学研究センター、理研計算科学研究センター（R-CCS）および東京大学情報基盤センターは、X線自由電子レーザー(XFEL)[1]施設「SACLA[2]」で得られた実験データの大規模解析のため、SACLAからHPCI[3]共用ストレージ[3]へのデータ転送サービスを5月14日より開始しました。

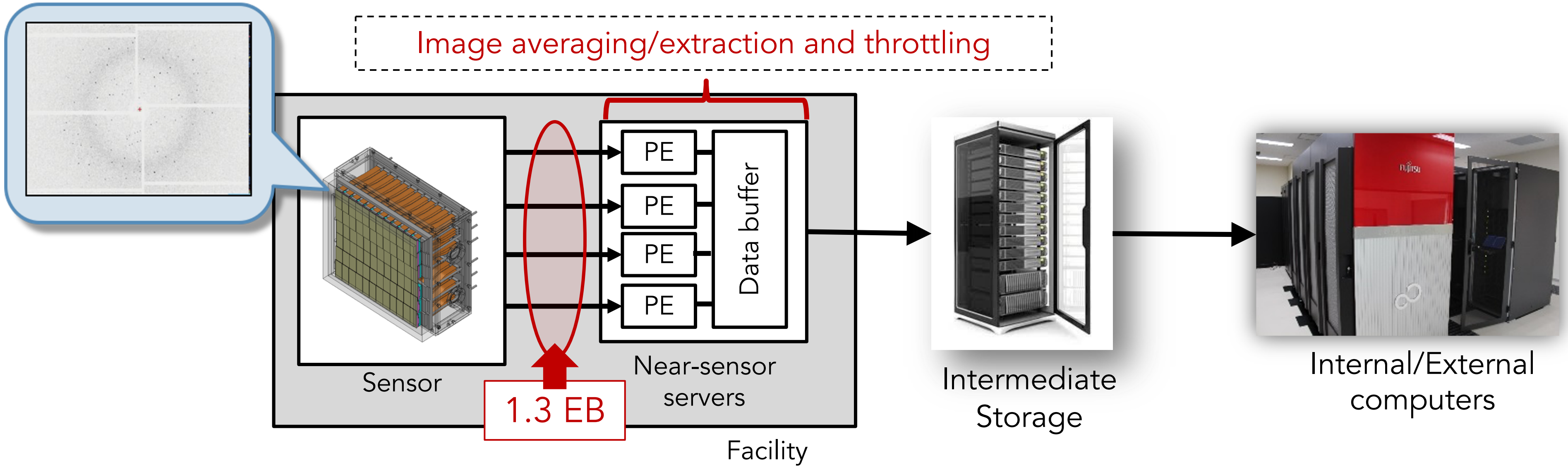
近年、SACLAで得られた大量の実験データを、外部の研究機関と迅速に共有し、高度な計算科学によって解析を行うニーズが急速に増えています。そこで本サービスでは、R-CCSと東大情報基盤センターが運用するHPCI共用ストレージを活用して、高性能・高信頼なデータ転送を実現します。HPCI共用ストレージで用いているオープンソース分散ファイルシステム「Gfarm」を活用した高速データ転送ツールを提供することで、幅広いユーザーが簡便に利用できる環境を整えました。これにより、スーパーコンピュータ「富岳[4]」「Wisteria/BDEC-01[5]」をはじめとしたHPCIを構成するスーパーコンピュータの能力を活用した大規模解析が容易になり、新たな研究成果が創出されることが期待できます。

関連リンク：放射光科学研究センター X線自由電子レーザー施設 SACLA

Source (May 14, 2021): https://www.riken.jp/pr/news/2021/20210514_1/

# Big data transfer in SPring-8

- SPring-8 public beamlines (26 BLs) generated 0.32 PB/year in 2017
- With the next generation detector (CITIUS), it is projected that the facility will generate 1.3 ExaB of raw data per year in 2025
  - Actual transfer size can be reduced to 100-400 PB by
    - Image averaging/extraction
    - Reducing duty ratio to throttle data generation rate




We are trying to further compress this big data to accelerate data transfer from sensors to HPC systems

# Compression of Time Evolutionary Image Data through Predictive Deep Neural Networks (CCGrid2021) [1]

Rupak Roy†1 , Kento Sato†2, Subhadeep Bhattacharya†2, Xingang Fang†2, Yasumasa Joti†3,
Takaki Hatsui†3 , Toshiyuki Nishiyama Hiraki†3, Jian Guo †2and Weikuan Yu†1
†1 Florida State University, †2 RIKEN Center for Computational Science, †3 RIKEN SPring-8 Center, †4 Anhui University of Finance and Economics

- We proposed new AI-driven data compressor (TEZIP) for time evolutaionary data
- We achieved higher compration ration compared to existing video encoder (Zstd, HFYU, FFV1, x.265 )

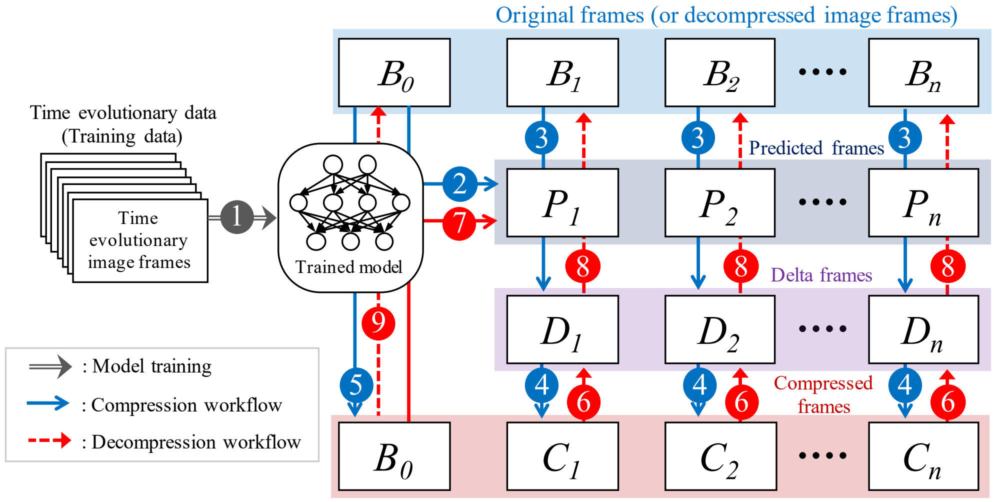



Fig. 1. Workflows of TEZIP (de)compression

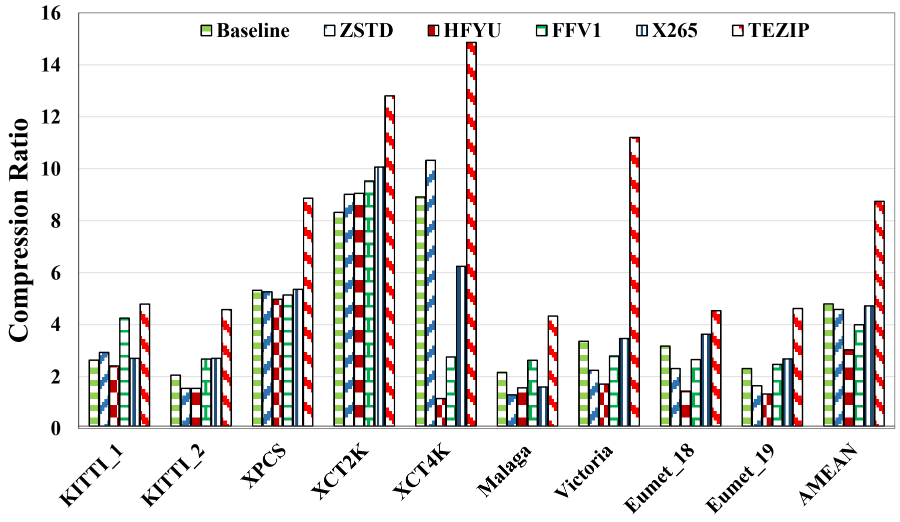
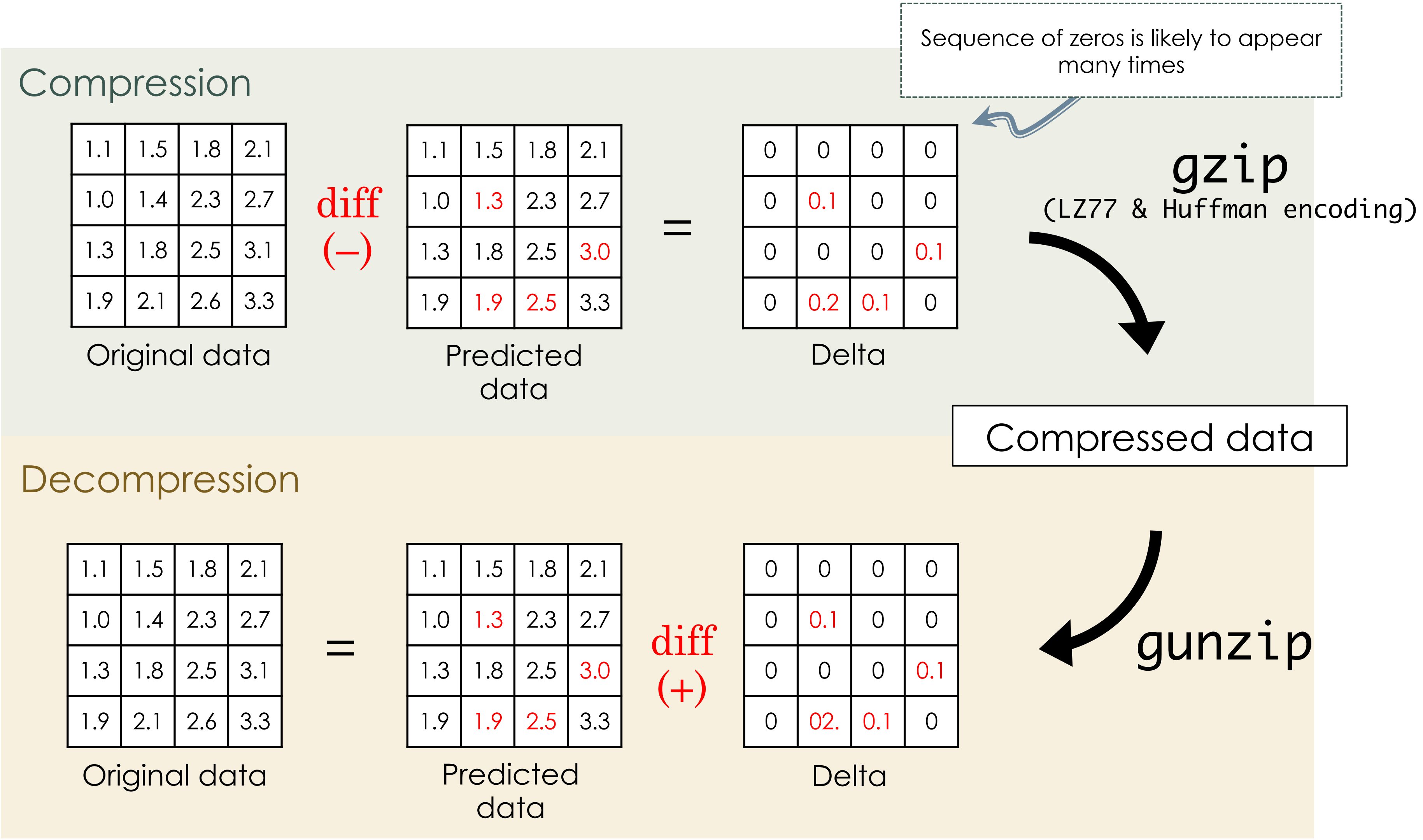



Fig. 8. Compression ratio with lossless compressors.

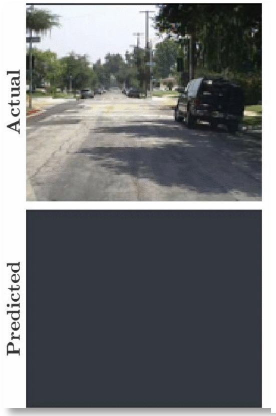
[1] Rupak Roy, Kento Sato, Subhadeep Bhattacharya, Xingang Fang, Yasumasa Joti, Takaki Hatsui, Toshiyuki Hiraki, Jian Guo and Weikuan Yu, "Compression of Time Evolutionary Image Data through Predictive Deep Neural Networks", In the proceedings of the 21 IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2021), May, 2021

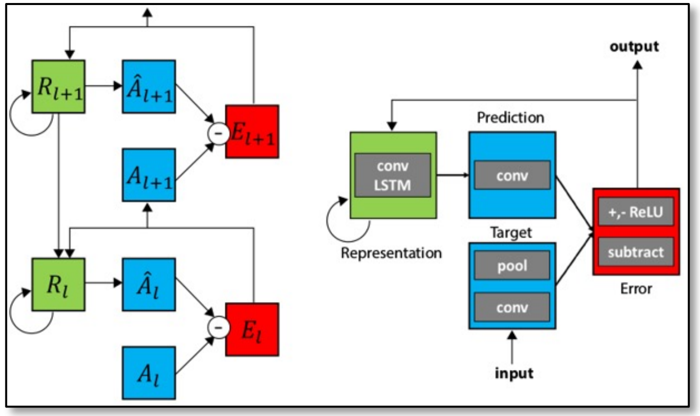
# Prediction is one of keys for good compression



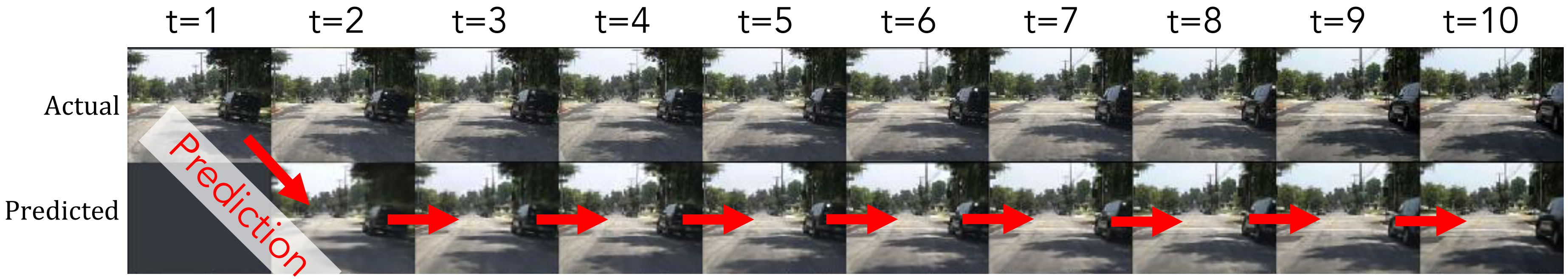
# We use deep neural network (PredNet) for prediction

- PredNet [1]
  - Deep recurrent convolutional neural network
  - Given a frame of pictures/video, this NN can predict multiple future frames




https://coxlab.github.io/prednet/



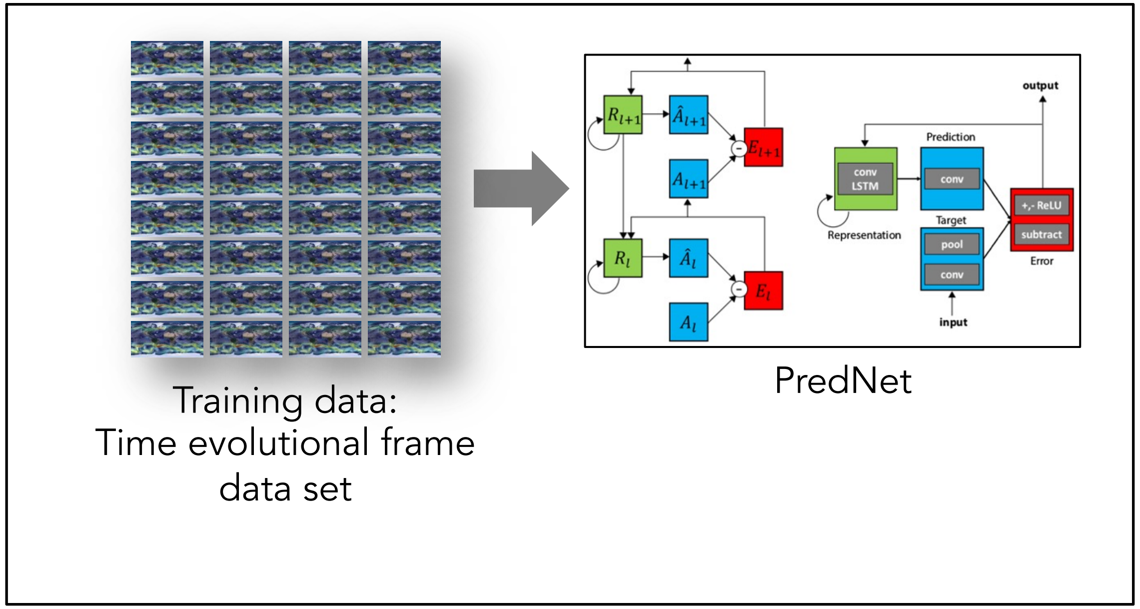


[1] Lotter, W., Kreiman, G., Cox, D.: Deep predictive coding networks for video prediction and unsupervised learning. arXiv preprint arXiv:1605.08104 (2016)

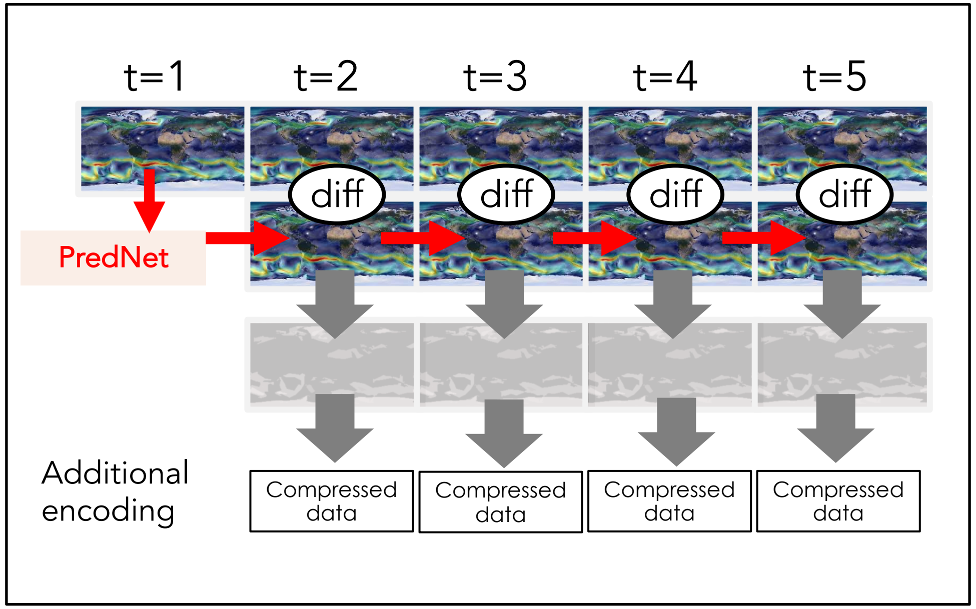
# Compression: Predict future frames and encode

- We train PredNet to learn how pixels move and how fast
  - i.e.) Giving a number of time evolutional frames to PredNet
- When compressing frames from t=1 to t=5, we predict future frames from original data (t=1)
- We compute diff, apply series of encoding
- We only store (1) base frame data (t=1) and (2) compressed data

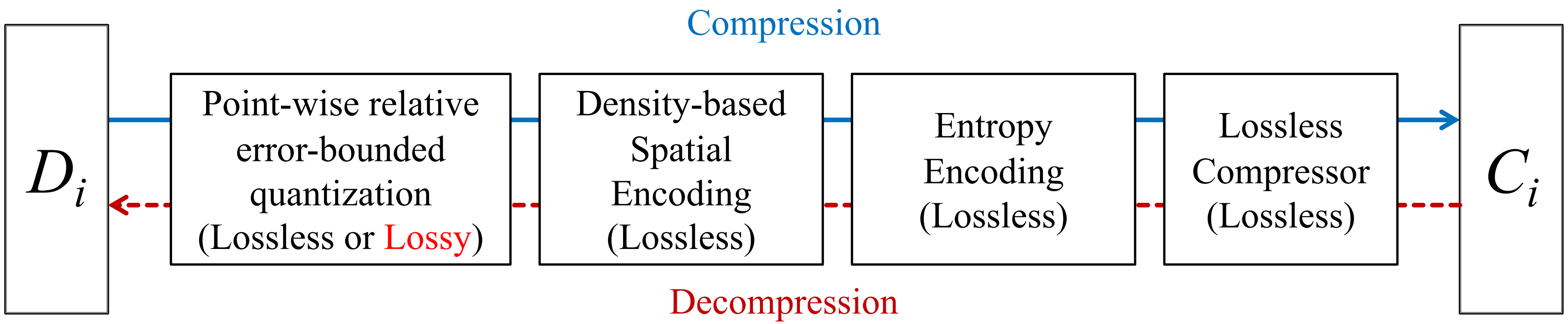
Training




Inference + Data compression



# Encoding workflow




- Quantization is a key data conversion to give good compression rate
  - This data conversion tries to maximize data compression rate while bounding certain-level errors
- Point-wise relative error bound
  - All the individual values are kept below a specified error bound threshold ($\varepsilon$)
  - Formulation
    - Give original data: $D = \{d_0, d_1, \ldots,\}$ and quantized data: $D' = \{d'_0, d'_1, \ldots,\}$
    - The following inequality holds for each data point:

$$\max_{d_i \in D,\, d'_i \in D'} \left| \frac{d_i - d'_i}{d_i} \right| \leq \varepsilon$$

# Point-wise relative error-bounded quantization (Lossless)

Example 1: error-bound ($\varepsilon$) = 0 (➔ lossless)

- Option: -M PW_REL -P 0

$$
\underbrace{\begin{bmatrix} 60 & 40 & 20 \\ 40 & 20 & 60 \\ 20 & 20 & 60 \end{bmatrix}}_{B_i}
-
\underbrace{\begin{bmatrix} 60 & 40 & 25 \\ 45 & 10 & 55 \\ 25 & 25 & 60 \end{bmatrix}}_{P_i}
=
\underbrace{\begin{bmatrix} Di(0)=0 & Di(1)=0 & Di(2)=-5 \\ Di(3)=-5 & Di(4)=10 & Di(5)=5 \\ Di(6)=-5 & Di(7)=-5 & Di(8)=0 \end{bmatrix}}_{D_i}
$$

Pass delta values to density-based spatial encoding as they are

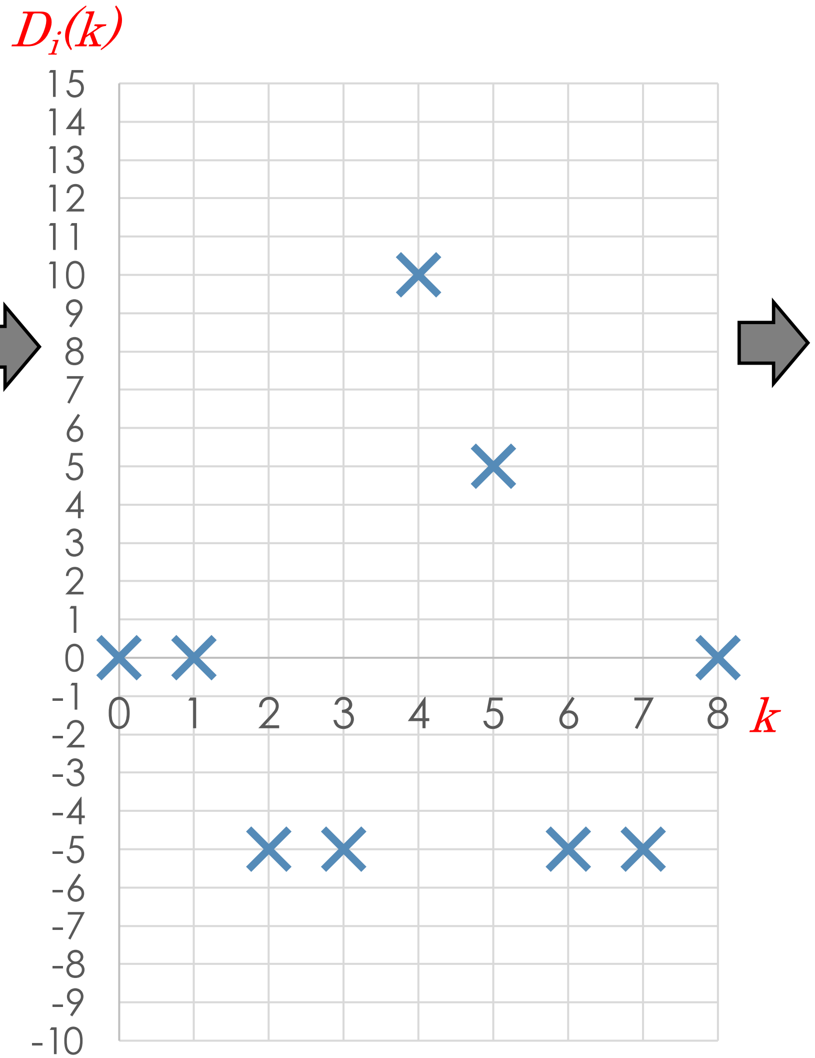



Fig. 2. $D_i(k)$: One-dementional vector expression of $D_i$

Density-based Spatial Encoding (lossless)

# Point-wise relative error-bounded quantization (lossy)

Example 2: error-bound ($\varepsilon$) = 0.1 (➔ Lossy 10% of errors )

- Option: -M PW_REL -P 0.1

$E_i$(k) is error bound of $D_i$(k)
For example, $E_i(0)$ = [-6, 6], $E_i(5)$ = [-1, 11],

| | | |
|---|---|---|
| 60±6 | 40±4 | 20±2 |
| 40±4 | 20±2 | 60±6 |
| 20±2 | 20±2 | 60±6 |

$B_i$

−

| | | |
|---|---|---|
| 60 | 40 | 25 |
| 45 | 10 | 55 |
| 25 | 25 | 60 |

$P_i$

=

| | | |
|---|---|---|
| 0±6 | 0±4 | -5±2 |
| -5±4 | 10±2 | 5±6 |
| -5±2 | -5±2 | 0±6 |

$E_i$

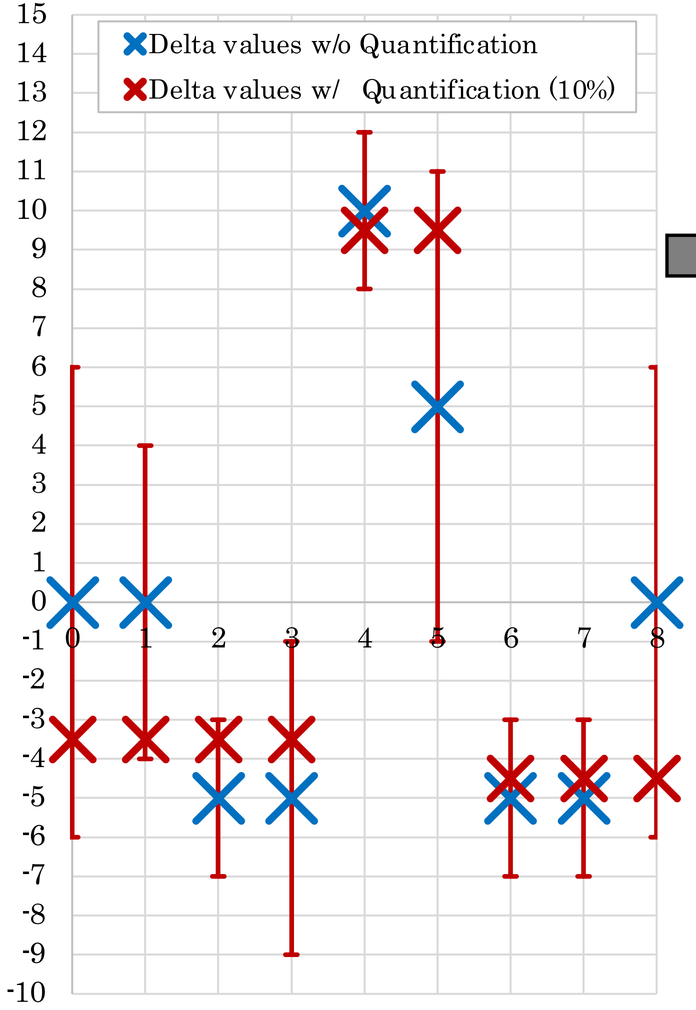



Fig. 2. $D_i(k)$: One-dementional vector expression of $D_i$

| | | |
|---|---|---|
| -3.5 | -3.5 | -3.5 |
| -3.5 | 9.5 | 9.5 |
| -4.5 | -4.5 | -4.5 |

Density-based Spatial Encoding (lossless)

The lossy mode allows delta values to be manipulated withint the error-bound (→ *window*)

Compute a common range of windows from the beginning until it becomes empty

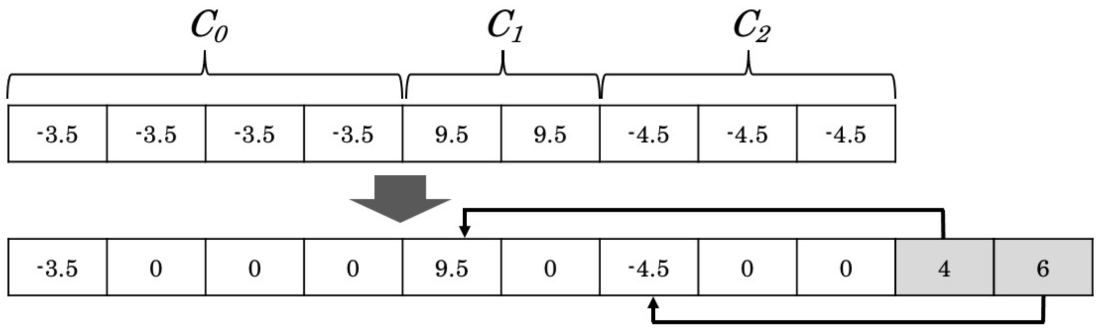
- e.g.) The first common range is [-3, -4]

Continue to the end values and manimipulate values to average values of each common range

- e.g.) [-3, -4] → -3.5, [8, 11]→9.5, [-7, -3]→ -4.5

# Following encodings

- Density-based Spatial Encoding
  - After the quantization, we see sequences of the same values (→ clusters)
  - We detect clusters and store delta in each cluster
    - It is likely that this encoding results in sequences of zeros




- Entropy Encoding
  - Replace highly recurrent values with smaller bits and replace less recurrent values with longer bits
  - e.g.) {0, 0, …,0} → 0 , {-3.5, -3.5, …, -3.5} → 1, {-9.5, -9.5, … ,  -9.5} → 2

- Apply lossless compressor
  - We used Zstd in this work

# TEZIP achieves high compression rate with comparable compression time

- TEZIP achieves an improvement up to 3.2× in terms of compression ratio.
- On average, lossless TEZIP delivers 2.1× better compression ratio compared to the second-best lossless compressor x265
- "Baseline" computes delta values from the previous frame

- TEZIP outperforms other lossless compressors for four datasets
- Overall, TEZIP performs 28% better than x265, while being comparable to FFV1

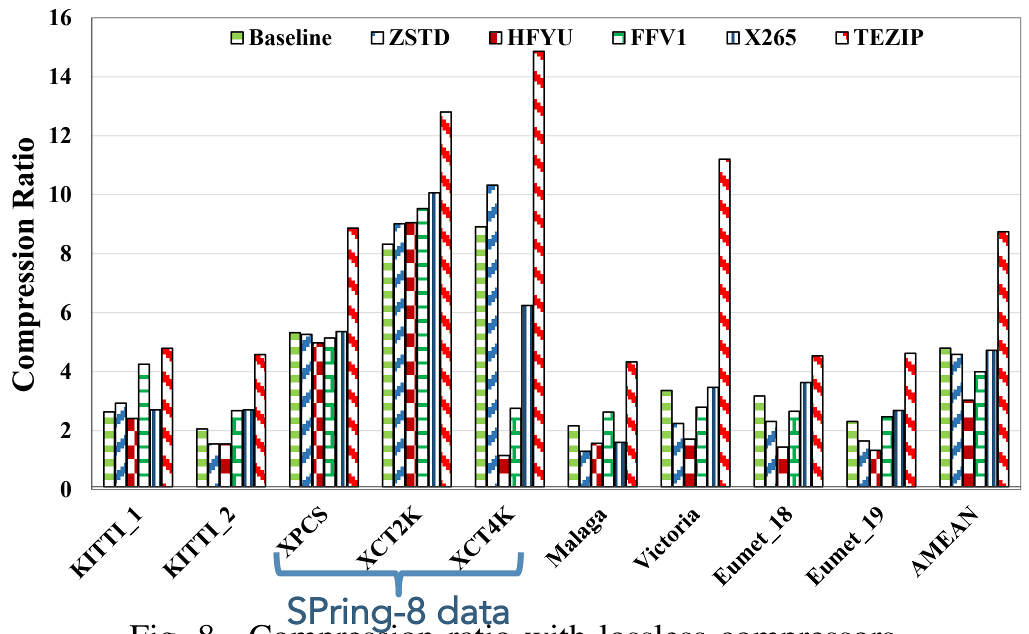



Fig. 8. Compression ratio with lossless compressors.

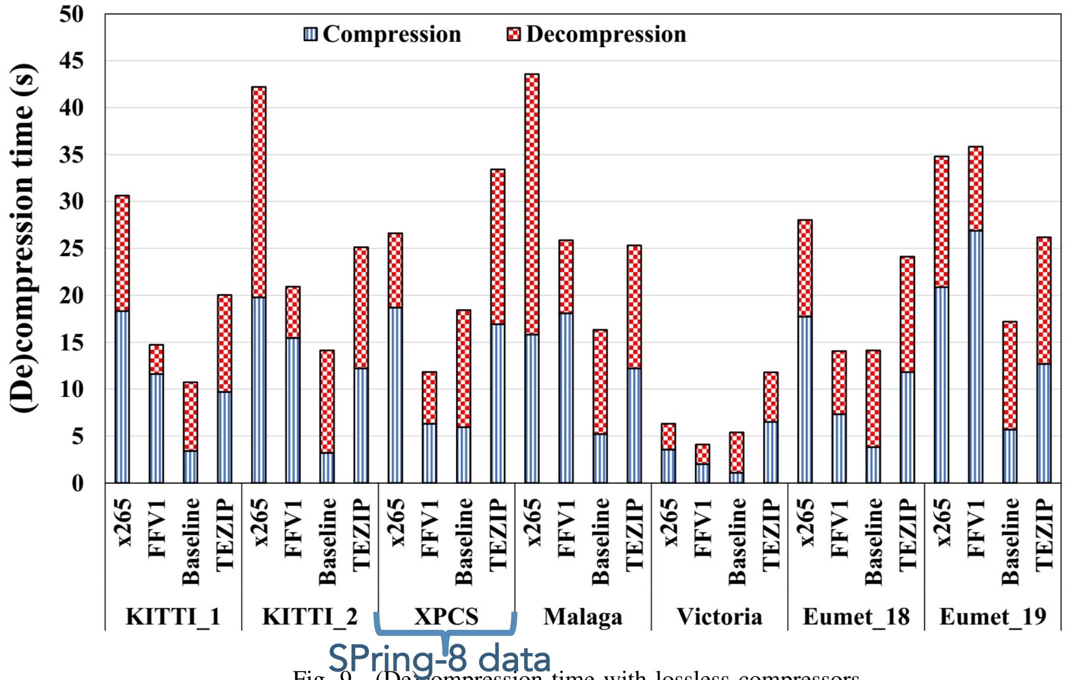



Fig. 9. (De)compression time with lossless compressors.

# Lossy compression mode in TEZIP futher improves compression ratio under the same error-bound

- TEZIP's lossy compression mode can set point-wise relative error-bound at quantization
  - The error-bound is set for SZ and TEZIP (configurable) based on errors in ZFP (unconfigurable)
- Results
  - TEZIP achieves an improvement up to 3.3x than the second best (SZ) in terms of compression ratio

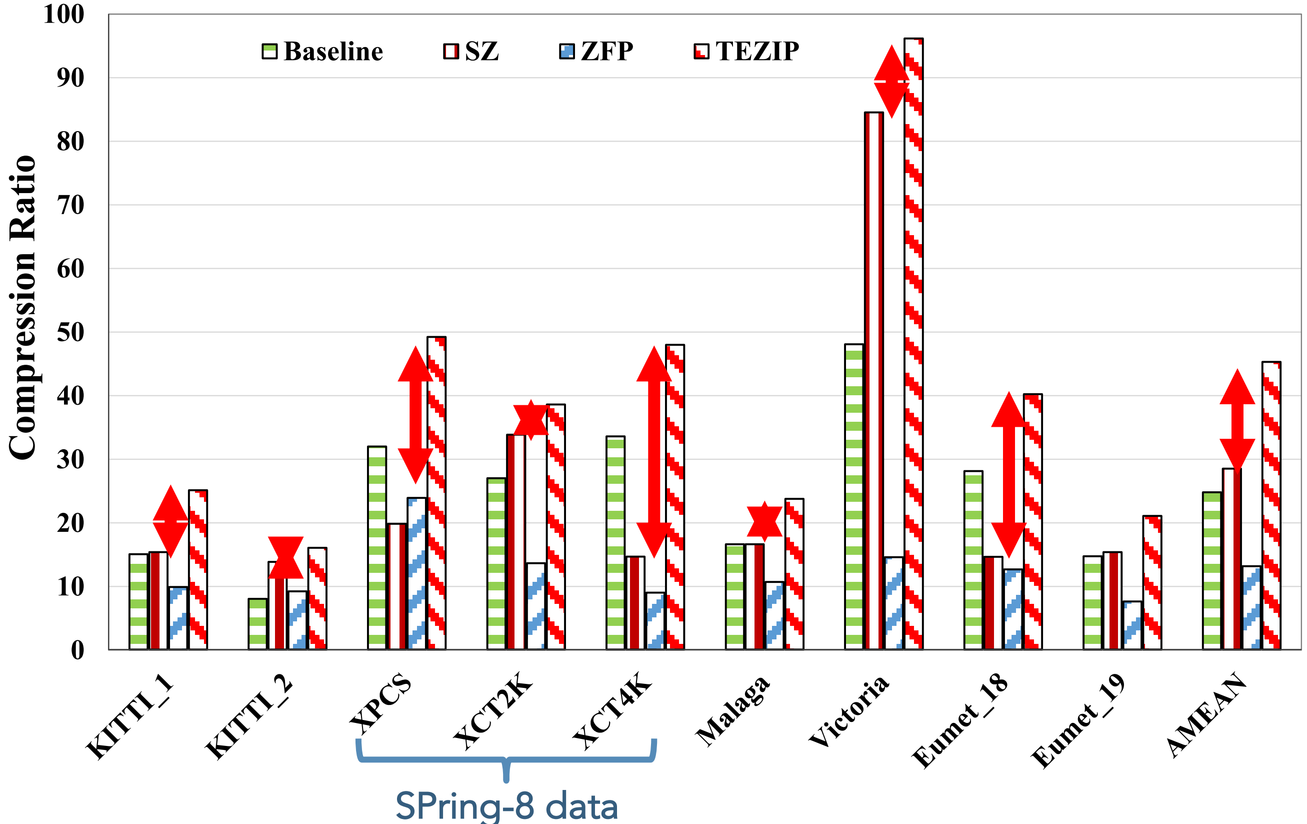
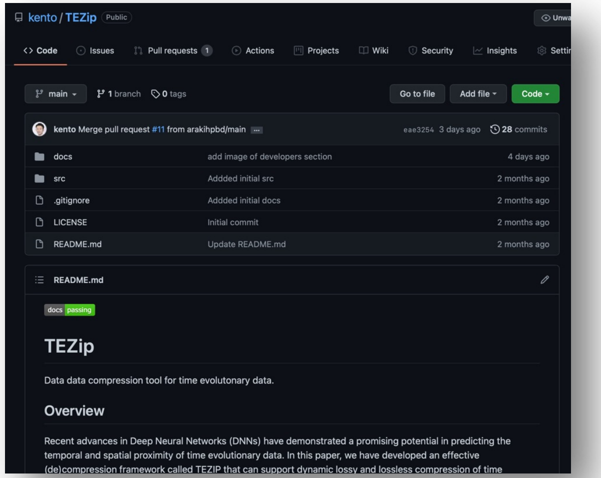



Fig. 10. Compression ratio with different lossy compressors

# TEZIP is open-source software and future works

- We open-sourced TEZIP and released documents

**Github:**
https://github.com/kento/TEZip




- Future works
  - Improvement in quantization
  - Improvement in Prediction
    - TEZIP relies on a generic predictor (PreNet)
    - The compression ratio will be futhrer imporved with domain-specific predictors

Generic predictor (PredNet)

Domain-specific predictor

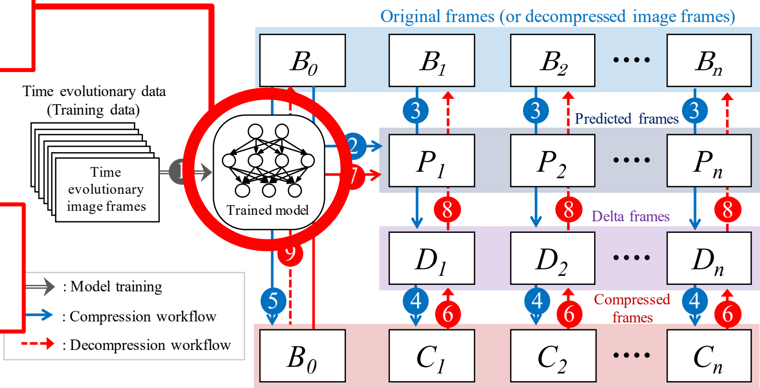
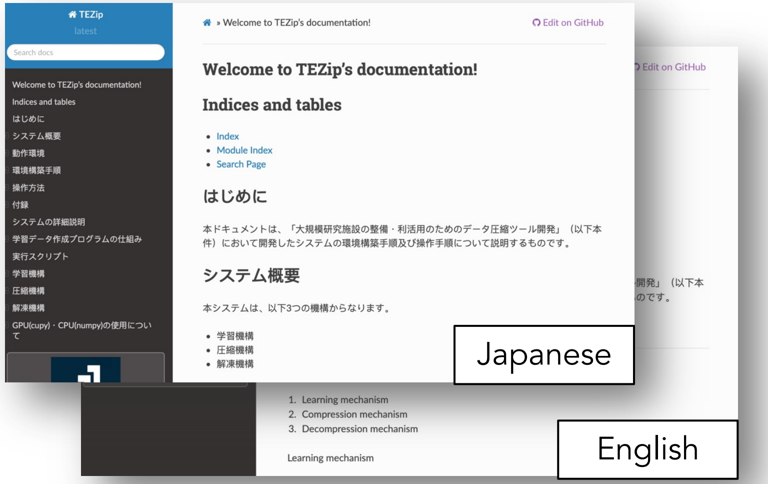



Fig. 1. Workflows of TEZIP (de)compression

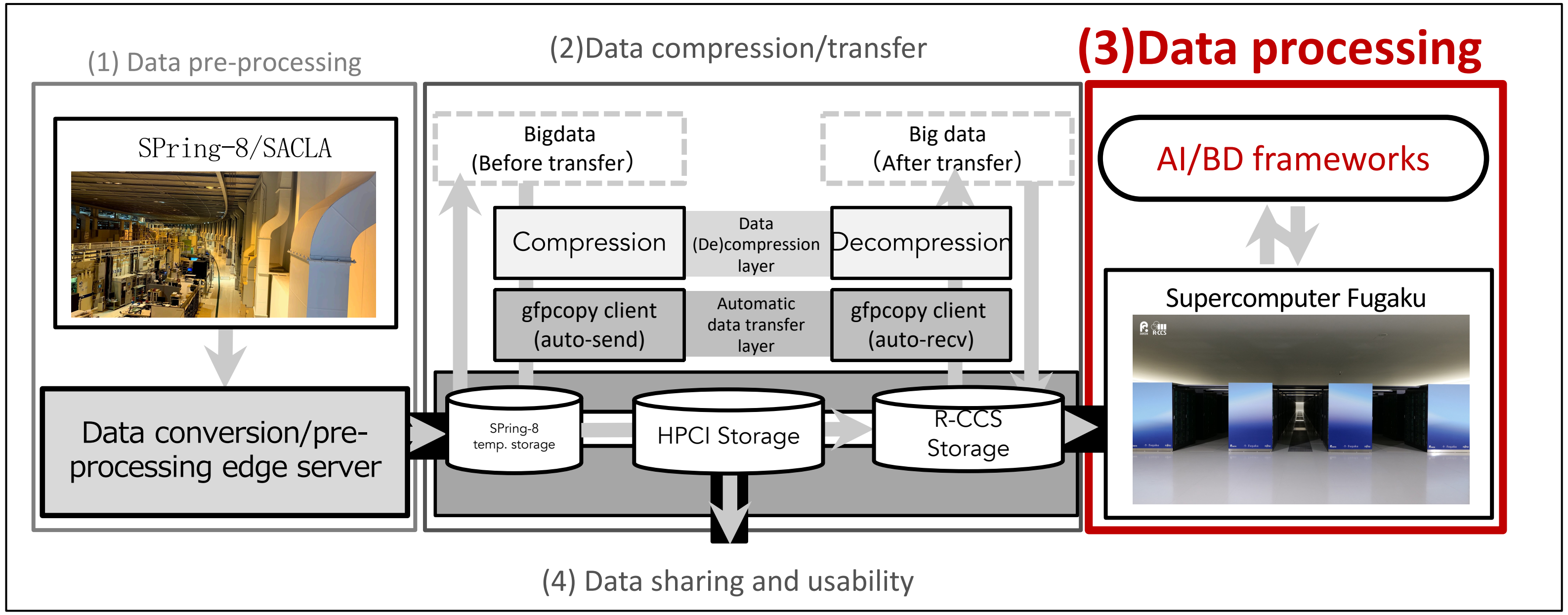
**Readthedocs:**
https://tezip.readthedocs.io/en/latest/?badge=latest

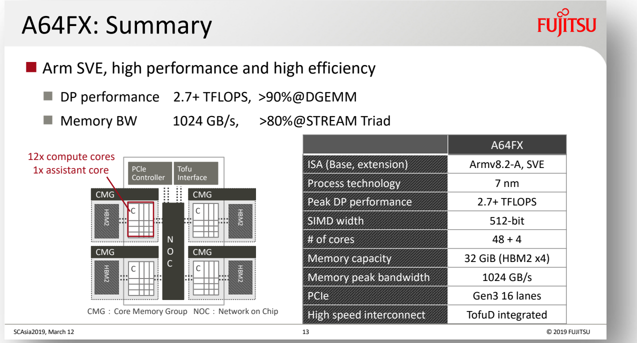



This project is seeking for a Postdoc or a Researcher !
https://www.hpbd.r-ccs.riken.jp/recruiting/

# Supercomputer Fugaku & Deep learning

- Once we move data to computers, the users will analyze the data and can use AI for the feature detection, Image recognition, segmentation etc.
- We must provide fast and scalable AI training environments on Fugaku
- GPU has become a popular platform for executing DL, but we revisit the idea of running DL on CPUs in Fugaku





A64FX: Summary

- Arm SVE, high performance and high efficiency
  - DP performance 2.7+ TFLOPS, >90%@DGEMM
  - Memory BW 1024 GB/s, >80%@STREAM Triad



| | A64FX |
|---|---|
| ISA (Base, extension) | Armv8.2-A, SVE |
| Process technology | 7 nm |
| Peak DP performance | 2.7+ TFLOPS |
| SIMD width | 512-bit |
| # of cores | 48 + 4 |
| Memory capacity | 32 GiB (HBM2 x4) |
| Memory peak bandwidth | 1024 GB/s |
| PCIe | Gen3 16 lanes |
| High speed interconnect | TofuD integrated |

SCAsia2019, March 12 13 © 2019 FUJITSU

Toshiyuki Shimizu, “Post-K Supercomputer with Fujitsu's Original CPU, A64FX Powered by Arm ISA”, Nov. 15th, 2018

→ High perf. FP16/INT8
→ High bw mem (1024 GB/sec)
→ Scalable TofuD net.

**To make use of Fugaku/A64FX performance, tuning AI software stack is indispensable**

# DL4Fugaku: Deep learning for Fugaku

- **Objective: Fast and scalable deep learning on Fugaku/A64FX**
  - Conduct porting, performance analysis and tuning
  - Deploy large-scale deep learning environment
  - Enhance the usability for production use in Fugaku

- **MOU for RIKEN/Fujitsu collaboration on AI framework development in Fugaku (Nov. 25, 2019)**

- **RIKEN R-CCS internal teams are working together**
  - Under collaboration with Industry & academia
  - Porting, tracing DL, performance analysis, tuning, merge to upstream

**RIKEN R-CCS**

| | |
|---|---|
| Operation team | ~~Application tuning development unit~~<br>→ Software Development Technology Unit |
| Research teams | High performance AI system research team |
| | High performance big data research team |
| | Large-scale parallel numerical computing technology research team |

**Collaborators**

| | |
|---|---|
| AIST | Fujitsu limited |
| ARM | Linaro |
| Cybozu | Tokyo Tech |
| ~~Fujitsu Laboratories~~ | (alphabetical order) |


https://www.hpbd.r-ccs.riken.jp/hpbd/en/dl4fugaku-project/

※ Some of software introduced in the rest of DL4Fugaku project slides is under development.
Experimental results will be changed in future in the course of tuning

# DL4Fugaku Project Menbers

FUJITSU

## Framework & oneDNN porting & tuning

**Naoki Shinjo**, Akira Asato, Atsushi Ike, Koutarou Okazaki, **Yoshihiko Oguchi**, Masahiro Doteguchi, **Jin Takahashi**, Kazutoshi Akao, Masaya Kato, Takashi Sawada, **Naoto Fukumoto**, Kentaro Kawakami, Naoki Sueyasu, Kouji Kurihara, Masafumi Yamazaki, Takumi Honda

Fugaku AI project

RIKEN

## Tuning for Fugaku

**Satoshi Matsuoka**, High Performance Artificial Intelligence Systems Research Team Leader

Kento Sato, High Performance Big Data Research Team Leader

Kazuo Minami, Application Tuning Development Unit Leader

Akiyoshi Kuroda, Application Tuning Development Unit

arm Linaro

Cybozu Labs

Technical support

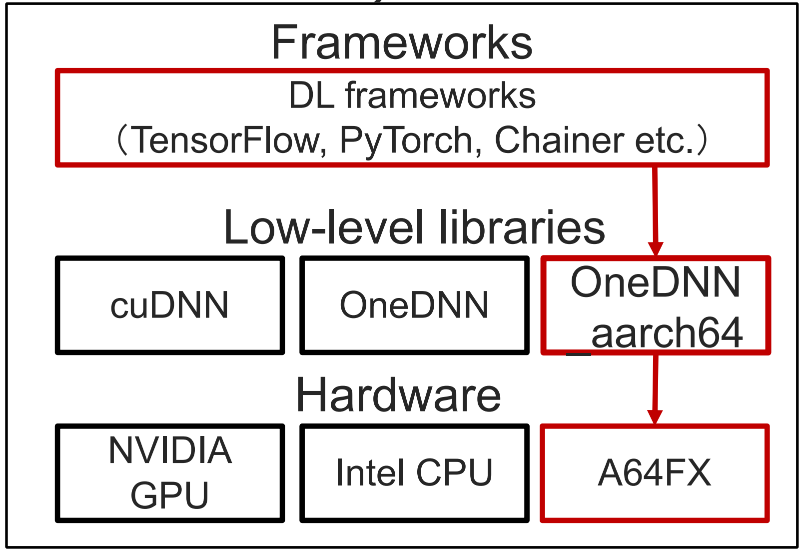
**Shigeo Mitsunari (Xbyak)**

# Porting and Tuning approach

- **Deep learning software stack**
  - Deep learning frameworks reply on low-level numerical libraries optimized for specific hardware
    - cuDNN for NVIDIA GPU, OneDNN for Intel CPU, ??? for A64FX

- **Approach**
  - We decided to tune OneDNN for Fugaku's A64FX CPUs (OneDNN_aarch64) instead of full scratch development

- **Current status**
  - The source codes are in a github repository
    - https://github.com/fujitsu/dnnl_aarch64
  - We also contribute to upstream of OneDNN repo

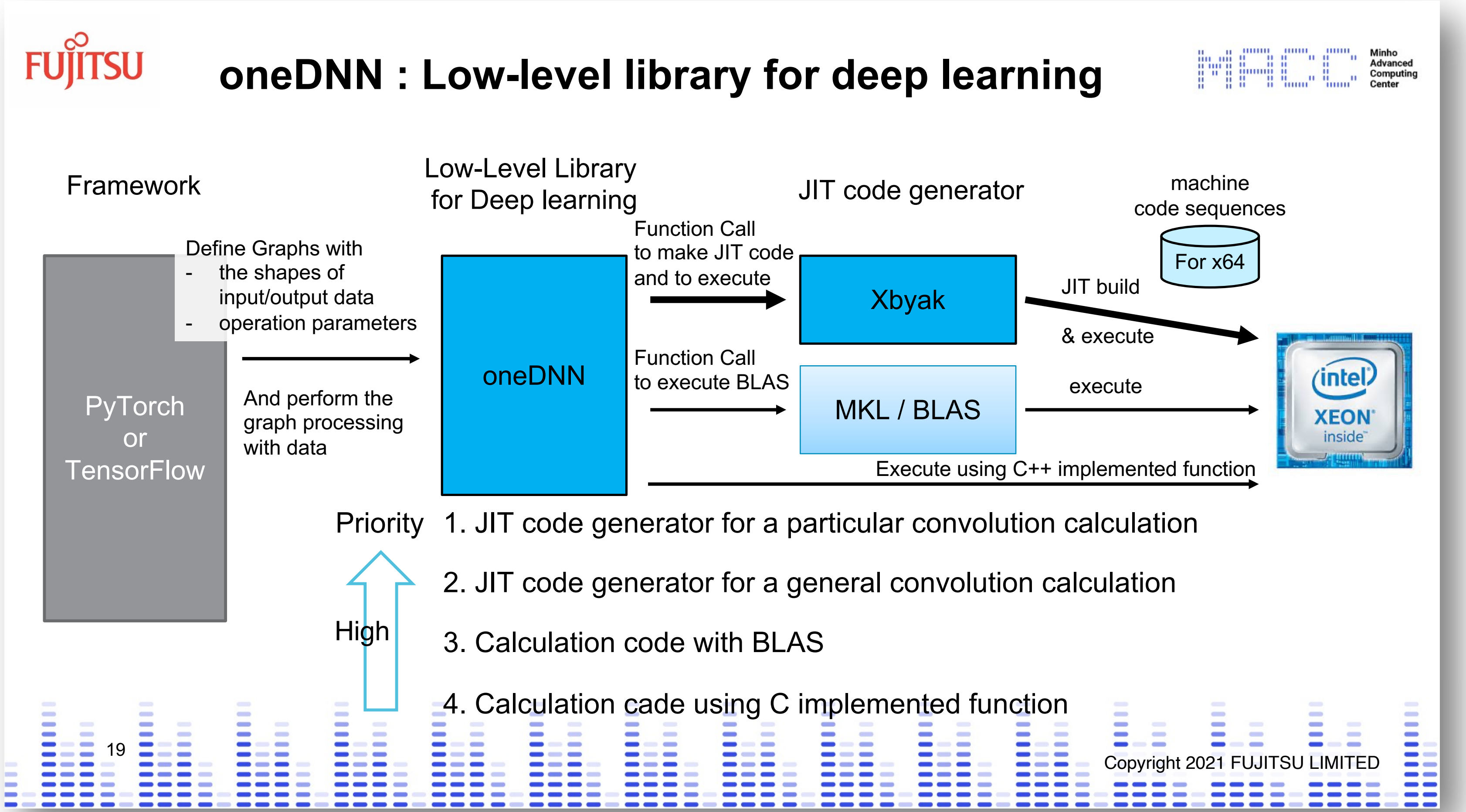



Intel Math Kernel Library for Deep Neural Networks (Intel MKL-DNN)
→ Deep Neural Network Library (DNNL)
→ oneAPI Deep Neural Network Library (oneDNN)



Slide courtesy of Jin Takahashi, Fujitsu laboratory ltd. with translation and modifications

# Original oneDNN@Intel logic



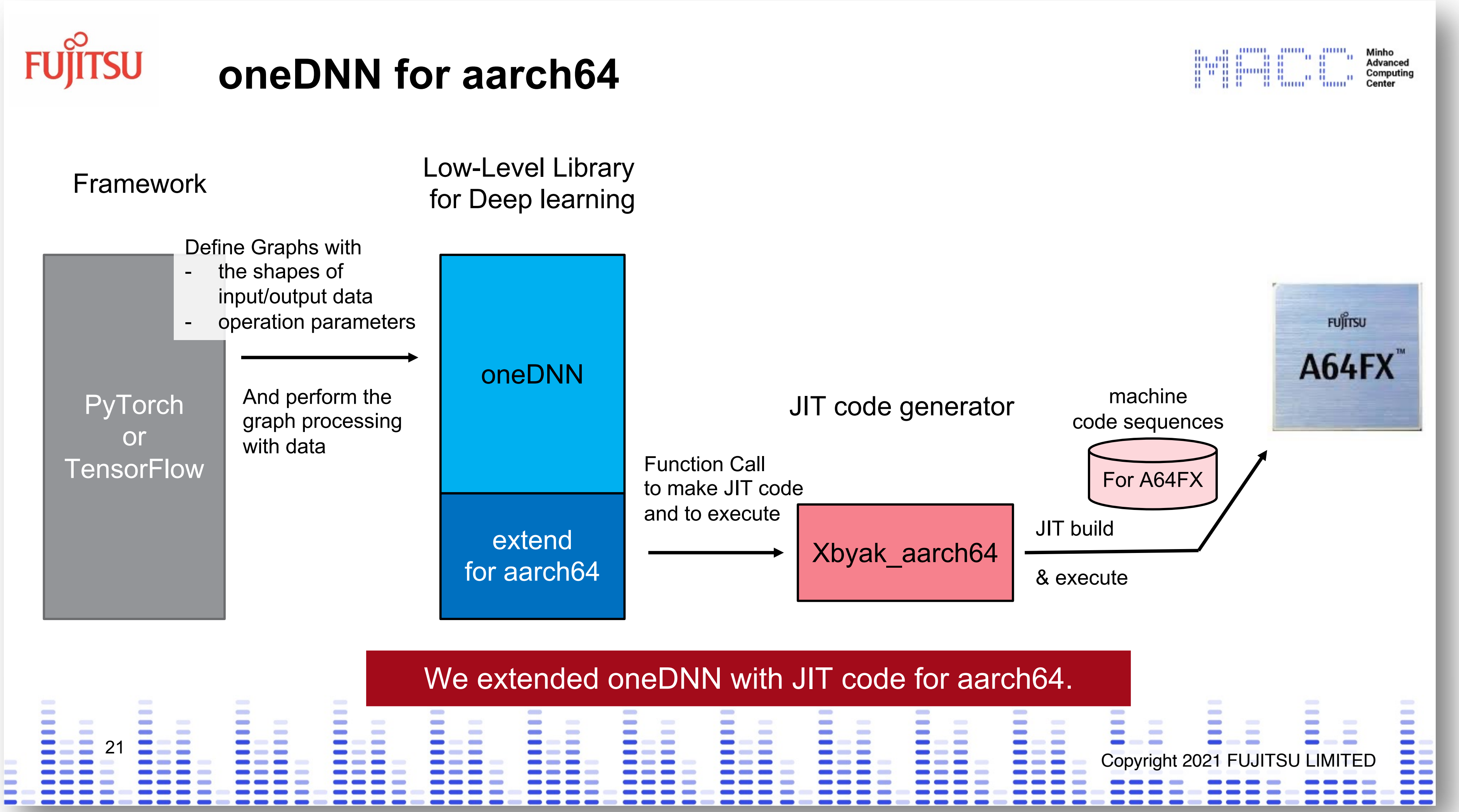

1. OneDNN gets information from a framework about (1) Shapes of input/output data; (2) Operation parameteers of each layer
2. OneDNN calls the fastest tensor routine based on the information
3. The priority is
   a) JIT-generated code
   b) BLAS
   c) C code implemented in OneDNN

- The generated code is cached and reused
- The same convolution kernels are called many time in deep learning
- The JIT-generation overhead is negligble for deep learning workloads

Slides: Masafumi Yamazaki (Fujitsu Ltd), “Deep learning on Fugaku”, MUG: MACC User Group Workshop, June 2021

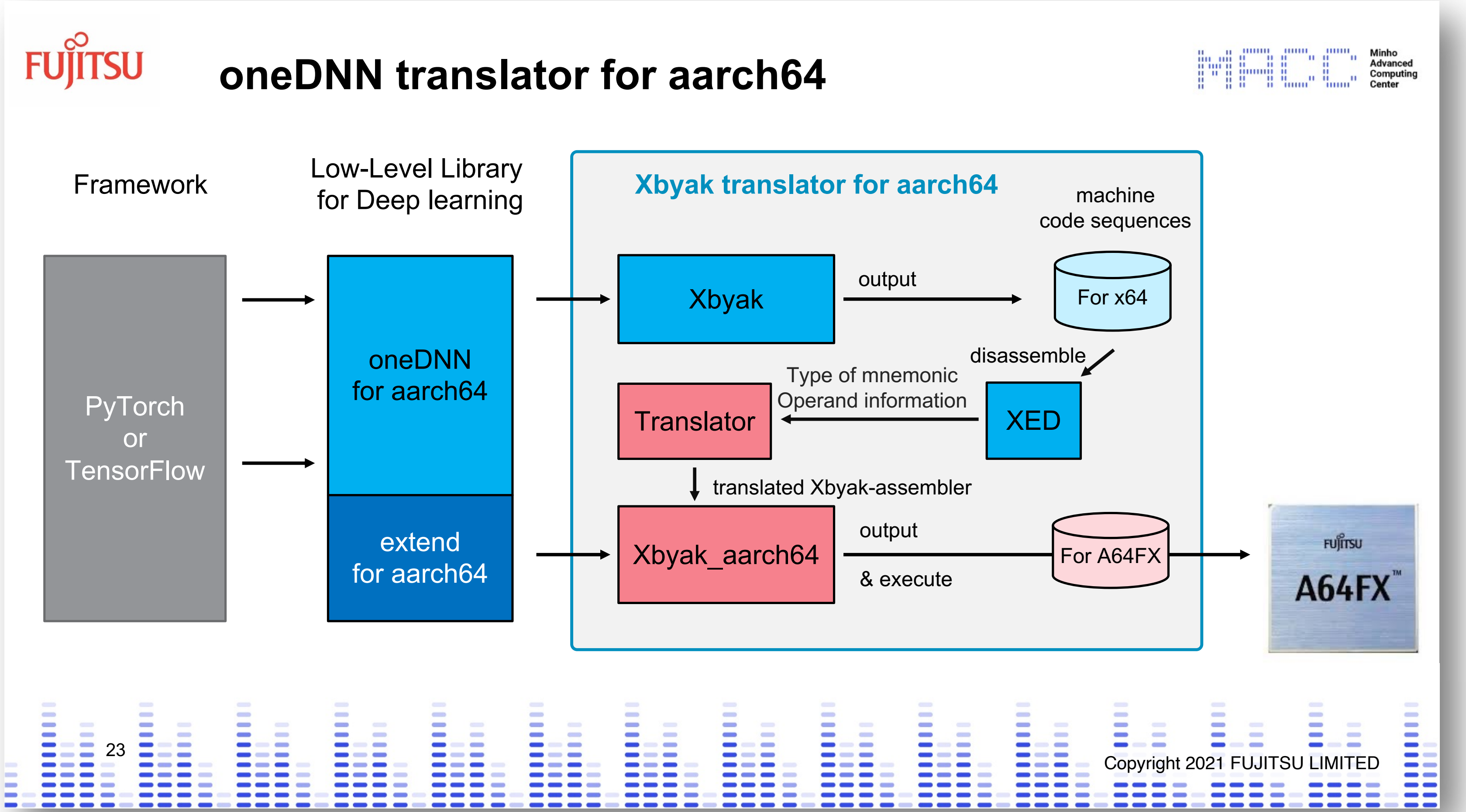
# oneDNN for A64FX/aarch64

- We extended OneDNN to generate aarch64 instructions via Xbyak




Slides: Masafumi Yamazaki (Fujitsu Ltd), "Deep learning on Fugaku", MUG: MACC User Group Workshop, June 2021

# Sustanable Porting Workflows




Slides: Masafumi Yamazaki (Fujitsu Ltd), "Deep learning on Fugaku", MUG: MACC User Group Workshop, June 2021

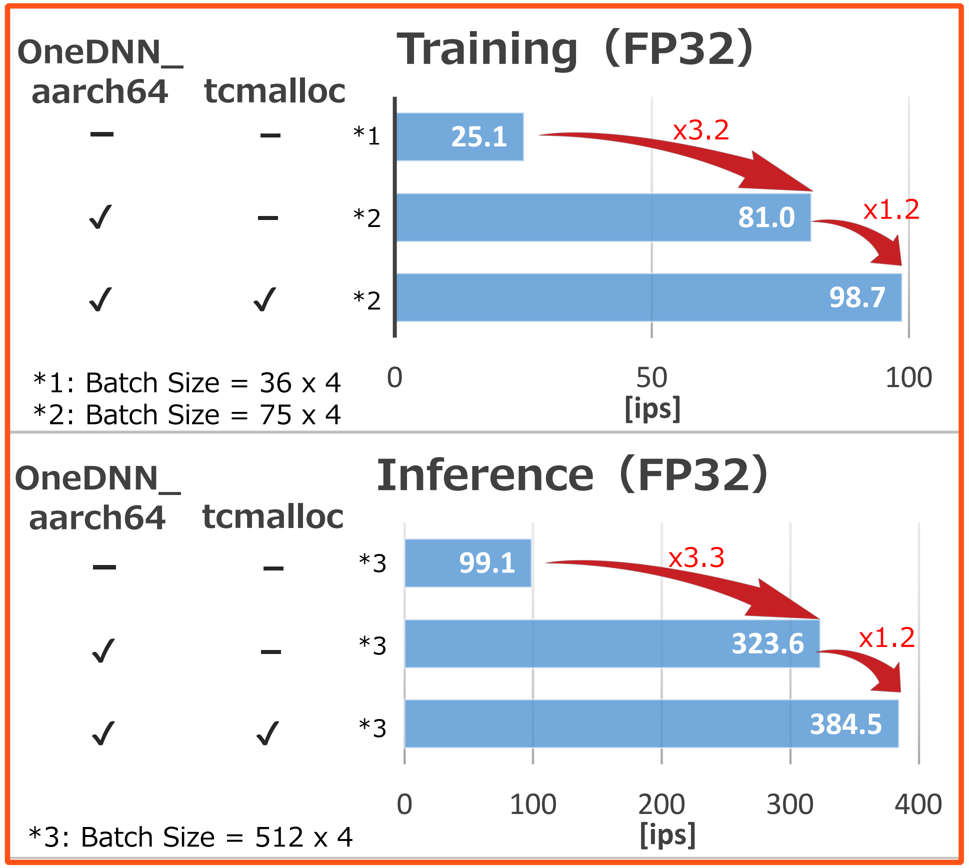
- By using the Xbyak, XED-Translator cascade, when the instruction set is extended, Xbyak and XED are replaced with the updated ones, and we only need to modify the mapping table between intel and Arm instructions in the Translator.

# Perfomrance Evaluation: ResNet-50 on A64FX (A single node)

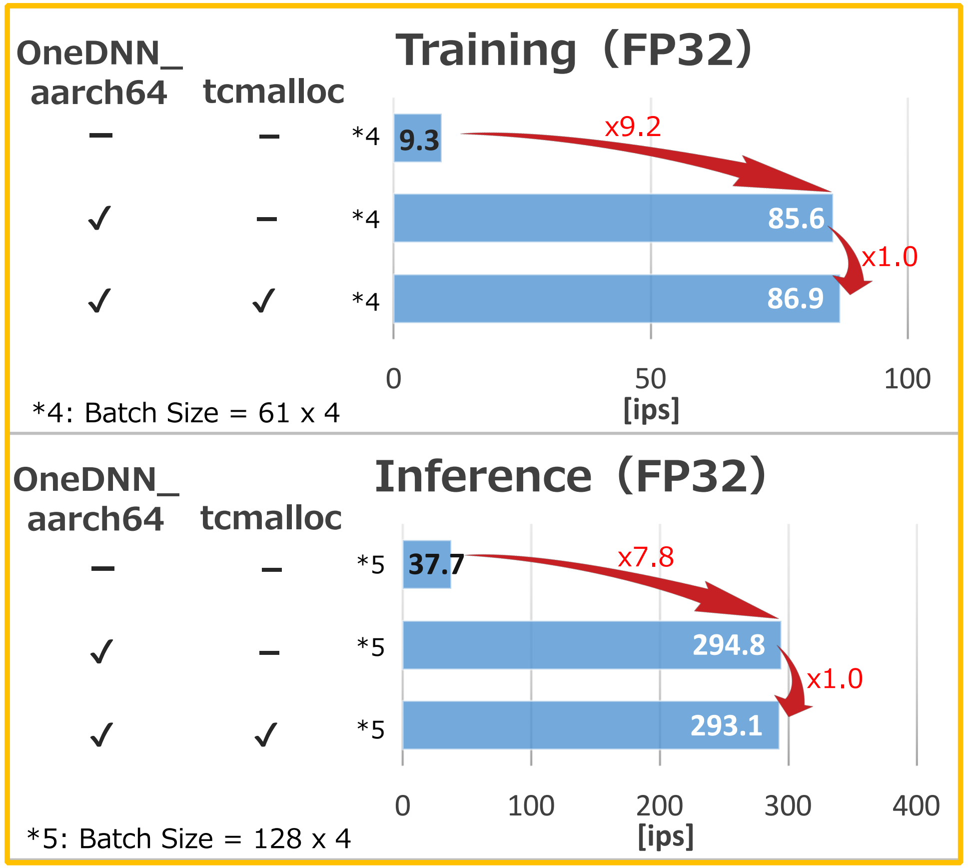
- **Environment**
  - HW: A64FX (2.2GHz, 48 cores, HBM2 32GB)
  - SW: Fujitsu compier (fcc), Fujitsu numerical libraries (SSL-II)

**Ref.) NVIDIA GPU V100: 905 ips [1]**
PyTorch/ResNet-50(training)/ImageNet2012

PyTorch v1.5.0




TensorFlow v2.1.0




[1] NVIDIA Data Center Deep Learning Product Performance, https://developer.nvidia.com/deep-learning-performance-training-inference

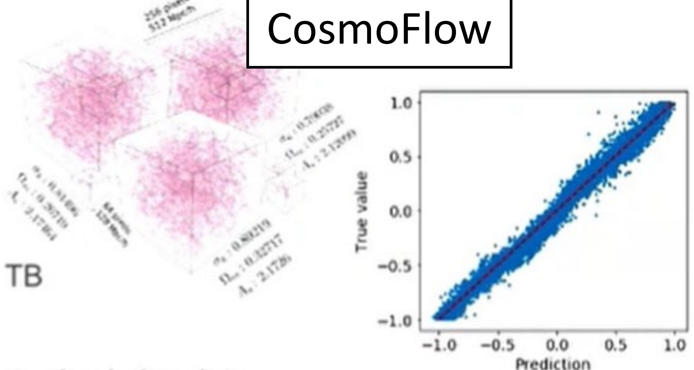


Slide courtesy of Jin Takahashi, Fujitsu laboratory ltd. with translation and modifications

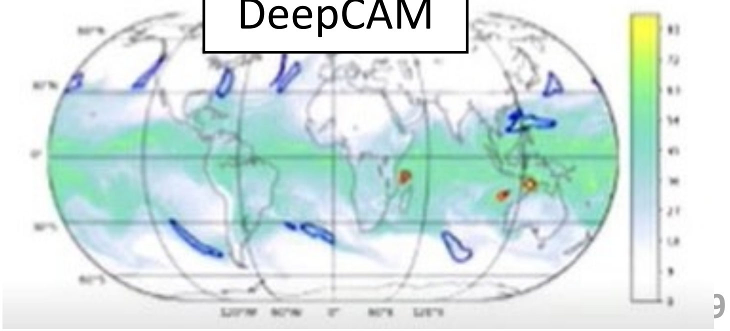
# MLPerf HPC v0.7 Benchmark

- **MLPerf HPC (v0.7) Benchmark**
  - One of deep learning benchmarks in MLPerf HPC
  - Repository: [https://github.com/mlcommons/hpc](https://github.com/mlcommons/hpc)
  - Benchmarks
    - CosmoFlow（宇宙科学）
      - Predict cosmological parameters from N-body cosmo simulation data
      - 3D CNN for regression of 4 parameters
      - Training data shape is（128, 128, 128, 4)
      - Training data size is 5.1TB
    - DeepCAM（気候・気象）
      - Indentify extreme weather phenomena in climate simulation data
      - 2D semantic segmentation with DeepLabV3+ model which predicts 3 classes per pixel (atomaspheric river, tropical cyclon or background)
      - Training data shape is (768, 1152, 16) and labeled with 3 per-pixel classes
      - Training data size is 8.8 TB

CosmoFlow




DeepCAM

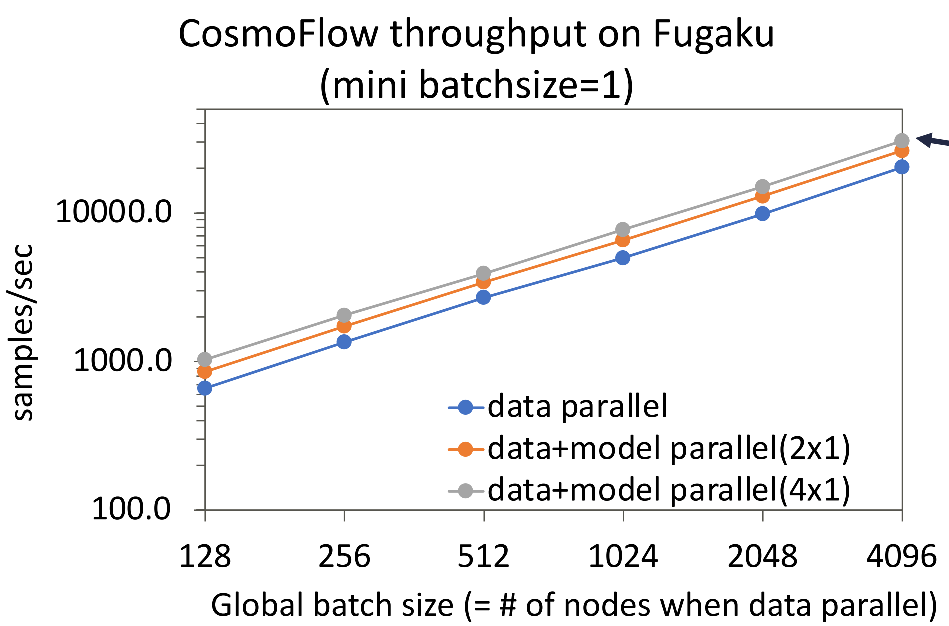


# Our process topology optimization enables scalable training

FUJITSU

## Throughput of Hybrid Parallelism

Minho Advanced Computing Center




Throughput scales almost ideally for global batch sizes when data + model parallelism is used

16,384 nodes(=processes)

=

4,096 models
(data parallel)

×

4x1 nodes/model
(model parallel)

36



Slides: Masafumi Yamazaki (Fujitsu Ltd), “Deep learning on Fugaku”, MUG: MACC User Group Workshop, June 2021

We achieved good scalability with a hybrid using of data&model parallel training

# MLPerf HPC (v0.7) ranking: CosmoFlow

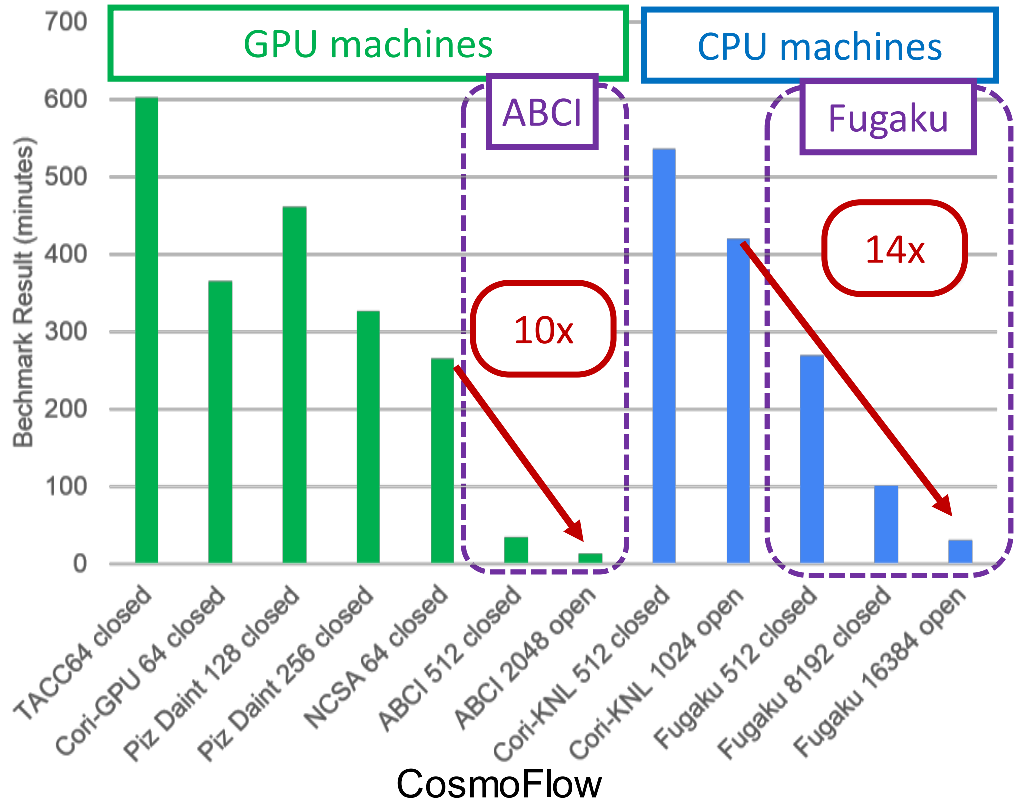
- **Fugaku was ranked at No 2. in MLPerf HPC ranking (Nov., 2020) even with "1/10 of Fugaku nodes"**
  - Fujitsu, AIST and RIKEN Achieve Unparalleled Speed on MLPerf HPC Machine Learning Processing Benchmark
    - https://www.hpcwire.com/off-the-wire/fujitsu-aist-and-riken-achieve-unparalleled-speed-on-mlperf-hpc-machine-learning-processing-benchmark/

| Submitter | System | Processor | # | Accelerator | # | Software | Benchmark results (minutes, smaller is better) CosmoFlow | DeepCAM |
|---|---|---|---|---|---|---|---|---|
| e on-premises | | | | | | | | |
| CSCS | daint_gpu_n128_tf2.2.0 | Intel® Xeon® Processor E5-2690 v3 @2.60 | 128 | NVIDIA P100-PCIE-16G | 128 | TensorFlow 2.2.0 | 461.01 | |
| CSCS | daint_gpu_n256_tf2.2.0 | Intel® Xeon® Processor E5-2690 v3 @2.60 | 256 | NVIDIA P100-PCIE-16G | 256 | TensorFlow 2.2.0 | 327.01 | |
| Fujitsu | ABCI PRIMERGY CX2570 M4 | Intel® Xeon® Gold 6148 Processor @2.40 | 512 | NVIDIA V100 | 1024 | PyTorch 1.6.0 | | 11.71 |
| Fujitsu | ABCI PRIMERGY CX2570 M4 | Intel® Xeon® Gold 6148 Processor @2.40 | 256 | NVIDIA V100 | 512 | TensorFlow 2.2.0 | 34.42 | |
| Fujitsu/RIKEN | fugaku_512xA64FX_tensorflow_closed | FUJITSU Processor A64FX | 512 | N/A | 0 | TensorFlow 2.2.0 + Mesh TensorFlow | 268.77 | |
| Fujitsu | fugaku_8192xA64FX_tensorflow_closed | FUJITSU Processor A64FX | 8192 | N/A | 0 | TensorFlow 2.2.0 + Mesh TensorFlow | 101.49 | |
| LBNL | corigpu_n64_pt1.6.0 | Intel® Xeon® Gold 6148 Processor @2.40 | 16 | NVIDIA V100 | 64 | PyTorch 1.6.0 | | 139.29 |
| LBNL | corigpu_n64_tf1.15.0 | Intel® Xeon® Gold 6148 Processor @2.40 | 16 | NVIDIA V100 | 64 | TensorFlow 1.15.0 | 364.73 | |
| LBNL | coriknl_n512_tf1.15.2 | Intel® Xeon Phi™ Processor 7250 @1.40G | 512 | N/A | 0 | TensorFlow 1.15.2 | 536.06 | |
| NCSA | hal_v100_n16_tf1.15.0 | IBM POWER 9 model 2.2 | 32 | NVIDIA V100 | 64 | TensorFlow 1.15.0 | 265.59 | |
| TACC | Frontera-RTX | Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10G | 32 | NVIDIA Quadro RTX 50 | 64 | TensorFlow 1.15.2 | 602.23 | |

**Division Times**

| Submitter | System | Processor | # | Accelerator | # | Software | Benchmark results (minutes, smaller is better) CosmoFlow | DeepCAM |
|---|---|---|---|---|---|---|---|---|
| e on-premises | | | | | | | | |
| Fujitsu | ABCI PRIMERGY CX2570 M4 | Intel® Xeon® Gold 6148 Processor @2.40 | 512 | NVIDIA V100 | 1024 | PyTorch 1.6.0 | | 10.49 |
| Fujitsu | ABCI PRIMERGY CX2570 M4 | Intel® Xeon® Gold 6148 Processor @2.40 | 1024 | NVIDIA V100 | 2048 | TensorFlow 2.2.0 | 13.21 | |
| Fujitsu | fugaku_16384xA64FX_tensorflow_open | FUJITSU Processor A64FX | 16384 | N/A | 0 | TensorFlow 2.2.0 + Mesh TensorFlow | 30.07 | |
| LBNL | coriknl_n1024_tf1.15.2 | Intel® Xeon Phi™ Processor 7250 @1.40G | 1024 | N/A | 0 | Tensorflow 1.15.2 | 419.69 | |

| Submitter | System | Processor | # | Software | Time [min] |
|---|---|---|---|---|---|
| Fujitsu | ABCI | Xeon Gold 6148<br>Tesla V100 GPU | 1024<br>2048 | TensorFlow | 13.21 |
| Fujitsu / RIKEN | Fugaku | A64FX | 16384 | TensorFlow + Mesh TensorFlow | 30.07 |



# The 16,384-node Parallelism of 3D-CNN Training on An Arm CPU based Supercomputer (HiPC2021) [2]

Akihiro Tabuchi∗, Koichi Shirahata∗, Masafumi Yamazaki∗, Akihiko Kasagi∗, Takumi Honda∗, Kouji Kurihara∗, Tsuguchika Tabaru∗, Naoto Fukumoto∗, Akiyoshi Kuroda†, Takaaki Fukai† and Kento Sato†
∗Fujitsu Limited, Kawasaki, Kanagawa, Japan †RIKEN Center for Computational Science, Kobe, Hyogo, Japan

- Xbyak-T which automatically generate tuned code for aarch64 from oneDNN that is originally tuned for x86 64.
- Highly scalable hybrid parallelism tuned for 6D mesh/torus network topology of TofuD interconnects with a rank mapping technique for MPI_Allreduce;
- I/O acceleration for data loading with data compression, data staging and data caching techniques;

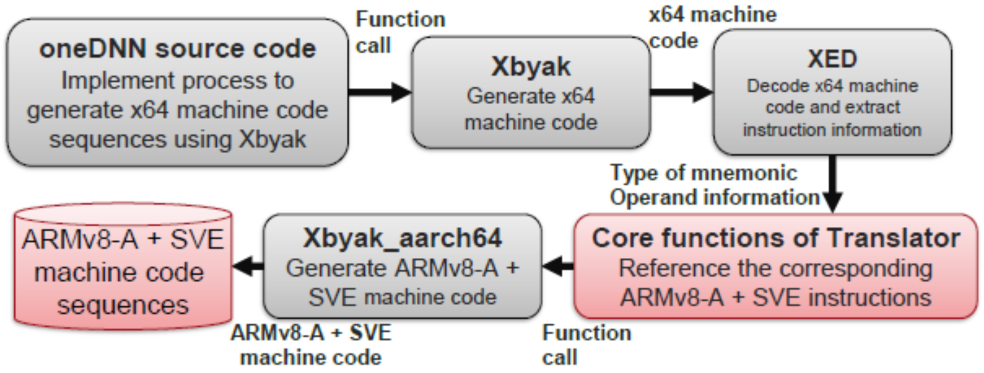



Fig. 3. Diagram of the process for converting an x86_64 machine instruction sequence into an Armv8-A + SVE machine instruction sequence.

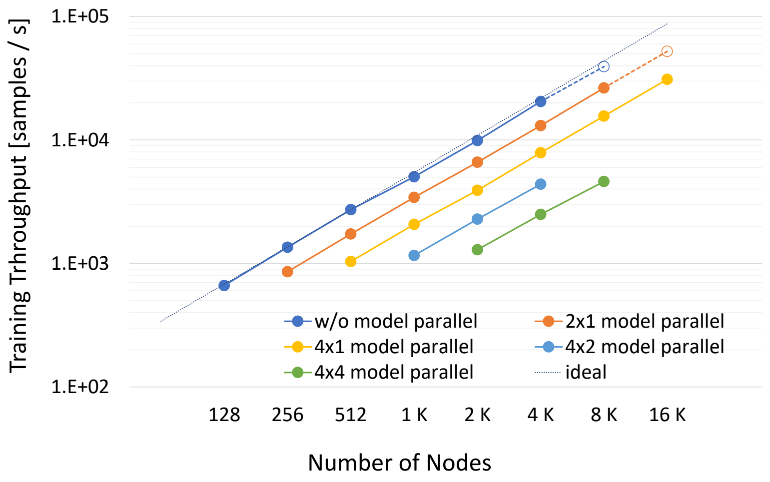



Fig. 11. The scalability of hybrid parallelism on Fugaku. Each node processed one sample per iteration.

[2] Akihiro Tabuchi, Koichi Shirahata, Masafumi Yamazaki, Akihiko Kasagi, Takumi Honda, Kouji Kurihara, Kentaro Kawakami, Tsuguchika Tabaru, Naoto Fukumoto, Akiyoshi Kuroda, Takaaki Fukai and Kento Sato, "The 16,384-node Parallelism of 3D-CNN Training on An Arm CPU based Supercomputer", 28th IEEE International Conference on High Performance Computing, Data, and Analytics (HiPC2021), Nov. 2021

# MLPerf HPC: Benchmarking Machine Learning Workloads on HPC Systems (MLHPC2021@SC21) [4]

Steven Farrell, Murali Emani, Jacob Balma, Lukas Drescher, Aleksandr Drozd, Andreas Fink, Geoffrey Fox, David Kanter, Thorsten Kurth, Peter Mattson, Dawei Mu, Amit Ruhela, Kento Sato,,Koichi Shirahata, Tsuguchika Tabaru, Aristeidis Tsaris, Jan Balewski, Ben Cumming, Takumi Danjo, Jens Domke, Takaaki Fukai, Naoto Fukumoto, Tatsuya Fukushi, Balazs Gerofi, Takumi Honda, Toshiyuki Imamura, Akihiko Kasagi, Kentaro Kawakami, Shuhei Kudo, Akiyoshi Kuroda, Maxime Martinasso, Satoshi Matsuoka, Kazuki Minami, Prabhat Ram, Takashi Sawada, Mallikarjun Shankar, Tom St. John, Akihiro Tabuchi, Venkatram Vishwanath, Mohamed Wahib, Masafumi Yamazaki, Junqi Yin and Henrique Mendonca
(Collaborations with LBL, ANL, HPE, CSCS, Indiana Univ., MLCommons, NVIDIA, Google, NCSA, TACC, Fujitsu, ORNL, Microsoft, Cruise)

- Summarize results across different organizations from the MLPerf HPC submission round in 2020
- These results feature measurements from leading supercomputing platforms around the world, innovations in scalable model-and-data-parallel training and learning algorithms, and the largest scale MLPerf submission to date

TABLE IV
PERFORMANCE METRICS (TIME TO SOLUTION IN MINUTES) FROM SUBMISSIONS IN CLOSED AND OPEN DIVISIONS

| Division | System | Submission | Software | #Processors | #Accelerators | Parallelism† | CosmoFlow | DeepCAM |
|---|---|---|---|---|---|---|---|---|
| Closed | Piz Daint | Piz-Daint-128 | TensorFlow 2.2.0 | 128 | 128 | 2 s/1 GPU | 461.01 | - |
| | Piz Daint | Piz-Daint-256 | TensorFlow 2.2.0 | 256 | 256 | 2 s/1 GPU | 327.01 | - |
| | ABCI | ABCI-1024 | PyTorch 1.6.0 | 512 | 1,024 | 2 s/1 GPU | - | 11.71 |
| | ABCI | ABCI-512 | TensorFlow 2.2.0 | 256 | 512 | 1 s/1 GPU | 34.42 | - |
| | Fugaku | Fugaku-512 | TensorFlow 2.2.0 + Mesh TensorFlow | 512 | 0 | 1 s/1 CPU | 268.77 | - |
| | Fugaku | Fugaku-8192 | TensorFlow 2.2.0 + Mesh TensorFlow | 8,192 | 0 | 1 s/16 CPUs | 101.49 | - |
| | Cori-GPU | Cori-GPU-64 | PyTorch 1.6.0 | 16 | 64 | 2 s/1 GPU | - | 139.29 |
| | Cori-GPU | Cori-GPU-64 | TensorFlow 1.15.0 | 16 | 64 | 1 s/1 GPU | 364.73 | - |
| | Cori-KNL | Cori-KNL-512 | TensorFlow 1.15.2 | 512 | 0 | 1 s/1 CPU | 536.06 | - |
| | HAL | HAL-64 | TensorFlow 1.15.0 | 32 | 64 | 1 s/1 GPU | 265.59 | - |
| | Frontera-RTX | Frontera-RTX-64 | TensorFlow 1.15.2 | 32 | 64 | 1 s/1 GPU | 602.23 | - |
| Open | ABCI | ⋆ABCI-1024 | PyTorch 1.6.0 | 512 | 1,024 | 2 s/1 GPU | - | 10.49 |
| | ABCI | ⋆ABCI-2048 | TensorFlow 2.2.0 | 1,024 | 2,048 | 1 s/1 GPU | 13.21 | - |
| | Fugaku | ⋆Fugaku-16384 | TensorFlow 2.2.0 + Mesh TensorFlow | 16,384 | 0 | 1 s/4 CPUs | 30.07 | - |
| | Cori-KNL | ⋆Cori-KNL-1024 | TensorFlow 1.15.2 | 1,024 | 0 | 1 s/1 CPU | 419.69 | - |

† Data-parallel granularity of train step: # samples (s) processed by number of compute units forming a data-parallel unit in each train step. E.g. Piz-Daint-128 processes 2 samples ("local batch size") on each GPU (pure data-parallelism, batch size $128 \times 2 = 256$), whereas Fugaku-8192 processes 1 sample in each group of 16 CPUs (through model-parallelism within this group, data-parallelism across these groups of which there are $8192/16 = 512 =$ batch size).

TABLE V
DATA STAGING TIME

| Benchmark | Submission | Staging time (minutes) | $\frac{T_{staging}}{T_{epoch}}$ |
|---|---|---|---|
| CosmoFlow | Cori-GPU-64 | $16.49 \pm 0.61$ | 2.55 |
| | ABCI-512 | $0.76 \pm 0.004$ | 2.27 |
| | ⋆ABCI-2048 | $0.20 \pm 0.004$ | 1.56 |
| | Fugaku-512 | $1.55 \pm 0.11$ | 0.64 |
| | Fugaku-8192 | $3.77 \pm 0.51$ | 3.59 |
| | ⋆Fugaku-16384 | $0.88 \pm 0.08$ | 4.93 |
| DeepCAM | ABCI-1024 | $2.20 \pm 0.01$ | 5.55 |
| | ⋆ABCI-1024 | $1.96 \pm 0.08$ | 5.45 |

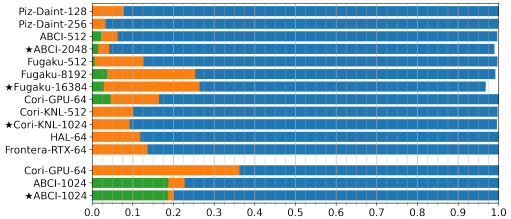



Fig. 1. Relative breakdown of time to train normalized to range [0-1], into staging (green), evaluation (orange) and training (blue). Lower three entries on y-axis are for DeepCAM, rest are for CosmoFlow.

TABLE VII
WORKLOAD CHARACTERIZATION: MEMORY BANDWIDTH (SINGLE CPU/GPU), NETWORK AND PER-WORKER I/O BANDWIDTH MEASUREMENTS

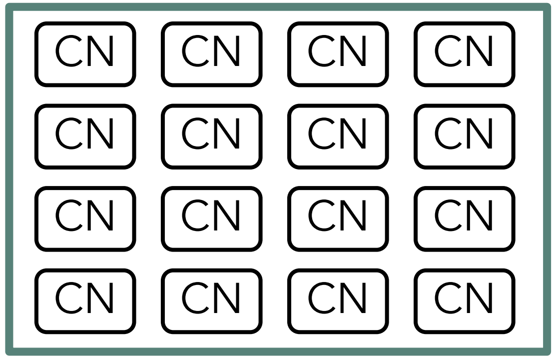
| Benchmark | System | Memory Tool | Memory BW (GB/sec) | Network Tool | # units | Network BW (GB/sec) | Size (MB) | I/O Tool | I/O BW (GB/sec) |
|---|---|---|---|---|---|---|---|---|---|
| CosmoFlow | ABCI† | Nvprof | 335.4 | Horovod TL | 512 GPUs | 3.41 | 19.97 | Nvprof | 1.65 |
| | Fugaku† | Perf | 110.8 | Mpitrace | 512 CPUs | 0.75 | 21.71 | Timer-based | 2.57 |
| | Piz Daint | Nvprof | - | Horovod TL | 256 GPUs | 1.86 | 2.21 | Darshan | 0.51 |
| | Summit | Nsight | 233.1 | Horovod TL | 510 GPUs | 2.24 | 22.0 | Darshan | 1.46 |
| | ThetaGPU | Nsight | 194.5 | Horovod TL | 128 GPUs | 1.95 | 15.20 | Darshan | 1.98 |
| DeepCAM | ABCI | Nvprof | 153.1 | Timer-based | 512 GPUs | 3.73 | 37.77 | Darshan | 2.36 |
| | Summit | Nsight | 254.7 | Timer-based | 510 GPUs | 4.50 | 225.0 | | |

[4] Steven Farrell, Murali Emani, Jacob Balma, Lukas Drescher, Aleksandr Drozd, Andreas Fink, Geoffrey Fox, David Kanter, Thorsten Kurth, Peter Mattson, Dawei Mu, Amit Ruhela, Kento Sato,,Koichi Shirahata, Tsuguchika Tabaru, Aristeidis Tsaris, Jan Balewski, Ben Cumming, Takumi Danjo, Jens Domke, Takaaki Fukai, Naoto Fukumoto, Tatsuya Fukushi, Balazs Gerofi, Takumi Honda, Toshiyuki Imamura, Akihiko Kasagi, Kentaro Kawakami, Shuhei Kudo, Akiyoshi Kuroda, Maxime Martinasso, Satoshi Matsuoka, Kazuki Minami, Prabhat Ram, Takashi Sawada, Mallikarjun Shankar, Tom St. John, Akihiro Tabuchi, Venkatram Vishwanath, Mohamed Wahib, Masafumi Yamazaki, Junqi Yin and Henrique Mendonca, "MLPerf HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems", The Workshop on Machine Learning in High Performance Computing Environments (MLHPC) 2021 in conjunction with SC21, Nov, 2021

# MLPerf HPC (v1.0) introduced scalability rules

- MLPerf HPC v0.7 in FY2020: Strong scaling metric
  - Strong scaling metric
    - Measures time to train one model on a system
  - Due to the large-batch problem, 1/10 of Fugaku nodes give the best performance
  - Benchmarks: CosmoFlow, DeepCAM

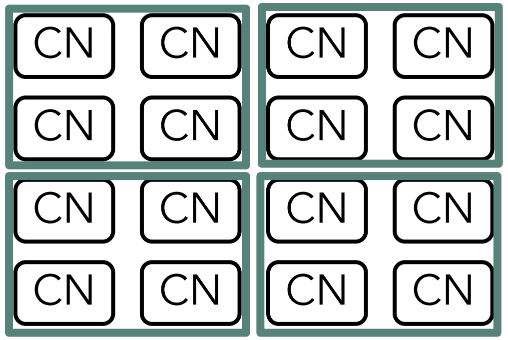
Training one model




- MLPerf HPC v1.0 in FY2021: Strong + Weak scaling metric
  - v1.0 introduces a new weak scaling metric (in addition to strong scale metric)
    - Time-to-train → Throghputs (models/second)
  - Weak scaling metric
    - Train multiple models on a system and measure # of trained models per sec
    - Models are independently trainined eath other ans it is scalable
    - We could use 1/2 of Fugaku nodes
  - Benchmarks: CosmoFlow, DeepCAM and Catalyst
  - Six metrics: {CosmoFlow, DeepCAM and Catalsyt} x {Strong, Week}

→ We targeted CosmoFlow & Week scaling metric

Training multiple models



# MLPerf HPC v1.0 result (CosmoFlow & Weak scaling metric)

- Fugaku took first place in MLPerf HPC v1.0 (CosmoFlow, weak scaling metric)
  - 82,944 CPUs are used (128 CPUs per model instance)
  - Trained 637 models in 8.26 hours (495.6 mins)
  - → 1.285 modesl / min

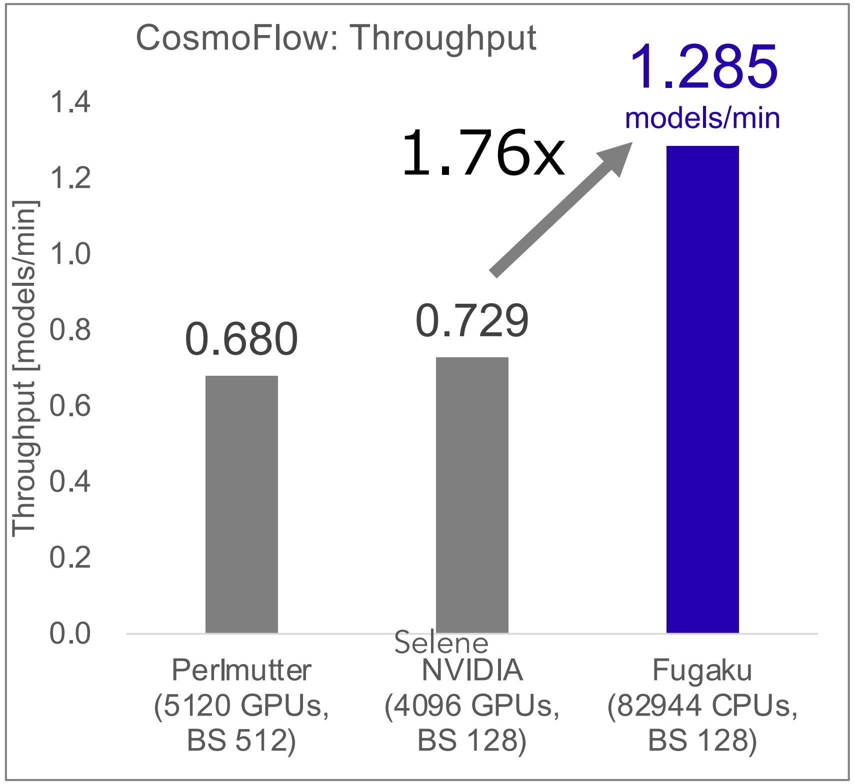
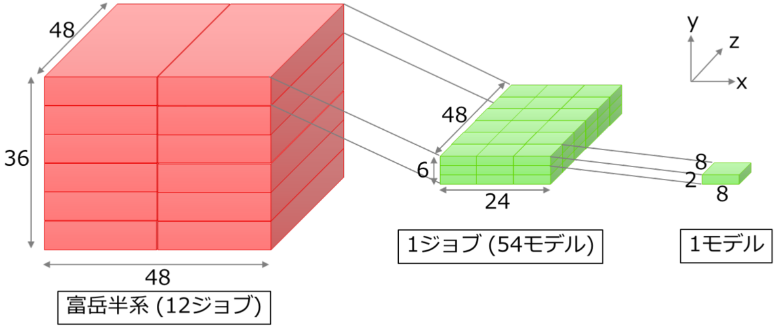
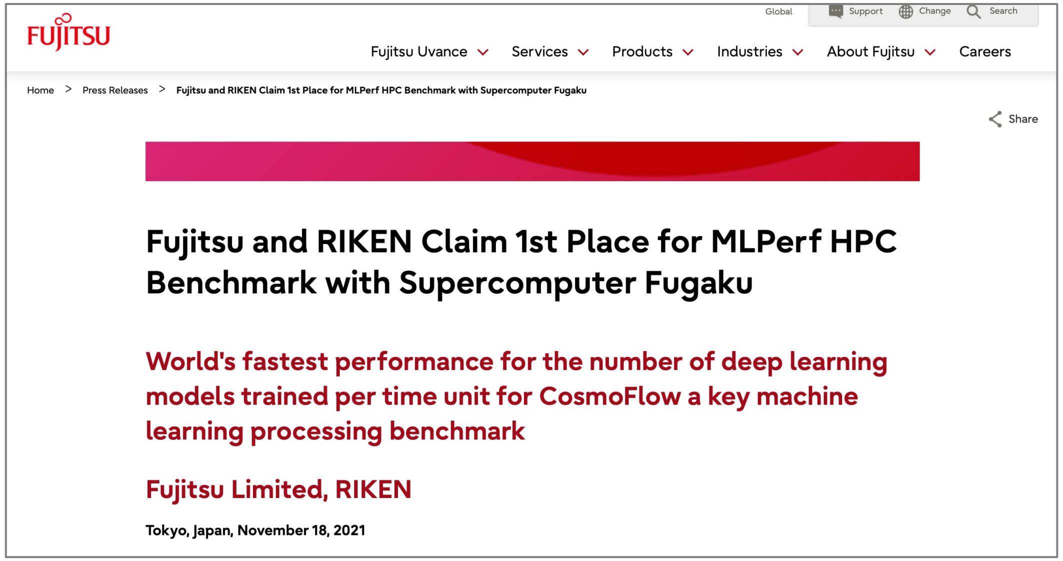



Figure from Koichi Shirahata (Fujitsu Ltd) presentation at SC21 BoF (MLPerf HPC)




12 x 54 = 648 models



FUJITSU

Global | Support | Change | Search

Fujitsu Uvance | Services | Products | Industries | About Fujitsu | Careers

Home > Press Releases > Fujitsu and RIKEN Claim 1st Place for MLPerf HPC Benchmark with Supercomputer Fugaku

Share

**Fujitsu and RIKEN Claim 1st Place for MLPerf HPC Benchmark with Supercomputer Fugaku**

**World's fastest performance for the number of deep learning models trained per time unit for CosmoFlow a key machine learning processing benchmark**

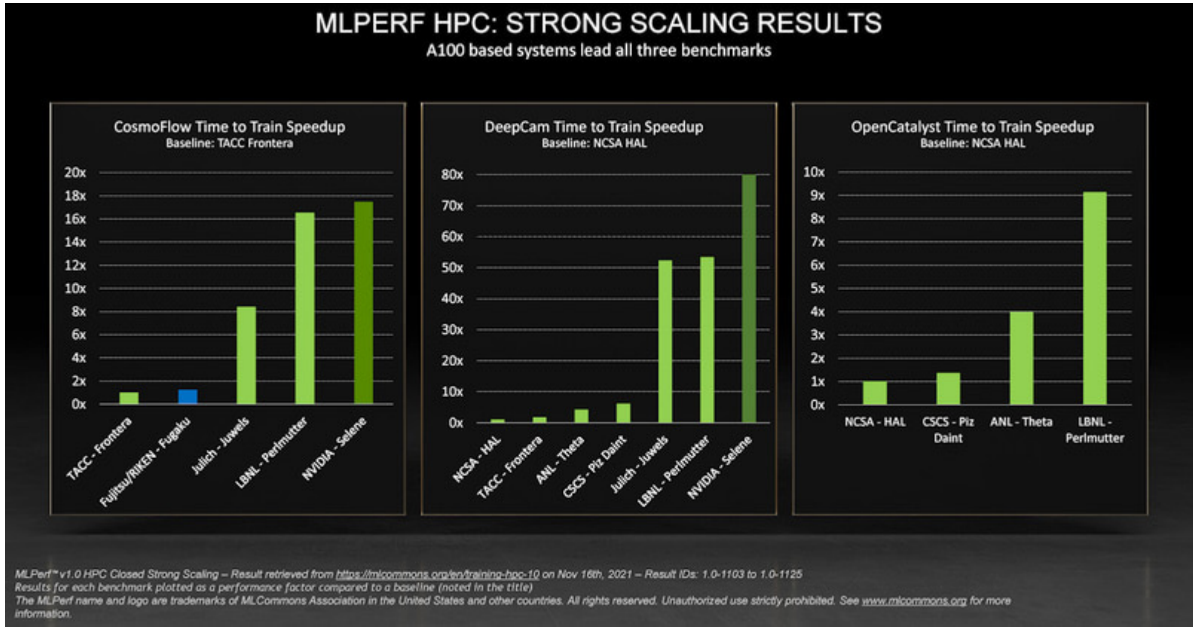
**Fujitsu Limited, RIKEN**

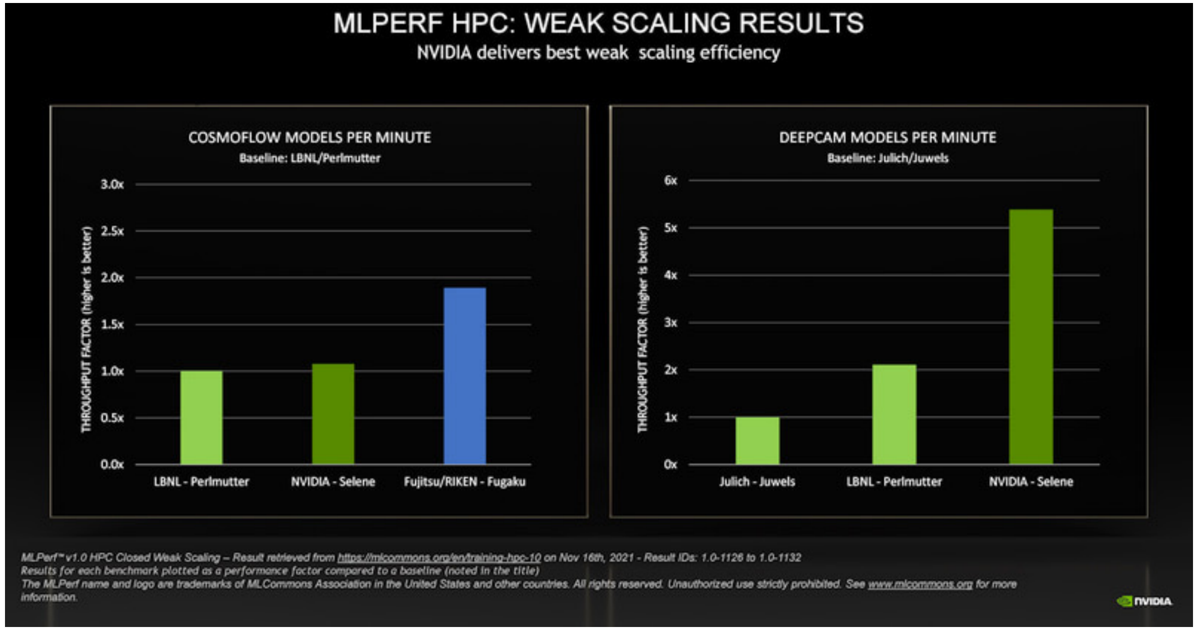
Tokyo, Japan, November 18, 2021

Source: https://www.fujitsu.com/global/about/resources/news/press-releases/2020/1119-02.html

# All results in MLPerf HPC v1.0

- Fugaku is the only CPU-based system amoung the submitters
- The rest of systems are NVIDIA GPU machines

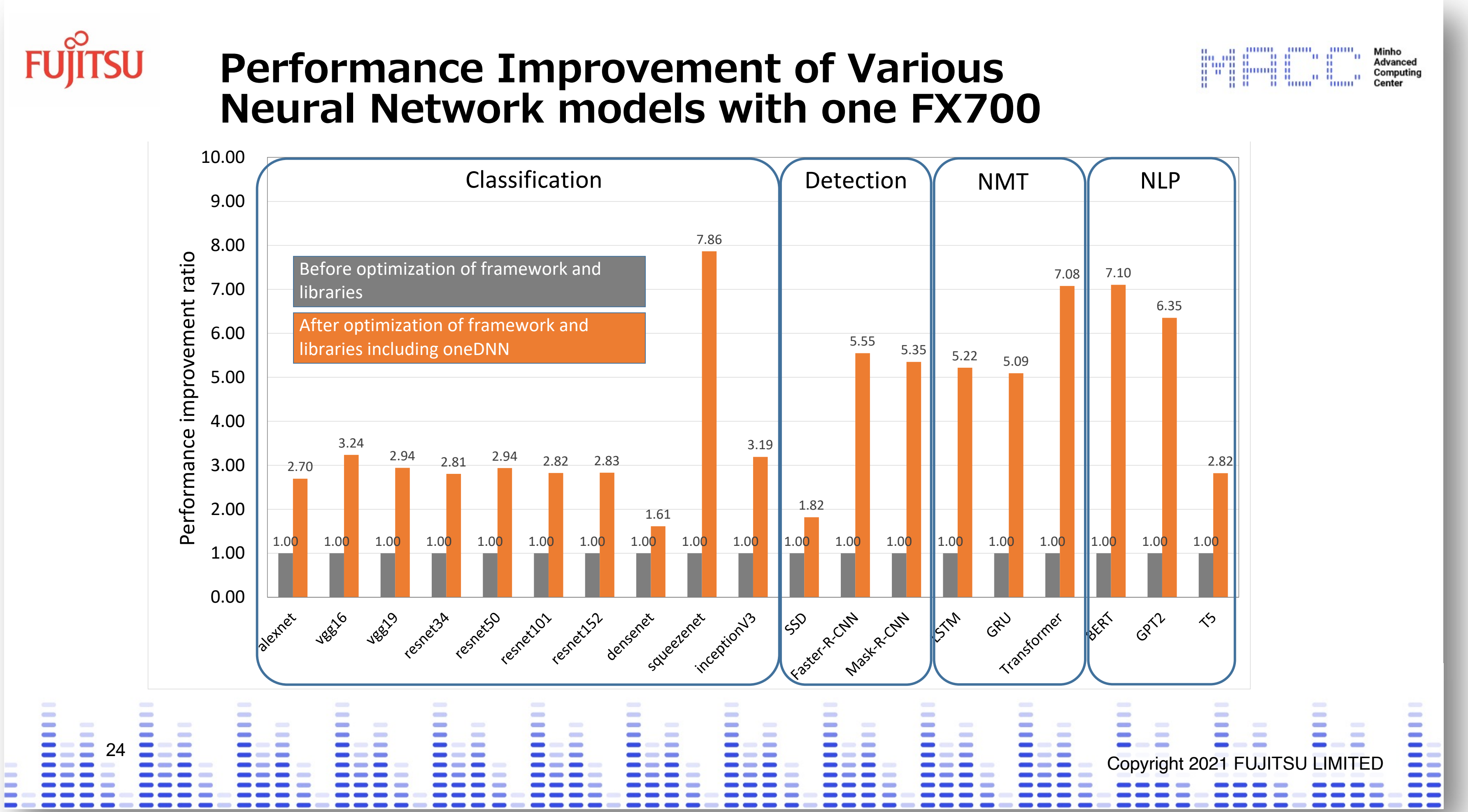



Figures: https://www.hpcwire.com/2021/11/19/mlperf-issues-hpc-1-0-benchmark-results-featuring-impressive-systems-think-fugaku/

# Performance results on other neural networks

- With tuned oneDNN for A64FX, we achieve1.6x to 7.8x performance improvement




Slides: Masafumi Yamazaki (Fujitsu Ltd), "Deep learning on Fugaku", MUG: MACC User Group Workshop, June 2021

# Supported TensorFlow and PyTorch versions on Fugaku

- Fugaku officially supports **TensorFlow-2.2.0**, **PyTorch-1.7.0**/1.6.0. These versions are linked to the Fujitsu's oneDNN library tuned for A64FX
- **Location**
  - FEFS storage : /home/apps/oss/...
- **Package versions**

| 環境 | | | モデル対応 | | | | | 提供状況 | |
|---|---|---|---|---|---|---|---|---|---|
| FW | OneDNN | Horovod | ResNet50 | OpenNMT | ResNetX | BERT | Mask-RCNN | 理研様提供 | Fujitsu github 公開 |
| PT v1.5.0 | v0.21.0 | v0.19.0 | ✓ | | | | | ✓ | ✓ |
| PT v1.6.0 | v1.6.0 | v0.20.3 | ✓ | ✓ | ✓ | | | ✓ | ✓ |
| PT v1.7.0 | v2.1.0 | v0.20.3 | ✓ | ✓ | ✓ | ✓ | | – | ✓ |
| PT v1.7.0 | v2.1.0L01 | v0.20.3 | ✓ | ✓ | ✓ | ✓ | ✓ | – | ✓ |
| TF v2.1.0 | v0.21.2 | v0.19.5 | ✓ | | | | | ✓ | ✓ |
| TF v2.2.0 | v2.1.0 | v0.19.5 | ✓ | ✓ | ✓ | ✓ | | – | ✓ |
| TF v2.2.0 | v2.1.0L01 | v0.19.5 | ✓ | ✓ | ✓ | ✓ | ✓ | – | ✓ |

  - Other: Python ver.3.8.2 + mpi4py ver.3.0.3, pandas ver.1.2.2, numpy ver.1.19.0, scipy ver.1.5.2, h5py ver.2.8.0, libtensorflow_cc.so ver.2.2.0, Batched BLAS ver.1.0, fapp ver.1.0.0 etc.

# Summary

- Data pratform is important for data-drive science
  - We launched a project to build data pre-processing/compression/analysis/utilization platform for RCS facilities (SPring-8, SACLA) and Fugaku
  - For data compression, we introduced TEZip for fast data transfer
    - AI-driven data compression tool designed for time evolutionary data
    - Compression rates are up to 15 in the lossless mode and 50 in the lossy mode in the real SPring-8 data
- DL4 Fugaku Project
  - We extended OneDNN library for A64FX by developoing the Xbyak translator
  - In MLPerf HPC v1.0 (CosmoFlow), Fugaku recieved No. 1 in the weak scaling metric
  - We also tuned many other NNs such as data classification, detection, NMT and NLP
- Working with the operation team, we woulid like to enahcen the usability of Fugaku and other systems

**Our team is seeking for researchers, postdocs and Ph.D. students.**
**If you are interested in joining our projects, pleasae feel free to contact me: kento.sato@riken.jp**