MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Performance Characterization of DNN Training using TensorFlow and PyTorch on Modern Clusters

Arpan Jain

Network Based Computing Laboratory (NBCL)

Dept. of Computer Science and Engineering , The Ohio State University

{jain.575, awan.10, anthony.301}@osu.edu, {subramon, panda}@cse.ohio-state.edu

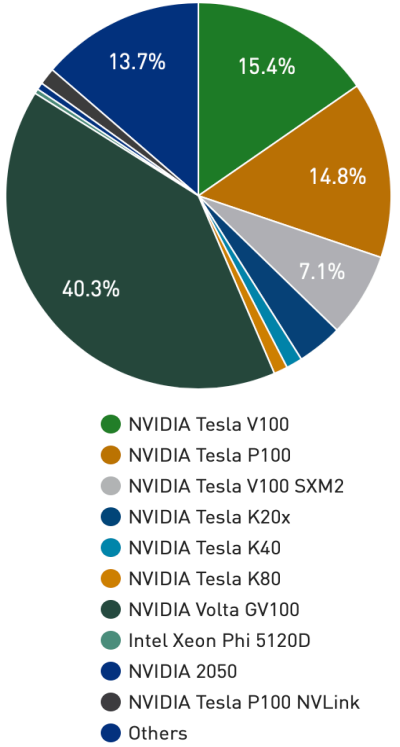
Agenda

- **Introduction**
- Background
- Research Challenges
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental Setup
- Performance Evaluation
- Conclusion

Deep Learning, CPUs, and GPUs

- *NVIDIA GPUs - main driving force for faster training of Deep Neural Networks (DNNs)*
- The ImageNet Challenge - (ILSVRC)
  - DNNs like AlexNet, ResNet, and VGG
  - 90% of the ImageNet teams used GPUs in 2014*
  - *And, GPUs are growing in the HPC arena as well!*
    – Top500 (June '19)
  - *CPUs still dominates HPC arena and can be used for Deep Learning*

Accelerator/CP Family Performance Share



https://www.top500.org/

*https://blogs.nvidia.com/blog/2014/09/07/imagenet/

Deep Learning Frameworks

- Easily implement and experiment with Deep Neural Networks
  - Several Deep Learning (DL) frameworks have emerged
- Caffe, PyTorch, TensorFlow, MXNet, and counting....
  - *Focus on **TensorFlow and PyTorch***
- Most frameworks - optimized for NVIDIA GPUs–
  - but CPU optimized implementations are also emerging
- Distributed Training support – still in infancy!

Deep Learning and TensorFlow

- The most widely used framework open-sourced by Google
- Replaced Google's DistBelief[1] framework
- Runs on almost all execution platforms available (CPU, GPU, TPU, Mobile, etc.)
- Very flexible but two major issues
  - Performance compared to other frameworks
  - Distributed Training support – fragmented (NCCL, MPI, Horovod, gRPC, etc.)
- https://github.com/tensorflow/tensorflow

Courtesy: https://www.tensorflow.org/

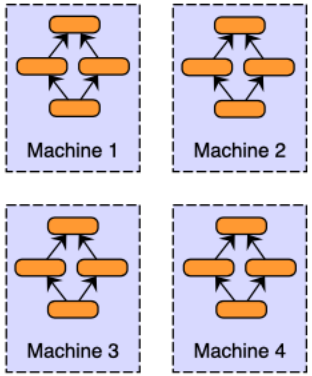
[1] Jeffrey Dean et al., "Large Scale Distributed Deep Networks" https://static.googleusercontent.com/media/research.google.com/en//archive/large_deep_networks_nips2012.pdf

Agenda

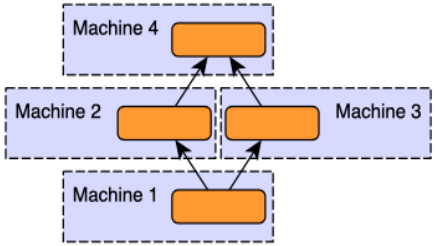
- Introduction
- **Background**
- Research Challenges
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental Setup
- Performance Evaluation
- Conclusion

Distributed DNN Training

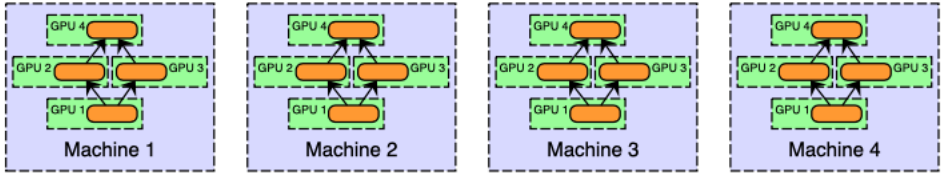
- Deep Neural Network training consists of two phases
  - Forward pass
  - Backward pass
- Training is a compute intensive task
  - Large datasets
  - Complex and large Deep Learning Models
  - Multiple iterations over dataset to reach SOTA
- Three approaches to Distribute DNN training
  - Data Parallelism (focus of this paper)
  - Model Parallelism
  - Hybrid Parallelism



Data Parallelism



Model Parallelism



Hybrid Parallelism

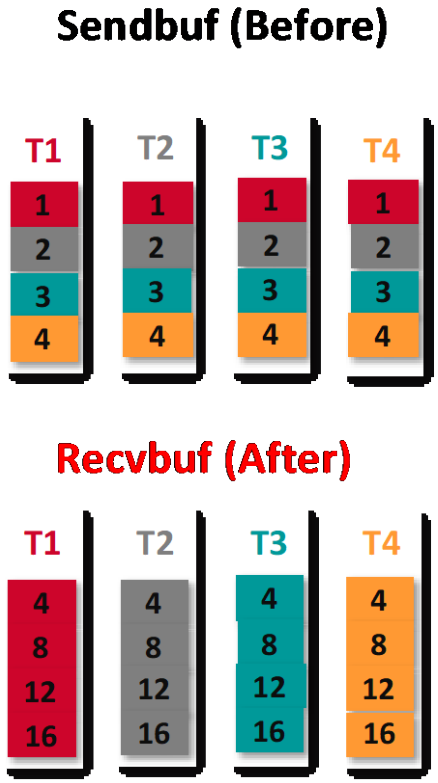
Courtesy: https://blog.skymind.ai/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks/

DL Frameworks and Communication Libraries

- Most ML/DL frameworks – started single-node/single-GPU design
  - Various multi-node design schemes have emerged since then!
- Distributed Training needs **communication libraries** to synchronize across nodes
- **DL Frameworks**
  - Caffe – single-node
  - Cognitive Toolkit – MPI-based from Day 1!
  - **TensorFlow and PyTorch with Horovod (focus of this paper)**
- **Communication Libraries** for DL
  - MPI Libraries: MVAPICH2, IntelMPI, OpenMPI
  - NVIDIA NCCL (GPU only)

What is Allreduce? And How DL frameworks use it?

- A generic group communication pattern – element-wise vector sum available to all participants in the group
- In the MPI world, we call it MPI_Allreduce
- Needed in DNN Training during gradient aggregation from different workers
  - Horovod uses several optimization strategies like Tensor Fusion to overlap compute and computation
- Horovod can be configured with MPI, IBM's DDL, Facebook Gloo, and NCCL



Agenda

- Introduction
- Background
- **Research Challenges**
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental Setup
- Performance Evaluation
- Conclusion

Broad Challenge

How to systemically characterize CPU-based DNN Training using TensorFlow and PyTorch at scale? And how to achieve best possible performance for different HPC systems?

Research Challenges

1. What is the impact of the number of cores of a CPU on the performance of DNN Training?

2. Does the batch size (BS) used to train a DL model impact the training performance on different CPU architectures?

3. Do single process (SP) per node and multiple processes (MP) per node configurations exhibit different perfor-mance behavior?

4. What are the performance trends for TensorFlow multiple nodes?

Let's characterize "Distributed Training on CPUs"!

Key Contributions

- Describe single-process (SP), multi-process (MP), and multi-node (MN) approach
- Highlight up to 1.47× better performance for MP approach over SP approach
- Evaluate five DNN architectures at scale (128 Xeon Skylake nodes)
- Report 125× speedup on 128 nodes for ResNet-152 with MVAPICH2
- Summarize key insights gained from the systematic characterization

Agenda

- Introduction
- Background
- Research Challenges
- **Characterization Strategy**
  - **Evaluation Platforms and Software Libraries**
  - Experimental Setup
- Performance Evaluation
- Conclusion

Evaluation Platforms

Architecture	Cluster	Speed (GHz)	Cores	Threads per core	Label
Skylake	RI2	2.6	28	1	Skylake-1
Skylake	Pitzer	2.4	40	1	Skylake-2
Skylake	Stampede2	2.1	48	2	Skylake-3
Broadwell	RI2	2.4	28	1	Broadwell
EPYC	AMD-Cluster	2.0	32	4	EPYC
K80	RI2	-	4992(Dual socket)	-	K80
P100	Owens	-	3584	-	P100
V100	Pitzer	-	Cuda: 5120 Tensor: 640	-	V100

Software Libraries

- Deep Learning Frameworks
  - Intel optimized TensorFlow (v1.12), -- details on the next slide
  - TensorFlow v1.12 (for GPUs and AMD processors)
  - PyTorch (v1.1)

- Horovod Distributed Training middleware
- MPI Library: MVAPICH2
- Scripts: *tf_cnn_benchmarks* and Horovod's *pytorch_synthetic_benchmarks*

Intel Optimized TensorFlow

- Optimized by Intel for Intel Xeon CPUs
- Uses Math Kernel Library for Deep Neural Networks –(MKL-DNN) primitives
- Can be installed easily using conda and pip
- Performance of tf_cnn_benchmarks can be improved by tweaking

  --num_intra_threads to set the number of TensorFlow's intra-operator (**intra-op threads**)

  --num_inter_threads to set the number of TensorFlow's inter-operator (**inter-op threads**)
- https://github.com/Intel-tensorflow

Agenda

- Introduction
- Background
- Research Challenges
- **Characterization Strategy**
  - Evaluation Platforms and Software Libraries
  - **Experimental Setup**
- Performance Evaluation
- Conclusion

Experimental Setup

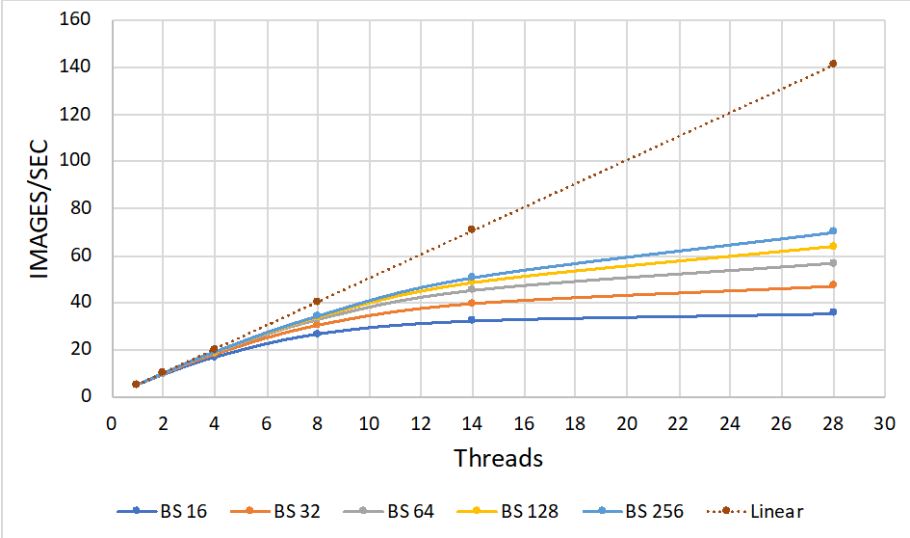
- Broadly, we perform four different types of experiments
  1. Single Node Single Process (SP) Experiments
  2. Single Node Multi-Process (MP) Experiments
  3. Multi-Node Multi-Process (MN) Experiments
  4. GPU vs. CPU Comparisons

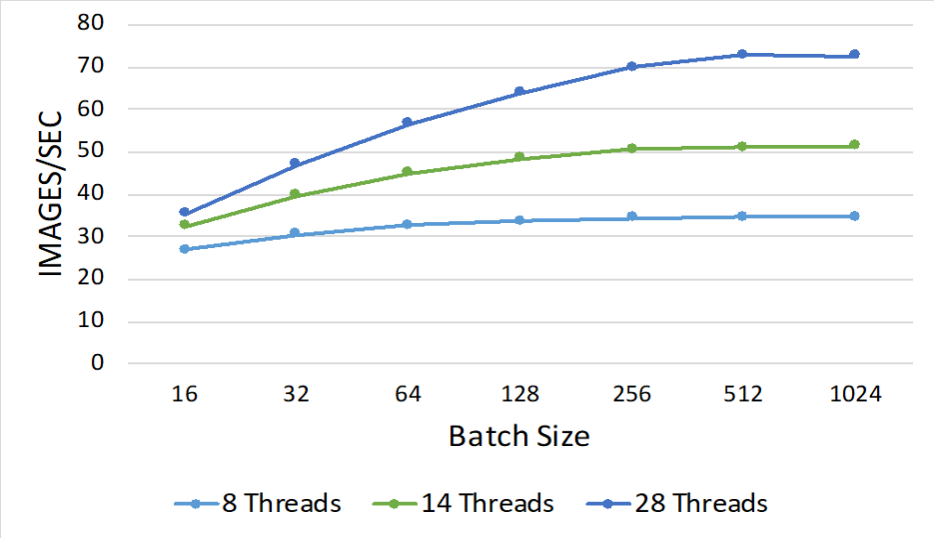
- In the end, we provide useful guidelines to extract best performance

Agenda

- Introduction
- Background
- Research Challenges
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental setup
- **Performance Evaluation**
- Conclusion

Single Node Single Process (SP) Experiments





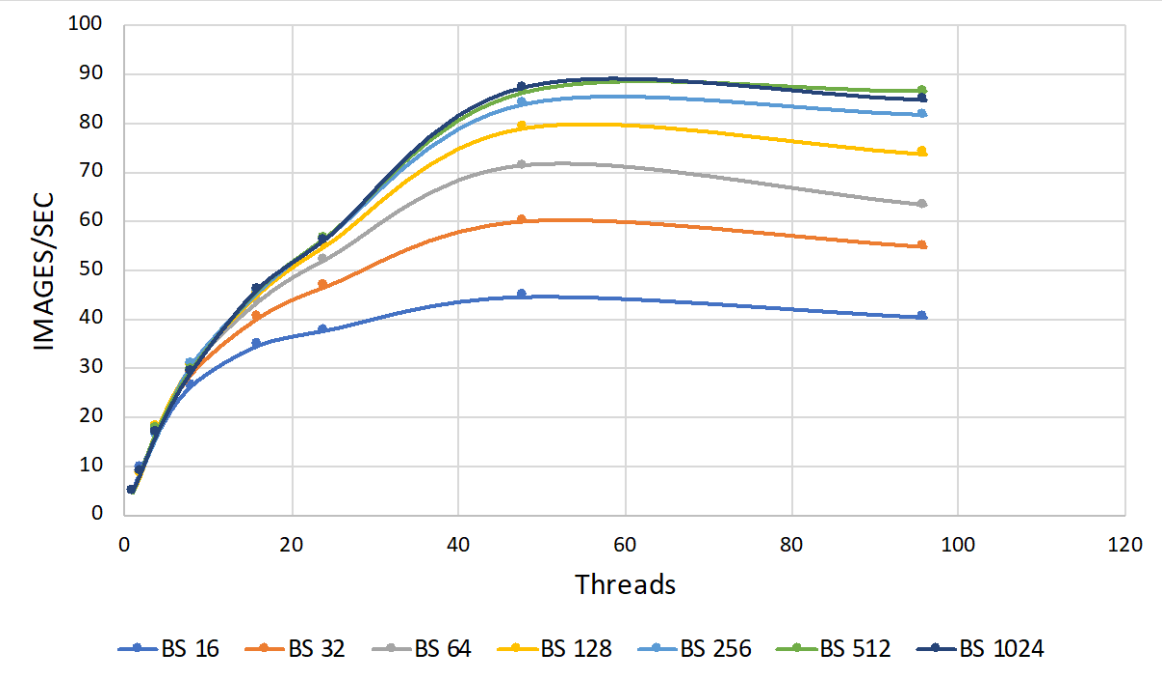
ResNet-50 Training performance

- Different configurations lead to different performance trends
- **Key Message: Process per node (PPN), Batch Size, and number of threads are tunable parameters**
- Parameters need to be determined and tuned properly!

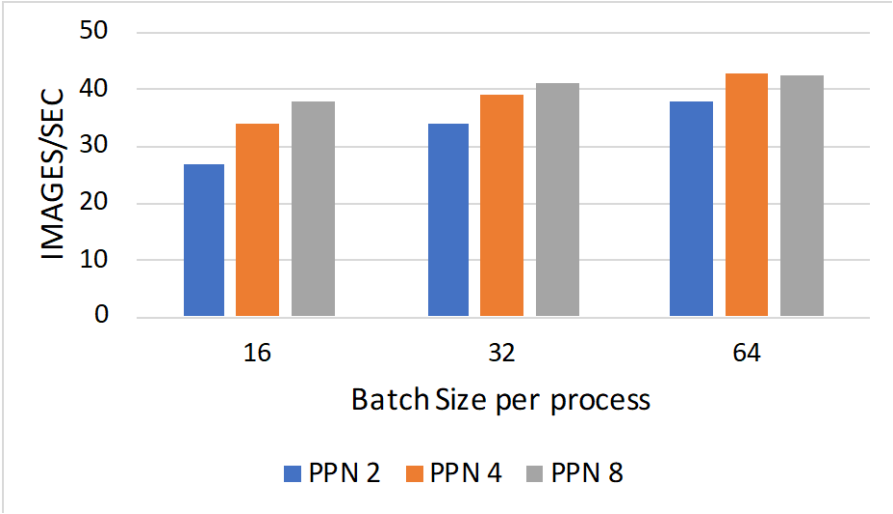
SP: Effect of Hyper-Threading

ResNet-50 Training performance

- Skylake-3 on Stampede2 is hyper-threaded (two threads per core)
- Possible to run TF on 96 threads
- But, performance degrades beyond 48 threads
  - **Why?**
    - Depends on the size and type of DNN

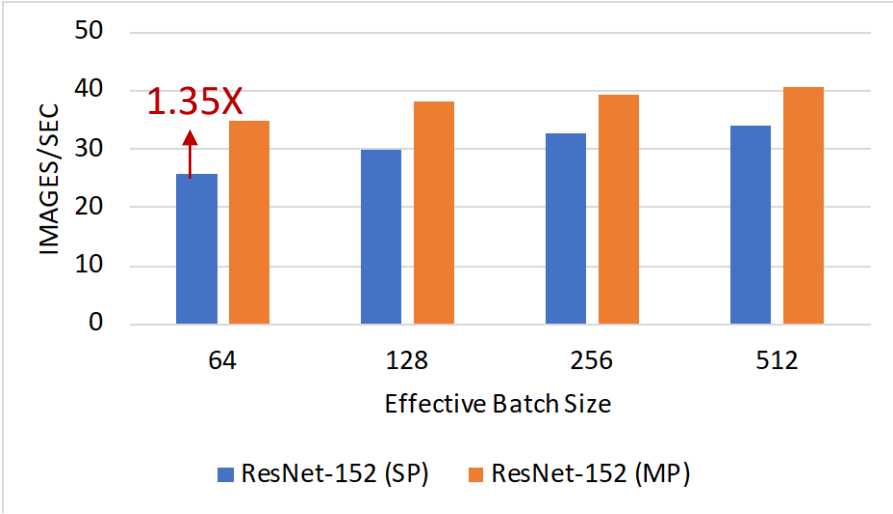


Single Node Multi-Process (MP) Experiments



ResNet-152 Training performance

- BS=64, 4ppn is better
- BS=32, 8ppn is slightly better
- **However, keeping effective batch size (EBS) low is more important! – Why? (DNN does not converge to SOTA when batch size is large)**

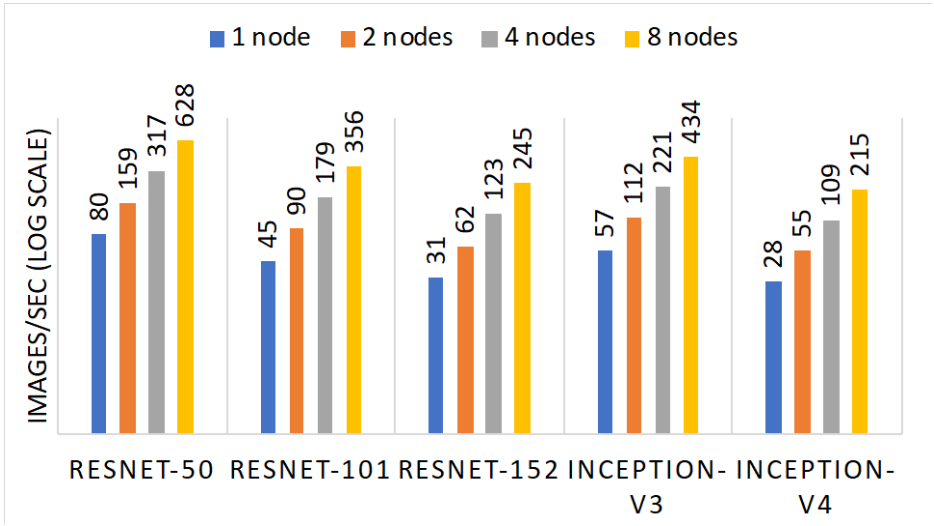


ResNet-152 (SP vs. MP)

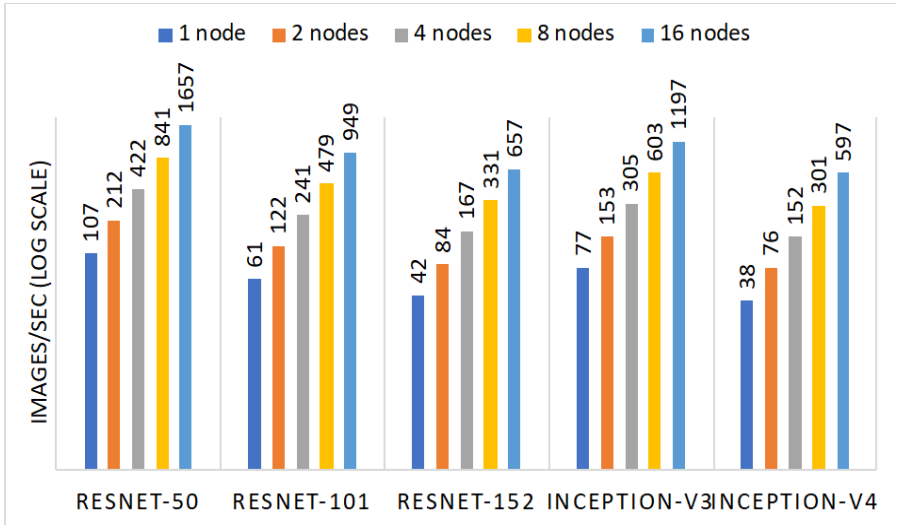
- **MP is better for all effective batch sizes**
- Up to **1.35X** better performance for MP compared to SP for BS=64.

Multi-Node Multi-Process (MN) Experiments

- **We use the best SP configuration to run Multi-node experiments**
- Evaluate five models to identify common trends
  - *All models give near-linear scaling on both platforms*



Skylake-1 (28 cores)

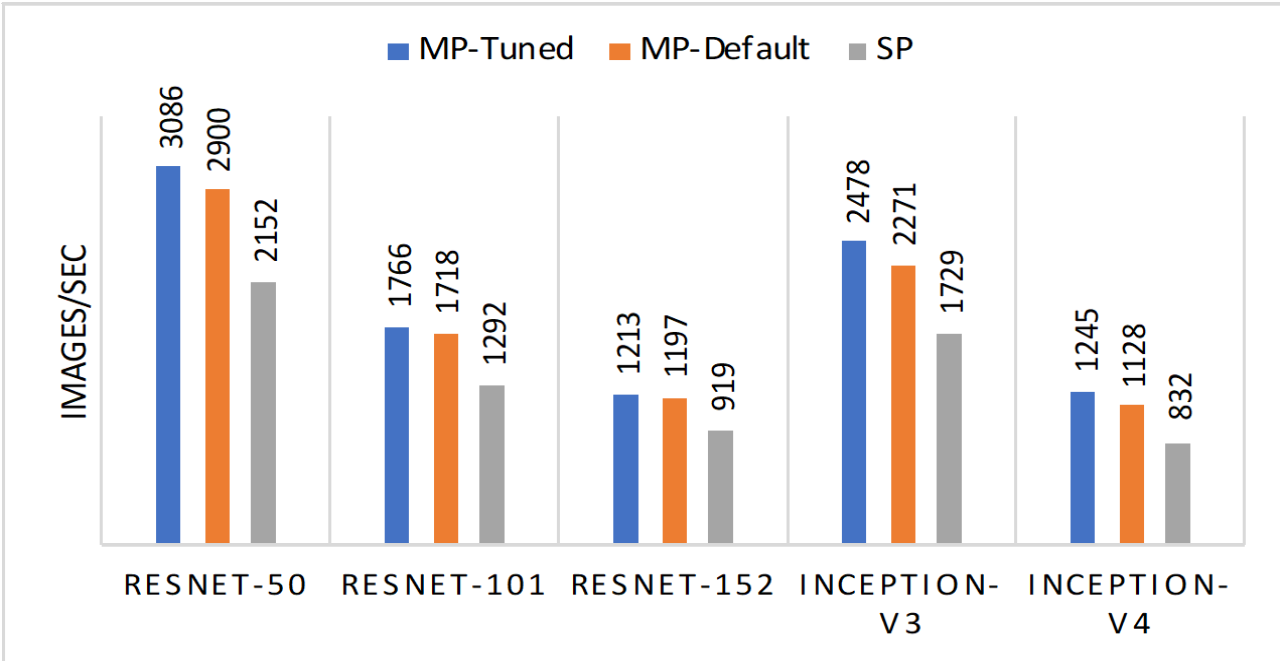


Skylake-2 (40 cores)

Multi-Node Multi-Process (MN): MP vs. SP?

Skylake-3 (48 cores, 96 threads)

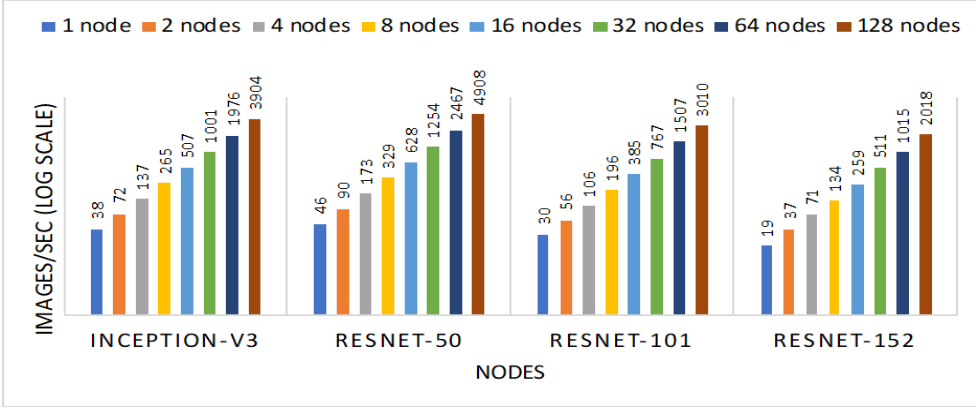
- Scale—32 nodes
- MP-Tuned—up to **1.5X** better than SP
- MP-Tuned—10% better than MP-Default
- **Why MP-Tuned is better?**
  - Uses the best possible number of inter-op and intra-op threads



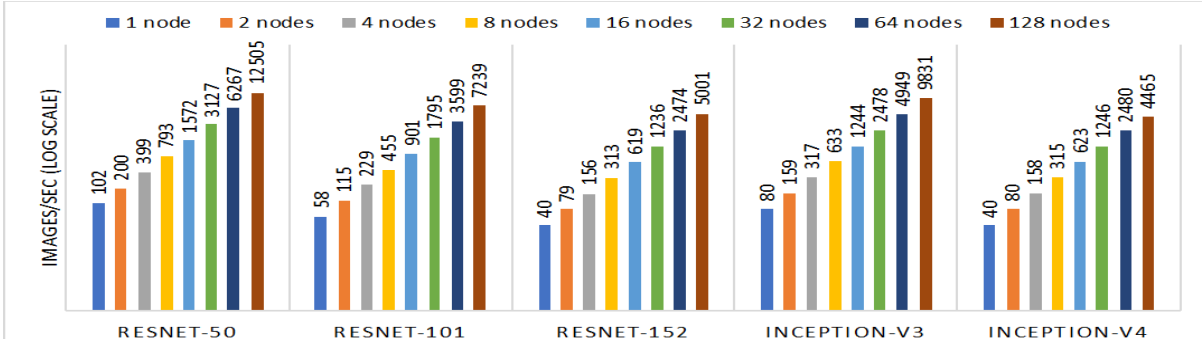
Multi-Node Multi-Process (MN): TF vs. PyTorch

- This is an early experience with PyTorch
- TensorFlow is up to **2.5X faster** than PyTorch for 128 Nodes.
- TensorFlow: up to **125X** speedup for ResNet-152 on 128 nodes
- PyTorch: Scales well but overall lower performance than TensorFlow

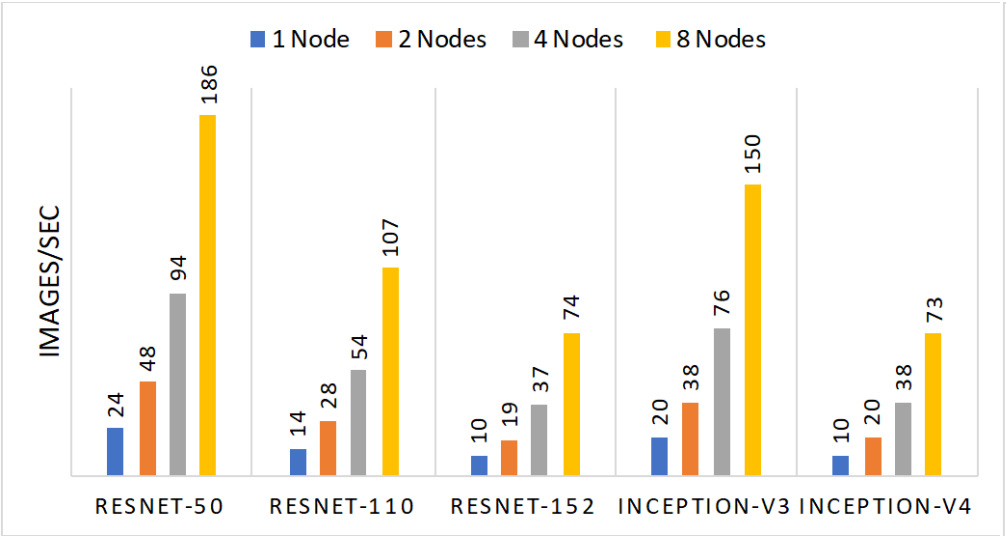
PyTorch



TensorFlow

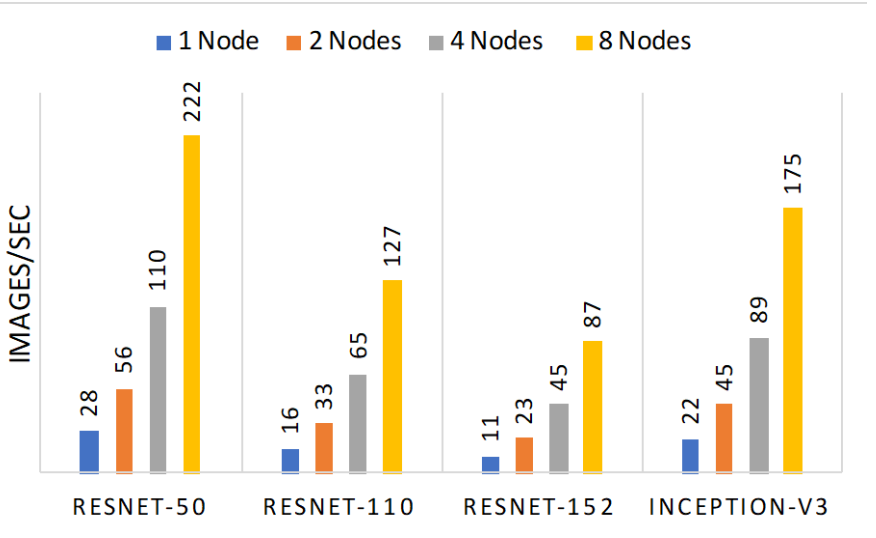


Multi-Node Multi-Process (MN): AMD Platform



EPYC for TensorFlow

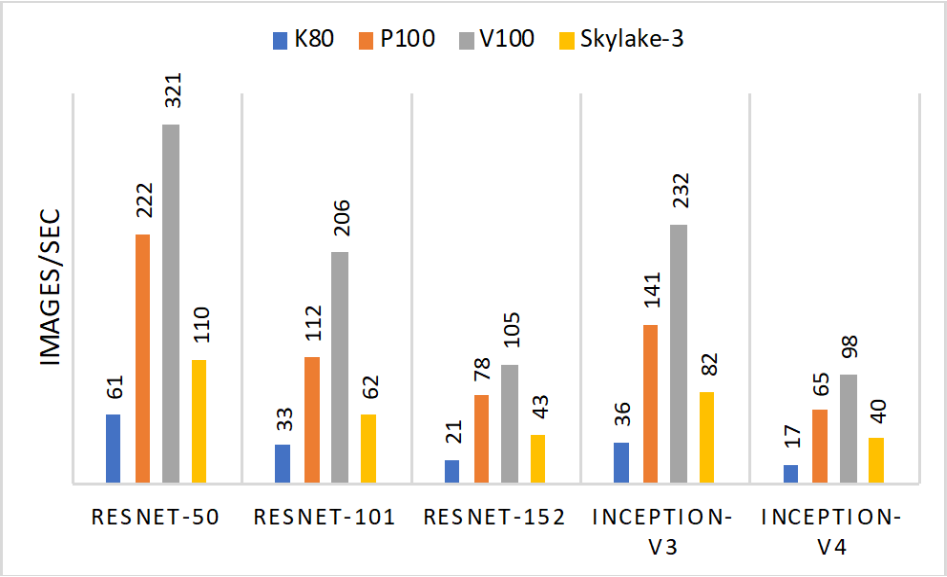
- TensorFlow is **4X slower** on EPYC compared to Skylake-3
- For EPYC, there is no optimized TensorFlow

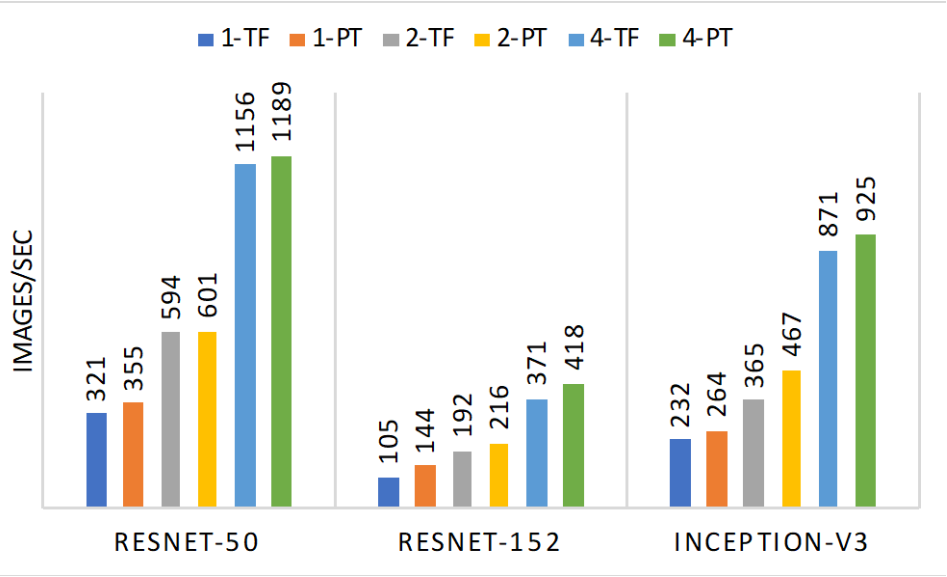


EPYC for PyTorch

- PyTorch—better than TensorFlow
- Up to **19% better** than TensorFlow on 8 nodes.

TensorFlow and PyTorch: CPU vs. GPU





TensorFlow on GPUs vs. CPUs

- Inception-v4 : Skylake-3 up to 2.35X faster than K80s
- ResNet-101: V100s up to **3.32X faster** than Skylake-3

Multi-Node: TensorFlow (TF) vs. PyTorch (PT)

- ResNet-50: PT slightly better TF
- ResNet-152, PT up to **12% better** than TF

Agenda

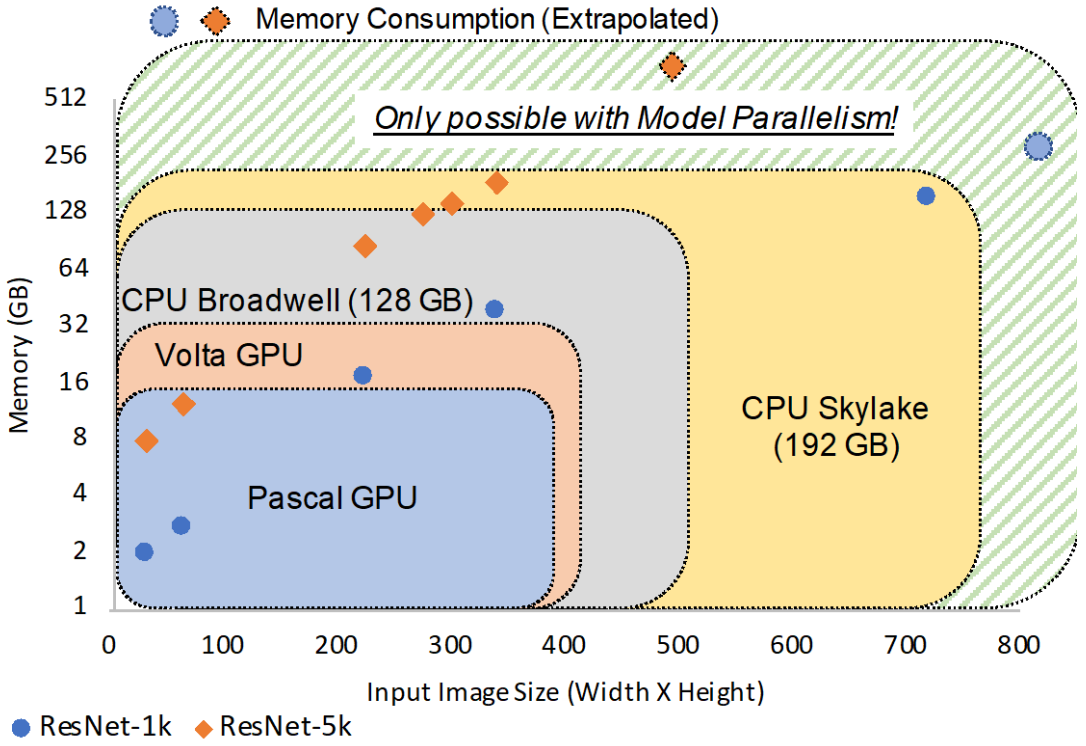
- Introduction
- Background
- Research Challenges
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental setup
- Performance Evaluation
- **Conclusion**

Conclusion

- In-depth Characterization for Distributed Training with TensorFlow and early results for PyTorch
  - Experiments on **five HPC clusters** including **Stampede2** and three different CPU architectures: Skylake, Broadwell, and AMD EPYC
  - Single Node Single Process (SP) and Single Node Multi Process (MP) to determine best performance for single node experiments
  - Use best single-node configuration for multi-Node experiments
  - Up to 128 nodes to show DNN training scaling
  - GPU vs. CPU comparisons for both TensorFlow and PyTorch
- Guidelines for the DL Researchers to get best performance on CPU platforms

HyPar-Flow: Hybrid Parallelism for TensorFlow

- Why Hybrid parallelism?
  - Data Parallel training has limits! →
- We propose HyPar-Flow
  - An easy to use Hybrid parallel training framework
    - Hybrid = Data + Model
  - Supports Keras models and exploits TF 2.0 Eager Execution
  - Exploits MPI for Point-to-point and Collectives

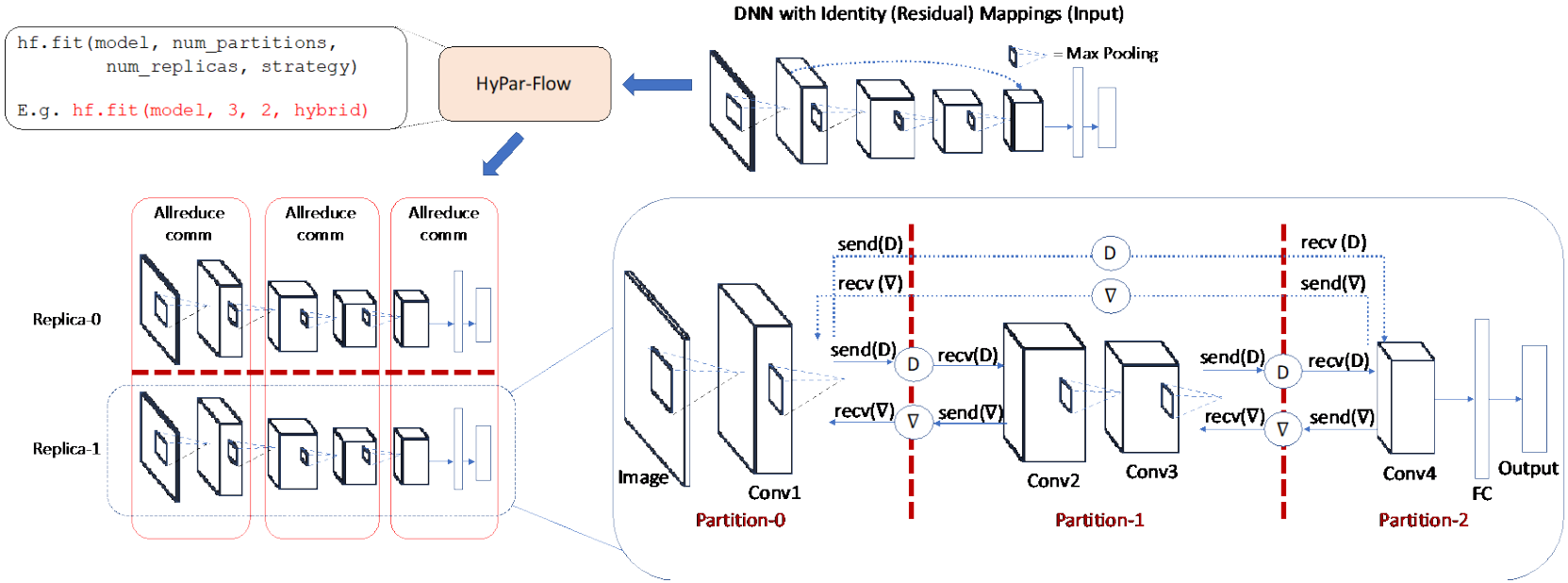


Benchmarking large-models lead to better insights and ability to develop new approaches!

*Awan et al., "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", arXiv '19. https://arxiv.org/pdf/1911.05146.pdf

HyPar-Flow: Design Overview

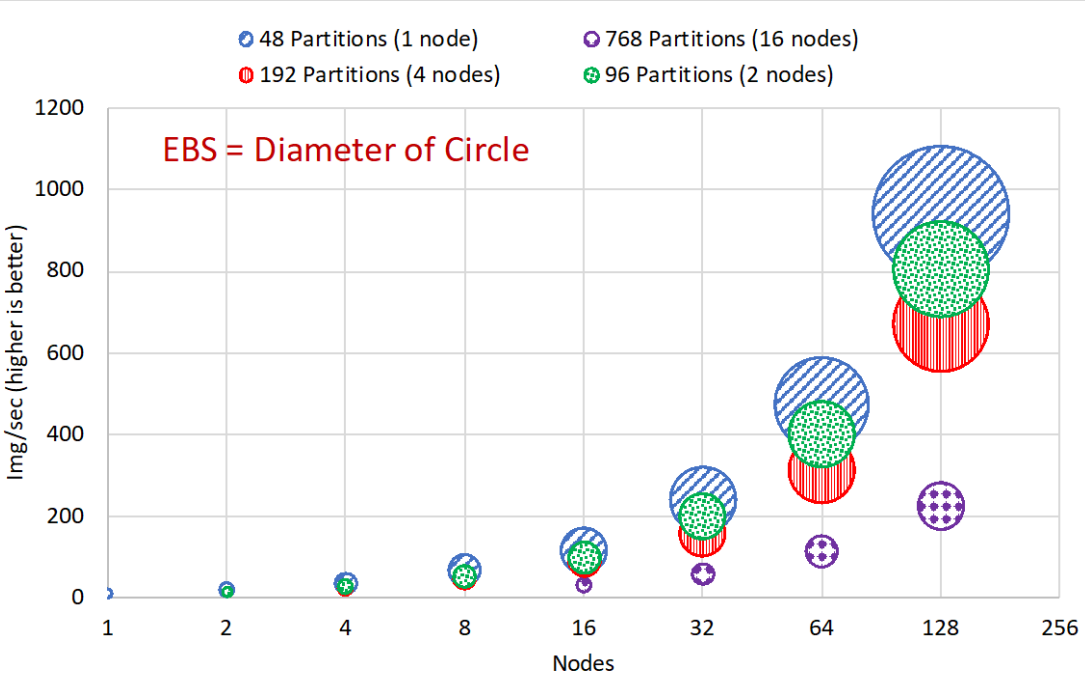
- HyPar-Flow: easy to use Hybrid parallel training framework
  - Supports Keras models and exploits TF 2.0 Eager Execution
  - Exploits MPI Pt-to-pt and Collectives for communication



*Awan et al., "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", arXiv '19. https://arxiv.org/pdf/1911.05146.pdf

HyPar-Flow (HF): Hybrid Parallelism for TensorFlow

- CPU based results
  - AMD EPYC
  - Intel Xeon
- Excellent speedups for
  - VGG-19
  - ResNet-110
  - ResNet-1000 (1k layers)
- Able to train "future" models
  - E.g. ResNet-5000 (a synthetic 5000-layer model we benchmarked)



110x speedup on 128 Intel Xeon Skylake nodes (TACC Stampede2 Cluster)

*Awan et al., "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", arXiv '19. https://arxiv.org/pdf/1911.05146.pdf

Thank You!

{jain.575, awan.10, anthony.301}@osu.edu, {subramon, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

High Performance Deep Learning
http://hidl.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

Multi-Node Multi-Process (MN): Tuning batch size

Skylake-3: 128 nodes

- Impact of BS: *depends on a DNN's type and size*
- ResNet-50: **near-linear increase** in performance as batch size increases