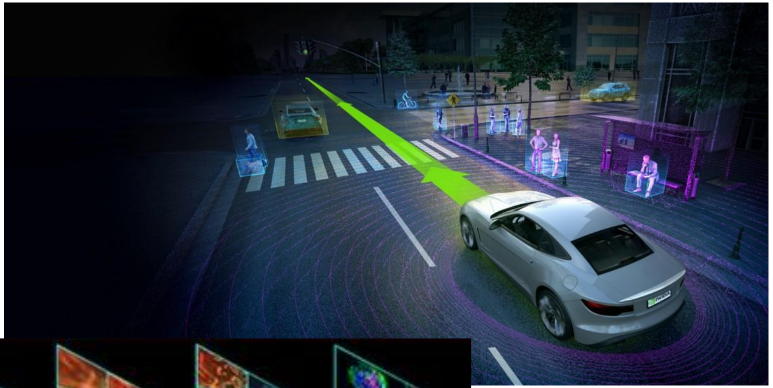
Performance Study on CPU-based Machine Learning with PyTorch

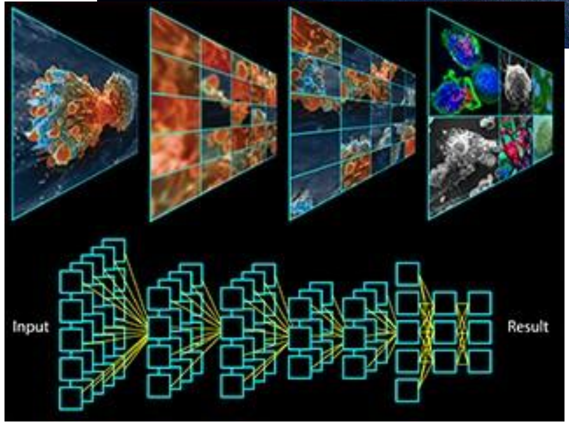
Smeet Chheda, Tony Curtis, Eva Siegmann, Barbara Chapman
IACS, Stony Brook University

Cray User Group Meeting 2023
Helsinki, Finland

Agenda

Introduction: Machine Learning





- ML algorithms and models are being continuously developed and deployed in multiple places.

- PyTorch is a popular open source machine learning library based on the Torch library.
  - It provides fast tensor computations with gpu acceleration and reverse-mode automatic differentiation through autograd.
  - Easy to use.

Software Stack and Communications setup

Library versions are cloned from GitHub or downloaded from respective websites (Python, PyPI) and kept consistent across all environments.

Library	Version
Python	3.8.2
Numpy	1.22.4
PyTorch	1.10.0
Torchvision	0.11.0
oneDNN	2.4.3
Horovod	0.24.3

Cluster	Compiler	UCX	MPI	Interconnect
Ookami	Fujitsu	-	Fujitsu MPI*	Infiniband HDR 100
Ookami	ARM / LLVM / GNU	1.11.2	Open MPI v4.1.2	Infiniband HDR 100
Stampede2	GNU	-	Intel MPI v19.0.9	Intel Omni-Path 100Gb/s

*Fujitsu MPI is based on Open MPI v4.0.1

oneDNN

- oneAPI Deep Neural Network Library (oneDNN) is an open-source cross-platform performance library of basic building blocks for deep learning applications.

- It includes highly vectorized and threaded building blocks for implementation of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) with C and C++ interfaces. This project is created to help the DL community innovate on the Intel(R) processor family.

- PyTorch is built with oneDNN v2.4.3 support (formerly known as MKL-DNN)
  - Fujitsu achieved this by creating an aarch64 version of xbyak JIT assembler.
  - *xbyak_aarch64* and *xbyak_translator_aarch64* have been primarily developed to enable assembly coding with full SVE support and porting oneDNN to aarch64.

CPU, Compiler and BLAS Libraries

Cluster	CPU	Compiler	BLAS Library	Compiler Flags
Ookami	Fujitsu A64FX	Fujitsu compiler v4.7	SSL2	*-Nclang -Kfast -Knolargepage -lpthread*
Ookami	Fujitsu A64FX	Arm v22.0.1	Arm Performance Libraries v2022.0.1	*-O3 -pthread -mcpu=a64fx -mtune=a64fx*
Ookami	Fujitsu A64FX	GNU v11.2.0	OpenBLAS v0.3.19	*-O3 -pthread -mcpu=a64fx -mtune=a64fx*
Ookami	Fujitsu A64FX	LLVM v16.0.0	OpenBLAS v0.3.19	*-O3 -pthread -mcpu=a64fx -mtune=a64fx*
Rusty	Intel Xeon Gold 6148 (40 cores)	GNU v10.3.0	OpenBLAS v0.3.19	*-O3 -lpthread -mtune=skylake-avx512*
Rusty	Intel Xeon Gold 6148 (40 cores)	Intel Compiler v2022.0.1	MKL v2022.0.1	*-O3 -lpthread -mtune=skylake-avx512*
Popeye	Intel Xeon Platinum 8358 (64 cores)	Intel Compiler v2022.0.1	MKL v2022.0.1	*-O3 -lpthread -mtune=icelake-server*
Stampede2	Intel Xeon Platinum 8160 (48 cores)	GNU v9.1.0	MKL v19.1.1	*-O3 -lpthread -mtune=skylake-avx512*

Evaluation: benchDNN

```
./benchdnn --DRIVER [COMMON-OPTIONS] [DRIVER-OPTIONS] PROBLEM-DESCRIPTION
```

benchdnn is an extended and robust correctness verification and performance benchmarking tool for the primitives provided by oneDNN.

- benchdnn itself is a harness for different primitive-specific drivers.

We evaluate the convolution driver in performance mode.

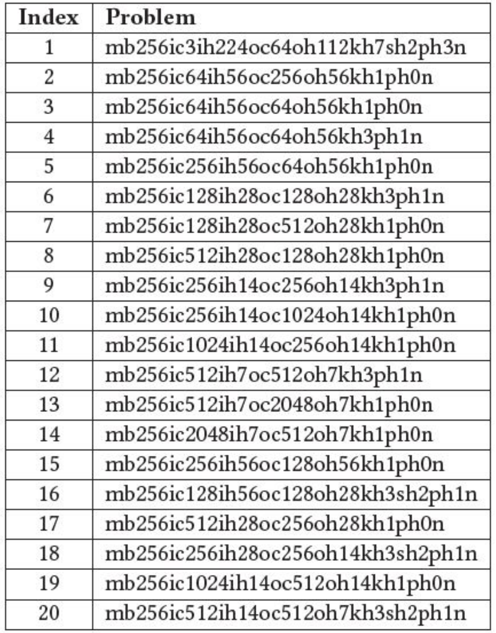
- Direction={forward with bias, backward with bias}
- Configuration={f32, u8s8s8} (f32 : 32 bit float, s8 : signed int8_t, u8 : unsigned uint8_t)
- Run configuration: `OMP_NUM_THREADS=8`

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Exascallab Exascale Programming Models Laboratory

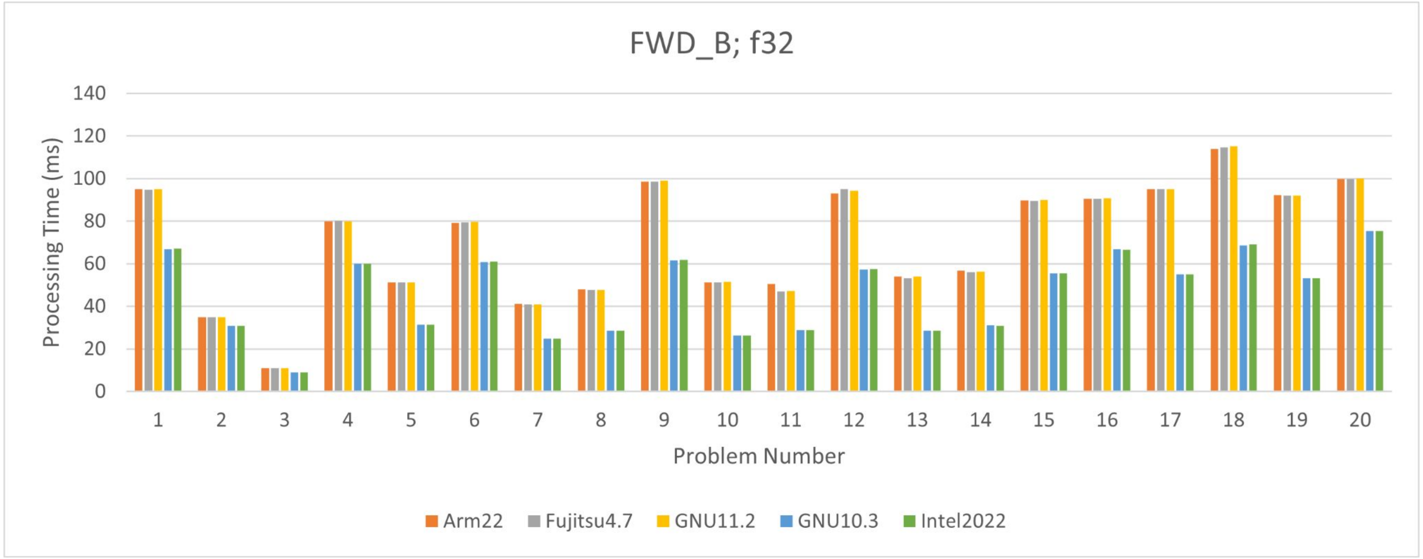
Evaluation: benchDNN

Entry	Description
mb	Mini batch size
ic, oc	Input and output channels
id, ih, iw	Input depth, height and width
od, oh, ow	Output depth, height and width
kd, kh, kw	Kernel depth, height and width
sd, sh, sw	Stride depth, height and width
pd, ph, pw	Front, top and left padding
n	Descriptor name



Index	Problem
1	mb256ic3ih224oc64oh112kh7sh2ph3n
2	mb256ic64ih56oc256oh56kh1ph0n
3	mb256ic64ih56oc64oh56kh1ph0n
4	mb256ic64ih56oc64oh56kh3ph1n
5	mb256ic256ih56oc64oh56kh1ph0n
6	mb256ic128ih28oc128oh28kh3ph1n
7	mb256ic128ih28oc512oh28kh1ph0n
8	mb256ic512ih28oc128oh28kh1ph0n
9	mb256ic256ih14oc256oh14kh3ph1n
10	mb256ic256ih14oc1024oh14kh1ph0n
11	mb256ic1024ih14oc256oh14kh1ph0n
12	mb256ic512ih7oc512oh7kh3ph1n
13	mb256ic512ih7oc2048oh7kh1ph0n
14	mb256ic2048ih7oc512oh7kh1ph0n
15	mb256ic256ih56oc128oh56kh1ph0n
16	mb256ic128ih56oc128oh28kh3sh2ph1n
17	mb256ic512ih28oc256oh28kh1ph0n
18	mb256ic256ih28oc256oh14kh3sh2ph1n
19	mb256ic1024ih14oc512oh14kh1ph0n
20	mb256ic512ih14oc512oh7kh3sh2ph1n

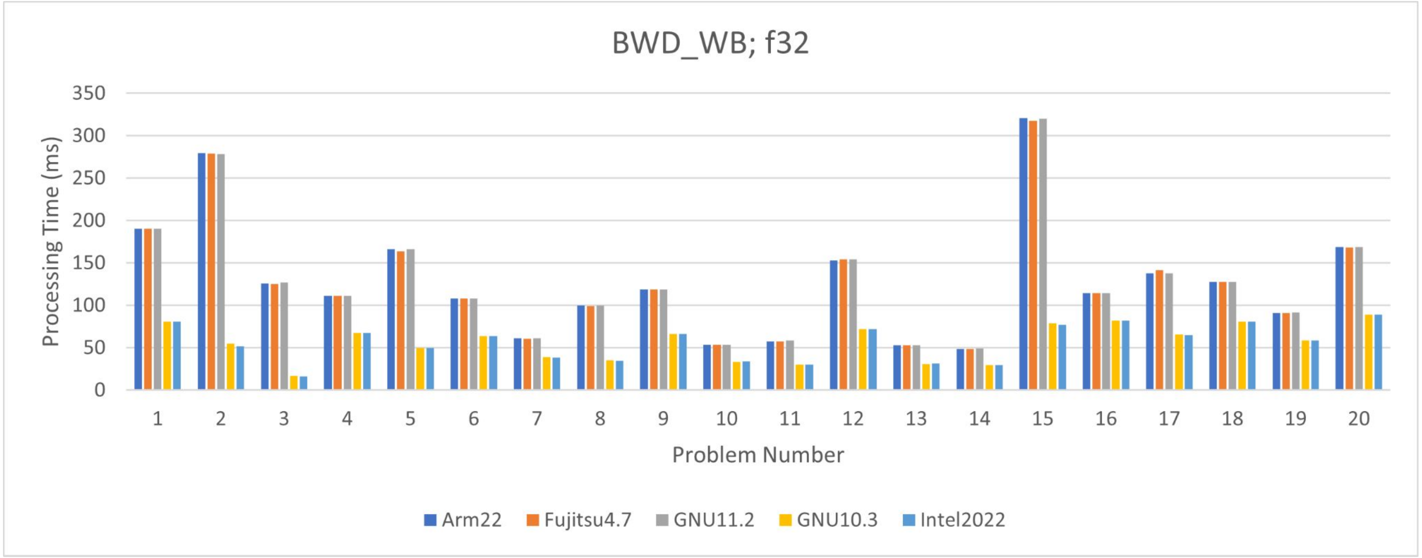
Evaluation: benchDNN (contd)



Blue and green bars represent runs on Intel Xeon Gold 6148.
Orange, Grey and Yellow bars represent runs on Fujitsu A64FX

AVX512 kernels perform better than SVE kernels

Evaluation: benchDNN (contd)



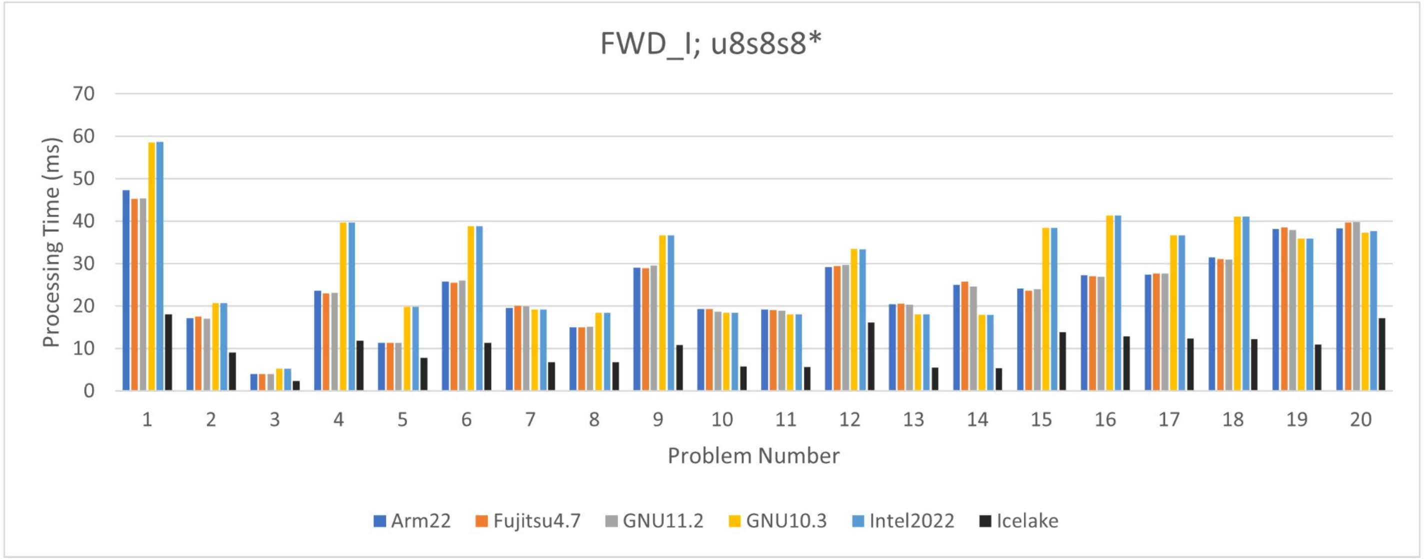
Yellow and light-blue bars represent runs on Intel Xeon Gold 6148.
Blue, Orange and Grey bars represent runs on Fujitsu A64FX

AVX512 kernels **significantly** outperform SVE kernels in some cases

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Exascalab Exascale Programming Models Laboratory

Evaluation: benchDNN (contd)



Yellow and light-blue bars represent runs on Intel Xeon Gold 6148.
Black bar represents Intel compiler on Intel Xeon Platinum 8358 (Icelake)
Blue, Orange and Grey bars represent runs on Fujitsu A64FX
SVE kernels have better inference, however are no match for VNNI optimized kernels

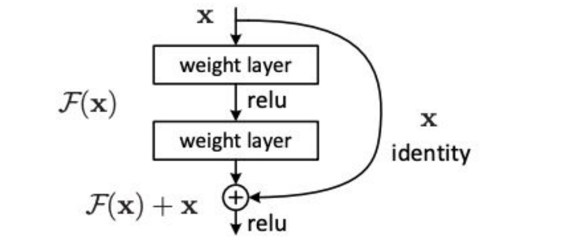
iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

ExascaLab Exascale Programming Models Laboratory

Evaluation: Models used

- We evaluate the performance of 3 different deep residual networks.
  - ResNet 34 (21.79M), ResNext 50-32x4d (25.02M), Wide ResNet 101 (126.88M)
  - Easy to convert to block format and therefore they can take advantage of optimized oneDNN kernels.

- Other models in *torchvision* do not support block formats at the moment
  - Unsupported operators for example, concatenation



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

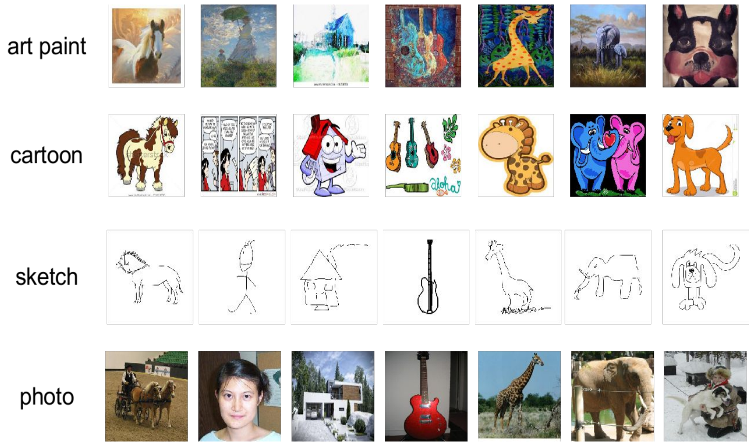
Image courtesy of He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Evaluation: Single Node

- Task: Image Classification
- Dataset: **P**hoto, **A**rt Painting, **C**artoon, **S**ketch (PACS*) - 4 domains, 7 classes, 9991 images

  Training : 6101 images
  Transforms : Resizing, Horizontal Flips, Color Jittering, Gray scaling, tensor conversion, normalization

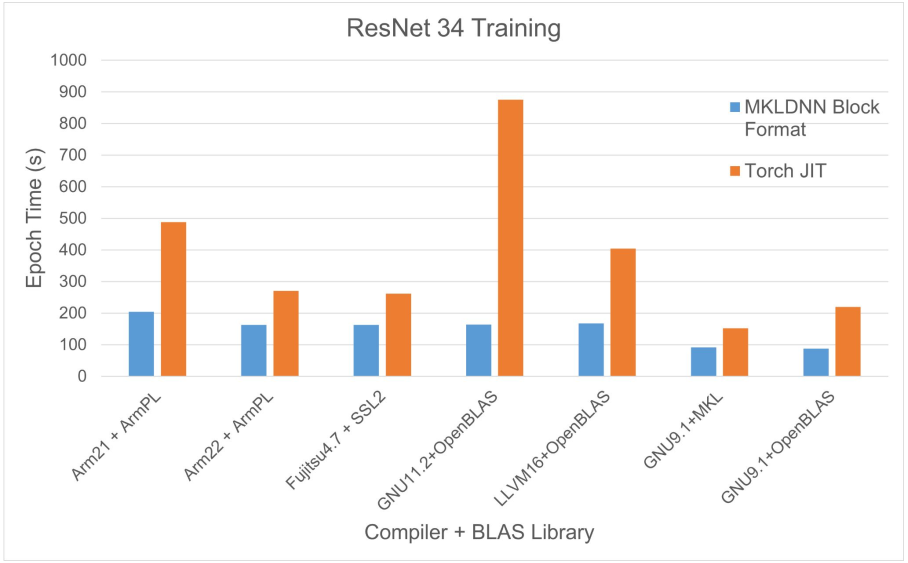
  Evaluation : 3942 images
  Transforms : Resizing, tensor conversion, normalization
- Model training & inference can be improved by using different memory formats (NCHW default, NHWC, nChw16c - mkldnn block format)
- Using TCMalloc for memory allocation.

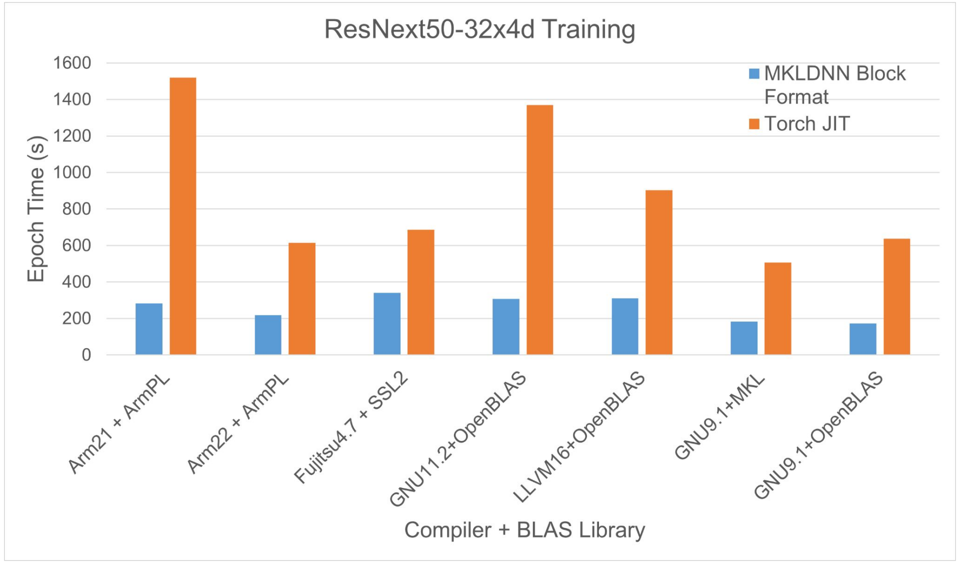


INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Exascalab Exascale Programming Models Laboratory

Evaluation: Single Node





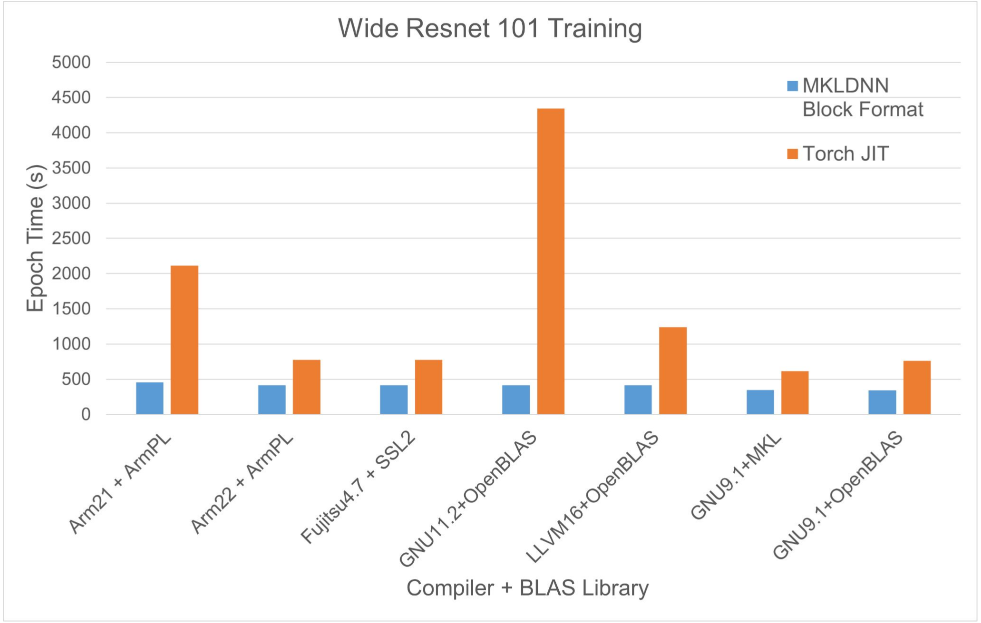
Training Time per Epoch (lower is better)

ARM and Fujitsu compiler binaries have similar performance for ResNet 34, however not in the case of ResNext50-32x4d

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

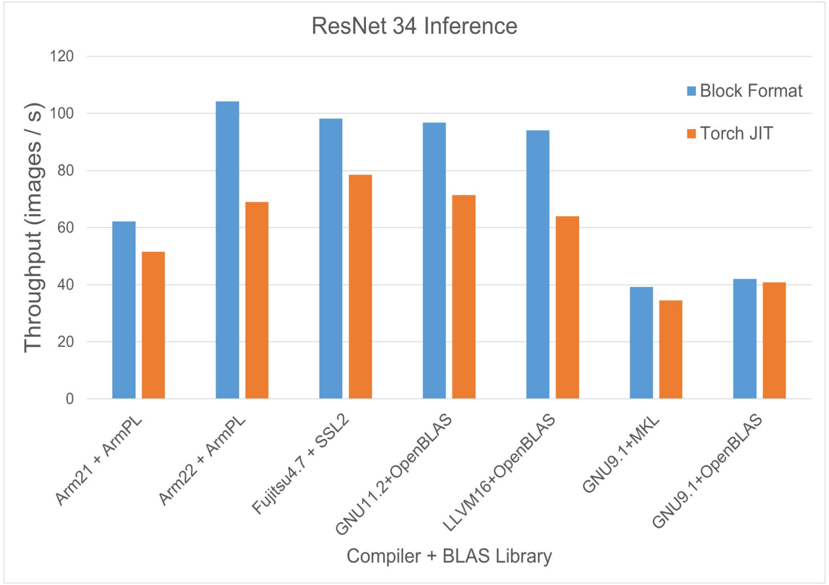
ExascaLab Exascale Programming Models Laboratory

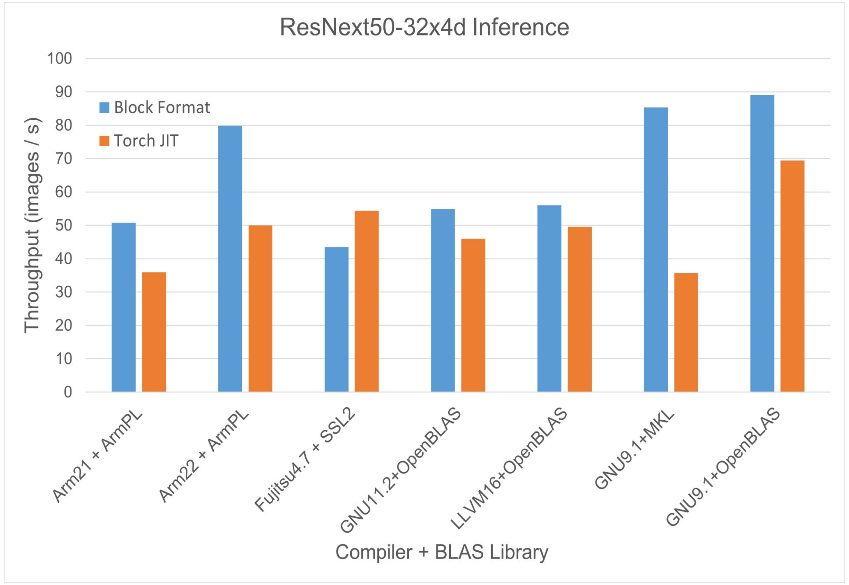
Evaluation: Single Node



Training Time per Epoch (lower is better)

Evaluation: Single Node





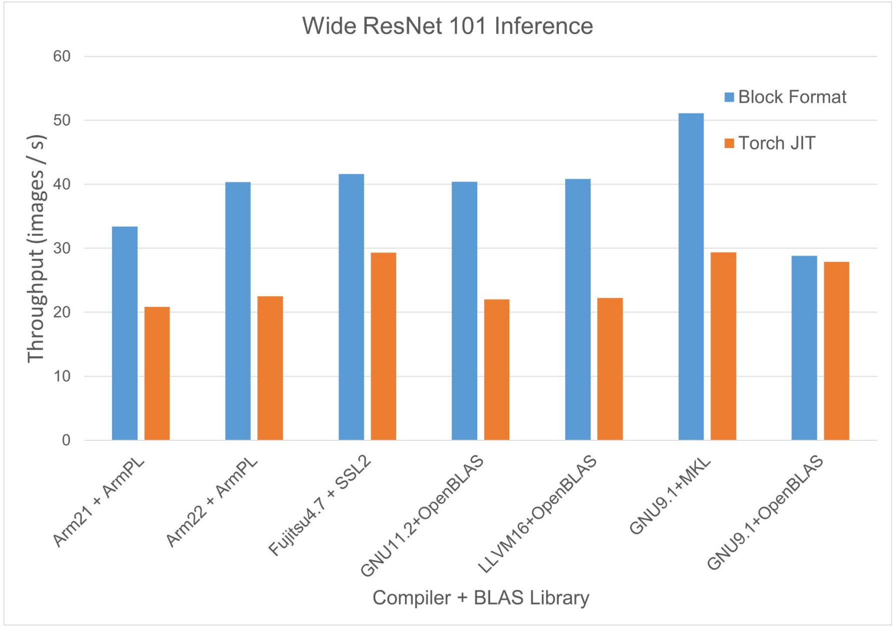
Inference Throughput (higher is better)

High throughput observed for A64FX builds compared to Skylake for ResNet 34, strange performance issue with ResNext 50-32x4d in Fujitsu compiler build

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

ExascaLab Exascale Programming Models Laboratory

Evaluation: Single Node



Inference Throughput (higher is better)

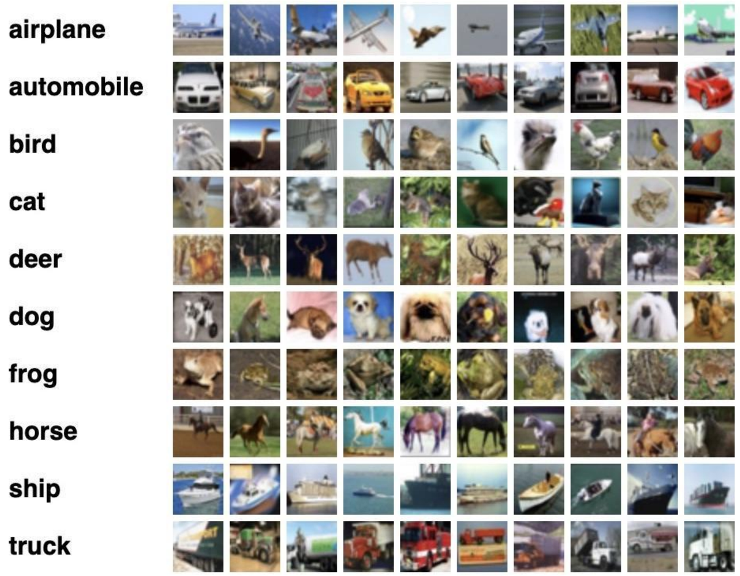
Evaluation: Multi Node

- Dataset: CIFAR-10* - 10 classes, 60000 images
- Train batch size = 128 images

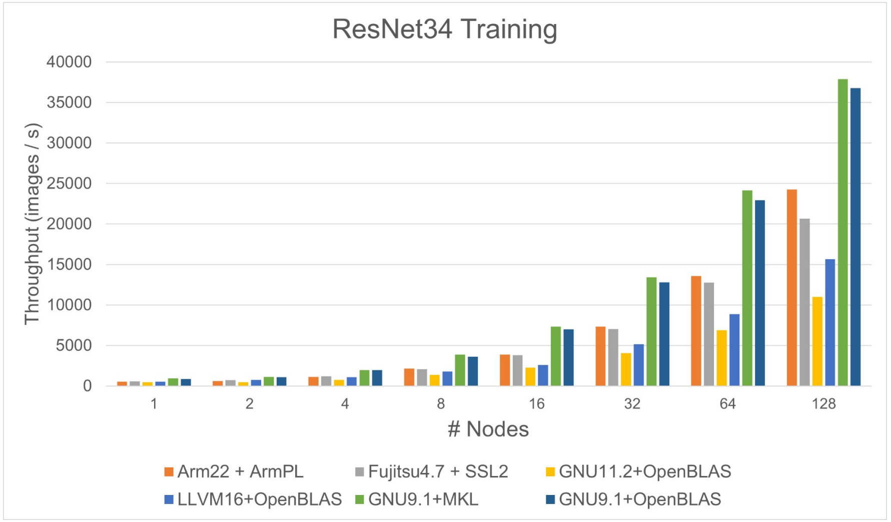
  Training : 50,000 images
  Transforms : Resizing, tensor conversion, normalization

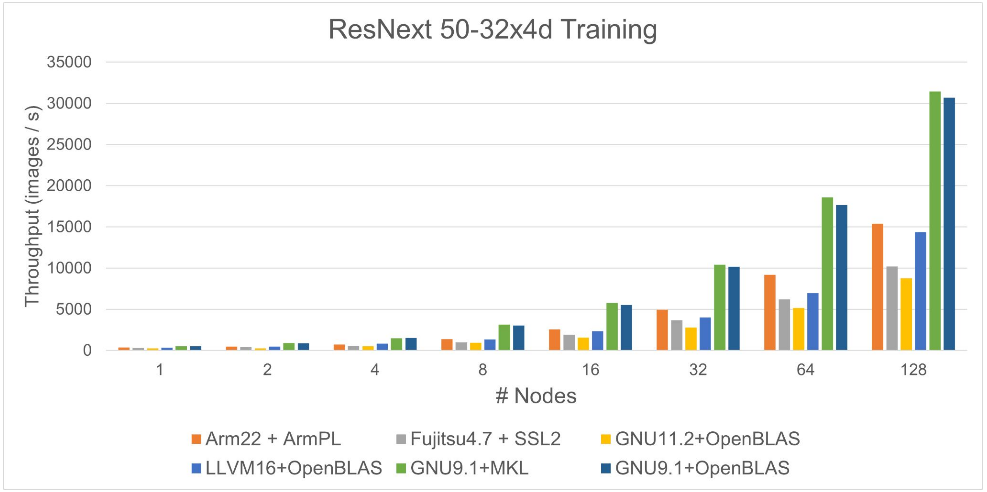
  Evaluation : 10,000 images
  Transforms : Resizing, tensor conversion, normalization

- Process mapping achieved by `--map-by` flag
- Weak-scaling runs upto 128 nodes



Evaluation: Multi Node





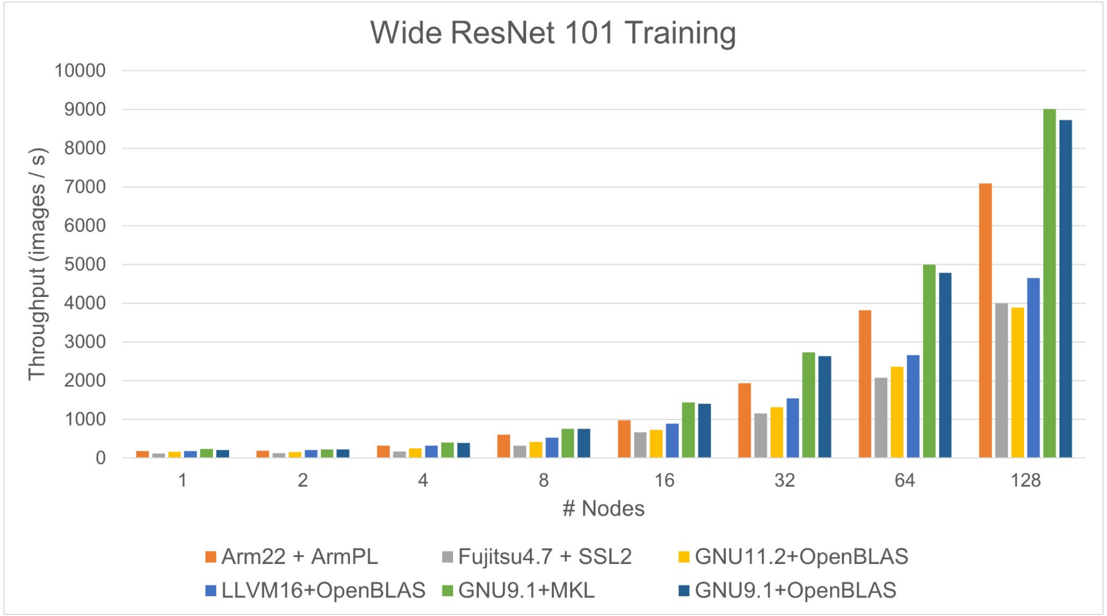
Training Throughput (higher is better)

Better scaling observed on Stampede2 cluster

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

ExascaLab Exascale Programming Models Laboratory

Evaluation: Multi Node



Training Throughput (higher is better)

Better scaling observed on Stampede2 cluster; ARM compiler build shows comparable results

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

ExascaLab Exascale Programming Models Laboratory

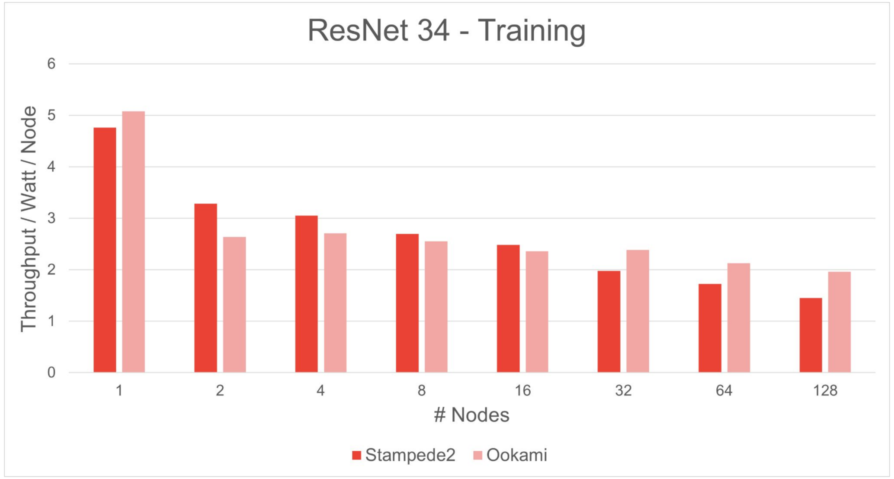
Power: Setup and Disclaimer

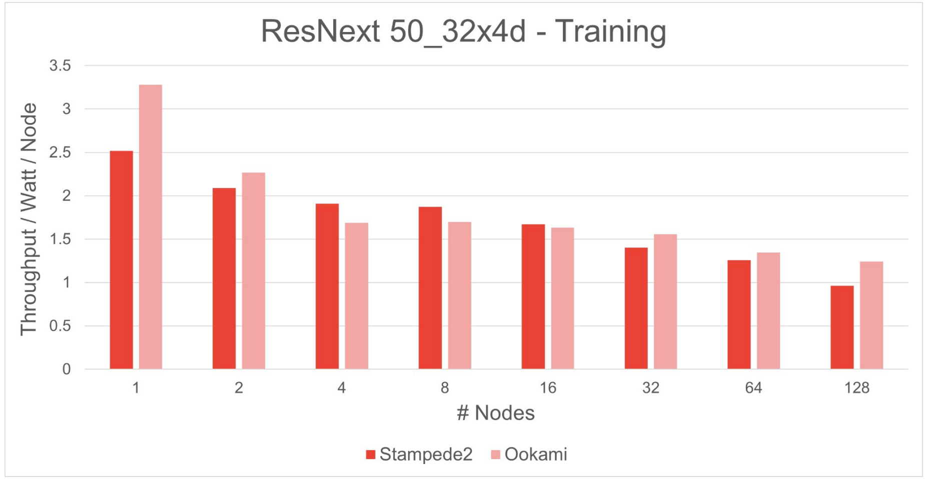
- Multi node runs compared between Intel 8160 (Xeon Platinum) in Stampede2 and Fujitsu A64FX in Ookami. We use the average power to train.

Power Measurement

- Ookami – Open XDMoD interface: Node power obtained from the IPMI DCMI Interface. Does *not* include power usage from network switches, || filesystem and cooling.
- Stampede2 – Intel RAPL interface: Node power obtained by manually calculating the difference in energy consumed per socket. Energy consumed since restart (or offset) found here - **`/sys/devices/virtual/powercap/intel-rapl/intel-rapl:{0,1}/energy_uj`**

Power: Comparative Study

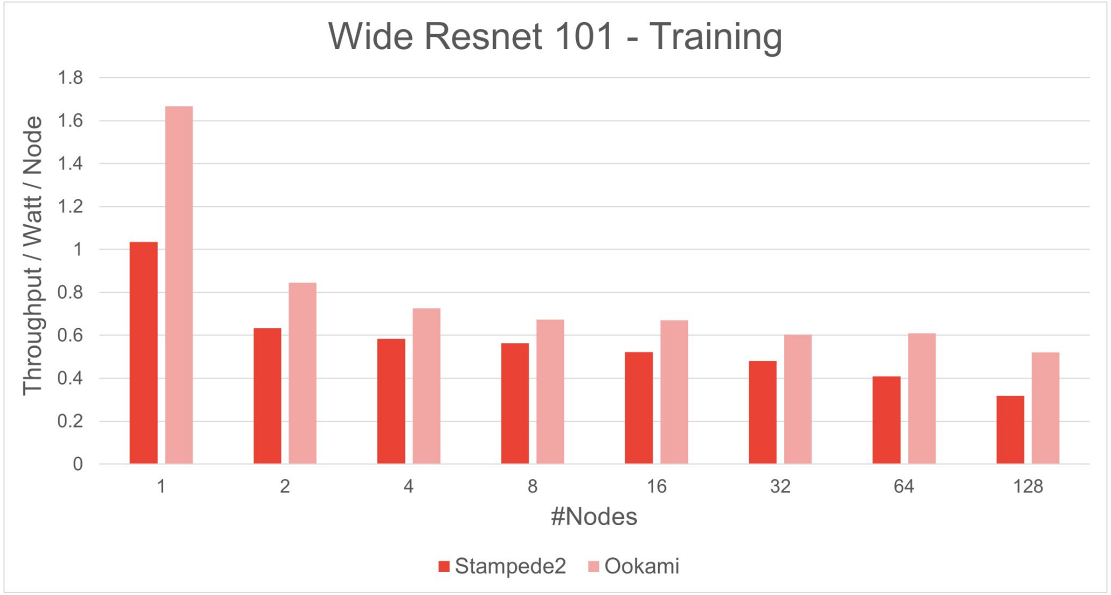




Training (images per second) throughput per Watt per Node

Higher is better

Power: Comparative Study



Training (images per second) throughput per Watt per Node

Higher is better

Conclusion

- A64FX processor can be used to scale ML workloads, and a tuned environment plays a crucial role.
- Power measurements on Ookami are fairly consistent. *Spikes are not observed*

  Power measurements on Skylake are dependent on Intel RAPL and have higher variability in measurements.
- While A64FX in Ookami is not as performant as Skylake-X in Stampede2, the reported power consumption is lower and overall the throughput per watt is generally higher for the A64FX processor.

- 512 bit SVE optimized kernels perform comparably with AVX512 optimized kernels in benchDNN.
- The VNNI instructions of the extended AVX512 ISA show significant reduction in processing time.

Acknowledgments

The authors would like to thank Stony Brook Research Computing and Cyberinfrastructure, and the Institute for Advanced Computational Science at Stony Brook University for access to the innovative high-performance Ookami computing system, which was made possible by a 5M National Science Foundation grant (**#1927880**).

The software was tested, in part, on facilities run by the Scientific Computing Core of the Flatiron Institute.
The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper.

The authors would like to thank the Ookami project and the community for their support. The authors also thank Géraud Krawezik for providing access to compute resources and helpful discussions in setting up the environment at the Flatiron Institute.

We thank the Stony Brook HPC team and XDMoD team at U. Buffalo for their support and help with understanding power measurements.We thank Kent Milfield @ TACC for suggesting power calculations using reported energy consumption in RAPL.

Thank you!

schheda@cs.stonybrook.edu