FUJITSU

shaping tomorrow with you

oneAPI AI Technical Advisory Board Meeting

oneDNN for A64FX SVE and MLPerfHPC challenge

Feb. 10, 2021

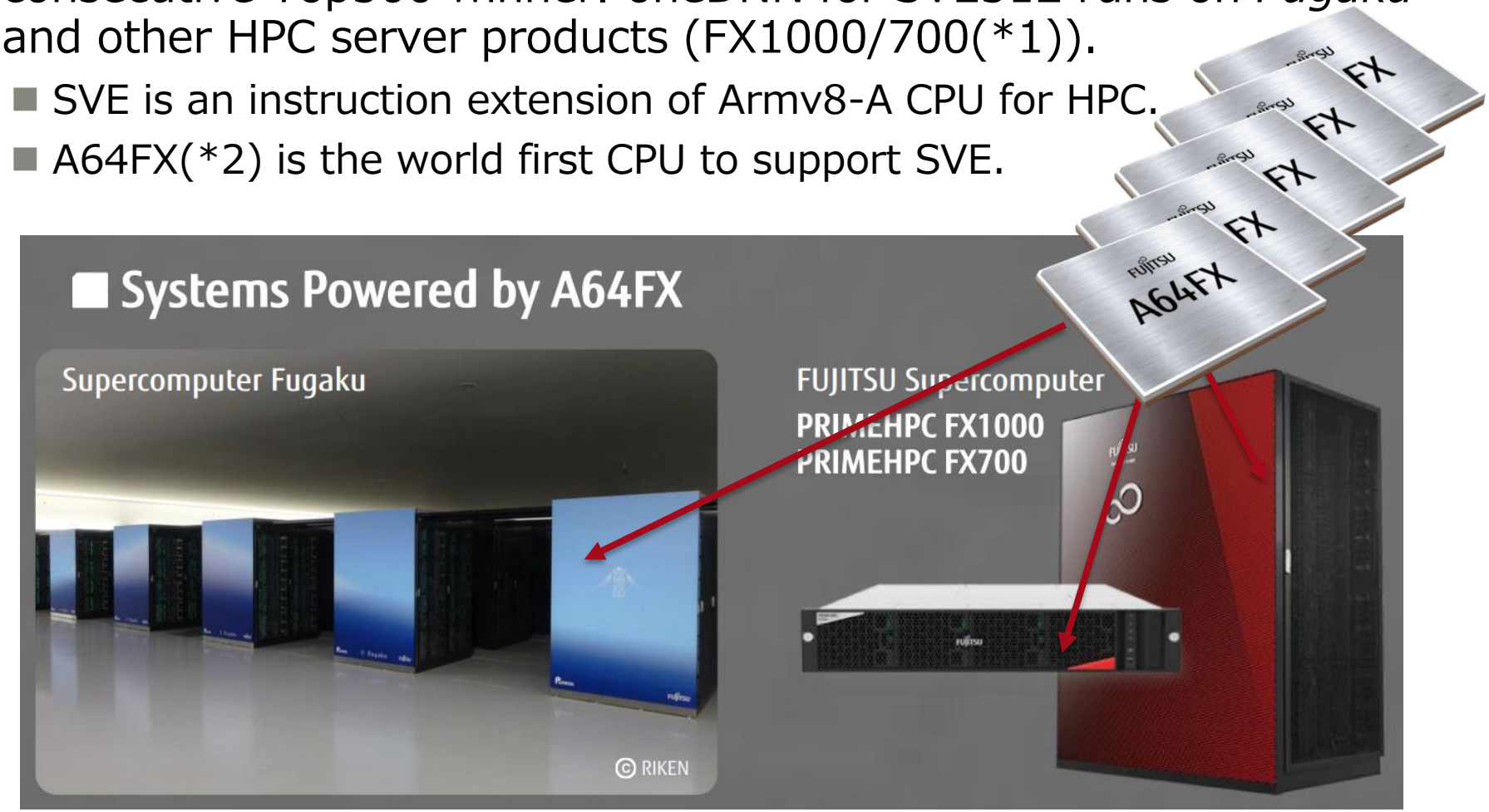
Kentaro KAWAKAMI(kawakami.k@fujitsu.com)
Senior Researcher
Fujitsu Laboratories Ltd.

Acknowledgements

- We would like to thank Intel for releasing the wonderful software, oneDNN, as OSS.
- Thanks to oneDNN, we have made great progress in developing AI processing software for Supercomputer *Fugaku* and Fujitsu's products.

Motivation

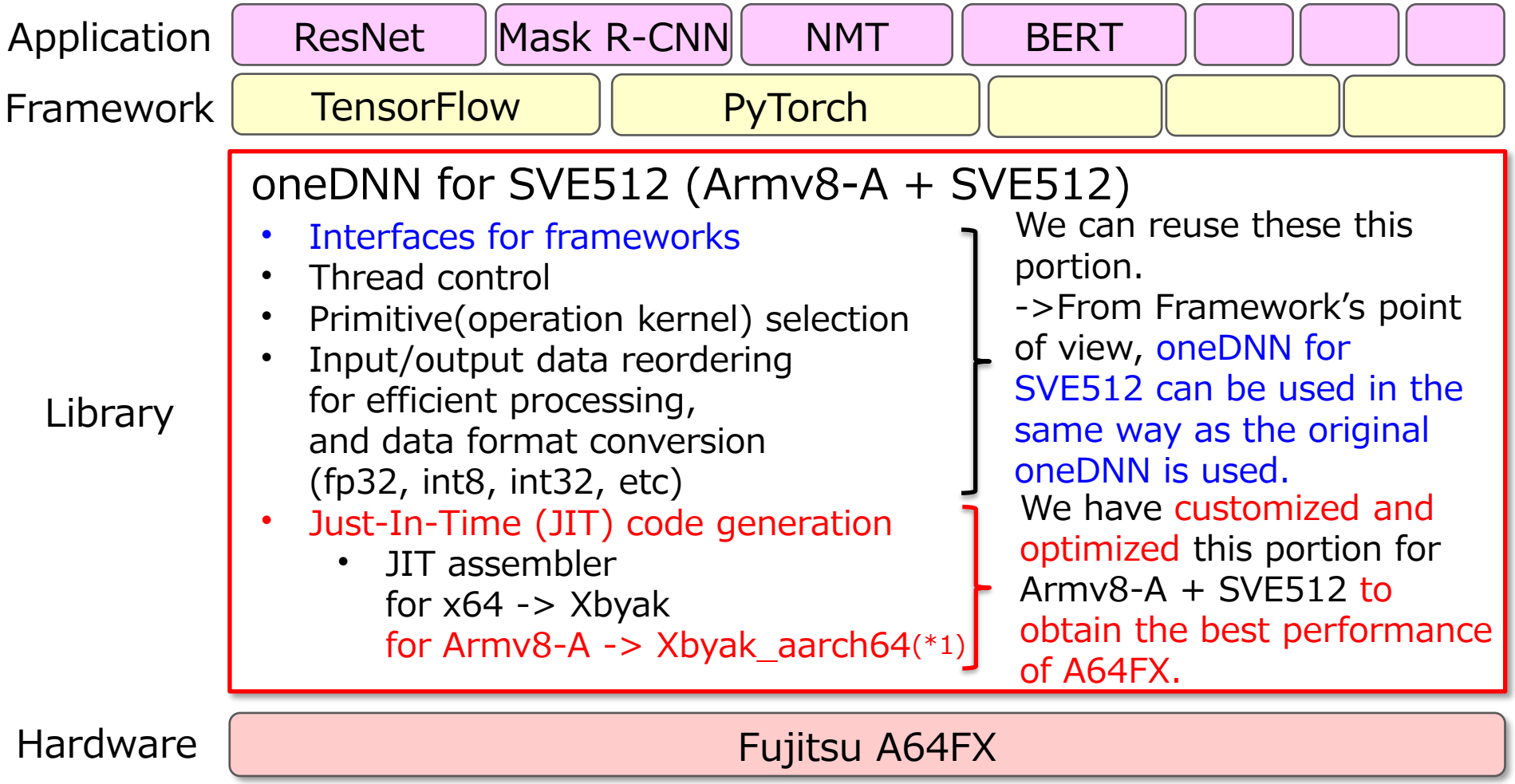
- We have been porting oneDNN for SVE512(Armv8-A + Scalable Vector Extension (SVE)) for Supercomputer *Fugaku*, two consecutive Top500 winner. oneDNN for SVE512 runs on *Fugaku* and other HPC server products (FX1000/700(*1)).
  - SVE is an instruction extension of Armv8-A CPU for HPC.
  - A64FX(*2) is the world first CPU to support SVE.



*1 https://www.fujitsu.com/downloads/JP/jsuper/a64fx/a64fx_infographics_en.pdf
*2 https://www.fujitsu.com/downloads/JP/jsuper/a64fx/a64fx_datasheet_en.pdf

Deep Learning Software Stack for A64FX

FUJITSU



*1 Mitsunari-san of Cybozu Labs, developer of Xbyak(kəi-bja-k) for x64, gave us technical advice and helped our development.

oneDNN for SVE512 Development Status

FUJITSU

Primitimve (Operation kernel)	Data type	JIT impl. for SVE512			To be ported
		Fujitsu's fj_main branch	Pulll requested to Intel?	Merged into Intel?	
Batch_norm	f32	Partially done(*1)	←(*1)	←(*1)	jit_uni_tbb_batch_nomalization.cpp
	s8	Done	←	←	
Convolution	f32	Done	Partially done	Ongoing	Expand acceptable tensor shapes
	s8/u8	Done	Partially done	Ongoing	Expand acceptable tensor shapes
Eltwise	f32	Partially done	←	←	alg=pow, sigmoid
	s8/u8/s32	Done	←	←	
Pooling	f32	Done	←	←	
	s8/u8	Done	←	←	
Reorder		Partially done(*2)	Partially done(*2)	←(*2)	jit_single_blk_kernel_t, jit_blk_reorder_t
Softmax		Done	←	←	
Injector (post_ops. fusion such as Eltwise after convolution, etc)		Done	Ongoing		

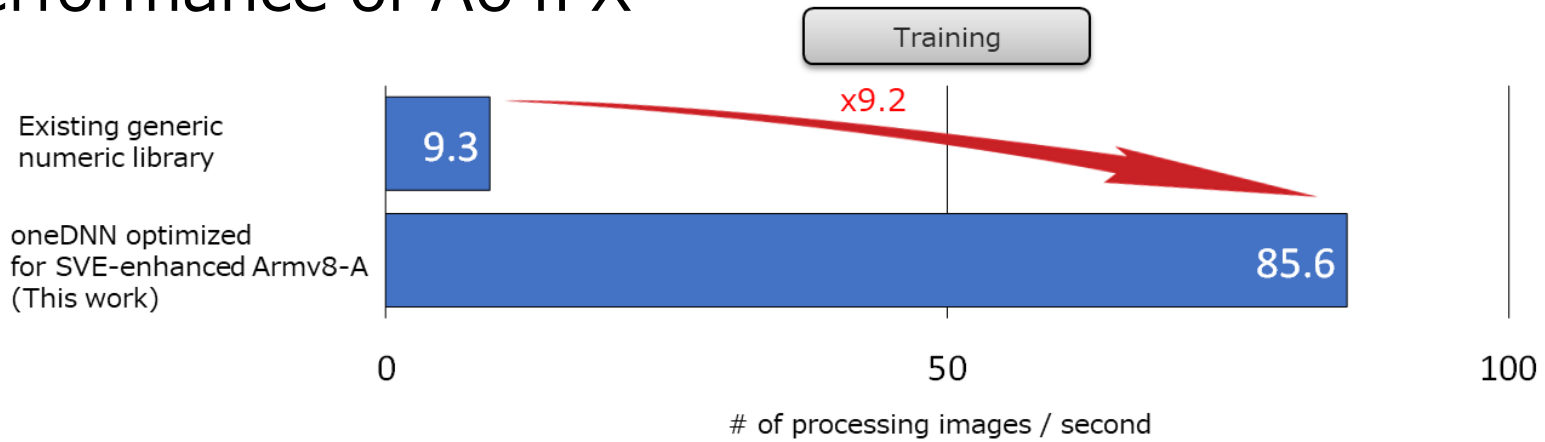
*1 JIT impl. for ASIMD is also partially done.

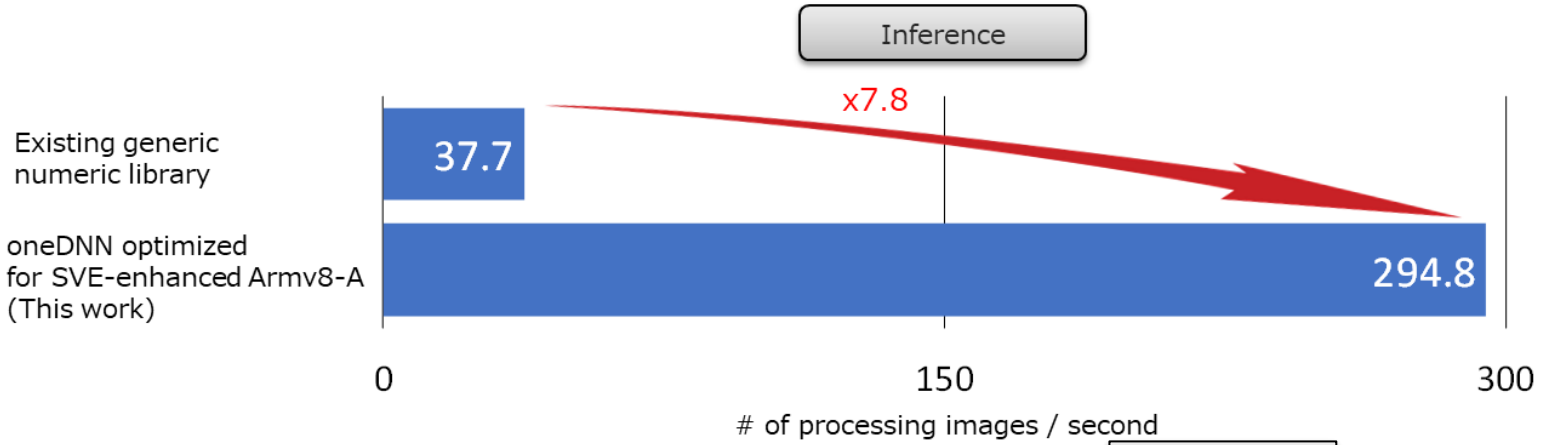
*2 No SVE instructions are used, but SVE 512 availability is checked. It may be possible to lower ISA limit to ASIMD.

Thanks to Intel's help, our source code has been merged into the original oneDNN. Eventually, we want to put everything of our port into the original.

Performance Improvement of oneDNN for SVE512

- Appropriate use of SVE instructions to fully exploit the performance of A64FX





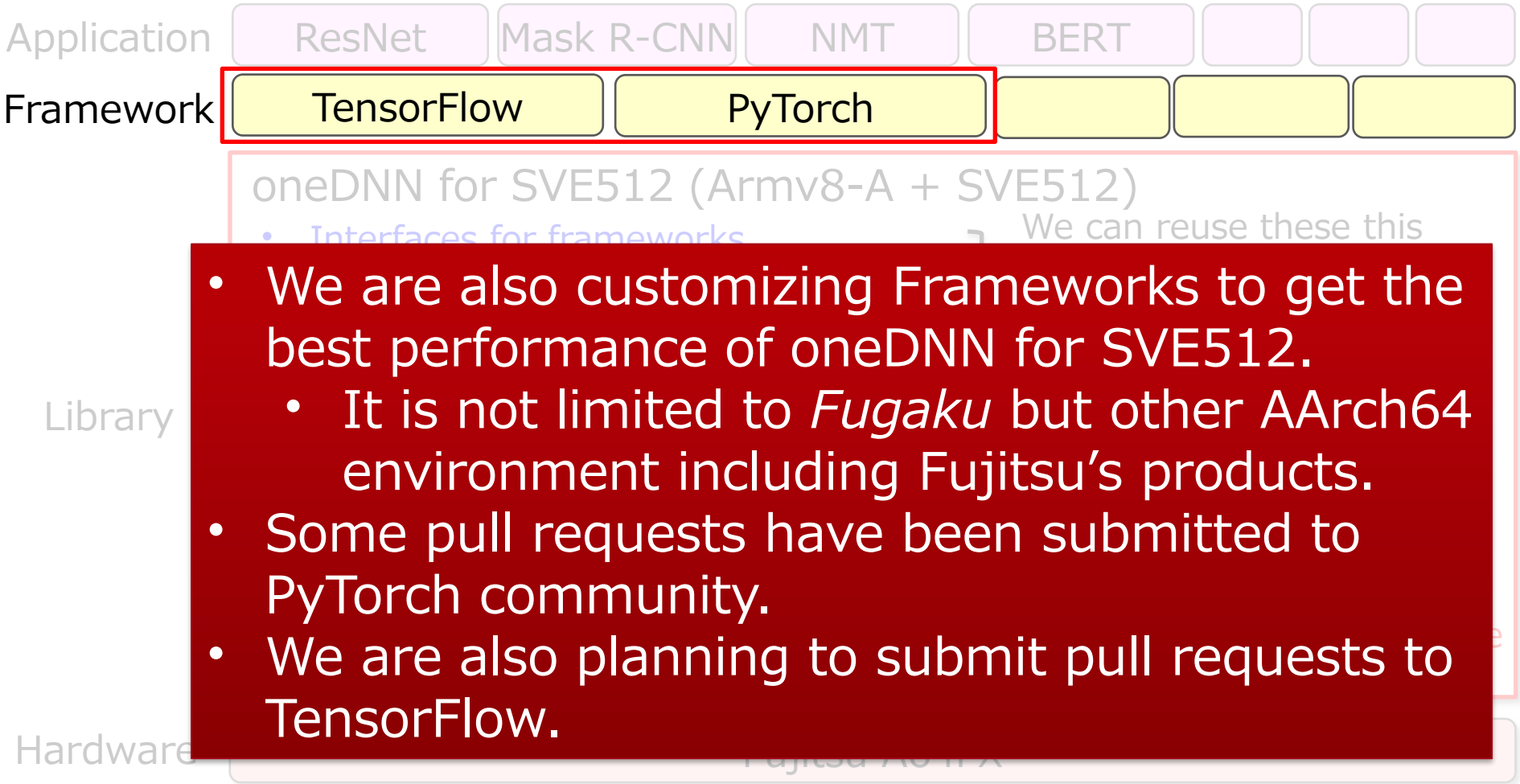
Measurement conditions
- Framework:TensorFlow
- Benchmark:Resnet-50
- CPU: A64FX

TechBlog

https://blog.fltech.dev/entry/2020/11/19/fugaku-onednn-deep-dive-en

Deep Learning Software Stack for A64FX

FUJITSU



- We are also customizing Frameworks to get the best performance of oneDNN for SVE512.
  - It is not limited to *Fugaku* but other AArch64 environment including Fujitsu's products.
- Some pull requests have been submitted to PyTorch community.
- We are also planning to submit pull requests to TensorFlow.

*1 Mitsunari-san of Cybozu Labs, developer of Xbyak(kəi-bja-k) for x64, gave us technical advice and helped our development.

MLPerf HPC v0.7 results

FUJITSU

- Both Closed and Open division, our results outperformed other sites.
  - Closed division

https://mlperf.org/training-results-0-7

Submitter	System	Processor	#	Accelerator	#	Software	Time [min]
Fujitsu	**ABCI**	**Xeon Gold 6148**	**256**	**NVIDIA V100**	**512**	**TensorFlow**	**34.42**
★ **Fujitsu/ RIKEN**	**Fugaku**	**A64FX**	**8192**	**-**	**-**	**TensorFlow + Mesh TensorFlow**	**101.49**
NCSA	HAL	POWER 9 model 2.2	32	NVIDIA V100	64	TensorFlow	265.59
★ **Fujitsu/ RIKEN**	**Fugaku**	**A64FX**	**512**	**-**	**-**	**TensorFlow + Mesh TensorFlow**	**268.77**
CSCS	Piz Daint	Xeon E5-2690 v3	256	NVIDIA P100	256	TensorFlow	327.01
LBNL	Cori GPU	Xeon Gold 6148	16	NVIDIA V100	64	TensorFlow	364.73

oneDNN for SVE512 inside!

  - Open division

Submitter	System	Processor	#	Accelerator	#	Software	Time [min]
Fujitsu	**ABCI**	**Xeon Gold 6148**	**1024**	**NVIDIA V100**	**2048**	**TensorFlow**	**13.21**
★ **Fujitsu/ RIKEN**	**Fugaku**	**A64FX**	**16384**	**-**	**-**	**TensorFlow + Mesh TensorFlow**	**30.07**
LBNL	Cori KNL	Xeon Phi 7250	1024	-	-	Tensorflow	419.69

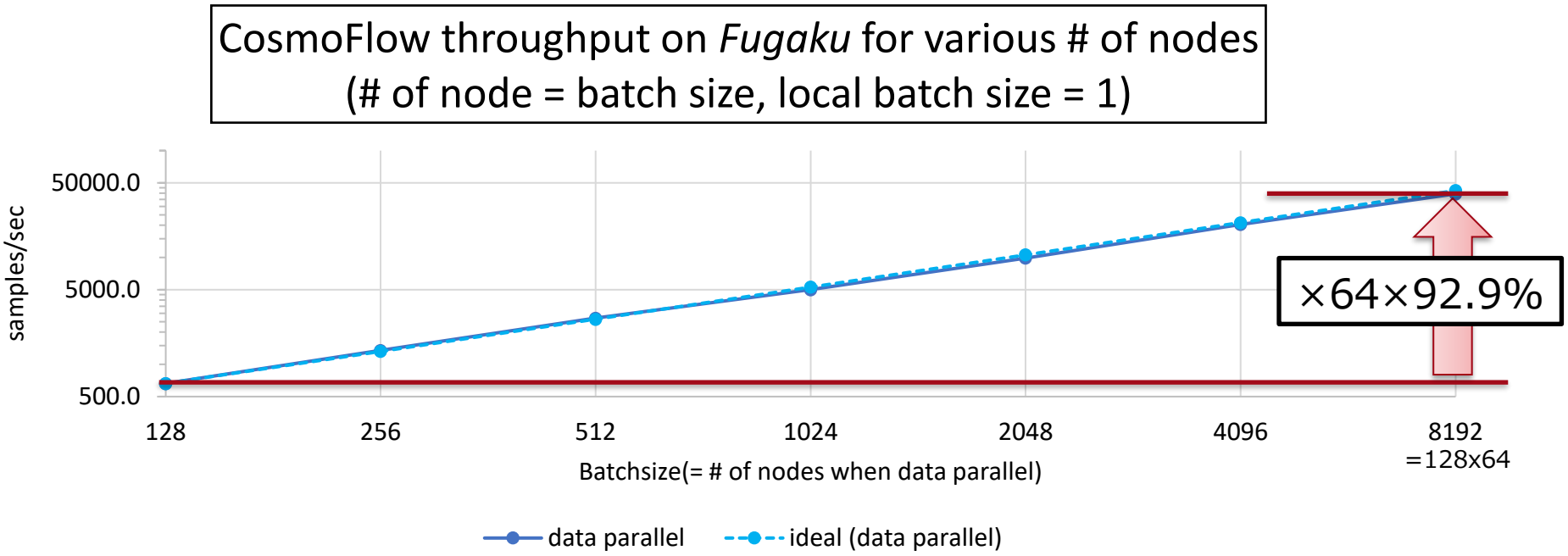
Improvements for MLPerf HPC other than oneDNN for SVE512

FUJITSU

- Improve data staging time
  - Shared SSD storage
  - Compress data
  - Data loader
- Use data pre-read for verification process
- Use Keras auto mixed-precision to reduce calculation workload
- Optimize batch size and other hyper parameters
- Use Mesh-TensorFlow to calculate in model parallel fashion
  →We are planning to submit a pull request of Mesh-TensorFlow soon.

For *Fugaku*, this technique was essential to obtain the best performance of very large-scale system.

Scaling of Throughput for Global Batch Sizes



- The total throughput scales out almost ideally with data parallelism. 8K nodes achieve 92.9% of "(throughput of 128 nodes) x 64)".
- But if we simply increase # of node only with data parallelism, # of epochs, # of iteration to achieve required training accuracy, also increases as shown in the next page.

The Number of Epochs to Converge for Different Batch Sizes

FUJITSU

Global batch sizes	# Epochs	Increase ratio of # epochs	Speed-up ratio
512	Around 45	1	1
×4 → 2048	×2.1 → 90 - 100	2.1	1.90 (≒ 4 / 2.1)
×2 → 4096	×1.63 → 150 - 160	1.63	1.23 (≒ 2 / 1.63)
×2 → 8192	×1.94 → 290 - 310	1.94	1.03 (≒ 2 / 1.94)

Throughput scales out as shown in the previous page.

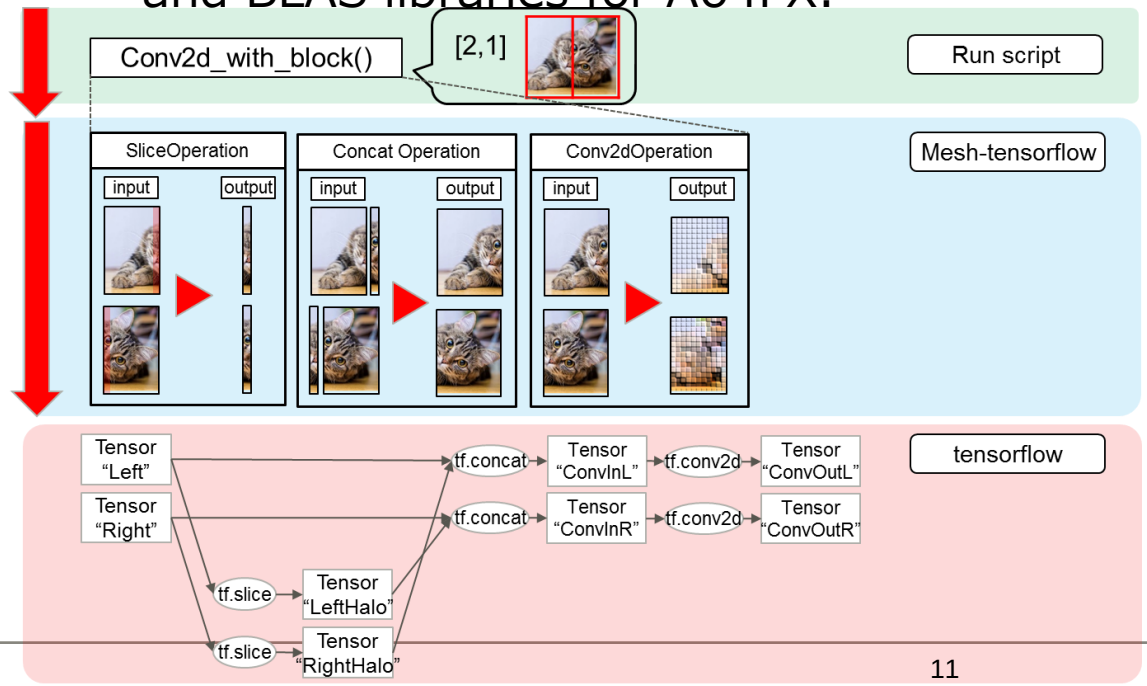
But # of epochs increases so as to global batch size.

Therefore, the total speed-up of deep learning training is limited despite the increased # of nodes (batch size).

- The total **speed-up ratio by data parallelism is limited to batch size = 4096** even if throughput scales ideally.
  - For DeepCAM, speedup ratio is limited to batch size **2048**
- In the current benchmarks, **model parallelism is necessary** in order to reduce runtime by scaling more than 4096 nodes (CPUs or accelerators).

Mesh-TensorFlow for (multi-process)×(model and data parallelisms)

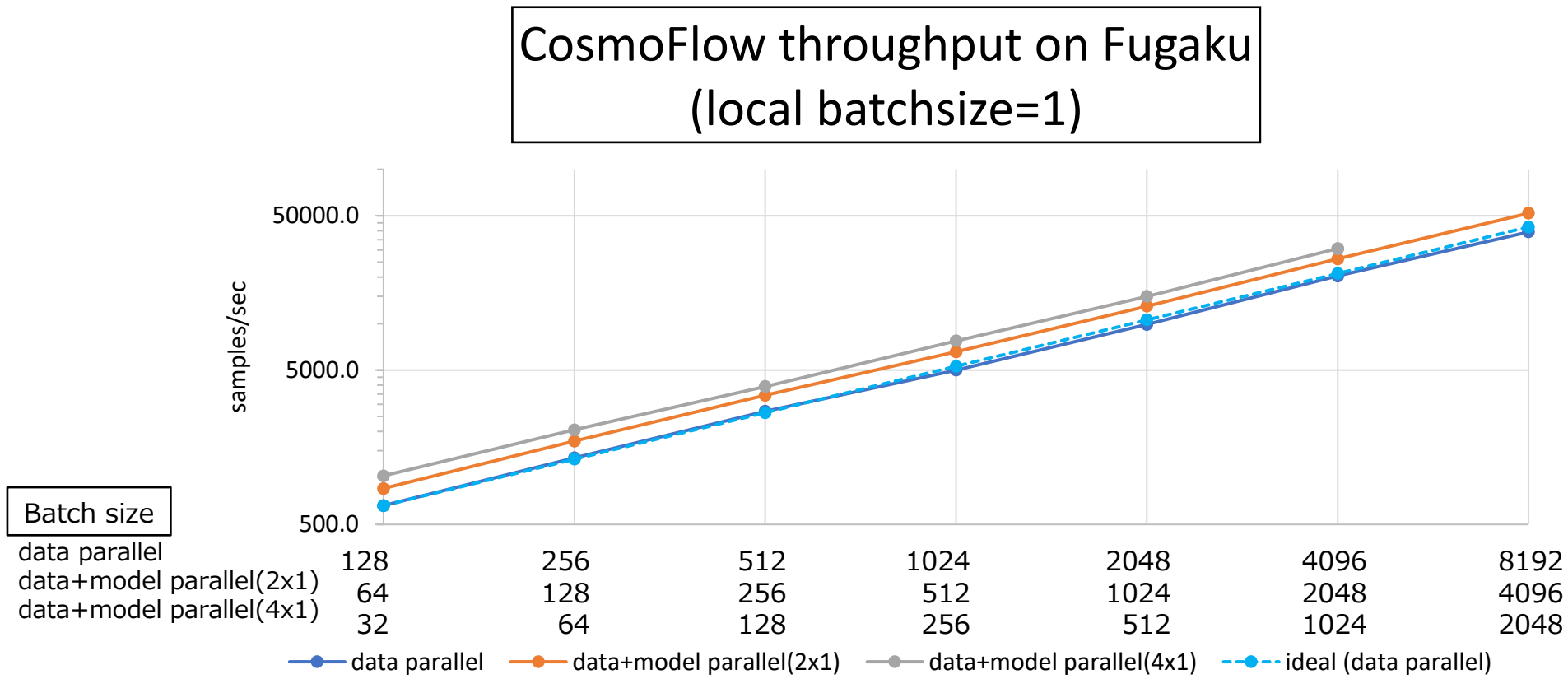
- Mesh-TensorFlow(MTF) converts NN into several operations in order to work in model parallel.
- We modified MTF in order to work in **multi-process model and data parallelisms**.
  - This MTF may work with not only A64FX, but also x64 and other Arm-based CPUs. **We can submit a pull request of our code if original MTF can accept it.**
  - TensorFlow framework is not modified except for oneDNN for SVE512 and BLAS libraries for A64FX.



Our changes from the original

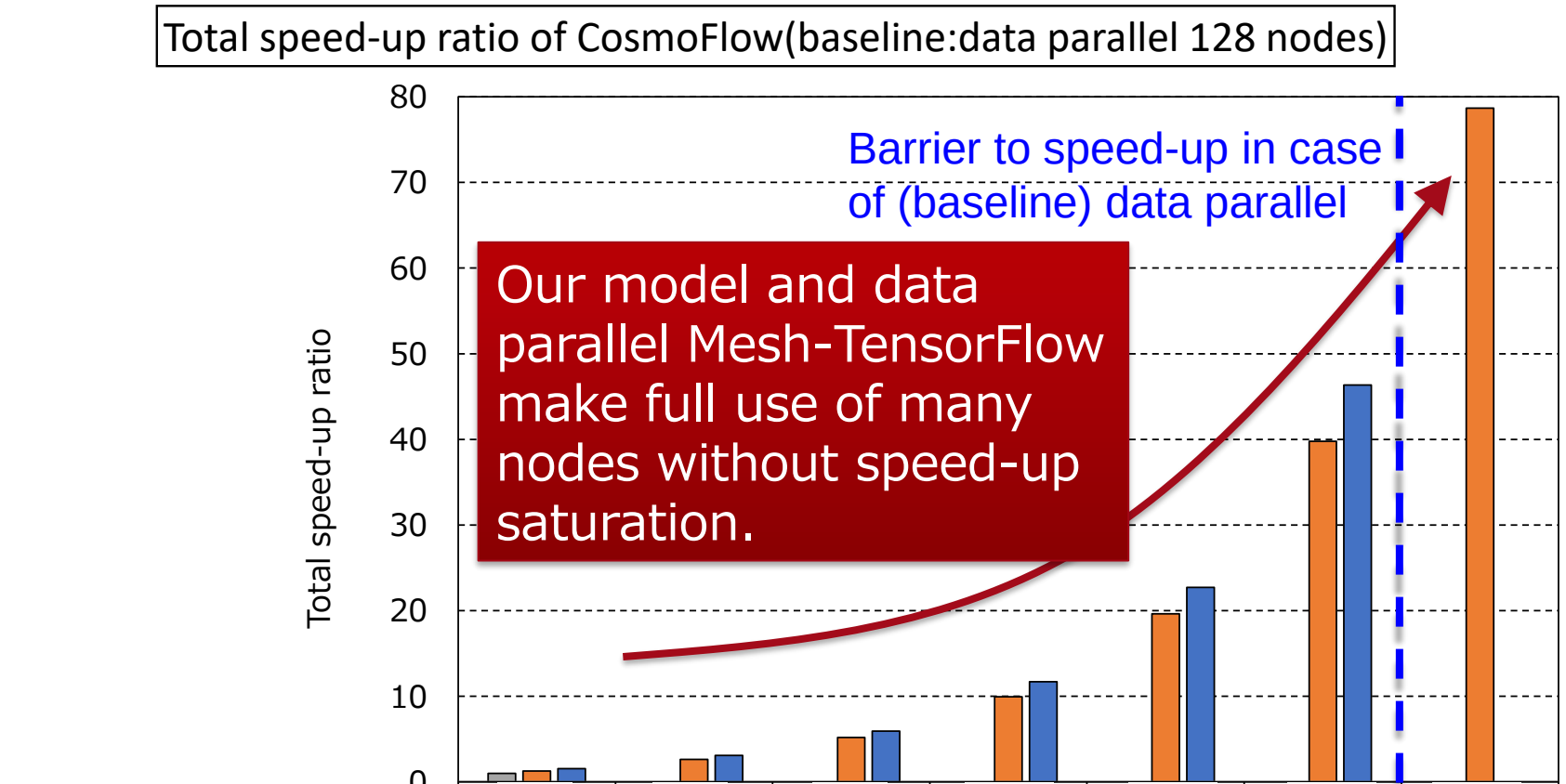
Support	Original MTF	Our MTF
2x2x2 conv3d	No	Yes
Model parallel	Yes	Yes
Data parallel	No	Yes
Multi process	No	Yes

Scaling for Global Batch Sizes



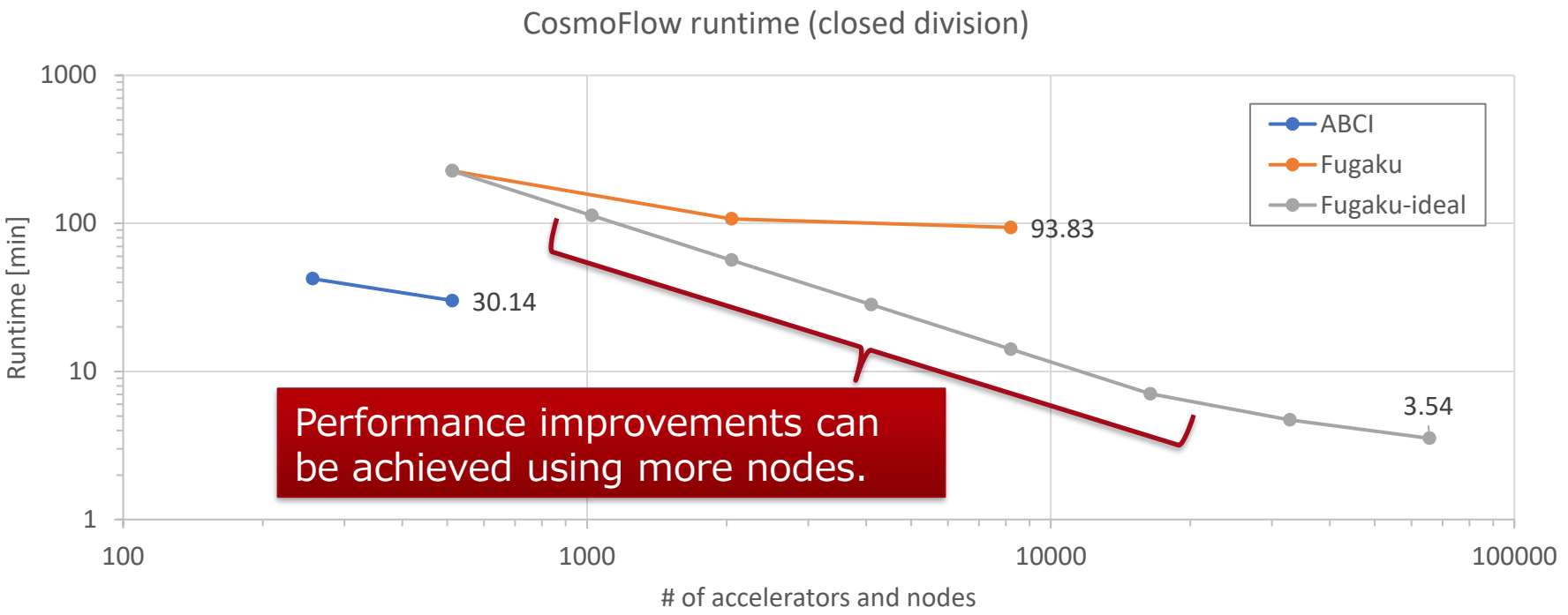
- Our modification for Mesh-TensorFlow, model and data parallelism, keeps the scaling out of total throughput.
  - In fact, our Mesh-TensorFlow achieves better throughput.

Speedup Ratio of Data + Model Parallelisms



# of nodes	128	256	512	1024	2048	4096	8192
Data parallel	1						
Data + model parallel(2x1)	1.29	2.62	5.19	9.93	19.62	39.78	78.68
Data + model parallel(4x1)	1.56	3.1	5.92	11.69	22.73	46.34	

Estimated Runtime Reduction, If Batch Size Limitation is Relaxed



Total # of nodes in *Fugaku* = 158,976

- For *Fugaku*-ideal, assume the number of epochs does not increase up to batch size 16384 and model parallelism scales by 2x using 4 processes per data
- If the rule was relaxed, lowering training accuracy, runtime of *Fugaku* (and other large-scale supercomputers) would be greatly reduced.

Future plans

FUJITSU

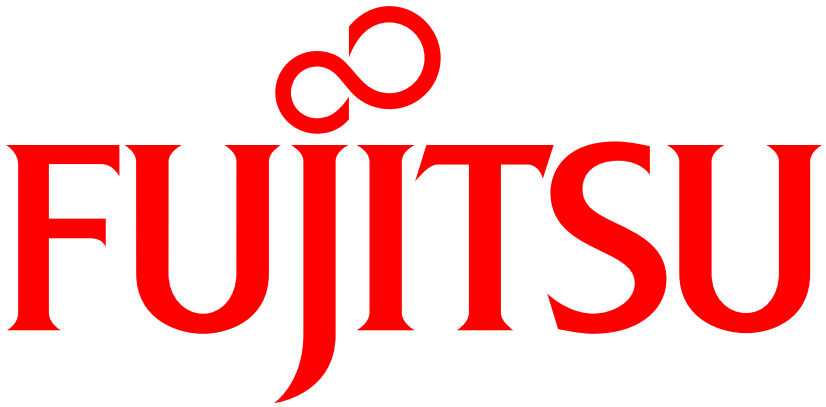
- We continue submitting pull requests of our porting work, oneDNN, PyTorch and TensorFlow.
- Our next target applications are shifting to Mask R-CNN, NMT, BERT. We will tune oneDNN for SVE512 to make these applications faster.
  - So far, we have been developing targeting to ResNET50 and MLPerfHPC.
- To accelerate development with Arm/Linaro/OSS community, we are also releasing the source codes and publishing TechBlogs.

Github repositories

https://github.com/fujitsu/oneDNN/tree/fj_main
https://github.com/fujitsu/xbyak_aarch64
https://github.com/fujitsu/xbyak_translator_aarch64
https://github.com/fujitsu/pytorch
https://github.com/fujitsu/A64FX
etc.

TechBlog

https://blog.fltech.dev/entry/2020/11/19/fugaku-onednn-deep-dive-en



shaping tomorrow with you

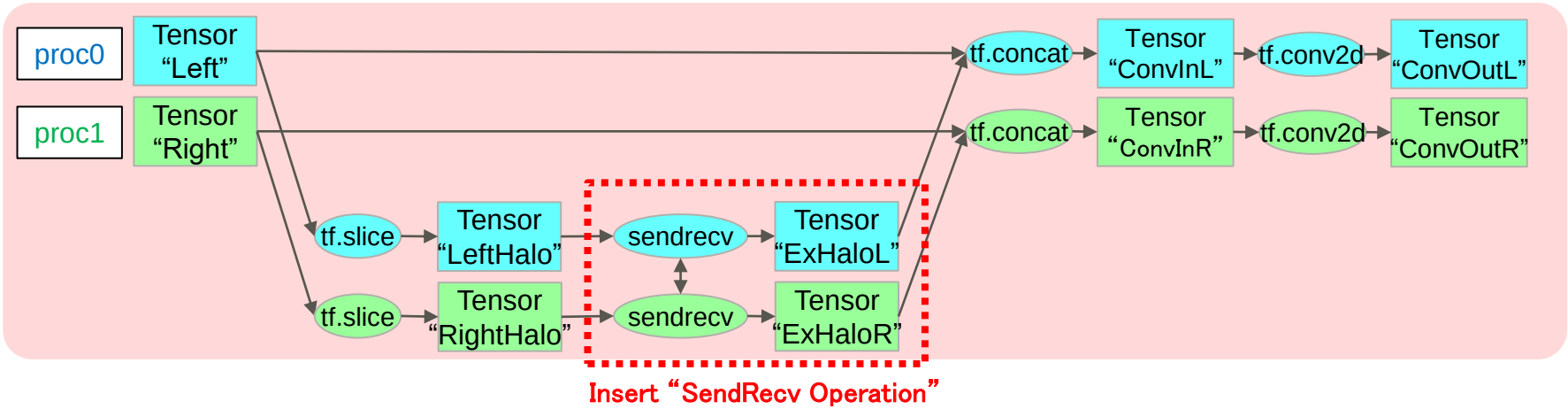
FUJITSU

Appendix

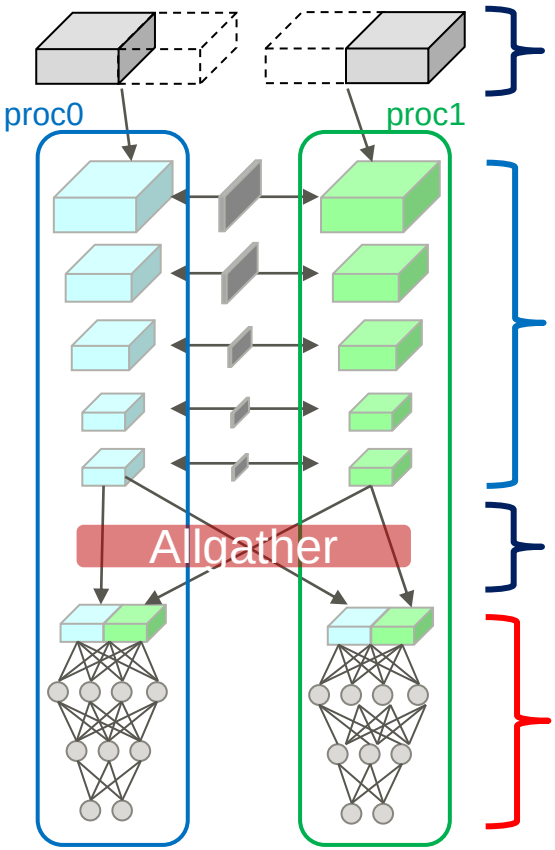
- If you are interested in our customize of Mesh-TensorFlow, please refer this appendix and our source code submitted to MLPerf Training v0.7 Results (https://mlperf.org/training-results-0-7).

An Alternative Mesh Model

- We do not use a mesh model of original MTF for multi-process support.
  - The mesh model of original MTF assumes a common address spaces.
- We defined an alternative mesh model using MPI communicator.
- e.g.) Halo exchange between multi processes
  - Our MTF has “SendRecv Operation” using mpi4py and exchanges the buffers between processes.



Model Parallelism in CosmoFlow

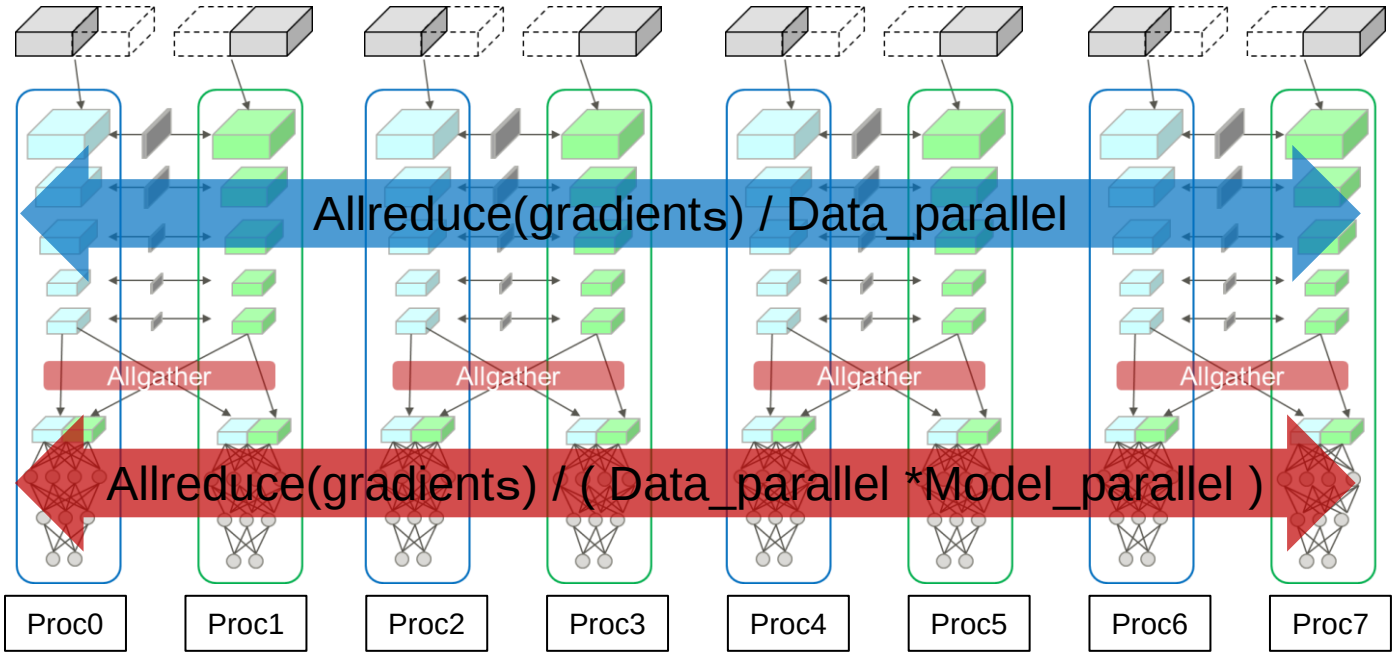


- **Data loader**
  - Each process in model parallelism has same input data and clips the input data according to the own process number in model parallelism.
- **Conv3d layers**
  - Model parallelism is applied.
  - Communicates halo using send/recv function of mpi4py.
  - Each process has same weights.
    - Using same initializer as reference model.
    - Each process initialized with same seed number.
- **Conv3d_to_dense layer**
  - Calls allgather communication and concatenates each received buffer.
- **Dense layers**
  - Model parallelism is **Not** applied
  - Each process in model parallelism is given the same input by allgather operation
  - Each process has same weights.

Computing Gradients in Data + Model parallelisms

FUJITSU

- Since each conv3d layer calculates gradients for the divided inputs, **the gradients are divided by the number of model parallel processes**.
  - For dense layers, the gradients are divided by the number of data + model parallel processes



e.g.)
- Data_parallel : 4
- Model_parallel : 2
  = Total 8 procs